Gathering of Mobile Agents in a Tree in presence of Black Holes

Submitted in partial fulfillment of the requirements for the degree of

Master of Engineering in Software Engineering

by

SAYAN RAY

CLASS ROLL NUMBER: 001711002019

REGISTRATION NUMBER: 140975 OF 2017-2018
EXAMINATION ROLL NUMBER: M4SWE19020

UNDER THE SUPERVISION OF

DR. SRUTI GAN CHAUDHURI

Assistant Professor

Department of Information Technology
Jadavpur University, Kolkata, India

May, 2019

CERTIFICATE OF SUBMISSION

I hereby recommend the thesis, entitled, "Gathering of Mobile Agents in a Tree in presence of Black Holes", prepared under my guidance by Sayan Ray, be accepted in partial fulfillment of the requirements for the degree of Master of Engineering in Software Engineering of this university.

_____

Signature of the Supervisor

Dr. Sruti Gan Chaudhuri

Assistant Professor

Department of Information Technology

Jadavpur University, Kolkata

Countersigned by:

_____

Head of the Department

Department of Information Technology

Jadavpur University

_____

Dean

Faculty of Engineering and Technology

Jadavpur University

CERTIFICATE OF APPROVAL

The thesis at instance is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned does not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve this thesis for the purpose for which it is submitted.

(Signature of Thesis Supervisor)
Dr. Sruti Gan Chaudhuri
Assistant Professor
Department of Information Technology
Jadavpur University

(Signature of Thesis Examiner)

DECLARATION OF AUTHORSHIP

I, Sayan Ray, declare that this thesis titled ”Gathering of Mobile Agents in a Tree in presence of Black Holes” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other Institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Place: Kolkata

Signature of the candidate
SAYAN RAY
M.E. (Software Engineering)
Jadavpur University
EXAMINATION ROLL NO.: M4SWE19020
REGISTRATION NO.: 140975 of 2017-2018

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Dr. Sruti Gan Chaudhuri, Assistant Professor, Department of Information Technology, Jadavpur University, for providing me the opportunity to work under her supervision. I am extremely thankful for the keen interest she took in advising me, for the reference materials provided and for the moral support and thoughtful guidance extended to me.

Also, my sincere obligation goes to all the teachers for providing me the technical skill that will always remain as my asset. Thanks to Jadavpur University, and all members of the staff; both technical and non-technical. A special thanks to All India Council for Technical Education (AICTE), for providing scholarship, without which the research would have been impossible.

Finally, I would like to thank my parents for their continuous and endless support.

ABSTRACT

Protecting agents from host attacks is a pressing security concern in networked environments supporting mobile agents. This communication network has been assumed to be an anonymous and un-oriented tree where mobile agents can move freely from one node to another node and reach a single node, called target node.

The black hole search problem that we consider in this paper is motivated by the following scenario: Mobile agents (software) can move through a network of computers, but some hosts, called black holes, terminate any agent visiting it, thus destroying the agent. We assume that the agents are a limited resource, so they should first locate black holes to avoid entering them and then dying. The mobile agents are assumed to be anonymous, oblivious and work collaboratively to achieve a common goal. They operate in Look-Compute-Move cycles and can move from one node to the adjacent node.

Initially, the agents are assumed to be placed randomly on some nodes which are not black holes (called safe node). No information about black holes is available in the safe nodes of the network (the nodes which are not black holes). The task of the agents is to meet at a safe node while avoiding the black holes. The location of the black holes must be detected before gathering.

In order to locate a black hole, at least one agent must visit it. An agent entering a black hole disappears there, so later this agent cannot show up where expected by the other agents, or communicates with them in any other expected way. Black hole search is a non-trivial problem, its difficulty is aggravated by the simultaneous presence of asynchrony of the agents and absence of any trace of destruction .We are interested in designing an efficient algorithm to detect the black holes as well as gather the mobile agents to the target node.

Keywords: *agents*, *black hole*, *detection*, *gathering*.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Background

Distributed Systems are present everywhere today when almost everything is connected to networks. Such systems are characterized by the presence of communities of networked entities communicating with each other and cooperating to achieve a common goal or solving a shared problem while acting autonomously and spontaneously. In distributed computing, the research focus is on the computational and complexity issues of the systems composed of autonomous computational entities interacting with each other to solve a problem or to perform a task.

While traditionally these autonomous computational entities have been assumed to be static, recent advances in a variety of fields, ranging from robotics to networking, have motivated the community to address the situation of mobile entities [1]. There are quite a large number of such systems and different models have been proposed in order to describe them.

The objective of this thesis is to design deterministic and distributed algorithms [8] for a group of mobile networked entities called agents, who cooperate between themselves to achieve a common goal. Few examples of such tasks are, spreading a message in a network by multiple mobile agents; gathering messages from several hosts to a single host in a network by multiple mobile agents; navigating or guarding a big area by multiple mobile or static sensors; leader election etc.

The computational problems related to these operations are definitely non-trivial, and a great deal of theoretical research has been devoted to the study of the conditions for the solvability of these problems under certain scenarios and to the discovery of efficient algorithmic solutions.

The environment can be described as a collection of autonomous mobile agents (or robots) located in a tree. The agents have limited computing capabilities and private storage, can move from one node to neighbouring node and perform computations at each node according to a predefined set of behavioural rules, which is the same for all agents. They are asynchronous, in the sense that every action they perform (computing, moving, etc.) takes a finite but otherwise unpredictable amount of time. The mobile agents are small, identical, and oblivious and operate in Look-Compute-Move cycles which are performed asynchronously for each agent [3]. Each node in the network, also called a host, may provide a storage area called whiteboard [8] for incoming agents to communicate and compute, and its access is held in fair mutual exclusion [4]. The research concern is on determining what tasks can be performed by such entities, under what conditions, and at what cost. In particular, a central question is to determine what minimal hypotheses allow a given problem to be solved.

In these environments, security is a pressing concern and possibly the most difficult to address. Among the severe security threats faced in distributed mobile computing environments, two are particularly troublesome: harmful agents (that is, the presence of malicious mobile processes) and harmful hosts (that is, the presence of harmful stationary processes at a network site). The former problem is particularly acute in unregulated non-cooperative settings such as the Internet (e.g., e-mail-transmitted viruses). The latter exists not only in those settings but also in environments with regulated access and where agents cooperate towards common goal (e.g., sharing of resources or distribution of a computation on the grid). In fact, a local (hardware or software) failure might render a host harmful.

This thesis addresses all the computational issues for the robots stated above in the presence of harmful hosts in a network that dispose of visiting agents upon their arrival, leaving no observable trace of such destruction [8]. Due to its nature, the site where such a process is located is called a black hole. The task is to unambiguously determine and report the location of the black hole. The research concern is to determine under what conditions and at what cost mobile agents can successfully accomplish this task. The network topology is considered to be a tree. The robots or mobile agents have to be gathered at a single node called target or destination node (which may have been decided in advance), using a suitable gathering algorithm.

1.2 Framework

The computation model can be sub-divided into two modules which are as follows: -

1. The model of swarm robots
2. The model of the environment.

The varying characteristics of the two sub-division models are blended with each other and tested to dig out suitable combinations to reach optimum results.

1.2.1 **Robot Model**

The mobile entities in distributed computing are popularly known as robots [3] or agents [2]. A swarm (group) of robots have changed the notion of solving problems by traditional robots where normally, a large complex machine (robot), with huge power demands, is used to solve one single problem. If that machine fails due to some reason, the process of solving that problem can only resume when the machine has been repaired. Repairing such a machine is expensive and requires both time and effort. This is where swarm robotics comes to our aid. By using small, inexpensive, mass producible, recyclable robots, which work collaboratively, not only do we solve the same problem with the same efficiency, we can also overcome the pitfalls of an overly complex singular machine.

Cooperative behaviour is the main strength of swarm robotics. The robots act in collaboration to execute a common task. This approach makes the system of swarm robots scalable and fault tolerant. The swarm robots as shown in Fig. 1.1 are a system of autonomous mobile robots. Each robot is capable of sensing its immediate surroundings, performing computations on the sensed data and moving towards the computed destination; its behaviour is an (endless) cycle of sensing, computing, moving and being inactive.

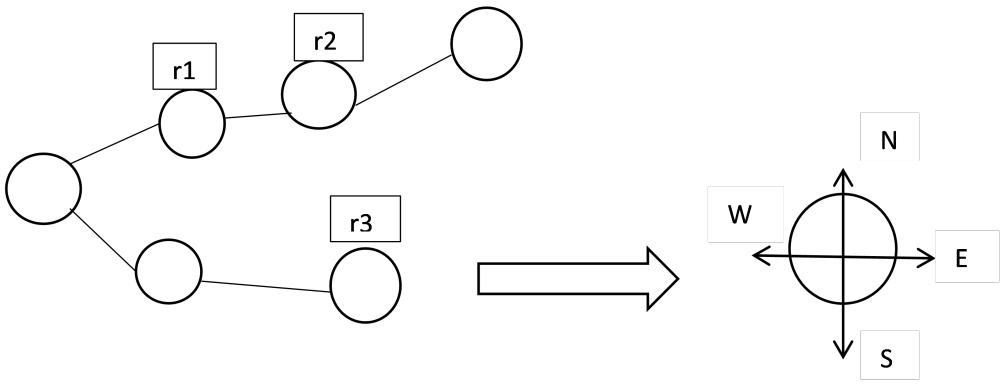

Fig. 1.1 r1, r2, r3 are three robots placed randomly on the nodes of a tree.

Since the robots are autonomous so there is no centralised mechanism to control them i.e. the robots perform their task in a decentralised (the robots do not have a common coordinate system) and asynchronous manner. The robots do not have an agreement on the coordinate axes i.e. there is no chirality which means the clockwise and anti -clockwise direction labels of all the robots are not same and secondly, the robots do not have a common Sense of Direction (S.O.D) as shown in Fig. 1.2

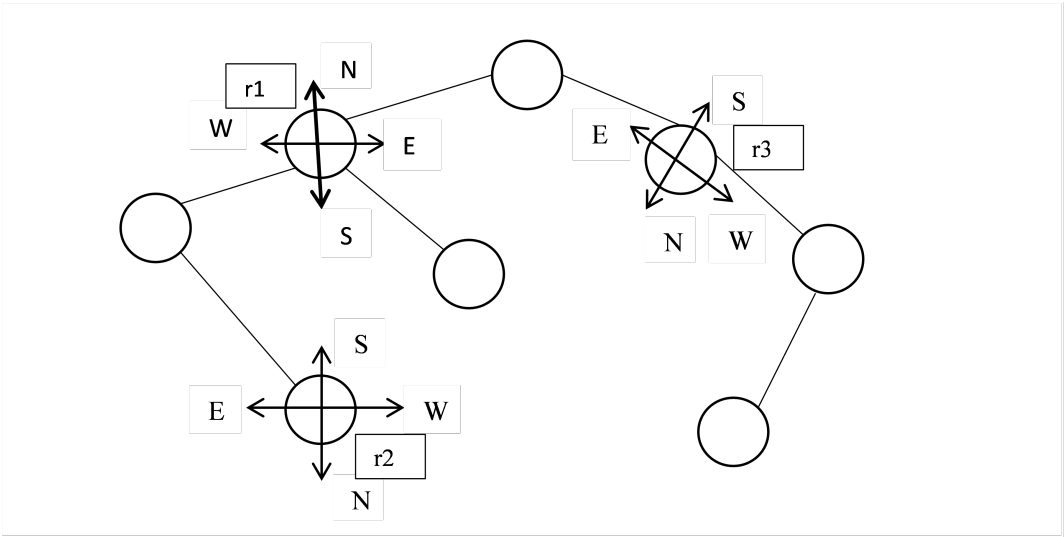

Fig. 1.2 r1, r2, r3 are three robots present on the tree. The black arrows depict the coordinate axes of the robots.

The robots are anonymous in nature i.e. they are identical in nature and exhibit the same deterministic algorithm (which takes the observed positions of the robots within the visibility radius as input and returns a destination point towards which the executing robot moves).

Each robot is either active or inactive at some point of time. When a robot is active, it operates in **Look-Compute-Move** cycles [3].

The Look-Compute-Move Algorithm: -

Phase 1 (look): - In this phase, the robot takes a snapshot or a view of the current configuration with its sensors according to its local coordinates system in Fig. 1.3.

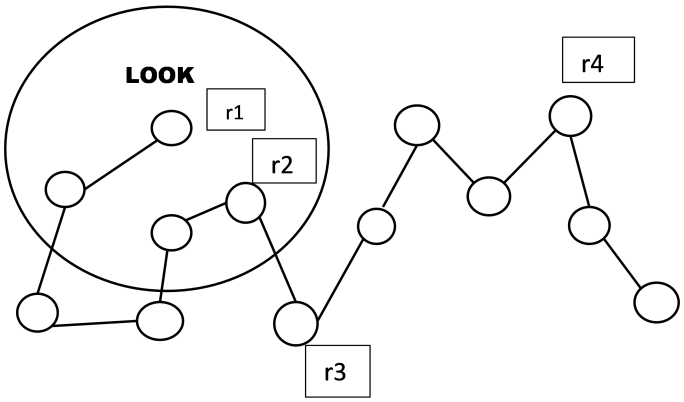

Fig. 1.3 The Look phase

Phase 2 (compute): - In this phase, based on the snapshot taken in the previous phase, the robot calculates its new destination node in Fig. 1.4.

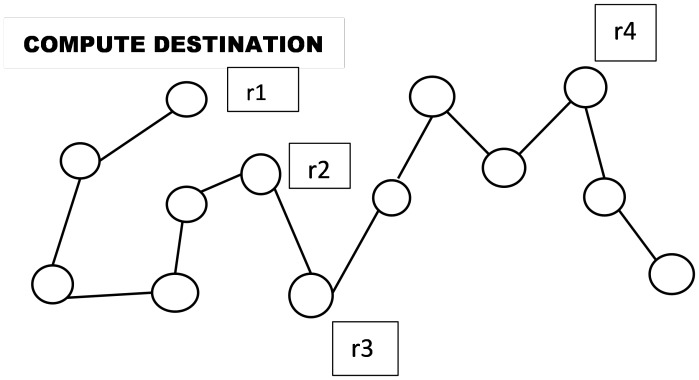

Fig. 1.4 The Destination phase

Phase 3 (move): - If the new computed destination node is the current node itself then the robot remains there or else makes an instantaneous move to the new destination node in Fig. 1.5.

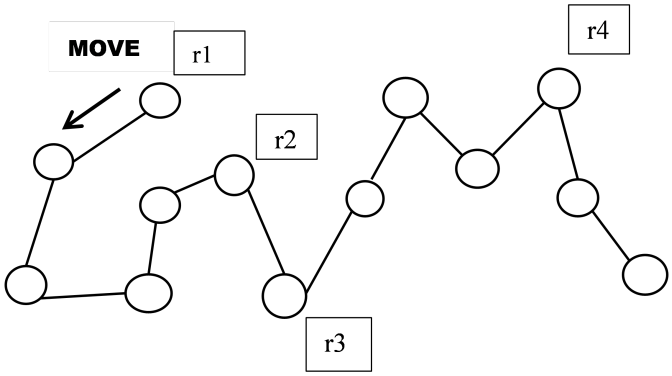

Fig. 1.5 The Move phase

The robots are fully asynchronous. In particular, the amount of time spent in a computation, in a movement and in inactivity is finite but not bounded or predictable. There are two assumptions which certify that the algorithms will stop in finite time which are as follows: -

Assumption 1: - Any robot will complete its cycle (wait-look-compute-move) in an amount of time which is finite but not bounded or predictable.

Assumption 2: - The distance travelled by a robot in a move is finite but unbounded. This assumption is for continuous domain. In discrete domain (graph) a robot will ei-

ther move to the neighbouring node or it will not move based on its computation.

The property of the robots which has played the pivotal role in the entire research is the multiplicity detection capability. It is the property by virtue of which a robot can determine whether there is none, one or more than one robots present in a node. If a node has more than one robots then we say there is strict multiplicity in that node. Multiplicity detection can be broadly classified into four types: -

1) Global-Strong version- In this case a robot is able to determine the exact number of robots occupying a node.

2) Global-Weak version- In this case a robot is able to determine only whether a node is occupied by one robot or more than one robot.

3) Local-Strong version- In this case a robot is able to determine only whether a node is occupied or not and is also able to determine the exact number of robots occupying the node in which it is currently residing.

4) Local-Weak version- In this case a robot can only determine if there is a multiplicity in the node in which it is currently residing but it cannot determine the exact number of robots.

The robots are **memory less** (oblivious) which means they do not have the capacity to remember their previous calculations or observations or leave any mark at the visited nodes i.e. all the information stored in the workspace is cleared at the end of each cycle.

Silence marks another property of the robots which states that the robots cannot directly communicate or send messages to each other and can only communicate with the environment just by observing it.

1.2.2 **Tree Model**

The tree or the configuration [3] is un-oriented in nature and the robots or mobile agents reside in the nodes. The edges or the links between the nodes do not have any label. The robots can be perceived only in the nodes and not on the edges or the links.
At every node, zero or more agents can reside at a time. The size of the tree is assumed to be finite. All the nodes are identical, i.e. none of them has any type of identification marks. Thus, robots cannot identify which nodes are the current node's children and which node is the current node's parent.

1.3 Objective of the Thesis work

The objective of this thesis is to design algorithms to detect black holes in the network which is assumed to be a tree and gather the agents at a destination or target node which may or may not be labelled. The mobile agents are initially placed randomly in various nodes of the tree. The node where to gather may or may not be given in advance.

1.4 Thesis Organisation

The organisation of this thesis is as follows: **Chapter 2** presents a brief overview of the earlier works reported in the field of gathering multiple robots as well as detecting black holes present in the network. **Chapter 3** presents the black hole detection algorithm. **Chapter 4** presents gathering algorithm for the agents when the destination node is labelled. **Chapter 5** presents gathering algorithm for the agents when the destination node is not labelled. Finally in **Chapter 6** we conclude by stating future scopes of the thesis work.

CHAPTER 2

LITERATURE SURVEY

2.1 Earlier works

In this section we will discuss some earlier works which are related to this thesis.

Fault tolerant gathering algorithms for autonomous mobile robots [1] presents a study of failure-prone robot systems through examining the gathering problem under the crash and Byzantine fault model. Under crash fault model, gathering problem is solvable in (semi-synchronous) SSYNC model, under Byzantine model, this problem is solvable in (fully synchronous) FSYNC model. The visibility range of the robots has been assumed to be unlimited and they are oblivious. The robots are treated as points (dimensionless objects) which do not obstruct each other's visibility or movement. The robots are anonymous and cannot communicate with each other. Each robot has its private coordinate system.

It may be useful to study other kinds of fault models in addition to the crash and Byzantine models, such as a model in which the robots might lose some of their movement control or a model in which robots might diverge from their original movement direction up to a certain percentage of error.

In Optimal tree search by a swarm of mobile robots [2], an algorithm for tree searching problem has been presented. After searching the tree completely, the robots gather at a specially designated node. In this target searching, all the robots in the swarm are required to gather at the specially designated node, termed as target node, which is distinguishable from the other nodes, within finite amount of time after exploring (collectively) the tree completely. Memory is assumed to be attached to each node of the tree called whiteboard (WB). Every node is unlabelled, only the target node is labelled. The robots, which are deployed over the nodes of the tree, are assumed to be anonymous, asynchronous, oblivious and observe only adjacent nodes. Robots carry out a sequence of computational cycles throughout the execution of the algorithm which consists of three phases, observe, compute, and move.

This problem differs from the conventional gathering problem in the sense that the target node is marked, whereas, in the conventional gathering problem, the gathering point is determined in run time.

Gathering of Asynchronous Mobile Robots in a Tree [3], presents a gathering algorithm for the gathering of robots in an anonymous and un-oriented tree, where the robots do not have the full structure of the tree. The robots search for the destination by probing. This algorithm does not guarantee that a robot will traverse a path only once, which is a desirable trait of any gathering algorithm, but it is always eliminating paths which force the robots to explore unknown paths, and thus converging on the destination faster. The robots are assumed to be anonymous, oblivious, asynchronous, silent and weak multiplicity detection. An algorithm is presented by which k robots deterministically gather at a single uniquely marked node in a tree with n nodes.

The future scope of this paper would be to modify this algorithm so that it can solve the same problem in a much more dynamic and real environment.

Searching for black hole faults in a network using multiple agents [4] discusses the designing of an efficient algorithm for the mobile agents to identify black holes present in a communication network assumed to be an undirected, connected graph. The agents are at the same start node s and know the topology of the whole network, but do not know the number and the location of black holes. The network is assumed to be synchronous and assume that agents can communicate and exchange their knowledge, only when they meet. They cannot leave any messages in the nodes of the network. In one synchronized step, each agent can either stay in its current host or move to a neighbouring one.

It remains a question for further research whether an algorithm can be devised that is optimal for arbitrary networks.
The case when the topology of the whole network is known in advance has been considered. It would be interesting to consider the case when initially no, or only partial, information about the network is available.

In Two convergence problems for robots on graphs [5], two robot convergence tasks in an asynchronous read/write shared memory, crash-prone system, where the base space is a finite graph has been studied. The tasks are the graph convergence and edge covering. In graph convergence, robots have to end up on one or two vertices of the same edge. In edge covering, where additionally it is required that not all robots end up on the same vertex. It is assumed that robots know the graph, and can communicate with each other their current vertex positions using the shared memory. The robots are

asynchronous and may crash.

In practice, robots have no shared memory to communicate with each other; typically, they communicate through a weaker communication medium. This is the limitation of this paper as it has been assumed that robots have shared memory.

Mobile search for a black hole in an anonymous ring [6] addresses the problem of mobile agents searching for a black hole in an anonymous ring network. The task is to have at least one surviving agent able to unambiguously report the location of the black hole. Two cases have been considered: when the agents start from the same home base (co-located) and when the agents start from different home bases (dispersed). The agents are asynchronous, have limited computing capabilities and limited storage; obey the same set of behavioural rules and can move from node to neighbouring node. Each node has a limited amount of storage, called whiteboard. Agents communicate by reading from and writing on the whiteboards; access to a whiteboard is done in mutual exclusion. The problem is solved if at least one agent survives, and all surviving agents know the location of the black hole.

Solutions to the blackhole search problem in anonymous rings have been provided for two settings: when the anonymous agents are co-located and when they are dispersed. The protocol for the dispersed agents setting, Algorithm DOR, may be employed to solve the black-hole search problem in the more general setting, when the anonymous agents are in more than one location in the ring but some locations contain more than one agent.

In Rendezvous of mobile agents in unknown graphs with faulty links [7], a group of agents has to be gathered at a single node in a graph when some of the edges in the network are faulty such that any agent entering one of these nodes would be destroyed. The objective is to minimize the number of agents that are destroyed and achieve rendezvous of all the surviving agents. The nodes start from a home base and they are identical. The system is totally asynchronous. The edges of the graph are labelled. The agents communicate by reading and writing information on whiteboards locally available at the nodes. No prior information about the network topology is required, except the size.

It has been assumed that faulty links do not disconnect the network, which may not be the case in a real world scenario.

Distributed Security algorithms for mobile agents [8] has described some problems and solution techniques developed in this investigation in the context of security. The focus has been on two security problems: locating a black hole and capturing an intruder. It

has described the computational issues and the algorithmic techniques and solutions. The environment is a collection of autonomous mobile agents located on a graph. They are asynchronous, have limited computational capabilities and private storage. Each node of the network may provide a storage area called whiteboard for incoming agents to communicate and compute, and its access is held in fair mutual exclusion. The research concern is on determining what tasks can be performed by such entities, under what conditions, and at what cost. Problems such as detection and decontamination of the harmful hosts have been discussed.

Further studies about multiple attacks, i.e., the general harmful host location problem when dealing with an arbitrary, possibly unknown, number of harmful hosts present in the system needs to be carried out. The problem of designing solutions when the harmful host represents a transient danger, i.e. when the harmful behaviour is not consistent and continuous but changes over time needs further research.

In Uniform Scattering of Autonomous Mobile robots in a grid [17], the uniform scattering problem for a set of autonomous mobile robots deployed in a grid network is considered where starting from an arbitrary placement in the grid, using purely localised computations, the robots must move so to reach in finite time a state of static equilibrium in which they uniformly cover the grid. The robots are randomly deployed in a region but it is required to cover the region uniformly so as to maximise coverage i.e. starting from an initial random placement on the SQUARE grid, the robots must position themselves in the grid at equal distance within finite time, without any collisions. The protocol used makes the assumptions that it does not require any direct or explicit communication between robots. The robots are fully asynchronous having a limited visibility range. Each robot uses a persistent memory similar to Finite State Machine. Global localisation system is not required, but only local orientation in the grid. Robots are anonymous and totally identical.

Each robot performs a sequence of operations consisting of LOOK-COMPUTE-MOVE which constitutes the unit cycle and each robot repeats this cycle. All operations are assumed to be instantaneous.

The uniform scattering algorithm SCATTER is asset of local rules for the robots. Each robot has a state variable initialised to -1. The algorithm is divided in three phases namely cleaning, collecting, deploying.

In Multi-robot Tree and Graph Exploration [18], a situation of multiple robots exploring an obstacle-dense environment, which is modelled as a graph, from a single starting vertex has been considered. The algorithm is called MULTIROBOT DFS algorithm based on the classical single robot DFS.

The graph is initially unknown; existence of edges becomes known only when a robot sees one end of the edge from a vertex, and the other end of the edge becomes known only when the robot actually follows that edge. This models an environment of sites with passages between them, where the passages are opaque: from either end, it is not clear where the passage goes. All edges have unit length, and each robot can follow one edge in each time step.
On trees it has been proven that the algorithm is optimal for two robots. For k robots, the algorithm has an optimal dependence on the size of the tree but not on its radius. Communication can be realized by bookkeeping devices being dropped by the robots at explored vertices, the states of which are read and changed by further visiting robots.
The robots move synchronously in time steps, and we want to minimize the total number of time-steps before the robots return to the start vertex and declare the search finished. In each time step, we assume that robots standing at the same vertex have an initial negotiation phase in which they decide which robot takes which edge. On trees, the algorithm becomes simpler since all robots enter a vertex by the same edge for the first time, coming from the root, and it cannot happen that a robot re-enters a vertex by a different edge than the one by which he left it.

The most-important problem for the practical applicability of this algorithm is to remove the assumption of robot movement in time steps; the real-robot movement is asynchronous, and the algorithm itself makes no assumption on synchronization, i.e., artifact of the analysis.

Distributed Algorithms for autonomous mobile robots [13] describes investigations on the interplay between robots' capabilities, computability and algorithmic solutions of coordination problems by autonomous mobile robots. The focus is to identify minimal robot capabilities that allow a problem to be solvable and a task to be performed.
Each robot performs the Wait-Look-Compute-Move computation cycle to reach its destination such as a gathering point. The robots can be classified as asynchronous or semi-synchronous. The robot may be able to sense a complete plane or a portion of it. They may not agree on the coordinate system and the robot may have a small amount or a large amount of memory. Based on this, a number of problems such as pattern formation, circle formation, line formation and gathering problem have been discussed. The goal of these studies is to gain a better understanding of the power of distributed control from an algorithmic point of view.

Since the operating capabilities of the robots are quite limited. Robots having complex capabilities may be utilised in these models. Little is known about the solvability of other problems like spreading and exploration, about the physical aspect of the models and about the relationships between geometric problems and classified distributed computations.

In the area of reliability and fault-tolerance, lightly faulty snapshots, a limited range of visibility, obstacles that limit the visibility and that moving robots must avoid or push aside, as well as robots that appear and disappear from the scene clearly are all topics that have not yet been studied.

Gathering of Six Robots on Anonymous Symmetric Rings [19] describes the case of only six oblivious, i.e. memoryless robots placed on the nodes of an anonymous ring in such a way they constitute a symmetric placement with respect to one single axis of symmetry, so as to make them gather at a single node. The ring is considered to be anonymous in which neither nodes nor links have any labels. There is at most one robot in each node. Robots operate in Look-Compute-Move manner. The target node is either the current position of the robot or one of its neighbours.

In Gathering asynchronous oblivious mobile robots in a ring [20], the problem of gathering identical, memoryless, mobile robots in one node of an anonymous un-oriented ring has been considered. Robots start from different nodes of the ring. They operate in Look-Compute-Move cycles and have to end up in the same node.
For an odd number of robots, it has been proven that gathering is feasible if and only if the initial configuration is not periodic, and a gathering algorithm for any such configuration has been provided.
For an even number of robots, the feasibility of gathering except for one type of symmetric configurations has been decided and a gathering algorithms for initial configurations proved to be gatherable.
Mobile entities (robots), initially situated at different locations, have to gather at the same location (not determined in advance) and remain in it. This problem of distributed self-organization of mobile entities is known as the gathering problem. The main difficulty of gathering is that robots have to break symmetry by agreeing on a common meeting location. This difficulty is aggravated when (as in our scenario) robots cannot communicate directly but have to make decisions about their moves only by observing the environment. An un-oriented anonymous ring of stations (nodes) has been considered. Neither nodes nor links of the ring have any labels. Initially, some nodes of the ring are occupied by robots and there is at most one robot in each node. The goal is to gather all robots in one node of the ring and stop. It has also been proven that without multiplicity detection i.e. the ability of the robots to perceive during the LOOK operation, if there is one or more robots in the given location.
It has been proposed that gathering of any 2 robots is impossible on any ring. If multiplicity detection is not available, then gathering any k $>$1 robots is impossible on any ring. Furthermore, gathering is impossible for any periodic configuration. For an odd

number of robots, gathering is feasible if and only if the initial configuration is not periodic.

Gathering Non-Oblivious Mobile Robots [20] describes a Gathering Problem where a set of n autonomous mobile robots is gathered at a point in the plane. This point is not fixed in advance. The robots are very weak, in the sense that they have no common coordinate system, no identities, no central coordination, no means of direct communication, and no synchronization. Each robot can only sense the positions of the other robots, perform a deterministic algorithm, and then move towards a destination point. It is known that these simple robots cannot gather if they have no additional capabilities. Hence, the robots have been assumed to be non-oblivious, i.e., they are equipped with memory.
In this paper it is shown that gathering problem is solvable for n $\geq$ 2 robots.
For n=2 robots, they are made to move on a line 'l' away from each other until both have seen the configuration at least once. Then they know the connecting line 'l' from their initial positions. In the next phase they both move on lines that are perpendicular to 'l' again, until both have seen the other robot at least once on the perpendicular line. Then they both know 'l' and its intersection with the two perpendicular lines, hence, they can gather on 'l' in the center between the perpendicular lines.
For n >2, the concept of smallest enclosing circle or SEC is implemented. If there are more than two robots on SEC, then each robot moves on a circle around the center of SEC until all robots have seen SEC. Thus, we use the fact that the smallest enclosing circle of the robots' positions does not change. Then all robots gather at the center of SEC. On the other hand, if there are only two robots on SEC, then the robots that are not on SEC move perpendicular to the line 'l' connecting the two robots on SEC, while the robots on SEC move on line 'l' away from each other. The smallest enclosing circle increases, but 'l' remains invariant. As soon as all robots have seen line 'l' and the configuration, they gather at the intersection between 'l' and a line k, which is the median perpendicular line of the robots, if n is odd, or the center between the two median perpendicular lines, if n is even.

This algorithm uses generous amount of memory to store the exact positions of robots. Further work can be done to minimise the amount of memory used by robots and find the minimum amount of memory necessary to solve the gathering problem.

2.2 Contribution of our work

We consider the model of robots (mobile agents) to have minimal capabilities. Even though it makes designing an algorithm more difficult, we did it so that the robots need the least amount of hardware. For example, if we had a fully synchronous system of robots [4], we need to add a timer module in every robot. Asynchronous robots do not need a timer module. More hardware gives rise to more expensive robots. The crux of swarm robotics is to have cheap, mass producible robots which can be easily replaced when a few are damaged. Having an expensive robot defeats the purpose of the system [3].

We have assumed the entire system to be asynchronous and the mobile agents have limited knowledge about the network configuration. We determine under what conditions this is possible and present algorithms for achieving rendezvous in such cases. We assume that the faults are permanent i.e. all the faulty nodes remain faulty throughout the duration of execution of the rendezvous algorithm and no new faults may develop during this interval. The locations of the faulty nodes are initially unknown to the agents. The task of the agents is to meet at a safe node, while avoiding the faulty nodes [7].

Obviously, not all agents would be able to meet, because some of the agents may die while traversing the faulty nodes. So, our objective is to minimize the number of agents that die and achieve the rendezvous of all the surviving agents whenever possible. If there are δ faulty nodes in the network then no more than k-δ agents may rendezvous, in the worst case. There are two algorithms required to solve the problem, namely, The Black Hole Search Algorithm and The Gathering Algorithm [3].

CHAPTER 3

THE BLACK HOLE SEARCH ALGORITHM

3.1 Definition of Problem

A group of mobile agents wandering among the nodes of a network has to gather at a single node of the graph [3]; this problem known as the Rendezvous problem has been studied extensively but only for networks that are safe or fault-free. In this thesis, we consider the case when some of the nodes in the network are dangerous or faulty such that any agent entering one of these nodes would be destroyed. Our objective is to minimize the number of agents that are destroyed and achieve rendezvous of all the surviving agents.

We determine under what conditions this is possible and present algorithms for achieving rendezvous in such cases. We assume that the faults are permanent i.e. all the faulty nodes remain faulty throughout the duration of execution of the rendezvous algorithm and no new faults may develop during this interval. The locations of the faulty nodes are initially unknown to the agents. The task of the agents is to meet at a safe node, while avoiding the faulty links.

Obviously, not all agents would be able to meet, because some of the agents may die while traversing the faulty edges. So, our objective is to minimize the number of agents that die and achieve the rendezvous of all the surviving agents whenever possible. If there are δ faulty links in the network then no more than k-δ agents may rendezvous, in the worst case.

3.2 Model Description and Assumptions

In a networked environment that makes use of mobile agents, security is a pressing concern, and possibly the most difficult one to address [10]. Actually, even the most basic security issues, in spite of their practical urgency and of the amount of effort, must still be effectively addressed [9].

The causes of this situation are not lack of interest and effort, but rather the (unexpected) difficulties found when developing solutions; the nature of these obstacles is

varied, most are technological, some computational. Witness, for example, the large effort to determine how to protect a network site (a host) from malicious agents, as well as to protect agents from host attacks.

We consider the issue of host attacks; that is, the presence in a site of processes that harms incoming agents. Obviously, the first step in any solution to such a problem must be to identify, if possible, the harmful host; i.e., to determine and report its location; following this phase, a ”rescue” activity would conceivably be initiated to deal with the destructive process resident there. The task to identify the harmful host is clearly dangerous for the searching agents and, depending on the nature of the harm, might be impossible to perform.

In this thesis, we consider a highly harmful process: a stationary process that disposes of visiting agents upon their arrival, leaving no observable trace of such destruction. Due its nature, the site where such an item is located is called a black hole [4][6][9]. The task is to unambiguously determine and report the location of the black hole, and will be called black hole search.

A black hole is a harmful node in the network that destroys any mobile agent visiting that node without leaving any trace [4]. The objective of the Black Hole Search problem is to identify the black hole without destroying too many agents and the main effort is to discover the minimal hypotheses under which it can be solved.

The network is assumed to be an anonymous and un-oriented tree. All the nodes are identical [3]. The locations of the black holes or unsafe nodes are not known to the mobile agents. Also, no information about black holes is available in the safe nodes of the network (the nodes which are not black holes).

The system is totally asynchronous, such that every action performed by an agent takes a finite but otherwise unpredictable amount of time. In an asynchronous network no global clock is available; the speed with which an agent computes or moves between nodes, while guaranteed to be finite, is not a-priori determined.

Thus, in order to locate a black hole, at least one agent must visit it. An agent entering a black hole disappears there, so later this agent cannot show up where expected by the other agents, or communicates with them in any other expected way. On this basis, the other agents may be able to deduce the exact location of a black hole.

We want to design an efficient communication algorithm for the agents to identify all black holes reachable from the start node. (If black holes disconnect the network into separate components, then we can only hope to explore the component which contains the start node.)

3.3 Description of Algorithm

The mobile agents are anonymous, i.e. they are indistinguishable from one another having no name or label and they execute the same algorithm. Any agent may wake up any time and start executing the algorithm and move from one node to the adjacent node by performing Look-Compute-Move cycles. The system is totally asynchronous, such that every action performed by an agent takes a finite amount of time.

We assume that the agents communicate by reading and writing information on public **whiteboards** locally available at the nodes of the network [9][10]. Thus, each node, say v, has a whiteboard (which is a shared region of its memory) and any agent visiting node v can read or write to the whiteboard. Access to the whiteboard is restricted by fair mutual exclusion, so that, at most one agent can access the whiteboard of a node at the same time, and any requesting agent will be granted access within finite time. An agent that is granted access to the whiteboard at node v, is allowed to complete its activity at that node before relinquishing access to the whiteboard (i.e. access control is non-preemptive). In order to locate a black hole at least one agent must visit it.

Case 1: The black holes are assumed to be present at the leaf nodes of the tree.

Initially, the k mobile agents are assumed to be placed randomly on nodes which are safe. All nodes of the tree are identical except the target node which may be labelled (Fig. 3.1, Fig. 3.2).

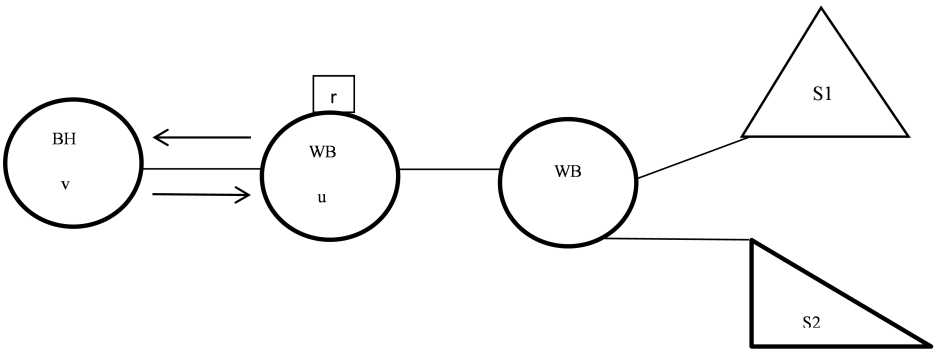

Fig. 3.1 Black Hole at leaf node.
BH: Black Hole, WB: Whiteboard
r: Mobile Agent S1,S2: Sub Trees

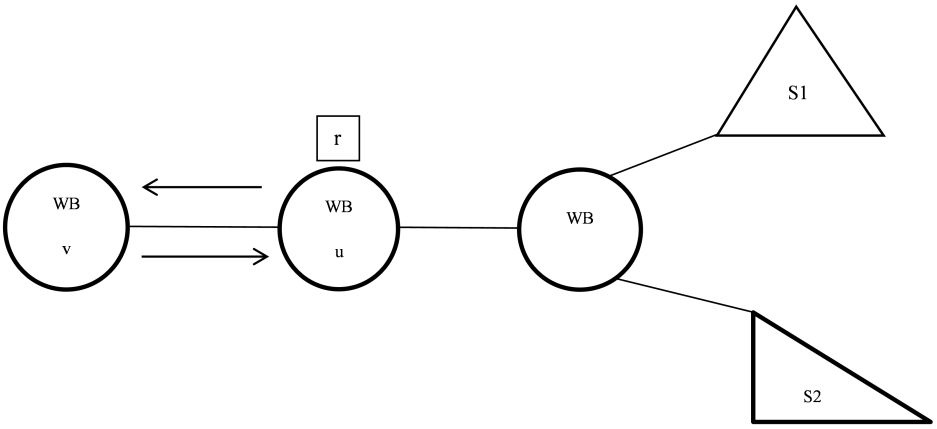

Fig. 3.2 Leaf Node, v is safe.
BH: Black Hole, WB: Whiteboard
r: Mobile Agent S1,S2: Sub Trees

Steps:

1. Each mobile agent should start traversing the graph, moving from one node to the adjacent node, using Look-Compute-Move cycles, preferably in DFS approach since the black holes are assumed to be present at the leaf nodes.

2. When an agent moves from node u to node v, it writes “active” on the whiteboard present in node u and proceeds to node v.

3. If node v is not a black hole, the agent turns 180 degree, returns to node u and writes “safe” on the whiteboard.

4. When any other agent reaches node u, it checks the information present on the whiteboard. If it sees “active”, it does not traverse from node u to node v. If the agent sees “safe” written on the whiteboard, the agent proceeds towards node v, having been convinced that v is not a black hole.

Pseudo Code: Detection of Black Holes

Input: A set of k agents, placed randomly on the tree and x number of black holes
Output: A tree consisting of safe nodes, devoid of black holes.

/* WB=region of whiteboard memory where information is stored. */

BEGIN
Let an agent be at node u, which is safe and next node be v.
if WB is blank
{ Write WB=”active” and proceed to the next node v }
else if WB=”active”
{ Next node being explored, DO NOT proceed to the next node v }
else if WB=”safe”
{ Safely proceed to the next node v }
END

After detecting the black holes, we get a smaller tree consisting of only safe nodes and the remaining k-x agents proceed to the rendezvous point, i.e. target node by executing the gathering algorithm.

Case 2: The black holes are assumed to be scattered all over the tree (Fig. 3.3).

Initially, the k mobile agents are assumed to be placed randomly on nodes which are safe. All the nodes of the tree are identical except the target node which may be labelled. The locations as well as the number of black holes are not known beforehand by the agents. The size and topology of the network are not known by the mobile agents.

In this case, some portions of the tree may get disconnected if a black hole is present in the path traversed by an agent which blocks the agent from moving to the next node. If an agent(s) is/are present in this disconnected component, it will never reach the target node present in another component and remain isolated from the rest of the tree.

If the target/destination node is marked beforehand, it may happen that the target node will be unreachable from the start node due to the presence of a black hole in the path leading to the destination node (Fig. 3.4).

Steps: The steps are same as in case 1.

Pseudo code: The algorithm is same as in pseudo code of case 1.

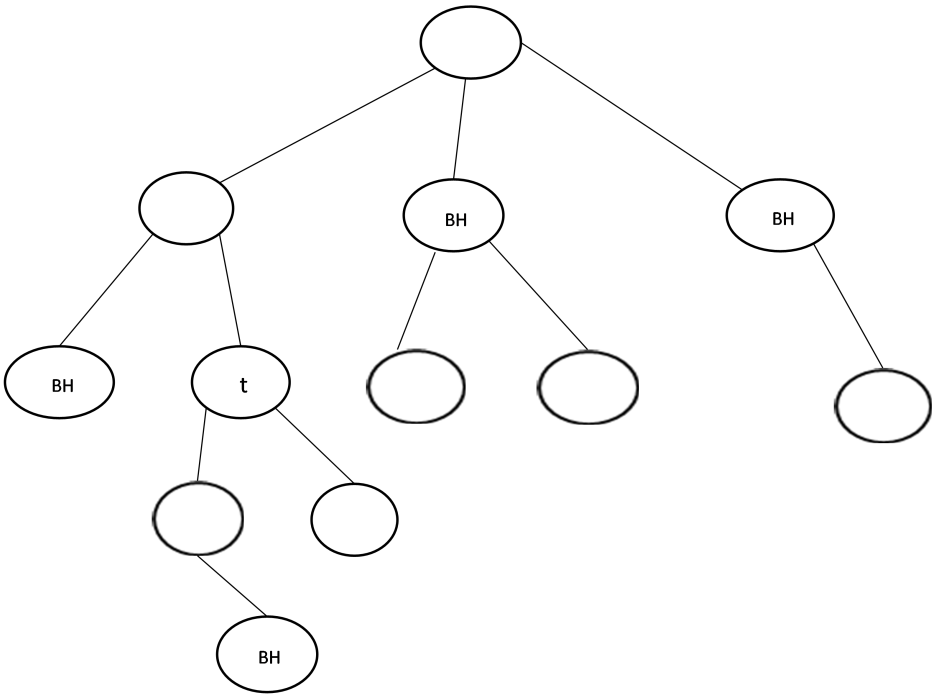

Fig. 3.3 The black holes are present randomly on the tree.
BH: Black Hole, t: Destination node

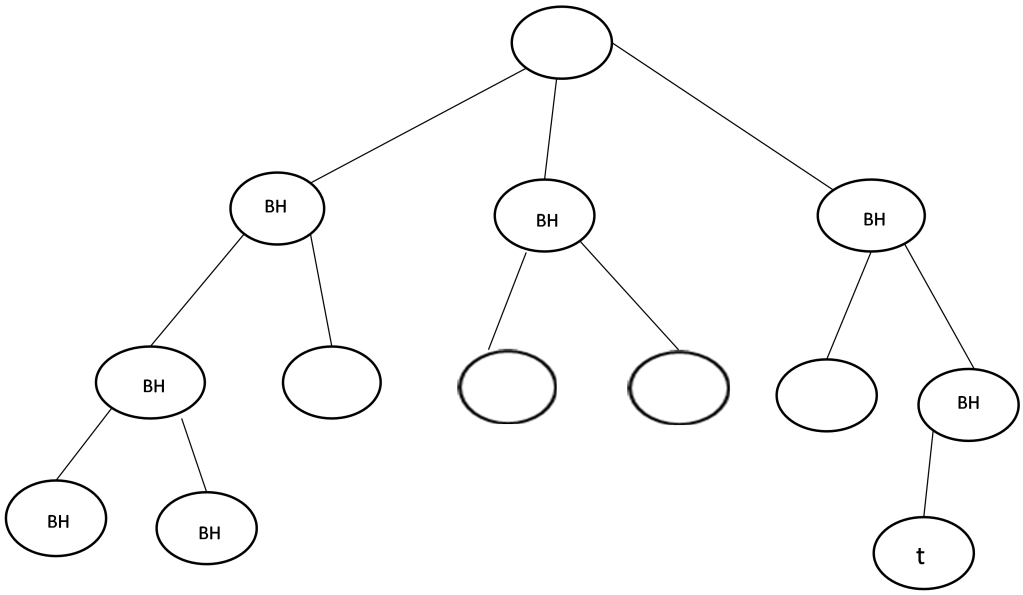

Fig. 3.4 The figure shows a configuration wherein if the target node is the one marked as 't', then it will be impossible for the mobile agents to gather at 't'.

3.4 **Correctness**

Condition: A safe tree is generated in finite time

According to the proposed model, every mobile agent performs the Black Hole search algorithm. The size of the tree is assumed to be finite. If an agent encounters a black hole then it is contaminated and other agents get informed that the node is unsafe by reading the information from the whiteboard placed in the adjacent node. After every execution cycle, a robot will either reach a black hole or a safe node.
If there are k mobile agents and x black holes, then after a finite number of execution cycles, x black holes will be detected and a safe tree will be generated.

3.5 **Complexity**

In this section, we will compute the number of possible Look-Compute-Move cycles for the Black Hole Search algorithm.

Let r be an agent placed at a node, u (say).
If the whiteboard at node u is blank, r will write ”active” on the whiteboard and proceed to adjacent node v.
Let the number of such movements be n1.
If the agent reads ”active” on the the whiteboard at node u, it will not move.
So, number of such movements is zero.
If the agent reads ”safe” on the whiteboard at u, it will proceed to the next node, v.
Let the number of such movements be n2.
These steps may be repeated k times.
Hence, total number of possible movements = k (n1 + n2).

CHAPTER 4

GATHERING OF AGENTS WHEN TARGET IS LABELLED

4.1 Definition of Problem

In this section, we present a distributed algorithm by which k robots deterministically gather at a single uniquely marked node in a tree with n nodes. The robots have limited visibility and have no common agreement in coordinate axes.

We consider that the robots move on a tree, that is, the robots will move from one node to another node and nowhere else, thus they execute discrete movements [3]. Initially, they are placed on some nodes of a given graph and they move through the edges and reach other nodes. However, they cannot stop on edges.

The robots can only see their immediate neighbours. Suppose any arbitrary tree is given. One node of the tree is marked uniquely. Rest of the nodes are identical. The robots are placed randomly on the nodes of the tree. The task of the robots is to gather at the uniquely marked node within a finite time. We propose a distributed algorithm for robots deployed in a tree, gathering at a single marked node, within finite time.

4.2 Model Description and Assumptions

We study the gathering problem in a distributed model of swarm robots that makes coordination of the robots' actions particularly hard, as robots cannot communicate directly but have to make decisions about their moves only by observing the environment. Moreover, they do not have memory of past observations.

Each robot is either active or inactive at some point of time. When a robot is active, it operates in Look-Compute-Move cycles.
In one cycle, a robot takes a snapshot of the current configuration (Look), then, based on the perceived configuration, makes a decision to stay idle or to move to one of its adjacent nodes (Compute), and in the latter case makes an instantaneous move to this neighbour (Move).

Cycles are performed asynchronously for each robot. This means that all the robots may not be active at the same time and the time spent in Look, Compute, and Move operations are not same for all the robots.

However, it is finite but unbounded.

The features of the robots are as follows:

1) Anonymous: The robots do not have any unique features by which they can be identified.

2) Oblivious: The robots have no extra storage capacity to remember data from past cycles. Everything is computed on-the-fly.

3) Silent: The robots cannot pass any explicit message to each other. Their method of communication is implicit, i.e. by observing the environment, gathering the position of other robots and nodes.

4) Local Axis: The robots do not have a global XY axes. They cannot determine the absolute position of other robots. Therefore, they cannot use the orientation of other robots to store the current status of the algorithm.

5) Weak Local Multiplicity Detection: The robots can only differentiate if there is 0, 1 or more than one robot in the current node. If there are multiple robots residing at the current node, then each robot cannot determine the exact number of robots at that node.

6) Instantaneous Movement: The movement of the robots is instantaneous in nature, which means the time taken for one robot to move from one node to another node can be considered negligible. Therefore, they are not visible at the edges and always seen at nodes.

7) Limited Visibility: A robot can only observe its neighbouring nodes. It means that the robot cannot observe the whole structure of the tree.

8) Asynchronous Activation Scheduling: The activation schedule of Look-Compute-Move will execute one after another but it will happen asynchronously.

The environment in which the agents will be travelling is a tree. Its features are as follows:

1) General Tree: The tree is denoted by T. It can have any number of children.

2) Anonymous nodes: Except the destination where the robots gather, all the nodes are identical.

3) Single unique node: The tree consists of a single destination which is marked. All the agents can identify the node as the gathering node. It is denoted by D.

The following definitions will be used in the algorithm:-

- Tower: If a node contains more than 1 robot, then we say a tower has formed at that node.
- Marking: Whenever a robot is sure that a certain path does not contain the destination, it cancels or blocks the path using an indicator which disallows other robots in traversing that path. This technique is known as marking.
- Linear leaf path: A collection of interlinked nodes, each of degree 2, which finally lead to a leaf node is defined as a linear leaf path (Fig. 4.1). The head of a linear leaf path is a node which has degree more than 2 (n4 in Fig. 4.1) but all of its subsequent children must be degree 2, with its tail as the leaf. It can be imagined as a linked-list.

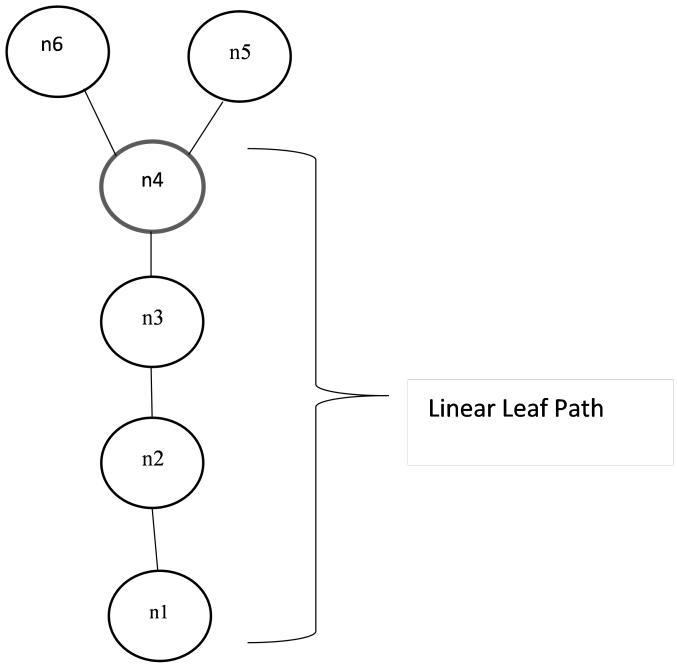

Fig. 4.1 Nodes n1 to n4 form the linear leaf path.
It starts from node n4 and ends at n1.
n4 is considered as the head and n1 is considered as the tail

- Junction: A junction is an intermediate node which leads to several linear leaf paths.
- Open and closed paths: An open path is a path which has not been marked by a robot, that is, it might lead to the destination.
  A closed path is a path which has been marked by a robot as it does not lead to the destination.

4.3 Description of Algorithm

The algorithm is as follows:

Step 1: The **roam and check phase** (Fig.4.2) restricts the robot in visiting an already visited node by not allowing the robot to turn 180 degree.
Thus the robot will either reach the destination or reach a leaf.

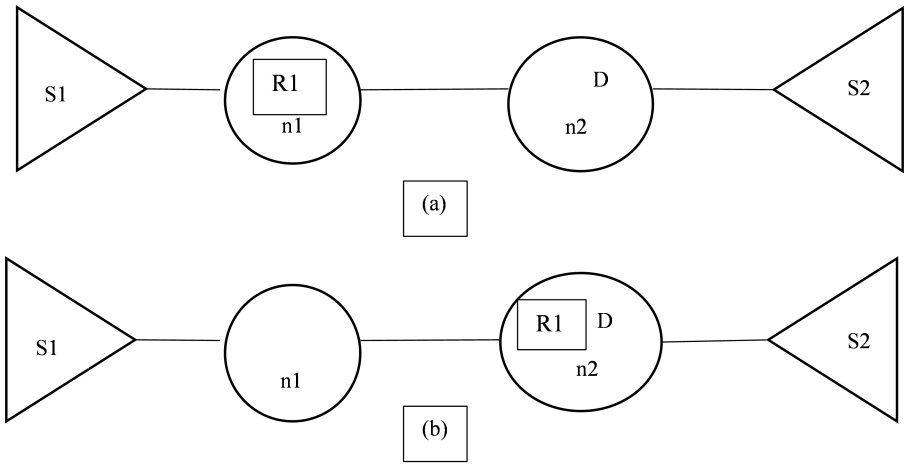

Fig. 4.2 (a) The robot R1 is at node n1 and n2 is the destination D. S1 and S2 are sub trees. The robot was randomly wandering in S1 and finally came to n1 but it cannot traverse back to S1. R1 detects that n2 is its neighbour.
(b) After R1 detects the destination, it moves to n2.

Step 2: The **leaf marking phase** (Fig. 4.3) marks linear leaf paths which do not contain the destination.

Thus, certain paths are blocked and robots will not tread upon a path which another robot has already visited, increasing the chance of other robots finding the destination quicker.

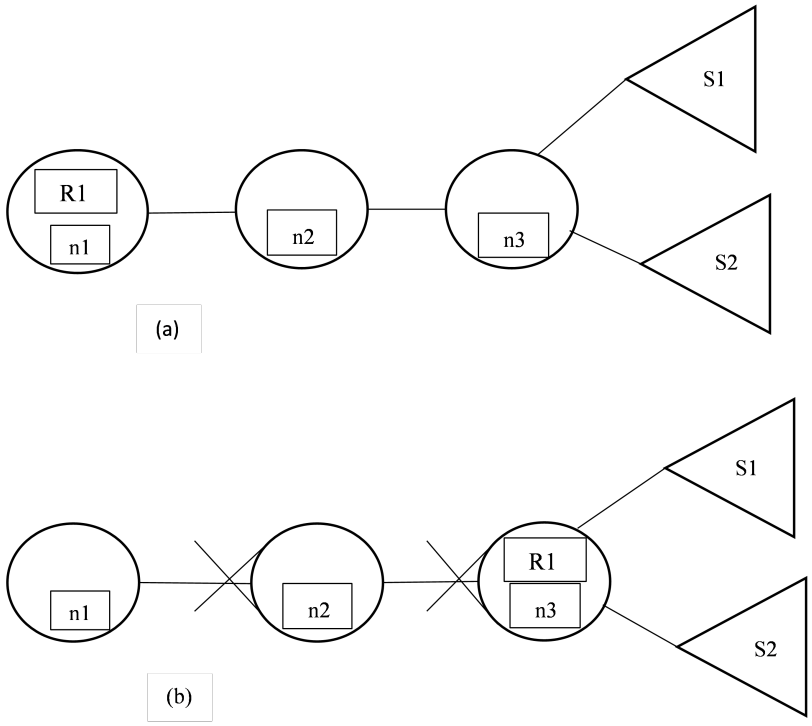

Fig. 4.3 (a) The robot R1 detects that it is at the end of a leaf.
(b) The robot turns around and proceeds to the nearest node and marks the previous path (indicated by the cross marks) as visited till it finds a node with number of open paths more than 1. In this diagram, it's node n3. Even though node n2 has 2 paths, one of its path is blocked, hence effective number of paths is 1.

Step 3: The **junction marking phase** (Fig.4.4) cancels nodes which are an intersection of multiple cancelled linear leaf paths (which were cancelled in the leaf marking phase). This blocks an entire sub tree with the purpose being same as the leaf marking phase.

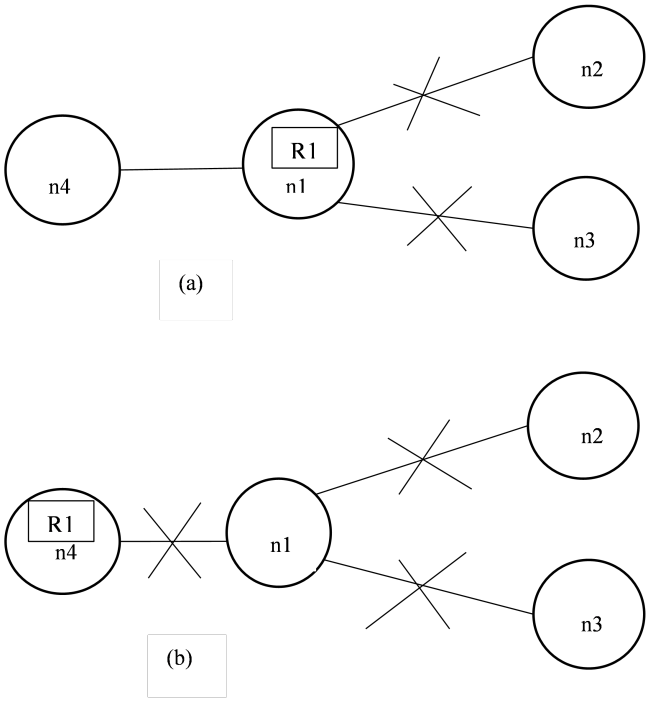

Fig. 4.4 (a) The robot R1 detects that it's in a junction but it finds out that, except one, all of its paths have already marked previously.
So in (b), it proceeds to the only available path and marks the path through it came, thus marking a junction.

Step 4: The **tower breaking phase** (Fig. 4.5) improves coverage of the robots and also prevents formation of deadlocks.

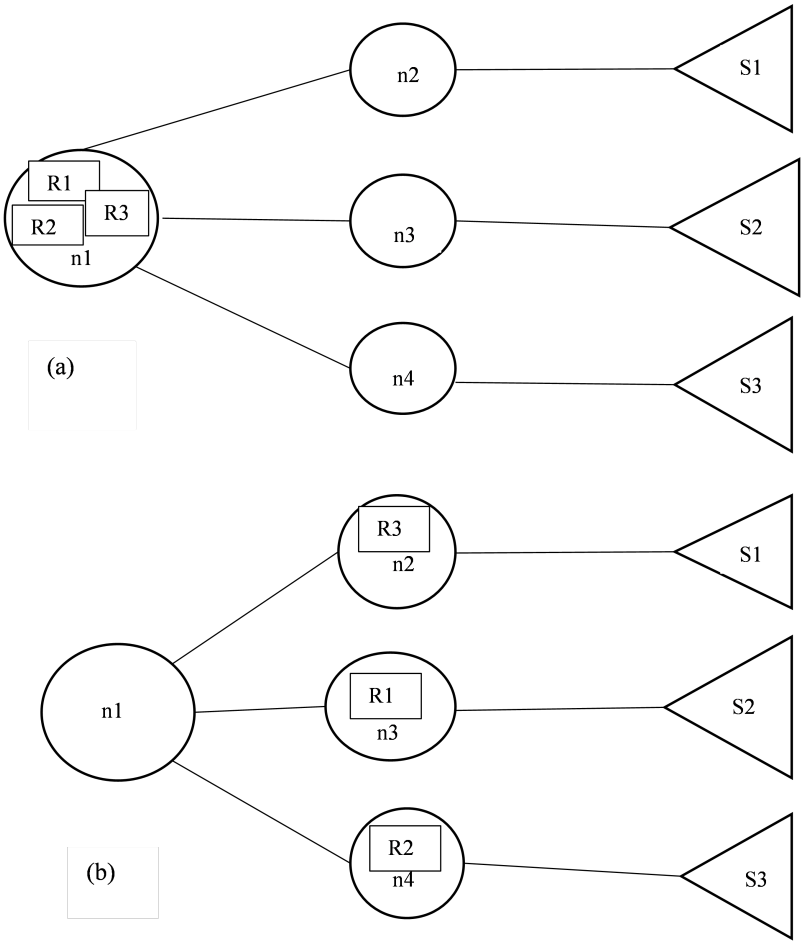

Fig. 4.5 (a) The robots R1, R2 and R3 form a tower at node n1.
(b) The tower at node n1 is broken. If the number of open paths for a node which contains a tower, say t1, is lesser than the number of robots t1, towers would have again formed in the neighbouring nodes. This process of breaking of towers would continue till there are no more towers or when the robots have reached the destination.

All robots are fed with the same set of instructions and they execute them independently. At every point, all the conditions are evaluated and only one condition will be evaluated as true.

None of these rules requires memory for storing computed data for future use. All decisions are taken with respect to the current snapshot.

Staring from its initial position, every mobile agent will execute the black hole search algorithm as well as the gathering algorithm to check whether the next node to be visited is a destination or an unsafe node.

The agents will continue its execution cycles until it reaches the target node or it gets destroyed / contaminated by a black hole.

Ultimately, a safe tree will be generated and the remaining agents will gather at the target node.

Pseudo Code: Gathering of Mobile Agents.

Input: $r_i \in R$
Output: $D(r_i)$: A node for r_i

/*GP = gathering point, CN = current node, NN = a neighbour node of CN, LF = leaf node, n = number of neighbours of CN with unmarked paths = number of possible paths from CN */

BEGIN

while (1) {
if CN == GP /*Condition only arises if the robot lies at destination */
then
    $D(r_i)$ = CN;
/*the robot does not move */
/*ROAM and CHECK PHASE */

if NN == GP /*Neighbour is the destination */
then
    $D(r_i)$ = NN;
/*ROAM and CHECK PHASE */

if CN contains tower then
    $D(r_i)$ = random unmarked NN;
/*TOWER BREAKING PHASE */

if CN has more than 1 path /*the robot is an intermediate node */
then
if (n-1) paths out of n paths are marked visited /*Robot is standing in a junction */
then
    D(ri) = only possible NN accessible from CN;
    Mark the path to from NN to CN as “visited”;
/*JUNCTION MARKING PHASE */

else /*Multiple open paths exist */
    $D(r_i i)$ = random unmarked NN other than previous node;
/*ROAM and CHECK PHASE */

if CN = LF /*The robots lies in a leaf node */
then {
Allow robot to rotate 180 degree;
$D(r_i)$ = only possible NN accessible from CN;
Mark the path to from NN to CN as"visited";
/*LEAF MARKING PHASE

It is the only possible NN because a leaf has only 1 neighbour */
}

r_i moves to $D(r_i)$;
if $D(r_i)$ == GP then
r_i stops executing Algorithm;
/*If the robot's destination is the gathering point, then the algorithm terminates */
}

END

4.4 Correctness

Condition 1: All mobile agents gather at the destination and the safe tree is created within finite time.

According to the proposed model, every mobile agent performs the Black Hole Search Algorithm as well as the Gathering Algorithm; hence it either reaches the destination or encounters a black hole. So we have to consider two cases:

Case 1.1: The mobile agent reaches the destination.

Let us assume an agent does not encounter a black hole. Starting from the initial position, the mobile agent will check whether the neighbouring node is a black hole or a destination. If it is a destination, it will move to the destination node and finish its execution cycle.

Let the hop distance from one node to another be ω and the agent wanders around the tree. In the worst case it will reach the leaf node which is the farthest location of the tree. Then it will turn 180° and return to the previous node thereby decreasing the hop distance ω by 1. By this time the tree may become smaller due to other agents locating

the black holes. Thus after finite number of execution cycles, ω will keep getting decreased by 1 and the agent will eventually reach the destination after a finite amount of time.

Case 1.2: The mobile agent encounters a black hole

According to our model, the mobile agents are placed randomly in the tree. The black holes are assumed to be scattered all over the tree, including leaf nodes. The agents either reach the destination or encounter a black hole. If an agent encounters a black hole, then it is contaminated and other agents get informed that the node is unsafe by reading information from the Whiteboard placed in the adjacent node which is at a hop distance of ω. After every execution cycle, a robot will either reach a black hole or a safe node (which may or may not be the destination).

If there are k mobile agents and x black holes, then after finite number of execution cycles, x black holes will be detected and a safe tree will be generated.

4.5 **Complexity**

In this section , we will compute the number of possible Look-Compute-Move cycles when the target node is labelled.

Let r1 be a mobile agent placed randomly on a node of the tree. During the Look operation, it will sense its adjacent nodes, the number of which is equal to the degree of the vertex on which r1 is present. The agent may Move to one of the nodes and continue the Look –Compute-Move cycle from the new node.

Hence, starting from the initial position, r1 can keep on moving to the next node till it reaches the leaf node.
Let the number of such movements be n1. Ref. Fig.4.6
From the leaf node, the agent will backtrack to the previous node.
Let the number of such backtracks be n2. Ref. Fig.4.7
These two steps will be repeated k times.
Let n3 be the move to reach the target node. Ref. Fig.4.8
Thus,
Total no. of movements possible= [k (n1 + n2) + n3]

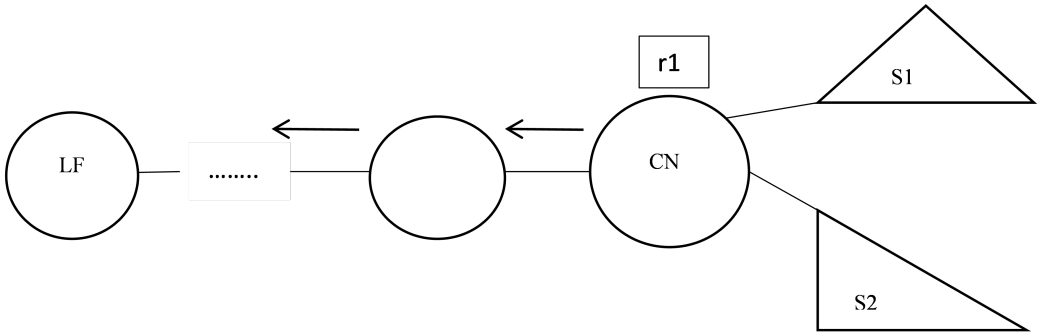

Fig. 4.6 r1 is placed at CN, the current node. S1,S2 are sub trees. LF is Leaf Node

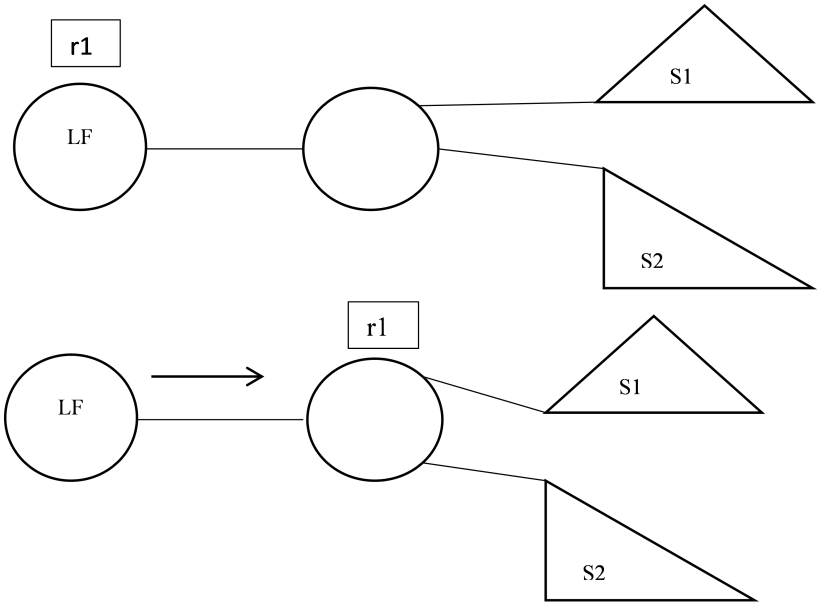

Fig. 4.7 (a) In the first figure r1 is at leaf node, LF.
(b) In the second figure r1 bactracks to the previous node.

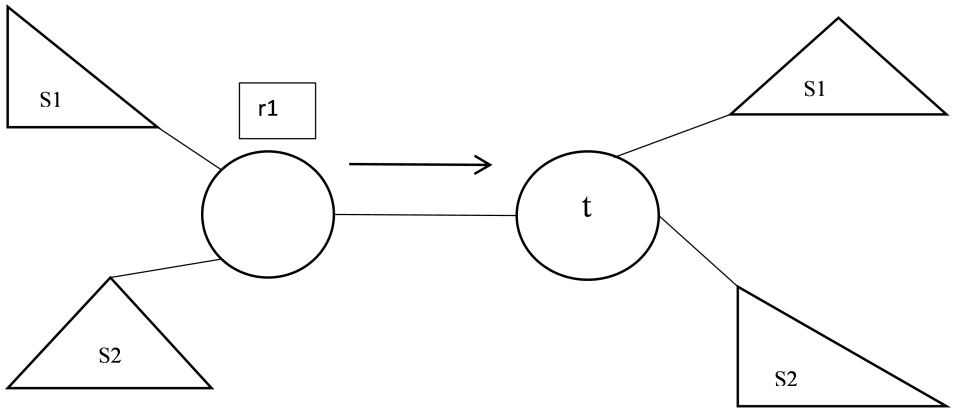

Fig. 4.8 The move n3, to reach the target node

We consider two cases:

Case 1: When initial node and target node are same.

When the target node is same as that of the node at which the robot r1 is placed initially, then there will be no movement to the adjacent node(s).

Thus n1=0, n2=0, n3=0, k=0.

Hence, total number of possible movements = 0.

Case 2: When initial and target nodes are different.

When the initial and target nodes are different, the best case will be that the target node is one of the adjacent nodes of the node in which the agent is currently placed.

Thus n1=0, n2=0, n3=1.

Hence, total number of possible movements = 1

CHAPTER 5

GATHERING OF AGENTS WHEN TARGET IS NOT LABELLED

5.1 Definition of Problem

We propose a distributed algorithm by which k robots gather at a single node which is not marked or labelled beforehand in a tree with n nodes. Unlike the gathering problem where the target or destination node was labelled and the agents gather at the designated node, we consider the scenario where the agents which are randomly placed on the tree, gather at a node which has no marking or labelling done prior to the initiation of the algorithm.

5.2 Model Description and Assumptions

The gathering task in robot-based computing systems represents one of the most fundamental problems widely considered in the literature. The basic requirement of the problem is to devise a distributed algorithm that allows a team of robots to meet at some common place [15][16].

In this section, we are interested in agents placed on the vertices of a tree [14]. Initially, the agents are placed arbitrarily on the nodes. Agents are equipped with visibility sensors and motion actuators, and operate in Look–Compute–Move cycles [15].

The Look–Compute–Move model assumes that in each cycle a robot takes a snapshot of the current configuration (Look), then, based on the perceived configuration, takes a decision to stay idle or to move to one of its adjacent vertices (Compute), and in the latter case it moves to this neighbour (Move).
Moves are assumed to be instantaneous and hence any robot performing a Look operation sees all other robots on nodes and not on edges.
Cycles are performed asynchronously, i.e., the time between Look, Compute, and Move operations is finite but unbounded. Hence, robots may move based on out-dated perceptions. In fact, due to asynchrony, by the time a robot takes a snapshot of the configuration, this might have drastically changed [16]. The scheduler determining the Look–Compute–Move cycles timing is assumed to be fair, that is, each robot performs

its cycle within finite time and infinitely often.

Robots are assumed to be uniform (running the same deterministic algorithm), autonomous (without a common coordinate system, identities or chirality), asynchronous (without central coordination), without the capability to communicate. Neither vertices nor edges are labelled (i.e., the graph is anonymous). They can see their adjacent nodes only (limited visibility). They are oblivious, that is, they cannot remember moves from their past cycles.

An important capability associated with robots concerns the so called multiplicity detection [13] [15][16]. During the Look phase, a robot may perceive whether a vertex is occupied by more than one robot in different ways. Here we assume the so called global strong multiplicity detection, meaning that robots perceive the actual number of robots among all the vertices. The global weak form considers robots able to detect only whether a vertex is occupied by more than one robot, but not the exact number. The local versions instead of global refer to the corresponding ability of a robot in perceiving the information about multiplicities only concerning the vertex where it currently resides.

The features of the robots (mobile agents) are as follows:

- Anonymous: The robots do not have any unique features by which they can be identified.
- Silent: The robots cannot pass any explicit message to each other. Their method of communication is implicit, i.e. by observing the environment, gathering the position of other robots and nodes.
- Limited Visibility: A robot can only observe its neighbouring nodes. It means that the robot cannot observe the whole structure of the tree.
- Instantaneous Movement: The movement of the robots is instantaneous in nature, which means the time taken for one robot to move from one node to another node can be considered negligible. Therefore, they are not visible at the edges and always seen at nodes.
- Asynchronous: There is no central control, i.e. they are asynchronous and move independently using Look-Compute –Move cycles.
- Oblivious: The robots have no extra storage capacity to remember data from past cycles. Everything is computed on-the-fly.
- Strong Multiplicity Detection: The agents have strong multiplicity detection wherein they can determine the exact number of agents occupying a node.

The agents do not have to remember the vertices that have been already visited; instead each node traversed by the agent is marked by writing information on the **whiteboard** (a small region of memory) present at the node. By doing this, the mobile agents do not need to remember a lot of information hence the amount of memory possessed by an agent is minimized to some extent. Black holes are present in the tree. The agents need to detect the black holes and finally gather at a destination node of the safe tree. Every agent will execute the same algorithm asynchronously.

The network in which the agents travel is considered to be a tree. Each and every node is identical and the node where the agents gather is not marked, i.e. the destination or target node is not labelled beforehand as in Fig. 5.1 and Fig. 5.2.

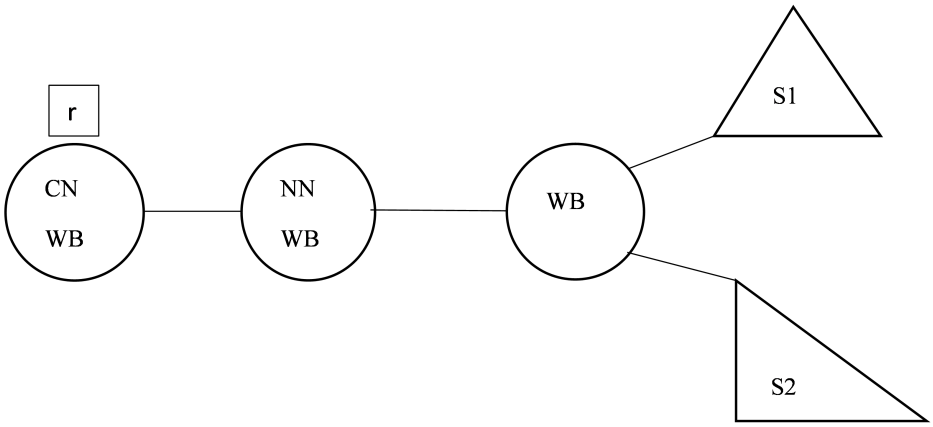

Fig. 5.1 (a) Agent r is present at a leaf node which is the current node, CN

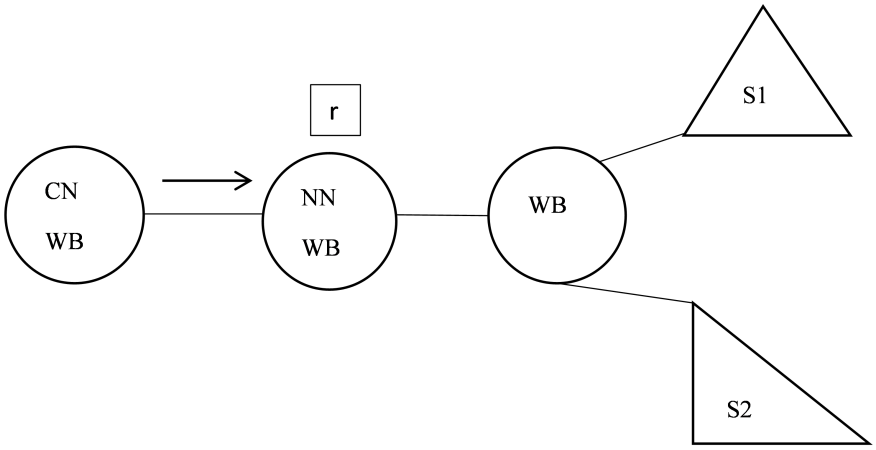

Fig. 5.2 (b) Agent r moves from current node to adjacent node, NN and writes WB=”VISITED” on CN

5.3 Description of Algorithm

The mobile agents are placed randomly on the tree network. The target node is not labelled hence every node of the tree is identical.
The objective is to gather the mobile agents to the **unlabelled** destination node.

The steps are:

The mobile agent will take a snapshot of the current configuration (Look). Based on the snapshot, it will take a decision whether to stay idle or move to an adjacent node (Compute).

1. If the current node is a leaf node, then write ”VISITED” on the whiteboard present at the leaf node and move to an adjacent non-leaf node. Each agent can mark the node visited by it, by writing on the whiteboard present on the node so that it does not go back to a leaf node already traversed.

2. If the current node has no agents and the adjacent node has one or more agents then the agent will move to the next node and mark the current node as ”VISITED” on the whiteboard. This step is required so that an agent keeps on searching for a vertex where one or more agents are present for gathering.

3. If the number of agents in the current node is less than the number of agents in the adjacent node, the agent will move to the adjacent node and mark on the whiteboard. This step is required for comparing nodes having different multiplicity.

After a finite number of cycles, the mobile agents will gather at an arbitrary unlabelled node.

Pseudo Code: Gathering of Agents (Destination not labelled)

Input: A set of k agents, placed randomly on the tree.
Output: Agents gathered at a node on the tree

BEGIN

```
/* r= agent placed arbitrarily on a node
CN= current node
NN= neighbouring node of CN
LF= leaf node
WB=region of whiteboard memory where information is stored on every node
r(CN)= number of agents present on current node
r(NN)=number of agents present on neighbouring node of CN */

for every r,

do
{
      if (CN == LF)
      then
         write WB = ”VISITED” on CN and move to NN

      else if (r(CN) == 0 and r(NN) >= 1)
      then
         write WB =”VISITED” on CN and move to NN

      else if (r(CN) <r(NN))
      then
         write WB =”VISITED” on CN and move to NN
}
```

END

Every agent will execute the same set of instructions independently of one another and try to gather at a node.

5.4 Limitations

Although the algorithm works on almost all configuration of a tree, but there are certain limitations where the algorithm fails to solve the gathering problem.
For instance, consider the Figure 5.3. In this figure, the tree has only two nodes, n1

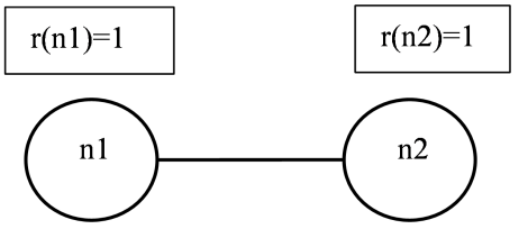

Fig. 5.3 node n1 and node n2 have one agent each

and n2. One agent is placed at each node. Our gathering algorithm fails in such a configuration. If we increase the number of agents, say r(n1)=8 and r(n2)=8, even then gathering is not possible in such configuration. This is a limitation of our proposed algorithm.

5.5 Correctness

Condition: The agents gather at a node within finite time.

Every mobile agent executes the same gathering algorithm independent of each other. The agents which are present at the leaf nodes move to non-leaf nodes and mark the leaf nodes so that they do not traverse them in future.
Thus, the tree gets shrunk and we get a smaller resultant network. Every node which has been visited by an agent is marked by writing on the whiteboard and multiplicity of each node is checked at every step.
The algorithm fails only in the case where the network comprises of two nodes and multiplicity of both nodes is same. Hence, after a finite number of cycles, the mobile agents will gather at an unlabelled node.

5.6 **Complexity**

In this section , we will compute the number of possible Look-Compute-Move cycles when the target node is not labelled.

Let r be an agent placed randomly on the node of a tree. It will wander around the tree and explore each node.

If the agent is in a leaf node, it moves to a non-leaf node.
Let the number of such movements be n1.

If the current node in which the agent is present has zero agents and its adjacent node has one or more agents then the agent will move to this adjacent node.
Let the number of such movements be n2

If the number of agents in the current node is less than the number of agents in the adjacent node, the agent will move to the adjacent node.
Let the number of such movements be n3.

These steps may be repeated, say k number of times.
Thus, total number of possible movements= [k(n1 + n2 + n3)]

CHAPTER 6

CONCLUSION

6.1 Summary of the Thesis

In this thesis, we have proposed algorithms to search for black holes in a network whose topology is tree, gather the agents at a labelled target node and an algorithm to gather the agents at an unlabelled node. Various cases have been considered with respect to the target node, the position of black holes in a tree network and subsequent detection of their positions by the mobile agents.
The gathering algorithm does not guarantee that a robot will traverse a path only once, which is a desirable trait of any gathering algorithm, but we are always eliminating paths which forces the robots to explore unknown paths, thus converging on the destination faster. It gives us an insight about the difficulties we have to face while encountering similar problems and the aspect of strict adherence to minimal hardware while designing robots.
We have assumed strong multiplicity detection in the case where the target node is not labelled beforehand. Instead of making the agents non-oblivious, we have taken the help of whiteboards to store data so that the agents are equipped with as minimum capabilities as possible.

6.2 Future Works

There are some open problems that still needs to be worked on are summarised as follows: -

- So far the research has been focussing on the feasibility of gathering but it would be interesting to investigate the minimum number of steps required for solving the gathering problem.
- It is not clear whether memory should be present on the nodes of the graph or on the robots itself to achieve maximum performance with minimum overhead.

- Despite its potential to promote robustness, scalability and flexibility, swarm robotics has yet to be adopted for solving real-world problems, for e.g. military applications like foraging, searching and rescuing; object clustering and assembling used in collective construction to produce 2D and 3D structures such as walls; obtaining a map of an environment using robot swarm and cereal harvesting.

- Some researchers have assumed that their solution involves some movement of the robots where a robot turns 180 degrees and take a move along the edge of the graph, but have not proposed any mechanism for this movement to take place.

- Further research needs to be done to overcome the limitation of the proposed gathering algorithm for unlabelled destination as it fails when the configuration comprises of two nodes of equal multiplicity.

REFERENCES

[1] Noa Agmon and David Peleg. Fault-tolerant gathering algorithms for autonomous mobile robots. SIAM Journal on Computing, 36(1):56–82,2006.

[2] Maitry Sinha, Srabani Mukhopadhyaya. Optimal Tree Search by a Swarm of Mobile Robots.In Avances In Intelligent Systems and computing,Proceedings of ICICT, 2016.

[3] S. Bhaumik. and S. Gan Chaudhuri. Gathering of Asynchronous Mobile Robots in a Tree. In Proc. IEEE 2nd International Conference, Applications and Innovations in Mobile Computing (AIMoC), 2015.

[4] Colin Cooper, Ralf Klasing, Thomas Radzik. Searching For black hole faults in a network using Multiple Agents.Springer, 2006.

[5] Armando Castaneda, Sergio Rajsbaum, Matthieu Roy. Two convergence problems for robots on graphs. In 7th IEEE Latin-American Symposium on Dependable Computing, Oct 2016.

[6] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Mobile search for a black hole in an anonymous ring. In Proc. of 15th Int. Symposium on Distr. Computing (DISC 2001), pages 166–179, 2001.

[7] Jeremie Chalopin, Santanu Das, Nicola Santoro. Rendezvous of Mobile Agents in Unknown Graphs with Faulty Links, Springer, 2007.

[8] Paola Flocchini , Nicola Santoro. Distributed Security Algorithms for Mobile Agents. In Mobile Agents in Networking and Distributed Computing, First Edition.2012

[9] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Searchng for a black hole in arbitrary mobile networks: Optimal mobile agent protocols. Springer, 2006.

[10] Euripides Markou. Identifying Hostile Nodes in Networks using Mobile Agents.In Bulletin of the EATCS no.108, pp. 93-129, October 2012.

[11] Shantanu Das. Mobile Agents in Distributed Computing: Network Exploration. In Bulletin of the EATCS no.109, pp. 54-69, February 2013.

[12] Pattanayak, Debasish, Kaushik Mondal, H. Ramesh, and Partha Sarathi Mandal. ”Fault-tolerant gathering of mobile robots with weak multiplicity detection.” In Proceedings of the 18th International Conference on Distributed Computing and Networking, p. 7. ACM, 2017.

[13] Giuseppe Prencipe, Nicola Santoro. Distributed Algorithms for Autonomous Mobile Robots. Fourth IFIP International Conference on Theoretical Computer Science-TCS 2006. Springer, Boston, MA, 2006.

[14] Suzuki, Ichiro, and Masafumi Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. In SIAM Journal on Computing 28.4 (1999): 1347-1363.

[15] Di Stefano, Gabriele, and Alfredo Navarra. Optimal gathering of oblivious robots in anonymous graphs and its application on trees and rings. Distributed Computing 30.2 (2017): 75-86.

[16] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of asynchronous robots with limited visibility. Theoretical Computer Science, 337(1-3), 147-168, 2005.

[17] Barriere L, Flocchini P, Mesa-Barrameda E, Santoro N. Uniform scattering of autonomous mobile robots in a grid. International Journal of Foundations of Computer Science. 2011 Apr;22(03):679-97.

[18] Peter Brass, Flavio Cabrera-Mora, Andrea Gasparri, and Jizhong Xiao, Multirobot. Tree and Graph Exploration. IEEE Transactions on Robotics, VOL. 27, No. 4, August 2011.

[19] D’Angelo, Gianlorenzo, Gabriele Di Stefano, and Alfredo Navarra. ”Gathering of six robots on anonymous symmetric rings.” International Colloquium on Structural Information and Communication Complexity. Springer, Berlin, Heidelberg, 2011.

[20] Ralf Klasing, Euripides Markou, and Andrzej Pelc. Gathering asynchronous oblivious mobile robots in a ring. Theoretical Computer Science, 390(1):27–39, 2008.

[21] Mark Cieliebak. Gathering non-oblivious mobile robots. In LATIN 2004: Theoretical Informatics, pages 577–588. Springer, 2004.