ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΪΑΤΡΙΚΗ

Κατανεμημένοι Υπολογισμοί: Κατανεμημένοι Αλγόριθμοι Ανεκτικοί σε Σφάλματα

Κόκκου Μαρία

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ
Υπεύθυνος
Μάρκου Ευριπίδης
Αναπληρωτής Καθηγητής

Λαμία, 2019

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΙΑΤΡΙΚΗ

Κατανεμημένοι Υπολογισμοί: Κατανεμημένοι Αλγόριθμοι Ανεκτικοί σε Σφάλματα

Κόκκου Μαρία

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Επιβλέπων
Μάρκου Ευριπίδης
Αναπληρωτής Καθηγητής

Λαμία, 2019

Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [1], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

1. *Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί* ***χωρίς να τα περικλείω σε εισαγωγικά*** *και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

2. *Δέχομαι ότι η αυτολεξεί* ***παράθεση χωρίς εισαγωγικά****, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

3. *Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια*

4. *Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.*

Ημερομηνία: ……/…../20……

Ο – Η Δηλ.

(Υπογραφή)

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.

Κατανεμημένοι Υπολογισμοί: Κατανεμημένοι Αλγόριθμοι
Ανεκτικοί σε Σφάλματα

Κόκκου Μαρία

Τριμελής Επιτροπή:

Μάρκου Ευριπίδης, Αναπληρωτής Καθηγητής (επιβλέπων)

Αδάμ Μαρία, Αναπληρώτρια Καθηγήτρια

Λουκόπουλος Αθανάσιος, Επίκουρος Καθηγητής

Περίληψη

Στην παρούσα πτυχιακή εργασία μελετάμε το πρόβλημα του Καθαρισμού Δικτύου από Μαύρο Ιό χρησιμοποιώντας μια ομάδα κινητών ανιχνευτών. Ένας μαύρος ιός είναι μια επιβλαβής διεργασία που βρίσκεται σε έναν αρχικά άγνωστο κόμβο του δικτύου και μπορεί να καταστρέψει οποιονδήποτε ανιχνευτή τον επισκεφτεί. Πιο συγκεκριμένα, όταν ένας ανιχνευτής επισκέπτεται κάποιον κόμβο στον οποίο βρίσκεται ένας μαύρος ιός, ο ανιχνευτής καταστρέφεται και ο μαύρος ιός αντιγράφει τον εαυτό του και μολύνει όλους τους γειτονικούς του κόμβους, αφήνοντας τον αρχικά μολυσμένο κόμβο καθαρό. Αν όμως κάποιοι από τους γειτονικούς κόμβους είναι ήδη κατειλημένοι από κινητούς ανιχνευτές, οι ανιχνευτές καθαρίζουν τα αντίγραφα του μαύρου ιού που επιχειρούν να μολύνουν τους κόμβους στους οποίους βρίσκονται. Ο μοναδικός τρόπος να καθαριστεί ένας μαύρος ιός είναι να περικυκλωθεί από ανιχνευτές, και στη συνέχεια ένας άλλος κινητός ανιχνευτής να επισκεφτεί το μολυσμένο κόμβο αναγκάζοντας τον ιό να επιχειρήσει να μετακινηθεί και κατά συνέπεια να καθαριστεί από τους ανιχνευτές που βρίσκονται ήδη στους γετονικούς του κόμβους. Σε αυτή την εργασία, μελετάμε το πρόβλημα του καθαρισμού από μαύρους ιούς τόσο σε συγχρονισμένους, όσο και σε ασύγχρονους δακτυλίους για μια ομάδα κινητών ανιχνευτών που αρχίζουν από τον ίδιο κόμβο. Προσδιορίζουμε τον ελάχιστο αριθμό κινήσεων και ανιχνευτών που χρειάζονται για τον καθαρισμού ενός συγχρονισμένου δακτυλίου και δίνουμε έναν αλγόριθμο που λύνει το πρόβλημα με το ελάχιστο πλήθος ανιχνευτών και κινήσεων. Στην περίπτωση του ασύγχρονου δακτυλίου, δίνουμε έναν αλγόριθμο που καθαρίζει το δίκτυο σε $O(n)$ κινήσεις, όπου n το πλήθος των κόμβων του δακτυλίου. Ακόμα, μελετάμε την περίπτωση στην οποία οι ανιχνευτές ξεκινούν από διαφορετικούς κόμβους και προσδιορίζουμε τον ελάχιστο αριθμό ανιχνευτών που επαρκούν για να λύσουν το πρόβλημα. Τέλος, δίνουμε έναν αλγόριθμο που λύνει το πρόβλημα του καθαρισμού χρησιμοποιώντας τον ελάχιστο αριθμό ανιχνευτών που αρχίζουν από διαφορετικούς κόμβους σε χρόνο $O(n)$.

Abstract

In this thesis we study the problem of Black Virus Decontamination using a team of mobile agents. A Black Virus is a harmful process that resides in an initially unknown node of the network and can destroy any agent visiting that node. More precisely, when an agent visits a node occupied by a black virus, the agent is destroyed and the black virus copies itself and infects all its neighbours, leaving the node it initially occupied clean. If, however, some of the neighbouring nodes are already occupied by agents, the agents clean the black virus copies that attempt to move to the nodes they occupy. The only way to clean a black virus is to surround it by agents and have another agent visit the infected node, forcing the black virus to attempt to move and consequently be cleaned by the agents already positioned at its neighbouring nodes. In this thesis, we consider the black virus decontamination problem in both synchronous and asynchronous rings with a team of initially co-located agents. We determine the minimum number of moves and agents needed to decontaminate a synchronous ring and provide an algorithm that solves the black virus decontamination problem using the minimum number of agents and moves. In the case of asynchronous rings, we provide an algorithm that solves the decontamination problem in $O(n)$ moves, where n is the number of nodes of the ring. Furthermore, we study the case of initially scattered agents in synchronous rings and provide a lower bound for the number of agents needed. Finally, we present an algorithm that solves the decontamination problem using the minimum number of initially scattered agents in $O(n)$ time.

Contents

1 Introduction

1.1 The Problem

The Black Virus Decontamination problem is the combination of two problems: Black Hole Search and Intruder Capture, both of which have been extensively studied. First, we provide a brief description of those two problems. A more in depth literature review will be presented in Chapter 2.

The Black Hole is a harmful node that destroys any agent visiting it without leaving any trace. A black hole resides at a node of the network not known to the agents in advance. The black hole is static, meaning it cannot infect any additional nodes and it cannot harm agents occupying other nodes of the network. The objective is to find the infected node using the least possible number of agents. Additional cost measures are the time and the number of steps needed to find the black hole.

The Intruder Capture problem, also known as Network Decontamination or Graph Searching, models a threat that can move in the network and is harmful to nodes but not to agents. In the Intruder Capture problem, a harmful mobile agent can move arbitrarily fast in a network infecting the nodes it visits. The objective is for a team of agents to decontaminate the network in finite time.

The Black Virus Decontamination problem, introduced in [1], combines the threat of a harmful node presented in the Black Hole Search problem, with the need for decontamination and the ability to infect additional nodes presented in the Intruder Capture problem. A Black Virus is a malicious entity initially residing at a node of the network. When an infected node is visited by an agent, the agent is destroyed and the black virus copies itself and infects all its neighbours not occupied by agents. The node originally containing the black virus is cleaned. If a black virus copy attempts to infect a node that is already occupied by an agent, the black virus copy is destroyed by the agent. The task is for a team of agents to remove all the black viruses from a network, cleaning it. This can be accomplished in three phases: locating each black virus, surrounding it, and finally having one agent move to the infected node in order to force the black virus to spread to its neighbours where it will be cleaned by the agents positioned there.

1.2 Contributions

The main contributions are presented below:

1. We study the problem in synchronous rings for initially co-located agents. We determine the minimum number of moves needed to decontaminate a synchronous ring and

we provide an algorithm that can decontaminate the ring using the minimum number of moves.

2. We study the problem in asynchronous rings and provide an algorithm that solves the problem using the minimum number of agents.

3. We prove a lower bound for the number of originally scattered agents that are needed to decontaminate a ring initially containing one black virus.

4. We provide a decontamination algorithm for originally scattered agents in synchronous rings that contain one black virus using a minimum number of agents possessing tokens.

1.3 Thesis Organization

The thesis is organized as follows:

- Chapter 2 contains an outline of problems related to the Black Virus Decontamination problem that is the main focus of this thesis. Furthermore, we present the work that has already been done concerning the Black Virus Decontamination. More specifically, we begin by describing the Black Hole Search problem and presenting several of the assumptions under which the problem has been studied. We also describe and discuss the Intruder Capture Problem, also known as Network Decontamination or Connected Graph Searching. Finally we review the already known results concerning the Black Virus Decontamination problem in various topologies.

- Chapter 3 introduces the model we use in the subsequent chapters. This chapter contains a detailed description of the basic notions we use in the rest of the thesis such as synchronous and asynchronous network, face-to-face communication, tokens and mobile agents.

- In Chapter 4 we address the Black Virus Decontamination problem in synchronous rings using initially co-located agents. In this chapter we present an algorithm that solves the problem along with a detailed description. Furthermore, we prove that our algorithm solves the problem in $O(n)$ time using the minimum number of moves, the minimum number of mobile agent casualties and the minimum number of mobile agents needed in total, where n is the number of nodes of the ring.

- In Chapter 5 we discuss the problem of Black Virus Decontamination in Asynchronous Rings using initially co-located agents. We propose a communication model and then present an algorithm that solves the problem. Moreover, we prove that our algorithm

uses the minimum total number of agents as well as the minimum number of agent casualties and solves the decontamination problem at most within $4n - 6$ steps.

- In Chapter 6 we present the Black Virus Decontamination in synchronous rings using initially scattered agents. The use of initially scattered agents has not been studied before in the literature. We propose an algorithm that solves the problem and we prove that the decontamination of a ring that initially contains one black virus requires at least ten mobile agents in the worst case. Next, we prove that our algorithm can always decontaminate the ring using ten agents in $O(n)$ time.

- Chapter 7 summarizes the main results of this thesis.

2 Related Work

2.1 Black Hole Search

The Black Hole Search problem was introduced in [2]. A Black Hole (BH) is a malicious entity that destroys any agent that visits it without leaving any trace. The Black Hole Search Problem (BHS) assumes the existence of either one or multiple black holes in a network and the goal is to determine the location of the black hole or the black holes using a team of mobile agents. In order to solve the problem at least one agent needs to survive knowing the location of the black hole. The agents can only infer the position of a black hole if it is located in the last unexplored node of the network or due to the loss of other agents in it. Therefore, the number of surviving agents after the location of the black hole can be smaller than the initial number of agents in the network. Since an agent lost in a black hole does not leave any trace, various communication methods are used between the agents in order to determine the position of the black hole.

The Black Hole Search problem has been studied with respect to the synchronicity of the network (synchronous, asynchronous), the initial location of the agents (co-located, scattered), the type of graph the agents operate in and the communication method used. Four communication methods are commonly used:

1. Whiteboards: The agents can leave messages for other agents at the nodes they visit.
2. Pure tokens: The agents can place tokens at the nodes they visit.
3. Enhanced tokens: The agents can place tokens at the nodes or edges they visit.
4. Timeout Mechanisms: The agents operating in a synchronous network can use the synchronicity and have one agent explore a new node. After some predetermined time the agent that visited the unsafe node is expected to meet another agent that is located at a safe node. If the agent that visited the new node does not return to the safe node, the position of the black hole can be inferred.

Although the BHS problem was originally studied in rings, it has also been studied in various other topologies in [3, 4, 5, 6]. Moreover, the black hole search problem has been studied in arbitrary networks, for example in [7, 8]. The most powerful communication mechanism among the agents are the whiteboards placed on every node of the network. The BHS problem has been studied in asynchronous networks for initially co-located agents that communicate using whiteboards in [6, 9, 10, 11], for agents that communicate using the enhanced token model in [12, 13] and for agents that communicate using the pure token

model [14, 15]. In the synchronous case the agents can detect a black hole without the use of the three communication methods described above. Instead the agents can employ a timeout mechanism as described in [8, 16]. Finally, the computational complexity of searching for a black hole, as well as hardness and approximation results are found in [17, 18, 19].

2.2 Intruder Capture

The Intruder Capture problem was introduced in [20]. In this problem it is assumed that a potentially harmful and invisible agent, called the intruder, is present in a network. The intruder can move from a node to a neighbouring unoccupied one with arbitrary speed infecting all the nodes it visits. Every node of the network is initially considered to be contaminated. The objective is for a team of mobile agents, that cannot be harmed by the intruder, to capture it. The intruder is captured when it comes in contact with an agent. The problem has been studied in various topologies such as trees [20, 21], hypercubes [22], tori and chordal rings [23]. The problem has also been studied for arbitrary networks in [24, 25].

2.3 Black Virus Decontamination

The Black Virus Decontamination problem (BVD) combines the threat of a Black Hole with the need for decontamination presented in the Intruder Capture problem. A Black Virus (BV) is a malicious entity that destroys every agent that visits it and subsequently clones itself and infects its neighbouring nodes. When a Black Virus clone attempts to spread to a node occupied by an agent, the black virus is immediately cleaned by the agent and the agent is not harmed. The objective is to clean all the nodes of the network, with the minimum number of casualties.

The BVD problem was introduced in [1] where the problem was studied in specific topologies: *q-grids, q-tori* and *hypercubes*. The problem was considered for a team of initially co-located mobile agents that are injected somewhere in the network and a different solution protocol was given for each topology. The main idea used in every strategy is to surround all the neighbouring nodes of the black virus and sacrifice an agent by making it move to the infected node in order to clean it. Furthermore, all solutions use two basic techniques: safe-exploration and shadowing. When a previously unexplored node is visited for the first time by an agent, all its explored neighbours are guarded by agents in order to minimize the spread of the black virus. This procedure is referred to as *Shadowing*. *Safe-exploration* is executed by two agents and is used to locate the black virus. One of the agents, the Explorer Agent (EA) goes to an unexplored node v while the Leader Explorer Agent, LEA, waits on

the last explored node (incident to v) for the EA to return. If v contains the black virus the EA is destroyed. Furthermore, the LEA is aware of the position of the black virus since instead of the EA a black virus clone returns to the node it occupies. When LEA learns the position of the black virus, the second phase of the algorithm begins. During the second phase all the shadowing agents surround the infected node. After all the neighbours of the infected node are occupied by agents, another agent moves to the black virus and cleans it. Next, in [26] the BVD problem was studied in arbitrary networks, where the agents have a map of the network. Once again the solution protocol uses two main techniques: shadowed exploration and elimination. Shadowed exploration includes the search for the black virus while keeping all the already explored nodes safe. Elimination is the final phase, in which all the neighbours of the infected node are occupied by agents and one agent moves to the black virus to clean it. In addition to those two phases, since the network is arbitrary, a path needs to be constructed indicating the order in which the nodes of the network will be explored by the agents. The optimal exploration sequence is found by selecting a node incident to the already explored nodes so that the residual degree of the selected node is smaller than a threshold. If no such node exists, the threshold is increased. In [27] a protocol providing a distributed optimal solution for arbitrary graphs is presented. The main difference in the model used in [27] is that the agents do not have a map of the network. However, the agents have "2-hop visibility" meaning they can see the nodes neighbouring to the node they currently occupy at distance two. The solution includes once more a shadowed-exploration and an elimination phase. In this case the shadowed exploration phase is also the phase in which the agents create a map of the network. Furthermore, this approach considers two types of black virus clones: *sterile* and *fertile*. In the case of sterile clones the clones cannot spread when visited by an agent. However they can be cleaned in the same way as any black virus. Fertile clones are the clones that maintain the same properties as the original black virus. Finally, a greedy strategy and a threshold strategy are described. In [28] the problem was considered for arbitrary networks that contain multiple black viruses for the first time. Once again, the techniques of shadowed exploration and elimination are employed. Furthermore, the agents do not have a map of the network but are provided with 2-hop visibility. In [28] both the cases of fertile and sterile clones are investigated. Moreover, in [29] the author presents an experimental approach to the black virus decontamination problem. Finally, the BVD problem has been studied in chordal rings in [30] and parallel strategies for the decontamination were given in [31]. In Table 1 we present the main results described above.

Network	Number of Infected Nodes	Number of Agents Needed	Number of Moves
q-Grid	$q+1$	$3q+1$	$O(m)$
q-Torus	$2q$	$4q$	$O(m)$
Hypercube	$logn$	$2logn$	$O(nlogn)$
Arbitrary (Greedy)	$\Delta+1$ (sterile), 2Δ (fertile)	Δ	$O(\Delta n^2)$
Arbitrary (Threshold)	$\Delta+1$ (sterile), 2Δ (fertile)	Δ	$O(\Delta^2 n)$
Arbitrary (Multiple BVs)	$d * n_{BV}$(sterile), $n-1$ (fertile)	$\Delta * n_{BV}$(sterile), $O(n)$(fertile)	

Table 1: Summary of Results (where q is the dimension of the graph, n is the number of nodes, m is the number of edges, Δ is the maximum degree in the graph, n_{BV} is the number of black viruses producing sterile clones and d is the degree of $d-$regular graphs)

3 Model

3.1 Network

The network the agents operate in is an $n-$node ring, R. The nodes of the ring are anonymous (i.e., they do not have distinct identifiers) and identical. However, all edges incident to a node have distinct port numbers, visible to the agents. This means that the ring is consistently oriented. The ring initially contains k mobile agents and one black virus. We consider both the case of a synchronous network and the case of an asynchronous network. In the case of a synchronous ring, an agent needs one unit of time to move from a node to a neighbouring one and the time needed for the agent to compute its next move is negligible. In the case of an asynchronous ring the move from a node to a neighbouring one may take an arbitrary but finite amount of time. Notice however that although an agent may take an unpredictable amount of time to start moving, the move itself is instantaneous. The communication of the agents at a node of an asynchronous ring is handled by the host in the following way. Each node contains k bits corresponding to each of the k agents. When an agent enters a node, its corresponding bit is set to one, and all agents at the node are notified instantly. Similarly, when an agent leaves a node, its corresponding bit becomes zero and all agents at the node are notified. Consequently, an agent knows how many and which agents are at the node it occupies at any time.

3.2 Agents

Mobile agents are computational entities that operate in the network and are able to move from one node to another. The agents can only move from a node to a neighbouring one. Although the agents do not have distinct identities initially, they can get numeric identities from an ordered set when they are at the same node. In the case of initially co-located agents, we assume that they all take distinct identities before starting to move. In the synchronous case, the agents we employ can always communicate when they are at the same node. More specifically, when two (or more) agents are at the same node they can view each other's *state*, *direction* and *ID*. Furthermore, the agents have constant memory. All the agents execute the same protocol, however they can assume different roles at different times depending on their position in the ring or their identity. In our algorithms we define the following states for the agents:

1. Explorer: An explorer moves in a region it assumes to be safe. A region can be delimited by two other agents, two tokens or an agent and a token.

2. Leader: An agent that is part of a pair or a team of agents and visits a new node first (i.e., before any of the other members of its team).

3. Companion: An agent that is part of a pair or a team of agents and visits a new node exactly one time unit after a leader has visited it.

4. Guard: An agent that is guarding one of the nodes incident to the black virus.

An agent can change states during the execution of the algorithms, however this is not always necessary. In fact, only in the case of initially scattered agents the agents change their state multiple times during the execution of the algorithms provided. Finally, in the case of originally scattered agents, the agents also possess tokens. The tokens allow the agents to have an additional form of communication even when they are not on the same node. The tokens can be placed only on nodes (pure token model) and they can also be removed by the agents. An agent can carry at most one token at any time unit.

3.3 Black Virus

A Black Virus, as we mentioned in the previous sections, is a malicious entity that resides at a node of the network. Similarly to a black hole, a black virus destroys any agent that visits the node it occupies. In addition to the ability to destroy agents that are on the same site, the BV is endowed with reactive capabilities, that is, it remains passive for as long as it is not visited by an agent. When, however, an agent enters an infected node, the black virus clones itself and infects all its unguarded neighbours. Before being destroyed, the mobile agent disinfects the node originally containing the black virus. Therefore, each time an agent visits a node containing a black virus, the number of black viruses in the ring is increased by at most one. If an agent already occupies a node the black virus is attempting to infect, the black virus copy is cleaned by the agent and both the node and the agent remain unharmed. We assume that when an agent and a back virus reach a node simultaneously the agent cleans the black virus and remains unharmed. We further assume that when two black viruses reach the same node, even at different time units, the black viruses merge. That is, each node can contain at most one black virus at any time t. The number of back viruses in the network can only be decreased in the following way. Since every node has exactly two neighbours, two agents occupy both of the neighbours so that the black virus cannot spread to its neighbours, and subsequently a third agent moves to the infected node. The black virus is cleaned by the mobile agent, the agent is destroyed and the two copies of the black virus are cleaned by the two agents placed on the neighbouring nodes. To sum up, when an agent moves to a new node the following cases are possible:

1. The node is clean and no other agent is at the node. In that case, the number of agents and black viruses in the network remains the same.

2. Another agent is located at the node. When two agents are at the same node, even if a black virus is at a node incident to the one occupied by the two agents, since at least one of the neighbours of the black virus is guarded by an agent, the number of black viruses in the ring will not be increased in the next step.

3. A black virus is already at the node. The agent is destroyed and the black virus copies itself and moves to its neighbouring nodes.

4. A black virus arrives at the node at the same time as the agent. The black virus is cleaned and the agent remains unharmed.

4 Synchronous Network

In this chapter we discuss the black virus decontamination problem in synchronous rings that initially contain one black virus in an unknown node of the network. The agents begin in the same node, called the homebase, at time t_0. We remind the reader that in the synchronous ring case, when the agents are initially co-located, the agents can use face-to-face communication when they are at the same node.

4.1 Impossibility Results

Lemma 1. One agent is not enough to decontaminate a ring, even if the location of the black virus and the size of the network are known.

Proof. Let us assume that one agent can decontaminate the ring. An adversary could place the *black virus* on the first node the agent will visit. Therefore, the agent is destroyed before cleaning the network.

Lemma 2. Two agents are not enough to decontaminate a ring, even if the location of the black virus and the size of the network are known.

Proof. In order to decontaminate the network, at least one agent must visit the infected node. There are three possible cases:

1. Both agents visit the node containing the *black virus* simultaneously. Both agents are killed before decontaminating the network.

2. The agents always move maintaining distance one from each other. When one of the agents visits the node containing the *black virus*, the other agent will be at a node neighbouring with the infected. The first agent is destroyed, and one of the clones of the *black virus* is cleaned by the agent still guarding the neighbouring node. However, one more instance of the *black virus* remains in the network and, as we showed in the previous lemma, the one agent remaining is not enough to clean it.

3. The distance between the two agents when one visits the node containing the *black virus* is at least two. The agent visiting the infected node is immediately destroyed by the *black virus*, which clones itself infecting its two neighbouring nodes. Now there is one more agent in the network and two instances of the black virus. The remaining agent is not enough to clean the network.

Lemma 3. Three agents are not enough to decontaminate a ring, even if the size of the ring is known.

Proof. The network cannot be decontaminated, unless at least one agent visits the infected node. We have the following cases:

1. If the algorithm moves only one agent, that agent will at some time t be destroyed by the *black virus*. After the agent's destruction, there are two more agents in the network and either one or two instances of the *black virus*, depending on the infected node's position. As we showed in Lemma 2, two agents do not suffice for the decontamination of the network.

2. If the algorithm only moves two of the agents, in the same direction, at least one of them will at some time t visit the infected node. The agent will be destroyed and the *black virus* will clone itself and infect either both of its neighbouring nodes, or just one, if the second agent that was moving was guarding one of the neighbouring nodes at t. In either case, at least one *black virus* remains and the two agents still in the network are not enough to clean it, even though its location is now known, as we proved in Lemma 2.

3. If the algorithm moves all three agents in the same direction, at some time t at least one of the agents will visit the node containing the *black virus*. Since all agents move in the same direction, at least one of the clones produced by the *black virus* will survive. The remaining instance of the *black virus* cannot be destroyed by the two agents still in the network as we showed in Lemma 2.

4. If the algorithm moves only two of the agents, in opposite directions, one of the agents will be destroyed by the infected node at time t leaving at least one more instance of the *black virus* and two more agents in the network. The two remaining agents are not enough to decontaminate the ring.

5. If the algorithm moves two agents in one direction and the remaining agent in the opposite, at some time t at least one of the agents will be destroyed by the infected node leaving at least one more instance of the *black virus* and two more agents in the network. The remaining agents are not enough to clean the network.

If however the location of the black virus is known to the agents, they can decontaminate the network. This can be accomplished by first having two agents guard the two neighbouring nodes of the node containing the *black virus*, and subsequently, having the remaining agent visit the infected node.

4.2 Algorithm and Algorithm Description

Four agents can decontaminate a ring using the following method. Before describing the method, we need to define the time-out mechanism used in the algorithm given below. In order to safely traverse the ring, the agents need to use a time-out mechanism, resembling the Cautious Walk technique.

Definition 4.1 (Cautious Traversal)**.** One of the agents, referred to as *leader*, moves one step towards a pre-determined direction known to its *companion* at time t, and then waits for one time unit. The *companion* at time t waits for one time unit and moves one step in the direction of the *leader* after the time unit has passed.

In order to find the *black virus* and decontaminate the network, the agents need to form two pairs, containing one *leader* and one *companion* agent each. The two pairs then set off on different directions, which is possible since the ports have discrete IDs visible to the agents. The first part of the algorithm (*Search Procedure*) presented below describes the Cautious Traversal technique, while the second part (*Ring Decontamination Procedure*) refers to the entire decontamination process, from the moment the agents start at the homebase until the *black virus* has been cleaned.

Algorithm 1 Cautious Traversal

```
1:  procedure SEARCH(dir)
2:      found ← 0
3:      if leader then
4:          Walk one step towards dir
5:          Wait(1)
6:      end if
7:      if companion then
8:          Wait(1)
9:          Walk one step towards dir
10:         if leader not on node then
11:             found ← 1
12:         end if
13:     end if
14:     return found
15: end procedure
```

Algorithm 2 Ring decontamination using four agents in a synchronous ring

1: **procedure** RING DECONTAMINATION
2: The two agents with the smallest IDs become leaders, and the remaining agents become companions
3: The agents form two pairs containing one leader and one companion agent each
4: Each pair picks a direction *dir*
5: **repeat**
6: $search \leftarrow Search(dir)$
7: **until** $search = 1$
8: **end procedure**

4.3 Correctness Analysis

Lemma 4. Algorithm RING DECONTAMINATION can always solve the decontamination problem in a ring, using four agents with distinct labels and constant memory, in $O(n)$ time, when the number of nodes and the position of the *black virus* are unknown.

Proof. Forming two different pairs and agreeing on the ring's orientation can be easily done, since the agents have different IDs, start at the same node and the edges incident to a node have different labels. Since the agents have numeric IDs, the two agents with the smallest IDs can become leaders, and the two remaining agents become companions. Furthermore, the leader with the smallest ID, picks a direction at random and defines the clockwise (cw) and counter-clockwise (ccw) direction for all agents. Finally, the pair containing the agent with the smallest ID heads to the clockwise direction, while the remaining pair heads to the counter-clockwise direction. The two pairs start exploring the ring employing the Cautious Traversal technique. The *leader* moves one step towards the assigned direction.

- If there is a *black virus* on that node, the agent is destroyed and the black virus clones itself and moves to its neighbouring nodes. Since the *companion* is one node behind the *leader*, the *companion* will clean one of the clones of the *black virus* trying to spread to the node occupied by the agent.

- If the node is safe (not occupied by the *black virus*), the *leader* waits for one time unit.

The companion moves one step towards the direction of the *leader*.

- If the *leader* is not on that node, the *companion* stops. Now, the companion can safely infer that the leader was destroyed by the *black virus* and therefore the *black virus* is on the next node.

- If the *leader* is on the node as expected, the two agents continue exploring the ring using the cautious traversal technique.

Let us assume that the *black virus* is on a node u and its two neighbouring nodes are v and w. Since the two pairs are moving in different directions, one pair will approach the infected node through v, while the other will approach it through w. Consider the leader that reaches the node containing the *black virus* first. Without loss of generality, let us assume that this leader is reaching the *black virus* node through v. When that leader gets on the infected node at time t, its companion is at node v. At time t, w can be:

1. Occupied by the *companion* of the second pair. This can happen if the two *leaders* arrived simultaneously at the infected node.
2. Occupied by the *leader* of the second pair.
3. Not occupied by an agent.

In the first two cases, since both neighbours of u are occupied, the *black virus* is cleaned when the *leader* of the first pair visits it. In the third case, the *black virus* clones itself after destroying the leader, and spreads to its neighbouring nodes w and v. The clone moving to v is immediately cleaned by the companion positioned there. The remaining instance of the black virus in now on node w. The companion, continuing to move using cautious traversal, moves to u and since its *leader* is not on u as expected, it stops. At some time t_1 the leader of the second pair will visit w, that now contains the black virus, forcing the *black virus* to spread to its neighbours that are both occupied by the two *companion agents*. Both new instances of the *black virus* are destroyed and the network is clean. Since the *black virus* is somewhere in the network and the agents move in specific and opposite directions exploring all nodes, the *black virus* will be located and cleaned during the agents' first traversal of the ring. Consequently, the time needed for the location and decontamination of the *black virus* is $O(n)$.

Observation 1. The network is decontaminated using the algorithm described above, however the surviving *companion agents* are not aware of the time the decontamination occurs. The agents only know that the infected node is located in the unexplored node with distance one of their current position. The algorithm stops when the last *companion* has stopped moving.

Lemma 5. The worst case with respect to the time needed for the decontamination of the ring is when the distance of the *black virus* to the homebase is 1.

Proof. In order to prohibit the *black virus* from spreading, one agent must be placed on each of its neighbouring nodes before the decontamination. The total number of nodes the two guarding agents must traverse is $n-2$, since all the nodes except the homebase and the one containing the *black virus* must be visited. Let x be the shortest distance of the *black virus* to the homebase, therefore $x \in [1, \frac{n}{2}]$. One of the agents will need to traverse $y = n - (x+1)$ nodes, while the other will have to traverse x nodes. Due to the use of the time-out mechanism each node needs two time units to be traversed, so the two agents need $2y$ and $2x$ time units to reach the nodes incident to the infected. Since we have defined x as the shortest distance of the *black virus* to the homebase, y will determine the time needed, so the worst case is when y is maximized. This happens when x gets its minimum value, meaning when $x = 1$.

4.4 Complexity Analysis

Lemma 6. The method described in this section requires $2n-4$ time units.

Proof. The agents explore new nodes in pairs. Each new node needs two time units to be explored. The pairs continue moving until at least one of the agents vanishes at a node, u, at distance x from the homebase. The remaining agent of that pair discovers the loss of the first agent when it moves to u. Up to that moment, x nodes have been explored and cleared towards each direction of the homebase, in addition to the homebase which was safe. Hence, $2x+1$ nodes are explored and cleared in $2x$ time units. Now at least one agent discovers it is the only one of its pair that survived and stays at node u. The other pair continues exploring the remaining $n-(2x+1)$ nodes. The time needed to explore the remaining nodes is $2(n-(2x+1))$ time units. After $2(n-(2x+1))$ time units, an agent of the remaining pair enters a node, v, and discovers that the other agent of the pair has vanished. The agent stops at v and the ring is clear. The total time needed is $2x+2(n-(2x+1)) = 2n-2x-2$. This expression achieves its maximum value when x is minimum. Since the minimum value for x is 1 as we showed in Lemma 5, the time needed by the algorithm is at most $2n-4$ time units. Notice that within at most $2n-4$ time units the ring is cleared, but the agents might not be aware of it. It is possible that only one agent of each pair has survived and all surviving agents eventually stop moving. However, each surviving agent is not aware whether it is the last agent to stop moving and therefore cannot know whether the ring is already clear at the time it stops moving. In the special case of a ring consisting of 3 nodes, although the ring will be cleared within 1 time unit, the agents move for a total time of 4 time units. In this special case the agents know after 4 time units that the ring is clear.

4.4.1 Properties of an algorithm that decontaminates the network

Definition 4.2. *State(u,A,t)* represents the state of node u, for an agent A at time t, when agent A is located at u or at a neighbour of u.

For each agent, we define the following possible states a node can be in at time t:

- *Unsafe:* The agent is not currently at that node, and no other agent had left with the intent to visit that node at time $t-1$.

- *Safe:* The agent is currently on that node, or another agent visited the node at time $t-1$. Note that it is possible the black virus was on the node at time $t-1$, but since the node was visited by an agent it is now clean as the black virus would have moved to the neighbouring nodes.

- *Black Virus:* The agent is aware of another agent's destruction and has inferred the position of the *black virus*. For example, suppose that an agent moves from a node u to an adjacent node v with the instruction to wait there. Suppose also that another agent moves from node u to node v and this second agent does not find an agent at v. If there are no other agents in the network, the agent infers that the black virus is at node $w \neq u$ which is adjacent to v.

Lemma 7. An agent cannot characterize a node as *safe*, for more than one time units, unless it is occupying that node.

Proof. Let us consider the following scenario: let u, v, w and z be four consecutive nodes, A and B be two agents located at u and the black virus be at v. Suppose that w is not occupied by any agent and let z be occupied by another agent, C. Suppose that at some time t, agent B moves to v and hence is destroyed by the black virus. Agent A categorizes v as safe at time t. The black virus clones itself and spreads to u, where it is destroyed by A, and w. Let agent C move to w at time $t_1 = t + 1$. Agent C is destroyed and the black virus spreads to v and z. Consequently, a node cannot be considered safe for more than one time units.

Lemma 8. An agent at node u cannot characterize a node v as black virus if v is not adjacent to u.

Proof. Let u be a node occupied by an agent A, v a neighbour of u containing the black virus at time t and w the other neighbour of v. Let us assume that A has categorized v as *black virus* at time t. Suppose A moves in the direction opposite to the node it has categorized as *black virus* at time $t_1 = t + 1$. Another agent can visit the infected node causing it to spread to its neighboring nodes at t_1.

At the beginning of any algorithm all nodes, except the homebase, are categorized as *unsafe* for all agents. Any algorithm that can decontaminate the network using four agents needs to fulfill the following conditions:

Property 4.1. An unsafe node cannot be visited by more than one agents simultaneously.

Proof. The adversary can place the *black virus* on the unsafe node the agents will visit, destroying them. Since at least two agents were destroyed, there are at most two remaining agents in the network. In view of Lemma 2, two agents are not enough to decontaminate the network.

Property 4.2. An agent cannot safely move to an unsafe node unless there exists at least one more agent at distance one from the unsafe node.

Proof. Let u, v and w be three consecutive nodes on the ring. Let an agent A be at u and let v be an unsafe node and w be either a safe or an unsafe node. Let us assume that no other agent is located at u or w at time t, and let A visit v at time t. If v is the node containing the *black virus*, A will be destroyed, and the black virus will spread to u and w. There are now three agents and two instances of the *black virus* in the network. Three agents are not enough to decontaminate the network, as shown in Lemma 3.

Property 4.3. In order to decontaminate a ring all nodes must be visited by some agent at least once.

Proof. Consider an algorithm that visits $n - 1$ nodes. An adversary can place the *black virus* at the node no agent visits. If the infected node is not visited, the *black virus* will never be cleaned.

Lemma 9. If a node is categorized as *black virus* by an agent A at time t, and A does not move, the black virus cannot move to a different node.

Proof. The only way for the black virus to attempt to spread, is for an agent, B, to visit the infected node. We have showed in Property 4.2 that an agent cannot move to an unsafe node unless it knows that there is one more agent at distance one from the unsafe node. Let us call the second agent at distance one from the unsafe node C. Both of the neighbours of the infected node are occupied by A and C at the time B visits the infected and the network is cleaned. Consequently, if the black virus node is not visited by an agent it cannot spread and if it is visited by an agent, according to Property 4.2 the network will be cleaned. In either case, the black virus will not move to a different node.

Lemma 10. The network cannot be decontaminated until an agent marks the infected node as *black virus*.

Proof. As it was shown in Lemma 8 and Lemma 9, if an agent has categorized a node as *black virus* and it does not move to a different node, the next agent visiting the infected node will decontaminate the network. It is necessary for both of the neighbours of the infected node to be guarded by agents, otherwise the black virus cannot be cleaned. If no agent marks the infected node as *black virus*, it cannot be certain that at any time, both neighbours of the infected node will be occupied simultaneously.

4.4.2 Optimality of our algorithm with respect to time

Any algorithm solving the *black virus* decontamination problem using four agents, needs to satisfy the properties described in the previous section.

Lemma 11. The algorithm we propose is optimal with respect to the time needed to decontaminate the network, for four agents that do not know the size of the ring and the location of the black virus in advance.

Proof. At least two agents and at least two time units are needed in order to safely explore one new node, as we showed in Properties 4.1 and 4.2. Therefore, four agents can explore up to two nodes within every two time units. When, however, one of the agents of a pair vanishes, the other agent should stay at the node where the first agent vanished, otherwise the black virus might eventually spread as we showed in Lemma 9 and Lemma 10. Consequently, the number of nodes which can be explored by the two pairs moving in parallel is at most $2x$, where x is the shortest distance between the homebase and the black virus. This exploration lasts $2x$ time units, while after that time, only one pair can continue the exploration and thus every two time units only one new node is explored. Furthermore, all the nodes in the network must be explored before the network is cleaned as we showed in Property 4.3. Hence, the optimal strategy with respect to time is not different than the strategy of our algorithm.

5 Asynchronous Network

In the case of an asynchronous network, the agents may need a finite but unpredictable amount of time to move to a node. Therefore, we need to determine a mechanism that will enable the agents to distinguish the case of a slow agent and an agent killed by the black virus. We consider the case where the agents may take an arbitrary amount of time to move to another node but the move itself is instantaneous. Furthermore, the communication between the agents is handled by the node and is therefore not subject to delays.

5.1 Impossibility Results

Given the fact that any negative result for the synchronous network remains a negative result in the asynchronous case, we get the following lemma.

Lemma 12. Less than four agents do not suffice to decontaminate an asynchronous ring, even if the size of the ring is known.

5.2 Algorithm and Algorithm Description

Like in the synchronous model, four agents suffice for the decontamination of a ring. Below we present an algorithm that can decontaminate an asynchronous ring using four agents starting at the same node.
The first phase of the algorithm includes the initialization and the traversal of the ring until the black virus is located. In the beginning of the algorithm all agents are located at the same node, called the homebase. The agents take distinct identities from a totally ordered set, such as the natural numbers, and thus assume different roles. The agent with the smallest ID will be denoted as *leader* and will be the first to explore an unexplored node. After the agents have taken their identities, the leader will pick a direction known to all agents and move one step towards it. The remaining agents on the homebase will wait until the leader has left and then two agents will move towards it, while one will remain in the homebase.

Algorithm 3 Ring decontamination using four agents in an asynchronous ring

```
1: procedure INITIALIZATION
2:     Get distinct IDs
3:     Agent with smallest ID is leader
4:     Leader chooses direction, dir
5:     if leader then
6:         Move one step towards dir
7:     else
8:         Wait until leader has left
9:         if you are not the agent with the biggest ID then
10:            Move one step towards dir
11:            CAUTIOUS EXPLORATION(dir)
12:        else
13:            dir ← oppposite_dir
14:            DECONTAMINATE(dir)
15:        end if
16:    end if
17: end procedure
```

The CAUTIOUS EXPLORATION algorithm describes the traversal of the ring until the black virus is located. The leader, will be the first agent to go to an unexplored node. Therefore, it will already be on the node that two of the remaining agents will visit. The two agents will wait until the leader has left and then move one step towards the predetermined direction. If the leader is not at the node as expected, the agents infer its loss and enter the decontamination phase, otherwise they continue to cautiously explore the ring using the same procedure.

Algorithm 4 Ring decontamination using four agents in an asynchronous ring

```
1: procedure CAUTIOUS EXPLORATION(dir)
2:     Let u be the current node
3:     while leader on u do
4:         Wait until 3 agents occupy u
5:         if leader then
6:             Move one step towards dir
7:             u ← current_node
8:         else
9:             Wait until leader has left
10:            Move one step towards dir
11:        end if
12:    end while
13:    dir ← opposite_dir
14:    DECONTAMINATE(dir)
15: end procedure
```

The second phase consists of the decontamination of the ring. Remember that in order to decontaminate the ring, two agents need to first occupy the infected node's neighbours so that the virus cannot spread. After each neighbouring node has been occupied by an agent, another agent needs to sacrifice itself by visiting the infected node. The virus will attempt to spread and will be cleaned by the agents already positioned at the neighbouring nodes. In the decontamination phase, one of the agents that is already in one of the nodes incident to the infected node will remain there, while the other agent will change direction and return to the homebase to meet with the agent that did not participate in the CAUTIOUS EXPLORATION phase. Then the two agents will approach the node containing the black virus through its unoccupied neighbour. The two agents move towards the black virus by first meeting on a node and then exploring the next one, in order to ensure that the node incident to the infected one will be protected by an agent before it is visited by the remaining agent and the network is decontaminated.

Algorithm 5 Ring decontamination using four agents in an asynchronous ring

```
1: procedure DECONTAMINATE(dir)
2:     Let u be the current node
3:     if number of agents on u = 2 then
4:         Move towards dir until you find a node v occupied by an agent
5:     else
6:         Wait until the number of agents on u is 2
7:     end if
8:     while true do                          ▷ Start moving using Cautious Traversal
9:         v ← current_node
10:        if you are the agent with the biggest ID then
11:            Move one step towards dir
12:            Wait until number of agents on v = 2
13:        else
14:            Wait until the other agent has left v
15:            Move one step towards dir
16:        end if
17:    end while
18: end procedure
```

5.3 Correctness Analysis

Lemma 13. Either an agent visits a node already occupied by the leader or it learns that the leader has been destroyed.

Proof. The algorithm ensures that the leader will be the first to visit a previously unexplored node, by making all agents wait on a node until the leader has left. Furthermore, the algorithm ensures that the leader will not move from a node until the two agents following it are on the node. Since the agents following and the leader all move in the same direction, and only move one step at a time, the agents can always expect the leader to be on the node they will visit, unless it has been destroyed.

Lemma 14. When the location of the black virus is found, only the leader has been destroyed.

Proof. Since the leader is the first agent to visit a new node, it will be destroyed by the black virus at some point while exploring the ring. As we showed in Lemma 13, if the leader is not at the node an agents visits, the agent knows that the leader was destroyed and will

stop executing CAUTIOUS EXPLORATION. Furthermore, the agent not participating in the exploration, does not move from the homebase and consequently cannot be harmed by the virus.

Definition 5.1 (Explored Region)**.** The explored region consists of all the nodes that have been visited by at least two agents and is delimited by the agent occupying the latest node visited by the leader and the agent furthest away from that node, in the path that contains the homebase.

Lemma 15. The explored region is always safe (i.e., does not contain the black virus).

Proof. Since the explored region is delimited by two agents, the black virus would need to pass through a node occupied by an agent which is impossible since any instance of the black virus attempting to spread to a node occupied by an agent is destroyed.

Lemma 16. During the decontamination phase, exactly one agent will be destroyed.

Proof. At the beginning of the decontamination phase, two of the agents know the position of the black virus. One of those agents will not move and therefore will not be harmed by the black virus. The other agent will move towards the opposite direction of the infected node's position. The already explored region is safe, as shown in Lemma 15. Therefore, while an agent moves in the explored region, until it returns to the homebase to meet with the remaining agent, it will not be harmed by the black virus. Finally, when the two agents start moving from the homebase towards the black virus, through its unexplored neighbour, they move one step towards the black virus and then wait until both agents are on the same node before moving again. This implies that the distance of the two agents is always at most one. Therefore, when one of the agents reaches the black virus, both of the infected node's neighbours will be guarded and the network will be cleaned, without the destruction of any other agents .

Lemma 17. The method presented above can always solve the black virus decontamination problem in an asynchronous ring using four initially co-located agents, even if the position of the black virus is unknown.

Proof. As shown in Lemmas 14 and 16, at most two agents will be destroyed by the black virus. We will show that when the second agent is destroyed by the black virus, the two neighbours are already occupied by agents and therefore the ring will be decontaminated. Let us assume that the black virus is on node u, its two neighbouring nodes are v and w, and z is the remaining neighbour of w. Initially all agents are located in the homebase. When the leader wakes up it will move one step towards the pre-decided direction. Let us assume that the leader approaches the black virus through v. There are two cases:

1. *The homebase is v.* In this case the leader is destroyed and the black virus clones itself and attempts to move to the neighbouring nodes. The instance of the black virus trying to move to the homebase will be destroyed by the agents there, while the remaining instance will infect w, leaving the initially infected node clean.

2. *The homebase is not v.* The leader waits until two more agents reach the node it currently occupies.

Next, two of the agents in the homebase, move towards the leader. Once again there are two cases.

1. *The two agents find the leader.* When the leader and the two agents following it are at the same node they begin exploring the next node. This means that the two agents wait until the leader moves to the next node before following it. This exploration method is repeated until the leader is killed by the black virus.

2. *The leader is not on the node.* The two agents learn that the black virus is on the next node and start the decontamination phase.

When the decontamination phase begins, the configuration is as follows: the black virus is at w, two agents are at the now safe node u and one agent is still at the homebase. The second agent that has reached u, changes its direction and starts traversing the safe region moving towards the homebase. Once the homebase is reached, the two agents start moving towards the black virus through z. The agents move one step towards their common direction, and wait until they meet before moving again. This ensures that the agents always maintain a distance of at most one with each other. When an agent reaches a previously unexplored node there are three possible cases.

1. *An agent is already there.* In this case the agents know that the black virus has not been cleaned and move towards the next node.

2. *The node is not occupied by an agent.* The other agent has either not reached the node yet or the network has been decontaminated. The agent however cannot decide which is the case and waits on the node.

3. *The node contains the black virus.* The agent and the black virus are destroyed and the network is cleaned.

Since the agents always have distance at most one from each other, it is ensured that there will be one agent guarding the explored neighbour of each unexplored node. Therefore, the black virus will not be able to spread in this phase. Finally, since the two agents won't stop

moving unless one of them is killed by the black virus, it is evident that in all cases the network will eventually be decontaminated.

5.4 Complexity Analysis

We measure the complexity of the algorithm with respect to the total number of steps performed by the agents in the network.

Observation 2. When the black virus is located at the node incident to the homebase in the direction opposite to the one chosen by the leader, the agents will manage to decontaminate the network in the CAUTIOUS EXPLORATION phase. Remember that during this phase one agent is located at the homebase and that the leader won't move to a new node unless its current node is occupied by two more agents. This means that when the leader enters the infected node both its neighbours will be occupied by agents and the ring will be cleaned before the beginning of the Decontamination Phase.

Lemma 18. An asynchronous ring that consists of n nodes and contains one infected node can be decontaminated using the method presented above in $O(n)$ steps.

Proof. Three agents actively participate in the Exploration Phase, therefore, three steps, one for each agent, are required for a new node to be explored. In the Decontamination Phase, only two agents move, therefore two steps are required for a node to be explored and for the exploration of the next node to begin. Let us assume that we have an n node ring, and the black virus is located in a node at distance x from the homebase, in the direction the leader will move. Since three steps are required for visiting each node $3x$ steps are needed in the first phase. In the first part of the Decontamination Phase, one of the agents traverses the ring in the opposite direction until it reaches the homebase, requiring x steps. Next, the two agents in the homebase will need to traverse $n - (x + 1)$ nodes. Since two steps are needed for the exploration of each node, in total $x + 2[n - (x + 1)]$ steps are required in the Decontamination Phase. In total, the ring will be decontaminated in $3x + x + 2[n - (x + 1)] = 2(n - 1) + 2x$ steps. The worst case will be when x is maximum. Since $x \in [1, n-1]$, and taking into account Observation 2, the maximum value for x is $n-2$. Consequently, in the worst case, the total number of steps will be $2(n-1)+2(n-2) = 4n-6$.

6 Scattered Agents in Synchronous Network

In this section we examine the problem of decontaminating a network initially containing one black virus, when the agents in the graph start at different positions. Each agent is equipped with a movable token. An agent occupying a node u can can see the state and the direction of any other agent located at u. Furthermore, any agent at a node u can count the agents on the node and pick up any token left at u.

6.1 Impossibility Results and Basic Observations

Lemma 19. The network cannot be decontaminated by any number of scattered agents without tokens.

Proof. Let us consider the case where $n-1$ agents are present in an n-node ring R, and R initially contains one black virus. Since all agents start at different nodes, each node of the ring is initially occupied by either an agent or the black virus. In order to find the position of the black virus at least one agent needs to visit the infected node, and since all agents execute the same algorithm, they all start moving in the same direction. Let u, v, w and z be four consecutive nodes and let v be the initially infected node, as shown in Figure 1.

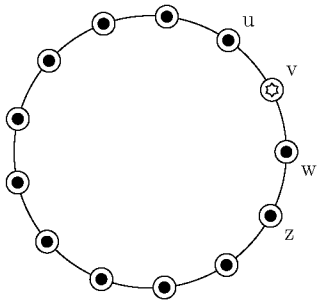

Figure 1: Starting Configuration for n-1 agents and one black virus

Suppose without loss of generality that the agents first move clockwise. Since each node is occupied by either an agent or the black virus, one of the agents is immediately killed by the black virus at v, v is cleaned, and the black virus moves to the now empty node w as shown in Figure 2. After this move, all remaining agents have the same input. Hence, they all execute the same instruction and move again. It is easy to see that they *always* have the same input, and therefore, they all always move either clockwise or counterclockwise. When an agent visits the now infected node w there are two possible cases:

1. The agent that visited w came from node v. In this case a copy of the black virus infects z, while the original black virus and the remaining copy are cleaned by agents. (Figure 3)

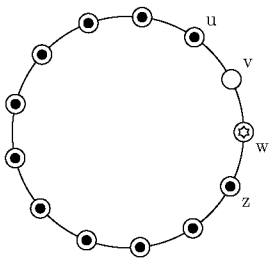

Figure 2: After first step

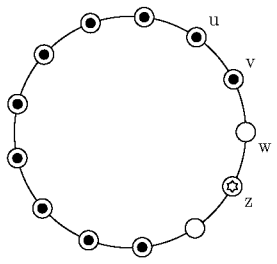

Figure 3: Clockwise Step

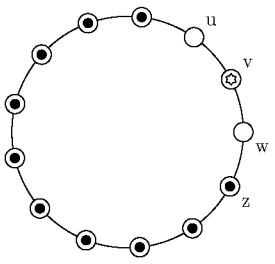

Figure 4: Counterclockwise step

2. The agent that visited w came from node z. In this case a copy of the black virus infects v, while the original black virus and the remaining copy are cleaned by agents. (Figure 4)

Since all agents move either clockwise or counterclockwise, each time an agent moves to a node containing the black virus, the black virus always moves to the next node in the direction the agent that visited it was moving.

6.2 Algorithm and Algorithm Description

The main idea of the algorithm is that each agent uses its token so that at least one group consisting of two or more agents is formed. That group explores the ring until the black virus is discovered and one of the agents starts guarding the node incident to the infected node.

Algorithm 6 Ring Decontamination from one BV using scattered agents

```
1:  procedure SCATTERED_BVD
2:      state ← INITIALIZE()
3:      switch state do
4:          case leader:
5:              repeat
6:                  Move one step ccw
7:                  Wait(1)
8:              until there is no token at the current node
9:              execute procedure LEADER(cw)
10:         case companion:
11:             pick up token from the current node
12:             repeat
13:                 Wait(1)
14:                 Move one step ccw
```

```
15:                 if a leader is not at the current node then
16:                     if number of agents at the node ≤ 2 then
17:                         Wait until a leader gets at the current node
18:                     else
19:                         if you have the smallest ID then
20:                             state ← leader
21:                             execute procedure LEADER(cw)
22:                         end if
23:                     end if
24:                 end if
25:             until there is no token at the current node
26:             execute procedure COMPANION(dir_of_leader)
27:         case explorer:
28:             execute procedure EXPLORER
29:         case guard:
30:             execute procedure GUARD
31: end procedure
```

The INITIALIZE procedure, is executed by each agent and has two main purposes:

1. To form a configuration which guarantees that at most three agents will be lost without decreasing the number of black viruses in the ring or decreasing the infected region.

2. To assign specific roles to all the agents.

First each agent places its token at the node it occupies. Then, each agent moves one step clockwise. If an agent A_i finds a token on the node it visits it waits for one time unit. If at the next time unit another agent joins A_i, A_i becomes a *companion*, otherwise, A_i becomes a *guard*. An agent that visits a node not containing any tokens moves one step back, to its homebase. An agent that finds another agent in its homebase, becomes a *leader*. Finally, an agent that finds no other agent in its homebase becomes an *explorer*.

Algorithm 7

```
1: procedure INITIALIZE
2:     Let v be the current node
3:     Release token at v
4:     Move one step cw at node w
5:     if there is a token at w then
6:         Wait(1)
7:         if an agent came back to w then
8:             state ← companion
9:         else
10:            state ← guard
11:        end if
12:    else
13:        Move one step ccw at node v
14:        if there is an agent at v then
15:            state ← leader
16:        else
17:            state ← explorer
18:        end if
19:    end if
20:    return state
21: end procedure
```

An explorer begins moving clockwise until it finds a token. Then, it changes direction and moves counterclockwise until it finds a token (i.e., until it has returned to its homebase.) The explorer picks up the token it finds and begins moving clockwise until it finds another token. If there is an agent at the node containing the token, then the explorer becomes leader, otherwise it becomes a companion and waits until another agent gets on the node it currently occupies.

Algorithm 8

```
1: procedure EXPLORER
2:     Let v be the current node
3:     Move cw until you find token
4:     Move ccw until you find token                    ▷ return to your homebase
5:     Pick up token from the current node
6:     repeat
7:         Move 1 step cw
8:         v ← current_node
9:     until (you see a member of a pair at v AND direction of pair = ccw) OR there is a
       token at v
10:    if there is a token at v then
11:        if there is a companion at v then
12:            switch to state leader
13:            execute procedure LEADER(cw)
14:        else
15:            switch to state companion
16:            Wait until an agent gets at v
17:            execute procedure COMPANION(dir_of_leader)
18:        end if
19:    else
20:        switch to state companion
21:        execute procedure COMPANION(ccw)
22:    end if
23: end procedure
```

A companion is an agent that moves to a new node exactly one time unit after a leader. If a leader is not at a node a companion visits and the pair is moving clockwise, the companion becomes guard. If a companion detects a guard on a node, it starts moving in the opposite direction along with its leader. If a companion is moving counterclockwise and a leader is not present at a node it visits, the companion waits until at least one more companion gets at the node it currently occupies, the companion with the smallest ID becomes leader and the decontamination continues.

Algorithm 9

```
1: procedure COMPANION(dir)
2:     Let v be the current node
3:     while true do
4:         Wait(1)
5:         Move 1 step towards dir
6:         if dir = cw then
7:             if leader not on node then
8:                 if number of companions on node > 1 then
9:                     if you are the companion with the smallest ID then
10:                        switch to state leader
11:                        execute procedure LEADER(ccw)
12:                    else if you are not the companion with the largest ID then
13:                        dir = ccw
14:                    else
15:                        switch to state guard
16:                        execute procedure GUARD()
17:                    end if
18:                else
19:                    switch to state guard
20:                    execute GUARD()
21:                end if
22:            else if guard on node then
23:                dir = ccw
24:            end if
25:        else if dir = ccw then
26:            if leader not on node then
27:                Wait until number of companions on node > 1
28:                Companion with smallest ID becomes leader
29:                if you became the leader then
30:                    execute procedure LEADER(ccw)
31:                end if
32:            end if
33:        end if
34:    end while
35: end procedure
```

A *leader* is an agent that moves first to a new node and is always followed by one or more companions. A leader moves clockwise until it is destroyed by the black virus, or until it meets the leader of another pair or a guard. If the leader is not destroyed, when it meets another agent it will start traversing the ring counterclockwise.

Algorithm 10

```
1: procedure LEADER(dir)
2:     Let v be the current node
3:     while true do
4:         Move 1 step towards dir
5:         Wait(1)
6:         if dir = cw then
7:             if you meet another leader on v then
8:                 switch to state companion
9:                 execute procedure COMPANION(ccw)
10:            else if guard on v then
11:                dir = ccw
12:            end if
13:        end if
14:    end while
15: end procedure
```

Finally, a guard is an agent that has inferred the position of a black virus and is guarding one of the infected node's neighbours in order to ensure that the black virus will not spread in that direction and that the network will eventually be cleaned. Since the guard is supposed to be at the node incident to a black virus in the counterclockwise direction, if a guard meets a pair moving counterclockwise, it learns that it was not at a node neighbouring to an infected node, and becomes a companion following the leader of that pair.

Algorithm 11

```
1: procedure GUARD
2:     Let v be the current node
3:     repeat
4:         Wait on v
5:     until leader at v AND dir_of_leader = ccw
6:     pick up token from current node
7:     switch to state companion
8:     execute procedure COMPANION(dir_of_leader)
9: end procedure
```

6.3 Correctness Analysis

Before analyzing our algorithm we need the following definitions.

Definition 6.1 (Consecutive Tokens)**.** Two tokens are called "consecutive" if there exists a path connecting them that contains no other token.

Definition 6.2 (Safe Region)**.** A part of the ring delimited by two consecutive tokens that does not contain a black virus.

Definition 6.3 (Unsafe Region)**.** A part of the ring delimited by two consecutive tokens that contains all the black viruses in the ring.

Definition 6.4 (Infected Region)**.** If there exists only one black virus, the infected region is just that node, and the size of the infected region is one. Otherwise, we call "infected region" a region delimited by the two black viruses that have the maximum distance from each other, including the delimiting black viruses, in the path that contains no agents. The size of the infected region is the number of nodes in the region.

Lemma 20. At most two agents are lost during INITIALIZE.

Proof. Let u, v, w and z be four consecutive nodes and let v be the node originally containing the black virus. The following starting configurations are possible.

1. An agent, A_1, starts at node u and no agent is initially located at w. Each agent places its token at its homebase and moves one step clockwise. A_1 is destroyed by the black virus at v, and the black virus clones itself and moves to u and w, leaving the initially infected node, v, clean. In the next step, each agent moves one step counterclockwise, however, since no agent was initially located at w, no agent will

return to the now infected node. Therefore, no other agent will be killed by a black virus during INITIALIZE.

2. An agent, A_1, starts at node u and another agent, A_2, is initially located at w. First, each agent places its token at its homebase and moves one step clockwise. A_1 visits the infected node v and is immediately destroyed. The black virus copies itself and moves to its two neighbouring nodes u and w, leaving v clean. A_2 is now located at z. There are two possible cases:

   (a) *A token is at node z.* In this case, A_2 remains at z for the next time unit.

   (b) *No token is at node z.* In this case, A_2 moves back to its homebase, w. However, the black virus moved to w in the previous time unit, therefore, A_2 is destroyed and the black virus copies itself and moves to v and z.

   No other agent can be killed during this step since each remaining agent that moved one step counterclockwise is at distance more than one from the black virus.

3. No agent is initially located at u. At the first step, each agent places its token at its homebase and moves one step clockwise. Since no agent was originally located at u, no agent will visit the infected node, and consequently, no additional nodes will be infected. In the following step, each agent moves one step counterclockwise. Since the black virus has not infected any additional nodes, no agent will be destroyed by moving back to its homebase.

Consequently, at most two agents can be killed by a black virus during procedure INITIALIZE.

Observation 3. Notice that after executing INITIALIZE and before all agents start executing the next procedure the following scenario might occur. Let L and C be a leader and a companion in a pair that moves counterclockwise before L and C have started executing LEADER and COMPANION respectively. In that case, the pair moves counterclockwise while it detects a token at the node it visits. Let u, v and w be three consecutive nodes, let L and C be located at node u, let u and v contain one token each and let w be a node containing a black virus. Since a token is located at u, L visits v and waits one time unit. Next, C visits v and since L and a token are on v the pair continues moving. L visits the infected node w and is destroyed by the black virus, w is cleaned, the black virus clone that attempts to infect v is cleaned by C and the remaining black virus clone infects the remaining neighbour of w. Finally, when C moves to w, since its leader, L, is not at w it stops and waits for another leader to get to w. Therefore, even if a leader is lost, the size of the infected region will either be decreased by one node if the size of the infected region is larger than one, or it

will remain one, since one of the copies of the black virus will be cleaned by the companion following the leader that was lost.

Lemma 21. At most one agent is lost during EXPLORER.

Proof. Let u, v, w and z be four consecutive nodes and let u, v and z be three nodes containing copies of the black virus during INITIALIZE, in the case presented in Item 2b of Lemma 20. The explorers begin moving clockwise until they find a token. Since the token left by A_1, the first agent that was destroyed by the black virus, was destroyed on a previous time unit, the Explorer that is moving towards the token left by A_1 will not find a token and will be destroyed by a black virus. Each other Explorer will find a token, return to its homebase, pick up the token it left there and move clockwise once again until it finds a token. Each Explorer then becomes either a Leader or a Companion and starts executing the next part of the algorithm.

Lemma 22. The size of the infected region is at most five at any time.

Proof. As we showed in Lemma 20 and Lemma 21, at most three agents are lost before becoming either leaders or companions, in the following way. Two agents are lost during INITIALIZE as we described in Item 2b of Lemma 20, and one more agent is lost during the execution of procedure EXPLORER. When the first agent visits an infected node, the length of the infected region is increased from one to three. Next, when the second agent visits a node containing a black virus, the black virus copies itself and infects its two neighbours. One of the neighbours is located inside the infected region, while the other is outside the previously infected region. Therefore, the size of the infected region is increased to four nodes. Similarly, when the third agent is destroyed by a black virus, the black virus copies itself and infects its two neighbours, only one of which was not previously in the infected region. Consequently, the infected region is once more increased by one node. In total, after three agents have been destroyed by a black virus, the size of the infected region is five. Notice that no other explorer can be destroyed by a black virus and a pair containing one leader and one companion will be formed. Furthermore, the leaders and companions only move using the cautious traversal technique. When a leader visits a node of an infected region of size more than one, its companion is guarding the node incident to the black virus that is outside of the infected region. Therefore, when the black virus copies itself and attempts to spread to its neighbours, it will only infect the neighbour that was already in the infected region, the node visited by the leader will be cleaned, and the size of the infected region will be decreased. Alternatively, if the size of the infected region is one, the size of the infected region will never increase. Finally, a guard does not move from the node it occupies.

Consequently, only the three original agents can increase the size of the infected region, and the infected region will not contain more than five nodes at any time.

Observation 4. By "leftmost node incident to the infected region" we denote the first node that does not belong to the infected region and is incident to the infected region in the counterclockwise direction.

Lemma 23. A guard occupies the leftmost node incident to the infected region before any pair begins moving counterclockwise, unless the agents have not started executing LEADER or COMPANION.

Proof. After all agents have finished executing INITIALIZE, they begin executing EXPLORER, LEADER, COMPANION or GUARD, depending on whether or not they had neighbours in the starting configuration. The agents executing EXPLORER first move clockwise until they locate a token, they return to their homebase to pick up their token and finally move clockwise once more until they reach a node containing a token, and begin executing either LEADER or COMPANION. Agents executing LEADER or COMPANION move together using Cautious Traversal clockwise. A pair only changes direction when it meets a *guard* or when it meets a pair already moving counterclockwise. Therefore, the first pair that begins moving counterclockwise must have changed its direction because it detected a *guard*. Consequently, a guard always occupies the leftmost node incident to the infected region before any pair begins moving counterclockwise.

Observation 5. When the first leader moving clockwise moves to a node containing a black virus, the size of the infected region is already at least three, since the originally infected node was already visited by an explorer.

Lemma 24. Each agent that is lost during the execution of procedure LEADER decreases the size of the infected region by one node.

Proof. An agent starts executing the procedure LEADER when it detects a token and a companion at a node, after it has picked up its own token from its homebase. The pair begins moving clockwise using the cautious traversal technique. At some time t, the leader of the first pair moving clockwise visits an infected node, v. That leader is destroyed by the black virus, and the black virus clones itself and attempts to infect its neighbouring nodes. One of the clones is cleaned by the companion of the leader that was lost, while the remaining clone infects the clockwise neighbour of v. The black virus nodes delimiting the infected region each have one neighbour outside and one neighbour inside the infected region. Since a companion of the leader that was last destroyed by a black virus cleans the

black virus copy that attempts to spread outside of the infected region, the remaining copy will move to a node already inside the infected region, leaving the originally infected node clean and decreasing the size of the infected region by one node. The companion moves to the now clean node v, and since its leader is not at v changes its state and begins executing the procedure GUARD. The second pair that begins moving clockwise will eventually reach the guard and start moving counterclockwise. A pair begins moving counterclockwise when it meets another pair that is already moving counterclockwise. Therefore, the first pair that starts moving counterclockwise, will form a team with all the pairs and explorers it encounters. Consequently, after a pair finds a guard, the pair along with all the remaining agents in the network will start moving counterclockwise in the following way. The leader moves to the next node in the counterclockwise direction and if a black virus is not present at the node, the leader waits for one time unit. The companion or the companions wait for one time unit and then visit the next node in the counterclockwise direction. If the leader is at the node the pair or team of companions explore the next node in the same way, otherwise, the companions elect a new leader and continue moving counterclockwise. When a leader is lost, the black virus copies itself and attempts to infect its neighbouring nodes leaving the originally infected node clean. Since one of the nodes incident to the infected node is occupied by a companion, only the node already inside the infected region is infected by the black virus and the size of the infected region is decreased by one node. Finally, as we showed in Lemma 23 when a pair starts moving counterclockwise, the leftmost node incident to the infected region is already occupied by a guard. Therefore, the black virus cannot spread outside of the infected region in the counterclockwise direction when an agent moving counterclockwise visits it. Consequently, when a leader visits a node containing a black virus, the size of the infected region is always decreased by one node.

Observation 6. Notice that a pair that moves counterclockwise before the execution of LEADER and COMPANION will visit at most one infected node and will decrease the size of the infected region by one node.

Theorem 1. The algorithm SCATTERED_BVD presented above solves the black virus decontamination problem in a synchronous oriented ring using ten scattered agents, even if the position of the black virus is unknown.

Proof. In Lemma 20 and Lemma 21 we show that at most three agents can be lost causing the infected region to be increased. In Lemma 22 we show that in the worst case, in which three agents are lost increasing the size of the infected region, the infected region consists of five nodes and its size cannot be increased further. Furthermore, in Lemma 24 we show that every leader that visits an infected node decreases the size of the infected region by one node,

therefore, the first pair that reaches the infected region moving clockwise decreases its size to four nodes. In Lemma 23 we show that the companion of the first pair to reach a black virus moving clockwise becomes a guard before any other pair starts moving counterclockwise. Furthermore, a pair begins moving counterclockwise when it meets a guard or when it meets another pair already moving counterclockwise. Consequently, after a guard is placed at the leftmost node incident to the infected region, all the remaining agents in the network will eventually form a single team and move counterclockwise using cautious traversal. When the team moving counterclockwise reaches the infected region, each node visited by a leader is cleaned as we showed in Lemma 24. Let v, u and w be three consecutive nodes and let u be the last infected node in the ring. The *guard* is located at v and the last *leader* and *companion* are located at w. In the last step of the algorithm, the leader moves to the infected node and is destroyed by the black virus. The black virus copies itself and attempts to spread to v and w. However, both nodes are already occupied by the last two agents and the ring is decontaminated. Therefore, $3 + 5 + 2 = 10$ initially scattered agents executing SCATTERED_BVD can always decontaminate the ring.

6.4 Complexity Lower Bound

In this section we determine the minimum number of scattered agents and tokens needed in order to decontaminate an oriented ring from a black virus.

Lemma 25. At least ten scattered scattered agents are needed to decontaminate a ring initially containing one black virus.

Proof. Let us suppose that u, v and w are three consecutive nodes, v is the infected node and u and w are the homebases of two distinct agents. Furthermore, let k agents operate in the ring, A_1 be the agent at u and A_2 the agent at w. As we showed in Lemma 19, as long as the agents move without placing their tokens at a node, all the agents are in danger of being destroyed by a black virus. In particular, consider an algorithm which instructs the agents to move without first leaving their tokens. In this case, an agent can visit an infected node in its first move causing the black virus to copy itself and infect its neighbours. Now there is one less agent in the ring, two black viruses and the surviving agents do not learn anything about the location of the black virus or the other agents. Therefore, there are two types of algorithms; those that instruct the agents to place their token at their homebase, and those that do not. The algorithms that instruct the agents to leave their tokens at their homebase are not worse than those that do not instruct the agent to place their token at their homebase. Consequently, in any algorithm that solves the black virus decontamination problem using the minimum number of agents, each agent can place its token at its homebase. When the

tokens are placed at the starting nodes, one unsafe and $k-1$ safe regions are created. In order to decontaminate the network, the agents must move and since the agents have the same input any algorithm should move them in the same direction. Let us suppose without loss of generality that the agents begin moving clockwise. In the first step, A_2 moves to its clean neighbour, A_1 is destroyed by the black virus and the black virus copies itself and infects u and w leaving the originally infected node, v, clean. Furthermore, the tokens that were placed at u and w by A_1 and A_2 are destroyed by the black virus. Since two tokens were destroyed, we now have $(k-1)-2 = k-3$ safe regions and one unsafe. After the first time unit at $t = 1$ the unsafe region contains two agents, A_2 and another agent, A_3 located at node x, that moved one step clockwise in the previous time unit. The configuration at $t = 1$ is that of Figure 5. We will show that both of the agents in the unsafe region, which lies between the two tokens represented as triangles in Figure 5, will eventually be destroyed by a black virus.

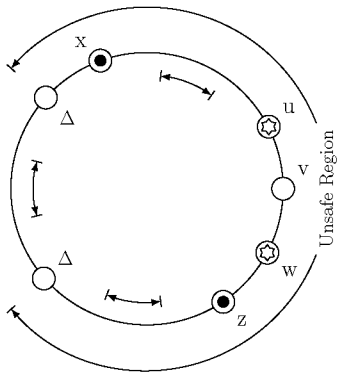

Figure 5: Configuration after the first step

At time $t = 1$, agents A_2 and A_3 are indistinguishable since they have the same input. If the agents move counterclockwise A_2 is immediately destroyed by the black virus located at w. If the agents move clockwise, they will once again have the same input at every step and eventually A_3 will be destroyed by the black virus located at u[1]. Therefore, one of the agents will eventually be killed by a black virus, and the other will reach a node containing a token. Let us suppose without loss of generality that A_3 is the agent destroyed by the black virus and A_2 is the agent that reached a node containing a token. A_3 causes the black virus to spread to v and the remaining neighbour of u. There are four possible next moves for an agent that reaches a node containing a token.

1. *Continue moving in the same direction until it finds a token.* If the agent continues

[1]In fact if the agents are instructed to move back and forth without needing a token, then the problem cannot be solved.

moving in the same direction even if it detects a token, the agent may enter an unsafe region and subsequently be destroyed by the black virus. In our case, A_2 will exit the unsafe region but another agent, A_4, that was previously in a safe region will enter the unsafe region and will eventually be destroyed by a black virus.

2. *Pick up the token.* If the agent picks up the token it finds, each agent can be destroyed by the black virus as we showed in Lemma 19.

3. *Stop.* Each agent will eventually reach a node containing a token and stop. Therefore the ring will not be decontaminated.

4. *Move in the opposite direction until it finds a token.* One agent, already in the unsafe region, in our case A_2, is destroyed by the black virus but no other agent is harmed.

Therefore, one of the optimal strategies that ensure that the ring will eventually be decontaminated is for the agents that detect a token for the first time to start moving in the opposite direction. However, moving in the opposite direction leads to the destruction of A_2. Now, the unsafe region is delimited by two tokens, and the remaining agents are all in safe regions. Consequently, three agents are lost before the unsafe region is delimited and all the remaining agents are in safe regions.
Since the number of agents that are destroyed by the black virus is known, we can calculate the size of the infected region. The first agent that is destroyed by the black virus, A_1, causes the black virus to copy itself and infect its two neighbours, increasing the size of the infected region from one to three. Notice that even though not all of the nodes in the infected region contain a black virus, the remaining nodes will be recontaminated before the infected region is completely cleaned. Next, A_3 visits one of the infected nodes, and the black virus copies itself and moves to a node previously in the infected region and to a node previously outside of the infected region. Therefore, the length of the infected region is increased by one. Similarly, when A_2 is destroyed by the black virus, the black virus copies itself and infects a node that was previously in the infected region and one previously clean node, increasing the size of the infected region by one node. Finally, the infected region contains five nodes in total, after three agents have been destroyed.
We will now show that seven agents are needed to clean the ring. Let y, u, v, w, and z be the five nodes of the infected region.

1. Without loss of generality let y be the first node to be cleaned. One agent is needed to guard the clean neighbour of y to ensure that no clean nodes will be infected. The second agent visits y cleaning the black virus there and forcing the black virus to

move to its two neighbours. Since the neighbour that was not in the infected region is guarded by an agent, the black virus will only move to u, which was already part of the infected region. Therefore, in order to clean the first node in the infected region, two agents are needed. At the end of the decontamination of the first node, one agent is destroyed and the size of the infected region is decreased by one.

2. In order to decontaminate the next infected node, u, two agents are needed. Once again, one agent will be in the clean node y to prevent the black virus from infecting it and one agent will visit u and clean it. The agent that visits u is destroyed and the black virus copies itself and moves to y, where it is cleaned by the agent already occupying the node, and v. Once more, one agent is destroyed and the size of the infected region is decreased by one.

3. The next infected node, v, needs two agents to be cleaned. One agent is positioned at u to prevent the black virus from spreading to clean nodes and another agent cleans the black virus at v. The agent that cleans v is destroyed, the agent at u cleans the black virus clone that attempted to move to u and another clone of the black virus moves to its remaining neighbour w. One agent is destroyed and the size of the infected region is decreased by one.

4. The next node that will be decontaminated is w. One agent guards w's clean neighbour v while a second agent cleans the black virus at w. The agent that cleaned w is destroyed and a copy of the black virus moves to z. The size of the infected region is decreased by one and one agent is destroyed.

5. Finally one more black virus remains in the ring. Since neither neighbour of z is in the infected region, three agents are needed to completely clean the ring. Two agents are positioned in each of the clean neighbours of z in order to ensure that the black virus will not be able to spread to its neighbours. Finally, one agent is needed to visit z. The agent that moves to z is destroyed, z is cleaned and the black virus attempts to spread to its neighbours. Since both of z's neighbours are guarded by agents the black virus is not able to spread, and all black viruses in the ring are cleaned.

Therefore, seven agents are needed to clean the ring from every black virus and ensure that no node outside of the infected region is infected. Furthermore, three agents were destroyed before the decrease of the infected region started. Consequently, ten initially scattered agents are needed to decontaminate a ring initially containing one black virus.

6.5 Execution Examples

In this section we give two examples of different starting configurations and we demonstrate how the starting configuration can affect the number of initially scattered agents needed to decontaminate a synchronous ring.
Let us first consider the starting configuration of Figure 6.

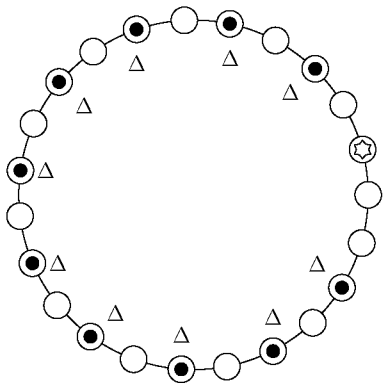

Figure 6

First each agent places its token, represented by a triangle, at its homebase. Then each agent moves one step clockwise, as shown in Figure 7. In the next step each agent returns to its homebase and we have once again the configuration of Figure 6. Next, the agents move clockwise until they find a token. In this step, one agent visits the node containing the black virus, causing it to copy itself and infect its neighbours leaving the originally infected node clean, as shown in Figure 8. Each surviving agent moves back to its homebase and picks up its token. Now, only one token, placed by the agent that was lost remains in the ring as we show in Figure 9.

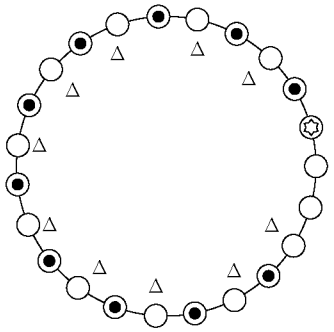

Figure 7

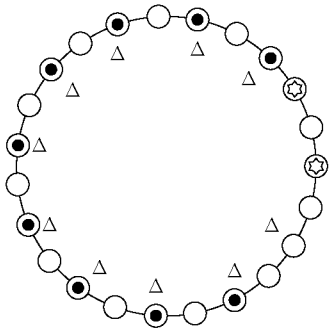

Figure 8

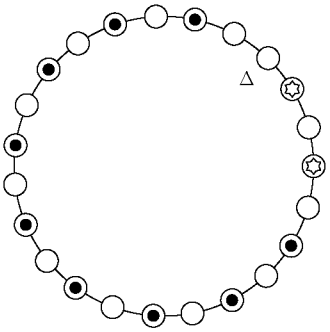

Figure 9

Next, each agent starts moving clockwise once again, until it detects the presence of a token at the node it occupies. The first agent to reach a node containing a token changes its state to *companion* and waits for another agent. When the first pair is formed, the configuration

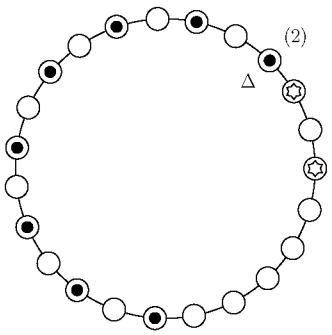

Figure 10

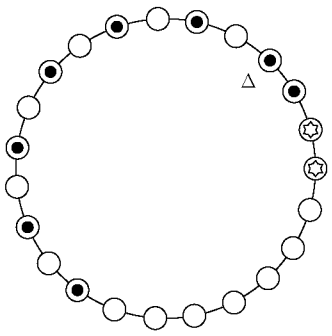

Figure 11

is that of Figure 10. Next, the pair starts moving using cautious traversal and the leader immediately visits an infected node. The leader is destroyed and the black virus copies itself and attempts to infect its neighbouring nodes. One of the black virus copies is cleaned by the companion located at one of the neighbouring nodes. The companion moves to the next node clockwise and since a leader is not there it becomes a guard (Figure 11). All the remaining agents in the ring change direction when they meet the guard or a pair already moving counterclockwise and move using cautious traversal until the ring is decontaminated. The size of the infected region is two, therefore after the network is cleaned, six agents remain in the ring.

Let us now consider the starting configuration of Figure 12.

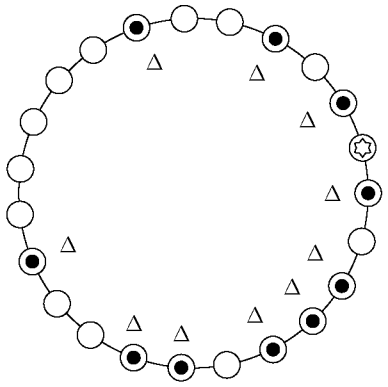

Figure 12

First each agent moves one step clockwise. One agent visits the infected node, causing the black virus to copy itself and move to its neighbours, as shown in Figure 13. Next, each agent that is not at a node containing a token moves one step counterclockwise. One more agent is lost at this step, causing the black virus to copy itself and infect its neighbours once again, as shown in Figure 14. Now, two pairs, consisting of one leader and one companion each, have been formed, the agents that returned to their homebase become explorers and

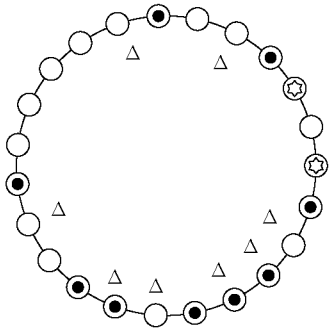

Figure 13

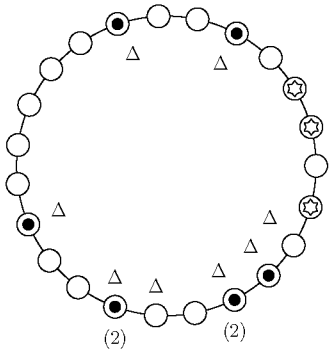

Figure 14

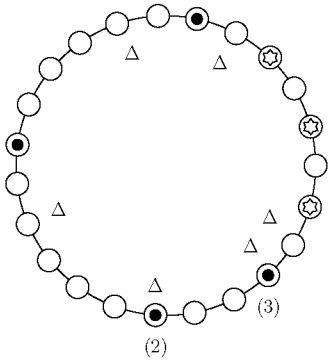

Figure 15

the agent that did not return to its homebase and did not meet another agent at its current node becomes a guard. Each explorer moves clockwise until it finds a token, and each pair moves counterclockwise while it detects a token on the node it currently occupies. During this step an explorer is lost causing the black virus to infect its neighbours (Figure 15) and a leader is lost decreasing the infected region by one node leading to the configuration of Figure 16.

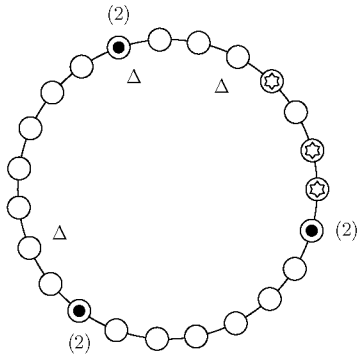

Figure 16

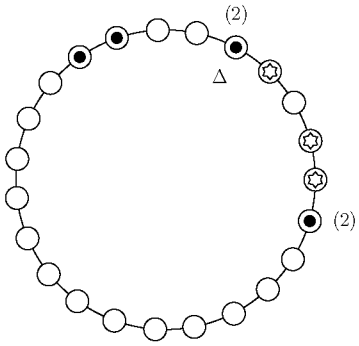

Figure 17

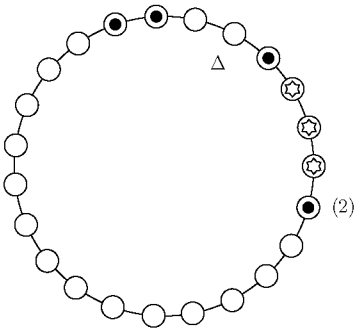

Figure 18

Now the two companions that were moving counterclockwise with the leader that was lost wait at their current node for another leader, the pair continues moving clockwise until it finds a guard or a pair moving counterclockwise, one of the explorers picks up its token and begins moving clockwise until it finds a token left by another agent and the last explorer returns to its homebase to pick up its token. The configuration when the two explorers form a pair is that of Figure 17. Next, the pair formed by the two former explorers starts moving using cautious traversal. The leader is immediately destroyed by a black virus and the black virus copies itself and attempts to infect its neighbouring nodes. One of the black virus copies is cleaned by the companion of the leader that was lost. In the next step, the companion of the leader that was lost moves to the next node clockwise and since a leader

is not at the node it changes its state to *guard*. The configuration is that of Figure 18. The leader of the remaining pair that is moving clockwise will reach the guard and the pair will begin moving counterclockwise. Since the infected region contains three nodes, three agents will be destroyed by the black virus and only two agents will remain in the ring after it has been cleaned.

7 Conclusions

7.1 Summary

In this thesis we study the black virus decontamination problem in rings and we propose solutions that employ mobile agents. We consider the problem in both the synchronous and asynchronous case, when one black virus is initially present in an unknown node of the ring. Furthermore, we address the case of initially co-located agents for both cases and the case of initially scattered agents in the case of synchronous rings. A Black Virus is a malicious entity that combines the harmful capabilities of a black hole to those presented in the intruder capture problem. We remind the reader that the black hole presents a threat to the agents in the network, whereas in the intruder capture problem the threat is to the nodes of the network. A black virus is dangerous to both the agents operating in the network and the nodes and therefore needs to be cleaned.

In Chapter 2 we present the related work. More specifically, we present the Black Hole Search problem, the Intruder Capture problem and the previous work done in the Black Virus Decontamination problem in various topologies.

In Chapter 3 we present the model and some of the basic terms used throughout the thesis.

In Chapter 4, we address the problem in the case of a synchronous ring, initially containing a team of co-located agents and one black virus. We first prove that four agents are needed in order to decontaminate the ring. Next, we give an algorithm that solves the decontamination problem using four agents in $2n - 4$ steps. Finally, we prove that no algorithm can solve the black virus decontamination problem in a synchronous ring in less moves.

In Chapter 5, we study the case of initially co-located agents in asynchronous rings, initially containing one black virus. First we prove that once again four agents suffice for the decontamination of the ring. Then, we give an algorithm that solves the decontamination problem using four agents in $4n - 6$ steps.

In Chapter 6, we consider the case of initially scattered agents in a synchronous ring. We prove that ten agents are necessary in order to solve the problem using scattered agents. Furthermore, the agents are equipped with tokens. We next give an algorithm that solves the problem using the minimum number of agents in $O(n)$ moves.

The following table summarizes the results presented above.

Network	Number of Agents	Number of Moves	Communication Method(s)
Synchronous Ring (co-located agents)	4	$2n-4$	face-to-face
Asynchronous Ring (co-located agents)	4	$4n-6$	bits in node
Synchronous Ring (scattered agents)	10	$O(n)$	face-to-face, tokens

Table 2: Summary of the results

7.2 Open Problems and Future Work

Some future research directions include:

- Consider the case of scattered agents in asynchronous rings.
- Investigate the problem using scattered agents in different topologies.
- Investigate the problem for multiple black viruses using scattered agents.
- Study the problem for black viruses with different capabilities such as:
  - The black virus copies itself and infects its neighbouring nodes at arbitrary times without being triggered by an agent.
  - The black virus can infect its neighbours at distance k when it is triggered by an agent.

References

[1] Jie Cai, Paola Flocchini, and Nicola Santoro. Decontaminating a network from a black virus. *International Journal of Networking and Computing*, 4(1):151–173, 2014.

[2] Stefan Dobrev, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Mobile search for a black hole in an anonymous ring. In *International Symposium on Distributed Computing*, pages 166–179. Springer, 2001.

[3] Jérémie Chalopin, Shantanu Das, Arnaud Labourel, and Euripides Markou. Black hole search with finite automata scattered in a synchronous torus. In *International Symposium on Distributed Computing*, pages 432–446. Springer, 2011.

[4] Jurek Czyzowicz, Dariusz Kowalski, Euripides Markou, and Andrzej Pelc. Searching for a black hole in tree networks. In *International Conference On Principles Of Distributed Systems*, pages 67–80. Springer, 2004.

[5] Stefan Dobrev, Paola Flocchini, Rastislav Kralovic, Giuseppe Prencipe, Peter Ruzicka, and Nicola Santoro. Black hole search by mobile agents in hypercubes and related networks. In *OPODIS*, volume 3, pages 169–180, 2002.

[6] Stefan Dobrev, Paola Flocchini, R Královič, P Ružička, Giuseppe Prencipe, and Nicola Santoro. Black hole search in common interconnection networks. *Networks: An International Journal*, 47(2):61–71, 2006.

[7] Stefan Dobrev, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Searching for a black hole in arbitrary networks: optimal mobile agent protocols. In *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pages 153–162. ACM, 2002.

[8] Colin Cooper, Ralf Klasing, and Tomasz Radzik. Searching for black-hole faults in a network using multiple agents. In *International Conference On Principles Of Distributed Systems*, pages 320–332. Springer, 2006.

[9] 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.

[10] Stefan Dobrev, Paola Flocchini, and Nicola Santoro. Improved bounds for optimal black hole search with a network map. In *International Colloquium on Structural Information and Communication Complexity*, pages 111–122. Springer, 2004.

[11] Paola Flocchini, Matthew Kellett, Peter C Mason, and Nicola Santoro. Searching for black holes in subways. *Theory of Computing Systems*, 50(1):158–184, 2012.

[12] Wei Shi. Black hole search with tokens in interconnected networks. In *Symposium on Self-Stabilizing Systems*, pages 670–682. Springer, 2009.

[13] Stefan Dobrev, Paola Flocchini, Rastislav Královič, and Nicola Santoro. Exploring an unknown dangerous graph using tokens. *Theoretical Computer Science*, 472:28–45, 2013.

[14] Paola Flocchini, David Ilcinkas, and Nicola Santoro. Ping pong in dangerous graphs: Optimal black hole search with pure tokens. In *International Symposium on Distributed Computing*, pages 227–241. Springer, 2008.

[15] Paola Flocchini, David Ilcinkas, and Nicola Santoro. Ping pong in dangerous graphs: Optimal black hole search with pebbles. *Algorithmica*, 62(3-4):1006–1033, 2012.

[16] Colin Cooper, Ralf Klasing, and Tomasz Radzik. Locating and repairing faults in a network with mobile agents. *Theoretical Computer Science*, 411(14-15):1638–1647, 2010.

[17] Jurek Czyzowicz, Dariusz Kowalski, Euripides Markou, and Andrzej Pelc. Complexity of searching for a black hole. *Fundamenta Informaticae*, 71(2, 3):229–242, 2006.

[18] Ralf Klasing, Euripides Markou, Tomasz Radzik, and Fabiano Sarracco. Hardness and approximation results for black hole search in arbitrary networks. *Theoretical Computer Science*, 384(2-3):201–221, 2007.

[19] Ralf Klasing, Euripides Markou, Tomasz Radzik, and Fabiano Sarracco. Approximation bounds for black hole search problems. *Networks: An International Journal*, 52(4):216–226, 2008.

[20] Lali Barrière, Paola Flocchini, Pierre Fraigniaud, and Nicola Santoro. Capture of an intruder by mobile agents. In *Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*, pages 200–209. ACM, 2002.

[21] Dariusz Dereniowski. Connected searching of weighted trees. *Theoretical Computer Science*, 412(41):5700–5713, 2011.

[22] Paola Flocchini, Miao Jun Huang, and Flaminia L Luccio. Decontamination of hypercubes by mobile agents. *Networks: An International Journal*, 52(3):167–178, 2008.

[23] Paola Flocchini, Miao Jun Huang, and Flaminia L Luccio. Decontaminating chordal rings and tori using mobile agents. *International Journal of Foundations of Computer Science*, 18(03):547–563, 2007.

[24] Lélia Blin, Pierre Fraigniaud, Nicolas Nisse, and Sandrine Vial. Distributed chasing of network intruders. In *International Colloquium on Structural Information and Communication Complexity*, pages 70–84. Springer, 2006.

[25] David Ilcinkas, Nicolas Nisse, and David Soguet. The cost of monotonicity in distributed graph searching. *Distributed Computing*, 22(2):117–127, 2009.

[26] Jie Cai, Paola Flocchini, and Nicola Santoro. Black virus decontamination in arbitrary networks. In *New Contributions in Information Systems and Technologies*, pages 991–1000. Springer, 2015.

[27] Jie Cai, Paola Flocchini, and Nicola Santoro. Distributed black virus decontamination and rooted acyclic orientations. In *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on*, pages 1681–1688. IEEE, 2015.

[28] J Cai, P Flocchini, and N Santoro. Decontamination of an arbitrary network from multiple black viruses.

[29] Jie Cai. Experimental analysis of black virus decontamination by disj. *INTELLI 2015*, page 54, 2015.

[30] Modhawi Alotaibi. Black virus disinfection in chordal rings. Master's thesis, Université d'Ottawa/University of Ottawa, 2014.

[31] Yichao Lin. Decontamination from black viruses using parallel strategies. Master's thesis, Université d'Ottawa/University of Ottawa, 2018.