MOBILE AGENT BASED BIO-INSPIRED MECHANISMS FOR SERVICING NETWORKED ROBOTS

Thesis submitted

in partial fulfillment of the requirements

for the award of the degree of

Doctor of Philosophy

in

Computer Science and Engineering

by

W. WILFRED GODFREY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI – 781 039 ASSAM INDIA

April 2012

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI 781039, ASSAM, INDIA

Dr. Shivashankar B. Nair
Associate Professor
Phone: 0361-2582356(Off)
E mail:sbnair@iitg.ernet.in

Certificate

This is to certify that this thesis entitled ***"Mobile Agent based Bio-inspired Mechanisms for Servicing Networked Robots"*** submitted by ***Mr. W. Wilfred Godfrey*** in partial fulfillment of the requirements for the award of the degree of Doctor of Philosophy, to the Indian Institute of Technology Guwahati, Guwahati, Assam, India, is a record of the bonafide research work carried out by him under my guidance and supervision at the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, Guwahati, Assam, India, and, to the best of my knowledge, no part of the work reported in this thesis has been presented for the award of any degree from any other institution.

Guwahati,
23-04-2012

Dr. SHIVASHANKAR B. NAIR
(Supervisor)

DECLARATION

I hereby declare that the work presented in this thesis entitled "**Mobile Agent based Bio-inspired Mechanisms for Servicing Networked Robots"** is based on the original research work carried out by me under the guidance and supervision of Dr. Shivashankar B. Nair, Associate Professor, Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, Guwahati, Assam, India, and, to the best of my knowledge, no part of the work reported in this thesis has been presented for the award of any degree at any other institution.

Guwahati
23 -04-2012

W. Wilfred Godfrey

DECLARATION ON PLAGIARISM

I, W.Wilfred Godfrey, (Roll No. 05610102), understand that plagiarism is defined as any one or the combination of the following: Uncredited verbatim copying of individual sentences, paragraphs or illustrations (such as graphs, diagrams, etc.) from any source, published or unpublished, including the Internet.

Uncredited improper paraphrasing of pages or paragraphs (changing a few words or phrases, or rearranging the original sentence order). Credited verbatim copying of a major portion of a paper (or thesis chapter) without clear delineation of who did or wrote what. (Source: IEEE, The Institute, Dec. 2004)

I have to the best of my abilities ensured that all the concepts, ideas, text, expressions, graphs, diagrams, etc., which are not the outcome of my research have been credited to those who own the same. Elaborate sentences used verbatim from published work have been clearly identified and quoted. I also affirm that no part of this thesis can be considered as plagiarism to the best of my knowledge and understanding and take complete responsibility if any complaint arises.

I am fully aware that my thesis supervisor(s) are not be in a position to check for any possible instances of plagiarism within this submitted work.

Guwahati
23 -04-2012

W. Wilfred Godfrey

Dedicated to

My Loving Parents and My Beloved Wife

ACKNOWLEDGEMENTS

First of all I thank the Almighty God for giving me the dream to pursue PhD and this opportunity and the strength to carry out this work.

I hereby acknowledge my sincere gratitude to my guide Dr. Shivashankar B. Nair for having taken me as his student and been consistently on my side all throughout this venture. I am also grateful to him for always patiently listening to my problems, ideas and doubts even when they were trivial or silly and for all his support and encouragement during the course of my PhD. Indeed, I have been inspired by his sincere and hard-working attitude, persistence and an out of the box outlook towards research. I sincerely hope that the research training that I have acquired while working with him will drive my efforts in future.

I would also like to thank the members of my doctoral committee, in particular Profs. P.K.Das, Chitraleka Mahanta and Pinaki Mitra, for their feedback and encouragement during the course of my PhD work. I am also grateful to the entire Computer Science faculty for their tremendous support and affection during my stay at IIT Guwahati. I would also like to thank the anonymous referees who have commented at various forums, on the work presented in this thesis. Their comments have been specically valuable in improving the rigour and presentation of this piece of work.

I also gratefully acknowledge MHRD, Govt of India and R&D Department, IIT Guwahati for supporting my research at IIT Guwahati.

I would like to thank engineers, Nanu, Bhirguraj, and office staffs namely, Hemanta, Rakthajit, Nabo, Malakar, Souwik Security who were warm and welcoming and were always helpful whenever I approached them.

I take this opportunity to express my heartfelt thanks to my friends and colleagues who made my stay at IIT Guwahati an enjoyable experience. Knowing and interacting with friends such as Alka, Rajendra, Agile, Nagesh, Suresh Babu, Sathyanarayana, Suddhasil De, Bidyut, Mohanty, Lipika, Mamata, Moushumi has been a great experience. These friends and many others have always been by my side whenever I have needed them.

I would also like to thank the Robotics Laboratory Friends, Soni, Ram, Sukumar, Sumit, Vibhor , Surjeet, Shrinivasa, Shashi Sekhar Jha and indeed it is a privilege knowing them all. Their help and kinship is something to cherish at all times. I would also like to make a special mention of and heartily appreciate Shashi who helped me with their hands on coding when I was time strapped at some crucial junctures.

I am deeply indebted to Roy Paily Sir and his Family for their love, affection, encouragement and support that they provided me at various junctures of my life here at IIT Guwahati.

It goes without saying that this journey would not have been possible without the tremendous support and encouragement I received from my Dad and my Mom. Indeed no words can express my gratefulness towards them who have unconditionally and whole-heartedly supported me in all my endeavours.

I am also grateful to my in-laws for their understanding, patience and tremendous support. I am also thankful to my sister-in-law Sherji for being a good friend at all times.

Last but not the least, I thank my wife Suji for her Sacrificial Love, constant encouragement, support and infinite patience. Without her, this journey could not have been completed so smoothly.

W. Wilfred Godfrey

CONTENTS

LIST OF TABLES

Chapter 7

LIST OF FIGURES

Chapter 2

Chapter 3

Chapter 4

Chapter 5

Chapter 6

Chapter 7

Chapter 8

LIST OF SYMBOLS

Symbols	Description
n	A node
t	Time
a	A mobile agent
P^n_a	Next node position of agent a at node n
ψ_n	Set of one-hop neighbours of node n
V_a	*Visited-Node* list of agent a
ψ'_n	Set of idleness values of one-hop neighbours of node n
ψ''_n	Set of propagated idleness values of k one-hop neighbours of n
r'	Idleness value at a node
r''	Propagated idleness value at a node
C	Pheromone concentration
C_{max}	Maximum value of concentration
ΔC	Concentration gradient
d	Pheromone spanning length
$P_{pc}(n)$	Diffused pheromone concentration at the n^{th} hop from the RRS
T	*Time-out* of a pheromone
T_{max}	Maximum value of *time-out*
$P_{t\text{-}o}(n)$	Diffused pheromone *time-out* at the n^{th} hop from the RRS
x	Service
δ	Pheromone re-laying interval
π_i	i^{th} pheromone
π_i^S	Service information in i^{th} pheromone

π_i^C	Concentration information in i^{th} pheromone
π_i^T	Time-out information in i^{th} pheromone
π_i^B	Neighbour information in i^{th} pheromone
ρ	Cloning pressure
Q_{Th}	Queue threshold
R	Cloning resource
R_{av}	Available *cloning resource*
R_{max}	Maximum *cloning resource*
Rew	Reward
τ_c, τ_r, σ	Non-zero positive constants
L	Life-time of an agent

GLOSSARY OF TERMS

Artificial Immune System (AIS):	These are Computational systems with sound methods and mechanisms inspired by vertebrate Immune system. These algorithms typically exhibit characteristics such as learning and memory in solving a problem.
Antigen:	This is a Foreign substances or microorganisms responsible for infection or disease in an organism which also triggers the proliferation of antibodies which neutralizes the antigen.
Antibody:	These are Y shaped proteins in the immune system which bind to infectuous antigens leading to their destruction. They are mainly produced in the Thymus.
Antigen Presenting Cells (APC):	APCs are accessory cells which act upon the antien and present them to T-cells in a form that they can recognize and destroy.
Clonal Selection Theory:	This is a widely accepted theory on how immune system responds to antigenic infection. According to this theory, antibodies of single specificity look for specific antigens with complimentary cell-surface receptors. When exposed, these antigens trigger generation of large quatities of specific antibodies while the total population of antibodies are maintained.
Cloning (Agent):	The biological process of generating genetically identical individuals is cloning. Agent cloning creates one or multiple copies of identical agents.
Cloning- Resource:	A variable which controls cloning but functions non-linearly due to its replenishment by rewards and constant recharging nature.
Complement System:	As part of the innate immune system, this set of proteins that act together with the antibodies in the

	attack against extra cellular pathogenic agents and complement the ability of the antibodies.
Immune Network Theory:	This theory explains the working of the adaptive immune system. According to this theory immune system is a network of variable regions of cells where cells may be either antigens or native cells.
Internet of Things:	Network technology which facilitates connection and communication between not only human-human but also including everyday objects.
Negative Selection:	The mechanism which explains how the newly generated specific antibodies are not specific towards the self. This is explained through a process by which self-specific immune cells are removed from the system.
Pheromone:	Chemical signals used by organisms as a mode of communication and interaction eliciting innate behaviours among them. Some vertebrates and even plants communicate using pheromones.
Secondary Response:	A secondary response occurs due to a subsequent exposure to the antigen. This results in an antibody response which is faster and higher than what occurred during the primary response.
Somatic Hypermutation:	This is the process of antibody tuning of its variable region through mutation of the antigen receptors randomly. This helps in antibody molecules detecting new types of antigens quickly. This also explains the adaptive response of the immune system.
Stigmergy:	This is a type of communication among agents performed in an indirect manner through traces left in an environment. It produces structures which are complex without any planning or control.

PUBLICATIONS

International Journals:

1. **Shivashankar B. Nair, W. Wilfred Godfrey**, "On Realizing a Multi-Agent Emotion Engine", International Journal of Synthetic Emotions, Vol.2, No.2, pp. 1-27, July-December 2011. (Publisher: IGI)

2. **W.Wilfred Godfrey, Shivashankar B. Nair**," Bio-inspired technique for servicing networked Robots", International Journal of Rapid Manufacturing, Vol. 2, No. 4, pp. 258-279, 2011. (Publisher: Inderscience).

International Conferences:

1. **W. Wilfred Godfrey, Shivashankar B. Nair**, *"An Immune System based Multi-Robot Mobile Agent Network"*, Proceedings of the 7th International Conference on Artificial Immune Systems (Lecture Notes in Computer Science, Springer Berlin, 5132/2008), 2008, Phuket, Thailand, pp. 424-433.

2. **W. Wilfred Godfrey, Shivashankar B. Nair**, *"Towards a Dynamic Emotional Model",* Proceedings of the IEEE International Symposium on Industrial Electronics,2009, Seoul, S. Korea, pp. 1932-1936.

3. **W. Wilfred Godfrey, Shivashankar B. Nair**, *"A Pheromone based Mobile Agent Migration Strategy for Robots*", Proceedings of the The 5th International ICST Conference on bio inspired models of network, information , and computing systems (Lecture Notes in Computer Science), , Boston, USA, 2010

4. **W. Wilfred Godfrey, Shivashankar B. Nair**, "*Mobile Agent Cloning for servicing Networked Robot*", Proceedings of the 13th International Conference on Principles and Practice of Multi Agent systems, Kolkata, India, 2010. (To appear in the Lecture Notes in Artificial Intelligence, Springer).

5. **W. Wilfred Godfrey, Shivashankar B. Nair**, "*Multi Mobile Agent Multi-Robot Network*", Proceedings of the International Conference on Emerging Trends in Robotic and Communication Technologies, Chennai, India, pp. 464-467.

6. **W. Wilfred Godfrey, Shivashankar B. Nair**, "A mobile agent cloning controller for servicing networked robots", Proceedings of the

International conference on Future Information Technologies, Singapore, 2011.

International Journals: (Communicated)

1. **W. Wilfred Godfrey, Shivashankar B. Nair**, ” On Stigmergically Controlling a Population of Heterogeneous Mobile Agents Using Cloning Resource”, International Journal of Computers, Communications & Control (Under Review).

ABSTRACT

The development of middleware for networked robotics offers challenges that are often quite different from those encountered in singular robots. It calls for the design and development of a software framework that can facilitate interactions amongst the distributed, interoperable, heterogeneous robotic entities that comprise the network and the simplification of complex robot control software systems which in turn can ease the associated application development process. Mobile agents form one of the virtual machine based paradigms that are ideally suited to the development of such middleware. A framework that exploits all features of mobile agents to form the basic middleware is still missing. The work described in this thesis aims at the realization of an *Artificial Being* (*AB*) comprising a network of mobile robots which are serviced by mobile agents. While the robots form the physical effectors of the *being* the mobile agents and the network facilitate the movement of information amongst them. The *being* is portrayed to use bio-inspired paradigms. Application scenarios where mobile agents carry services in the form of programs for networked robots have been depicted. Bio-inspired mechanisms - *PherCon* and *PherCon-C* have been proposed to provide for faster servicing by the mobile agents. Both simulation and real-time experiments were carried out and their results have been portrayed and discussed. An increase in the number of agents through cloning can enhance the performance of such networked systems. However, uncontrolled cloning can lead to excessive consumption of system resources including network bandwidth. The work reported in the thesis addresses this problem and proposes a novel *stigmergy* based method for control of populations of heterogeneous mobile agents.

As an offshoot of this research, a mobile agent framework for use in conjunction with an *Internet of Things* (IoT) has been proposed and its implementation described.

Chapter 1
INTRODUCTION

1.1 Networked Robotics

As per the IEEE technical committee on Networked Robots [1] , a networked robot is –

"… a robotic device connected to a communications network such as the Internet or LAN. The network could be wired or wireless, and based on any of a variety of protocols such as TCP, UDP, or 802.11".

Networked robots allow for multiple robots and entities to perform tasks that are well beyond the abilities of a single robot. In addition to coordination and cooperation, networked robots can exploit the inherent parallelism culminating in higher efficiency and improved fault-tolerance. Such a networked paradigm often brings together components that are complementary in nature and harnesses the potential benefits of each to achieve much more than their individual utilities.

Networked robots are broadly classified into two classes viz. *Teleoperated* and *Autonomous*. In the former, human users send data and receive feedback via a network while in the latter, the robots and the sensors send and receive information mutually over the network. Autonomous networked robots could thus include mobile sensor networks [2]. A networked robotic system may consist of robotic nodal elements which could be either static or mobile. The topology of this network can vary from a simple star over a wired LAN to an advanced multi-hop wireless mesh. Each of the individual nodes facilitates routing of data towards their respective destinations.

With robots becoming ubiquitous, the utility of networked scenarios has increased manifold. This is so because such robots are capable of working in

conjunction with other devices and appliances that also populate the network. Non-domestic applications such as those in healthcare [3], military [4], space research [5], undersea operations [6], domestic household [7], [8] domains, to name a few, benefit greatly from the use of networked robots. Typical application scenarios include *Search and Rescue* (SAR) missions in dangerous and inaccessible terrains, human assistance for the elderly or physically challenged and medical/surgical operations where the robots have a common goal

1.2 Behaviour based Robots

The work carried out by Rodney Brooks, pioneer researcher in the field of robotics deserves a special mention. He revolutionized the area of robotics by turning away from traditional notion of intelligent algorithms for robots by simplifying the parts and code and incorporating concepts from sociology and biology. Instead of imparting high-level human intelligence into the robot's brain, Brooks wanted to program only basic behaviors into the device so as to give it a way to experience sensory perception and allow it to learn from experience just the way human infants do [9], [10], [11], [12], [13], [14]. This is mainly because he believed that *the external world is its own best representation*, and the robot does not necessarily need to create an internal model of it in order to engage in *intelligent* behavior [15]. This paradigm shift in thinking led to the development of the subsumption architecture for robots [15]. He also hypothesized that robots should not have a single centralized processor that does everything but rather have different functional units concentrating on specific functions yet coordinated to give a general subsumption architecture. According to him

"…evolution builds a hodge-podge of capabilities that are adequate for the niche in which a creature survives. It is possible that with a few additional wiring changes in our 'normal' brains we would have newfound capabilities ...

Just as some of our modules have capabilities that are not present in chimpanzees, a supersapiens might have modules and capabilities that are not even latently present in us" [16].

Hence by emulating the modularized design of the brain instead of a single centralized brain, Brooks proposed his concept of Subsumption [17].

1.3 Multi-Robot Systems

All tasks cannot be always accomplished by a single robot system. Some distributed tasks require the use of multi-robot systems. Multi-robot systems are also advantageous in that their inherent parallelism results in faster problem solving and the presence of redundant robots results in fault tolerant systems. Most of the works in multi-robot systems can be categorized into key areas such as architectures, communication, swarm robotics, heterogeneity, task allocation and learning.

Here architectures and communication of multi-robot systems specify how the individual robot members are organized and how they interact. Multi-robot team architectures may be designed based on one of several existing philosophies which include - centralized, hierarchical, decentralized and hybrid. Centralized architectures coordinate the entire team from a single point of control. Nerd Herd [18] by Mataric is a representative architecture for centralized control of swarm robots using large number of homogeneous robots. ALLIANCE [19] is a related architecture developed by Parker for fault-tolerant task allocation in heterogeneous robot teams. Simmons et. al. have developed a hybrid architecture called the Distributed Robot Architecture (DIRA).

In a multi-robot scenario, knowledge coherency is an important requirement if these independent entities are required to cooperate. It is generally assumed that globally coherent and efficient solutions can be achieved through interaction of robots lacking complete global information. The three

most common techniques for robot interaction are stigmergy [20], passive action recognition [21] and explicit communication [19].

Swarm robots are inspired from biological societies such as ants, bees and birds. The major goal of this thesis is to develop similar evolvable behaviours in multi-robot teams. This is because biological societies are able to accomplish impressive group capabilities such as the ability of termites to build large complex mounds or the ability of ants to collectively carry large preys. Hence robotics researchers aim to reproduce these capabilities in robot societies. The work referred before by Mataric [18] involved about 20 physical robots performing aggregation, dispersion, and flocking. More recently, McLurkin [22] developed a swarm behavior software and demonstrated these behaviors on about 100 physical robots (called the SwarmBot robots), developed by iRobot.

Robot heterogeneity is defined in terms of differences in robot behavior, morphology, performance quality, size and cognition. This property of heterogeneity was first demonstrated in the work carried out by Parker [19] for the ALLIANCE architecture. Sukhatme et al [23] have demonstrated a helicopter robot co-operating with two ground robots. They have experimented with tasks involving marsupial-inspired payload deployment and recovery, cooperative localization and reconnaissance and surveillance tasks. Parker and Tang [24] developed ASyMTRe (Automated Synthesis of Multi-robot Task solutions through software Reconfiguration), which enables heterogeneous robots to share sensory resources to accomplish tasks that would be impossible without tightly coupled sensor sharing.

Task allocation is another area which defines how the mission of the robot team in terms of individual tasks are allocated to individual robots. Gekey and Mataric [25] define a taxonomy for task allocation that provides a way of distinguishing task allocation problems along these dimensions, which is referred to as the multi-robot task allocation (MRTA) taxonomy.

Multi-robot learning is the problem of a system learning new co-operative behaviors, or learning in the presence of other robots. The other robots in the environment, however, have their own goals and they too may learn in parallel. The types of applications that have been studied for multi-robot learning include multi-target observation [26], [27] air fleet control [28], predator–prey [29], [30], [31], box pushing [32], foraging [33] and multi-robot soccer [34], [35].

Many real-world applications also benefit from the use of multiple mobile robot systems. Example applications include container management in ports [36], extra planetary exploration [37], search and rescue [38], mineral mining, transportation, industrial and household maintenance, construction [39], hazardous waste cleanup [19], security [40], [41], agriculture, and warehouse management[42]. Multiple robot systems are also used in the domain of localization, mapping, and exploration.

RoboCup multi-robot soccer domain is a challenging problem for studying co-ordination and control in multi-robot systems [43]. This domain involves many challenging aspects of multi-robot control such as collaboration, robot control architectures, strategy acquisition, real-time reasoning and action, sensor fusion, dealing with adversarial environments, cognitive modeling and learning. A key aspect of this domain that is not present in the other multi-robot test domains is that robots must operate in adversarial environments. Later on the RoboCup competitions have added an additional search-and-rescue category to the competition [44], which has also become a significant area of research.

1.4 Middleware

In order for networked robots to work efficiently, they may require to cooperate, coordinate among themselves, share information with each other, utilize the services of complimentary devices or external hardware and also cooperate with human beings. These functionalities and operations make a networked robotic system perform in a complex and distributed manner. Such

systems are often heterogeneous which makes interoperability and management of the networked robots and devices an increasingly difficult task. This makes it mandatory to devise new, efficient and simple techniques to integrate the heterogeneous set of robots and devices so as to ease the development of real world applications for networked robots. Such techniques call for the formulation of a middleware that can act as a communication bridge between all devices populating the network

The term *Middleware* denotes a class of software technologies which creates or acts as an interconnection between the various hardware/software components and the applications. It may facilitate the management of inherent issues in a heterogeneous distributed system. Heterogeneity in distributed systems could manifest in the form of differences in the type of hardware, operating systems, networks or programming languages.

The use of a middleware for networked robots helps simplify complex robot control software systems and eases the associated application development process. In the domain of networked robotics, the middleware facilitates communication, interoperability, integration with other systems, collaborative operations among different robots, robot services and automatic resource discovery, among others [45].

1.5 Related Work and Challenges

Considerable research has gone into the development of such middleware to glue robotic components together in an efficient manner supporting concurrency-intensive operations, robustness and modularity. Middleware such Miro[46], Orca[47], RT-Middleware[48] and WURDE[49] offer modular design mechanisms and high level abstractions for component-based development. Reusability of components is offered by UPnP[50], Orca[47] and MARIE[51]. UPnP [50], RT-Middleware[48], and Distributed Humanoid middleware[52] offer support for resource utilization. PEIS Kernel[53], ORIN[54], AWARE[55]

and sensory DP middleware[56] feature integration with external systems (heterogeneity). Miro[46], Player/Stage[57] and MARIE[51] offer a flexible enhancement (adaptive nature) in terms of functionalities. These works have contributed to definite enhancements in the development of robotic systems. However, they do not offer holistic solutions to several issues. Mohamad et al. [58], [59] point out several features lacking in such middleware. Some of the pertinent features required of a middleware for networked robots and devices have been listed below:

1) Generic architecture for heterogeneous devices: A networked robotic system may involve robots and hosts of other inter connected devices. Communication and cooperation between them become paramount when one needs to make the best use of these entities populating the network. It is thus necessary for the middleware to provide an appropriate layer of abstraction that can hide the differences among the entities and act as a coherent interface.

2) Application development environment: Developing applications for networked robots is generally an arduous task. The associated middleware should thus provide an application development environment which allows for rapid development and testing.

3) Scope for self-adaptation: Adaptive programs and their deployment enable robots to adapt and reconfigure, either at hardware or software level, based on the environmental conditions and constraints.

4) Resource and Bandwidth management: The use of robots often involves computationally intensive processing such as that encountered in vision, mapping and navigation etc. In networked robotics, the robots may share these information and processing tasks over the network making resource and bandwidth management a vital task.

5) Scalability: Scalability is a key issue in networked robotics. More and different types of robots and devices may be required to be tethered to the system with ease.

6) Security: With the network acting as the main means of communication between the robots, securing the same is of vital importance lest a hacker gain control on the same.

7) Autonomy: Embedding autonomy into the middleware can allow for scheduling parallel and sequential tasks in a multi-robot networked environment. It can also facilitate the searching of free resources (robots) on the network that can execute a given task. Further such middleware can also express a robot's willingness to engage in the execution of a scheduled task on the network.

The development of a robust middleware for networked robots thus seems to require a paradigm shift. This can be achieved from insights gained by inspecting not only middleware design approaches used in networked robotics but also those used in allied areas. Since sensor networks are akin to networked robotics, it may be worthwhile to inspect the design approaches of middleware used therein. Few of such approaches have been discussed in the subsequent section.

1.6 Middleware Design Approaches

Middleware in both robotic systems or sensor networks plays a similar role. It provides a software infrastructure that glues together the network hardware, operating systems, network stacks and applications. Moreover middleware offers standardized system services to various applications, provides an environment supporting and coordinating multiple applications and also tenders mechanisms to achieve adaptive and efficient utilization of system resources. In robotics, especially in multi-robot systems, the choice of a middleware plays a crucial role in determining the cooperation approaches, communication methods, support for heterogeneity, scalability, portability and usability properties for that system.

In order to overcome the challenges mentioned in section 1.5, various solution approaches have been developed. Based on their individual

functions and their level of abstraction, these challenges can be classified into four types viz. Classic, Data-centric, Virtual machines and Adaptive [60]. Classic middleware approaches hide the complexity of network communication and data transfer. Data-centric middleware approaches provide an abstraction by considering the network as a database. Virtual-machine middleware approaches consider the network as a collection of code executers where programs/scripts are executed. The main focus of adaptive middleware approaches is to, as the name suggests, provide adaptability.

Virtual machine based middleware involves hosts which provide for the flexibility of containing a complete computing system in every node. In addition the other entities include smart and active messages and mobile agents. Smart ©[61]and Active messages are objects capable of processing on their own. These act as a medium for transfer of executable but fixed length data. Some of the common middleware approaches include distributed databases, agent based approach, virtual machine approach, application-oriented approach, message-oriented approach, component-based approach and macro-programming approach. In this thesis, we have opted to take-up an agent based middleware approach which uses mobile agents modeled on immune systems for multi-robot systems.

1.7 Mobile Agents

Mobile agents are software programs comprising code and data that can act autonomously and also can migrate carrying their execution state and data. Mobile agents provide the functionality just like other distributed computing paradigms. In addition to this they also feature mobility, cloning and autonomy. Furthermore, mobile agents just as their static counterparts have the ability to be embedded with behaviours and learning abilities that can make them exhibit intelligence. Mobility, cloning and payload carrying abilities of a mobile agent can facilitate a heterogeneous and faster flow of information across a

network. Scalability issues may also be tackled by injecting new agents, into the network on-the-fly, that carry relevant payloads so as to support new devices and robots. Thus intelligence, autonomy, scalability and asynchronous execution [62], make mobile agents prominent candidates for middleware for Networked Robotics.

This makes them ideal candidates for transferring and also embedding intelligence in distributed systems. These agents have been used in network management, monitoring, routing and load balancing and also in variety of other applications including information retrieval, robotics, etc. Mobile agents share all characteristics of their static counterparts but stand apart in their capability to migrate and clone autonomously. The concept of cloning featured in mobile agents helps increase their population and indirectly allows for parallel operation and information transfer across a network. It can also aid in the upward scalability of a distributed system. Having multiple copies of an agent can not only enable parallel execution but can also provide fault-tolerance, thus increasing both robustness and efficiency of the system.

The technique of creating mobile agents has evolved from Remote Procedure Calls. A mobile agent migrates from one node to another through data duplication which is carried out by saving its own states and transporting this saved state to the new host and then resuming execution from the saved state.

Mobile agents have been proposed to be used in conjunction with wireless sensor networks [63] to perform operations such as data integration [64], [65], [66] and tracking [67]. Sensor network middleware based on mobile agents have been found to be energy efficient [68], [69]. Fok et al. [70] report an implementation of the use of such agents on a real sensor network.

1.8 Mobile Agent for Networked Robots

The use of mobile agents as a glue in the world of networked robotics is still rare [71]. Cragg and Hu initially describe an architecture[72] that uses mobile agents in conjunction with teleoperated robots. Cragg and Hu[73],[74] elaborate on how the mobile agents augment robots to create an ALLIANCE like [19] distributed computing system. They have proposed and used a multi-robot system along with mobile agents to form patterns of robots. These robots can share knowledge and information to achieve complex tasks. However, their architecture does not highlight the actual need or utility behind the use of mobile agents nor is any functional enhancement, due to their use, portrayed. Kambayashi et al.[75],[76] have also used mobile agents on multi-robot systems to perform experiments including control of biped walking. The same author has also used a mobile agent approach for an interesting trolley collector problem in an airport[77]. Nevertheless most of these applications fail to exploit the inherent and more important capabilities of a mobile agent system viz. migration and on-demand cloning. Other features such as communication and adaptiveness have also remained unutilized.

A paradigm suited to networked robotics, that can exploit all features of mobile agents to form the basic middleware, still seems missing. This thesis describes bio-inspired paradigms that make use of most of the salient features of mobile agents and provide autonomous *search and service* mechanisms for networked robots. The sections that follow describe the major contributions of the work carried out and a chapter-wise summary of the thesis.

1.9 Contributions of the Thesis

The work reported in this thesis attempts to portray bio-inspired mechanisms for use in conjunction with Networked Robots and Mobile Agents. The major contributions made in this work are listed herein–

1. Formulation of a Mobile Agent based architecture for Networked Robotics using Immune System metaphors

This architecture was deemed necessary due to the absence of a holistic architectural framework which can support heterogeneous devices, be scalable and offer support for modularity and flexibility. The architecture is mainly inspired by the Biological Immune Systems (BIS) [78] whose metaphors justify the use of mobile agents within it. Agents are looked upon as *immune cells* while the robotic nodes form metaphors for the *organs* of a massively complex network or an *Artificial Being* (*AB*). The mobile agents (*immune cells*) migrate from one node to another providing the services (*antigen detection, neutralization* and *triggering* the *complement system*) they carry within as payload. The architecture also supports wireless robotic nodes that constitute the network.

2. Mechanisms for faster Service of Robots using Virtual Pheromones and Localized Cloning

Mobility is an inherent and significant feature of mobile agents that has not been well exploited. In the work described, mobile agents carry services in the form of programs for robotic tasks. Robots tethered to the network by novice programmers need not essentially possess programs for all the tasks they are capable of executing. This rids novices from the arduous task of writing code, considerably. It is thus essential for these agents to *search and service* such robots requesting for a service (or program) on the network, much like their immune system counterparts that *seek and destroy* pathogens within the body. The mechanisms proposed and evolved in the thesis are bio-inspired and make use of a near bi-directional search performed by both the robotic node and the mobile agent, eventually culminating in a faster service. While the robotic nodes diffuse virtual pheromones to signal mobile agents of a service requirement, the agents in turn use a conscientious and localized cloning approach to converge on the requester.

3. *Stigmergy* based Cloning Control of the Mobile Agent Population

Cloning is a function which is unique to a mobile agent. On-demand cloning reduces the service times of the robots and hence improves the efficiency of the system. However increased cloning can consume system resources including network bandwidth and lead to a reduction in system performance. An attempt to solve this issue by controlled cloning using *stigmergic* means of sensing the population of agents forms one of the major and novel contributions of this thesis. The proposed mechanism which is distributed in nature was found to be useful in controlling cloning while also ensuring that the available bandwidth is effectively used without compromising system performance.

4. Realization of Simulators

Though there are many commercial and open source mobile agent platforms, testing new algorithms and mechanisms using such platforms, is extremely tedious. This is so because performing measurements of time, resources available, etc. is difficult in real-time environments. It was thus felt that the realization of the relevant simulators will allow researchers to initially test their theories on mobile agents before going to the implementation phase. Mobile agent simulators provide an opportunity to model the behaviour of the mobile agents and test and verify them before porting them to real life applications. The thesis describes mobile agent simulators that were developed to test the proposed bio-inspired mechanisms as also some of the existing ones.

5. Real-world Implementations of the Service Mechanisms to cater to the *Internet of Things*

Real world implementations offer challenges which are often marred in the closed world of simulation. In order to check the validity and robustness of the proposed mechanisms they were implemented on a network of nodes using physical robots and sensors. In addition, as an off-shoot of the research, the proposed mechanisms were also found to be applicable to the new and evolving

field of the *Internet of Things*[79]. An application scenario comprising a sensor and a robot and mobile agents was tried and tested.

1.10 Outline of the Thesis

The thesis comprises nine chapters. A chapter wise organization of the thesis is given below.

Chapter 1 discusses the motivation for the work, followed by a survey and state of the art in the areas of networked robotics, associated middleware and the use of mobile agents for relevant applications. It also briefly describes the contributions made in this thesis.

Chapter 2 addresses a paradigm for middleware using an AIS based model. A more comprehensive version of the model using wireless networked robots is also portrayed.

Chapter 3 provides a description of various existing mobile agent migration mechanisms. It brings out the need to develop mechanisms that perform far better than the naïve and grossly inefficient round robin based migration strategy. The chapter describes the random and conscientious migration strategies and also popular patrolling algorithms viz. CLInG and EVAP. Gaber and Bakhouya's (G-B) algorithm for resource discovery in peer to peer systems is also explained in this chapter. The positive and negative aspects of these algorithms have also been explained.

Chapter 4 covers the motivation for exploring better and more efficient bio-inspired migration and *search and service* mechanisms for mobile agents. It then describes two of the proposed mechanisms for mobile agent migration using *Pheromone diffusion* and *Localized Cloning*. The chapter also compares the performances of these mechanisms with the ones described in the previous chapter.

Chapter 5 tries to bring out issues that arise out of excessive cloning. It highlights the need for population control and discusses the existing mechanisms

and the problems faced in their real implementations. A new concept of *cloning resource* has been proposed and used to achieve a *stigmergy* based population control.

Chapter 6 describes the need for a mobile agent simulator along with the one developed to test the existing and proposed mechanisms.

Chapter 7 explains a real implementation of a mobile agent based networked robotic system. The efficacy of the proposed mechanism for different network topologies has also been portrayed.

Chapter 8 introduces the concept of an Internet of Things and explains how the proposed mechanism can be used in conjunction with it. A simple implementation of the same is also described.

Chapter 9 highlights the conclusions arrived at, together with a brief summary of the contributions made and suggestions and scope for future work.

Chapter 2
AIS Paradigms for Mobile Agent based Networked Robotic Systems

As mentioned in the previous chapter there is a dire need for a paradigm shift in the middleware to be used in conjunction with Networked Robots. The use of Mobile Agents for such middleware was surveyed and the related advantages and deficiencies were also highlighted. This chapter throws more light on these issues and tries to put forth an architecture for the same using mobile agents that could in turn emulate Artificial Immune System (AIS) based paradigms. Before discussing the actual architecture, the chapter presents an overview of the Biological Immune System (BIS) and the related work on its artificial counterpart and its relevance to robotics.

2.1 Biological Immune Systems - A Brief Overview

The immune system serves to protect the body against foreign organisms. The innate immune system [80] is the first line of defense against a foreign attack. After an interim period, it initiates the adaptive immune system [81]. This largely comprises of the B- and T- type cells produced mostly in the bone marrow and the thymus. These cells are generated as precursor cells in the bone marrow and migrate to the thymus where they eventually mature before being released into the blood stream.

Bone marrow models and Thymus models of the immune system are used to explain the generation of a repertoire of immune cells and molecules. It is important that only those cells that can recognize foreign substances thrive while the others need to be removed from the system. Monitoring of cells that can perform self/non-self discrimination is performed in the thymus.

The T-cells are known to stimulate their B-counterparts when an antigen is detected. An antigen could be any substance that induces an immune response. This causes the B-cells to release a large number of antibodies with a single specificity that is capable of tackling this antigen. An antibody is an element of the immune system. The genetic material to produce an antibody molecule is stored in the component libraries [82]. Random selection of the genetic material from these libraries results in the production of an anti-body molecule [83]. Better binding of an antigen with the antibody receptors results in the promotion of new antibodies. This is the core of the adaptive immune response. The behaviour of the adaptive immune system to produce a large number of single specific antibodies over a huge range of antigens is explained by the Clonal Selection theory [78], [84].

A typical antibody molecule is made up of a variable region and a constant region as shown in Figure 2-2 [85]. The variable region is responsible for antigenic recognition and binding while the constant region is responsible for effector functions. A unique shape on the surface of an antigen which triggers an antibody response is known as the epitope [86]. The portion of antibody molecule that recognizes an epitope is called the paratope [87]. The process of refinement of the antibody specificity affecting the genes occurring in the B-cell is called somatic mutation [88]. Somatic hypermutation, is responsible for generating random genetic changes which result in diverse antibody patterns. Somatic hypermutation, which results in diverse antibody patterns, is used to increase the Antigen-antibody affinity. The strength of this affinity or the degree of match is determined by the intensity and surface area of binding between both the complementary binding sites viz. the cell receptor and the epitope. The variations in these binding areas determine the degree of match between them.

An infection causes an increase in the number of antigen cells. Learning in the immune system involves increasing the number of antigen specific antibody population in response to the number of antigens. The immune system stores high affinity antibody producing cells termed Memory cells. Such cells

proliferate faster in case of a repeated attack by the same antigen thereby providing a quicker secondary response. This provides for an inherent learning mechanism.

The clonal selection theory regards immune cells to be at rest and triggered only by foreign invasions. The Immune network theory [89] propounds that the immune system is dynamic and reacts even in the absence of a external stimulus. It views the immune system as a network of regulated antibodies that stimulate or suppress one another and are activated by an antigen attack. Communication between antibodies is performed using an idiotope which functions similar to an epitopes.

2.2 Related Work

Natural processes are characterized by their complex dynamics and interactions [90]. The complexities involved in these processes produce behaviours that are non-trivial and highly sophisticated. The BIS is a typical example of such a process. It provides high level biological processing capabilities and acts independently [78]. The properties of the biological immune system are highly appealing and have diverse applications in the world of information processing [91], [92], [93].They also have found their way into the domain of robotics. The BIS has features that are distributed, robust and easily adaptable. They are thus well suited for controlling robots. Besides, the recent surge in the use of robots has forced many a researcher to employ AIS based algorithms for robot control. Ishiguro et al. have applied AIS principles in robotics mainly for behaviour arbitration [94], [95], [96], [97] and for gait control of walking robots [98], [99], [100]. Dong and Kwee [101] describe an immune network theory based co-operative control of autonomous mobile robots, termed Distributed Autonomous Robotic System, in which the desired effect is produced as an emergent behaviour of the robots. Hart [102] has used the AIS based approach to create ‘growing up’ of rules for accomplishing

complex tasks. More AIS based approaches used in robotics are demonstrated in [103], [104]. An AIS based multiple autonomous mobile robot control is discussed in [101]. Sathyanath et al. [105], [106] implemented the problem of mine detection in a multi-robot scenario by modeling mine locations as antigens and robots as antibodies. The movement of the Robot towards the mine and away from the mine is controlled by stimulation and suppression. Chingtham and Nair [107] explore an application of adaptive learning mechanism for robots based on the natural Immune system. They have used innate and adaptive learning to arbitrate the behaviour of two robots.

Researchers [86] have also integrated Immune system using Farmer's model with behaviour based robotics to create a garbage-collecting robot with conflicting objectives. Similar work in this direction has been carried out by Vargas, Zuben et al. [94] etc. In the Farmer based systems [108], antigens represent environmental situations while antibodies represent competence modules and the variations in them govern the dynamics of these robots.

Most of the works reported do not seem to model the BIS in the true sense. Antibody concentrations are modeled as a mere variable which is manipulated in time while in the biological world this is the actual number of antibodies. Immune cell to cell interactions too seem grossly simulated. A true emulation of the BIS will greatly help in studying the distributed and asynchronous immune system paradigms. This however calls for an *artificial being* that emulates its biological counterpart whose body is to be defended by the AIS. A networked robotic system can be imagined to be such a *being* with the robots and their associated processors forming the actuators and organs respectively. In order to emulate a true BIS there is a need for independent entities that participate in the antigen *seek and destroy* process.

Mobile agents could easily play the part of immune cells if AIS paradigms were to be copied and used in conjunction with networked robots. In such a scenario, a robot requiring a service could be treated as an antigen which is serviced by mobile agents that carry with them the desired service. This thesis

describes an attempt at realizing an *artificial being* using a networked robotic system, mobile agents and bio-inspired paradigms. The mobile agents within the *being* act as the immune cells that migrate and provide services within the network.

2.3 The Artificial Being

This work provides motivations to build an *Artificial Being* (*AB*). An *AB* in this context is formed by the integration of

1) A Networked robotic system: This system forms the main physical body of the *being*. While the robots form the actuators that allow the system to move and locomote, the onboard processors act as various organs that perform and control various functions. The network acts as the plasma that facilitates the flow of information amongst these processors.

2) The Mobile Agents: These emulate the immune cells, allowing flow of information and control amongst the networked robots.

3) Control mechanisms: These mechanisms provide for regulating the flow of information amongst the entities that make up the being. It also help in governing the dynamics within the *being*.

The entities (organs, cells, and the plasma etc.) that comprise a biological being act in a distributed, parallel and asynchronous manner. These features can be emulated using a networked robotic system, mobile agents and the set of control mechanisms. The creation of such an *AB* would allow us to test various biologically inspired mechanisms in the real world.

In the subsequent sections attempts to model such a being using AIS metaphors has been described.

2.4 An $\mathcal{AB}$ based on AIS paradigm

A rudimentary $\mathcal{AB}$ architecture [109] described in Figure 2-1 constitutes the basis of the work described in this thesis. The system shown in this figure comprises computing nodes, mobile agents and robots.

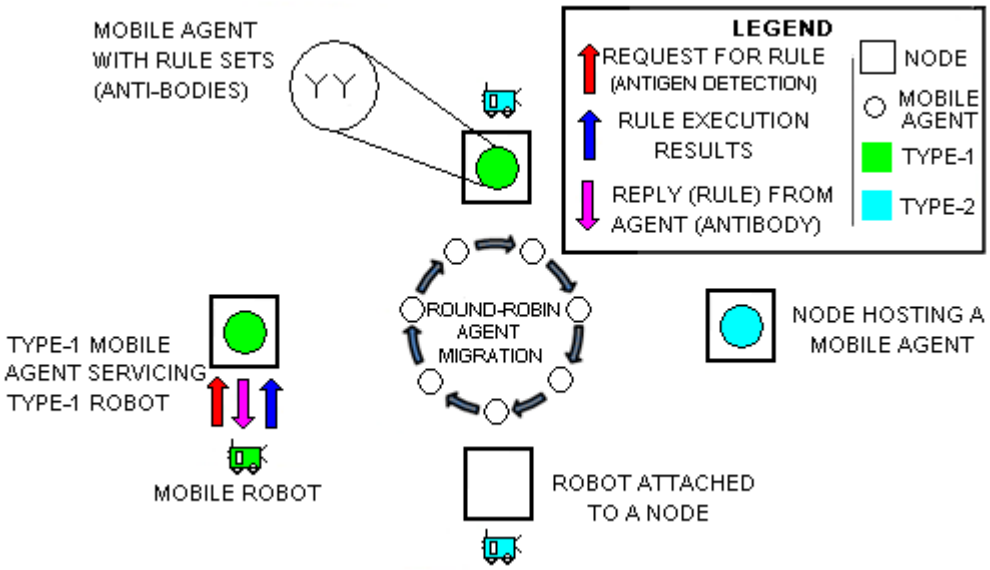


Fig. 2-1 The Immune system based Networked Robot System based on [109]

The *nodes* in Figure 2-1 act as placeholders for the mobile agents. A placeholder is more like a software platform that facilitates the hosting of mobile agents, their migration to other nodes, cloning and processing. Robots communicate with the agent through the node they are tethered to. Agents move in a round robin manner within the network and carry *rule-sets* required by the robots. The information that the robot receives from the agent is in the form of a set of rules termed the *rule-set*. The Robots depend on the agents to provide information when they encounter situations that they cannot tackle on their own.

A robot could be of a certain *type* defined based on its configuration, [110], [111] the task assigned and its operating environment. The *type* of an

agent is defined by the family of robots it can provide services to. Naturally an agent of one *type* cannot respond or support a robot of a different *type*.

The robots are autonomous but not totally intelligent in themselves. For instance a robot may encounter a situation it cannot comprehend such as an obstacle in front for which it does not have any recovery *rules* within itself. Under such conditions the robot contacts the associated node. If a mobile agent within that node has the requested *rule* or *rule-set* then the node passes it onto the robot. If the resident agent within the node cannot satisfy the robot's request, the latter waits till a new agent that can provide the required *rule-set* migrates and populates the node. By executing the rules within, it is hoped that the robot will be able to overcome an unforeseen situation. In its attempt to overcome the situation, the robot utilizes the *rule-set* and relays to the agent whether or not it was successful in doing so. The agent in its turn analyzes this information and modifies the rule-sets if required. The robot's behaviour is akin to that of self-preservation in biological beings. The term self-preservation denotes that instinct which helps an animal to survive fear and pain. In the current scenario the self-preservation behaviour of the $\mathcal{AB}$ is exhibited by its attempts to execute the *rules* so as to come out of the unforeseen situation that caused the *discomfort*. Discomfort or pain is expressed as a function of the various sensory perceptions of a robot. The sensory perceptions may be the internal state of the robot or the environmental conditions perceived by it.

The rudimentary architecture portrayed herein functions in a manner similar to that of the immune system in a biological being. The networked robotic system resembles the BIS both structurally and functionally.

An agent represents an immune cell that carries within itself the rule-set which forms the antibodies. The antigen in this case is the unforeseen situation encountered or pain experienced by the robot.

The rule-set has been portrayed as an antibody in Figure 2-2 as in [109]. A rule-set (antibody) is a collection of rules. As can be seen from Figure 2-2, each

variable limb of the Y-shaped antibody has two *paratopes* each of which represents a rule. Thus an antibody in this case carries a rule-set containing four rules R_1, R_2, R_3 and R_4. A rule applies to a robot only if the information contained in the constant region matches that of the robot. The constant region of the antibody specifies the *type* or family of robots which can use this rule. It could also indicate the type of *services* the robot is capable of. Imagine that a specific *type* of mobile robot capable of performing a *service* of picking an object, requires a rule. Such a rule can be supplied to it only by those antibodies that have the same *type* and *service* viz. *mobile robot with picking an object capability*, stored within its constant region. The antibody provides specificity to four different antigens or sensory conditions conforming to the four rules. When the sensory conditions reported by a robot are similar (high affinity) to those on the antibody then the rule is provided to the robot, subject to the condition that the robot's *type* and *service* requested conform to what is represented by the constant region of the antibody. The rule sets constituting the antibodies, are then used by the robot and graded based on their effectiveness in the real world. The robot thus actually executes the right hand side of the rule premise. Effectiveness is a measure of the amount of pain-reduction. Their effectiveness is relayed back to the agent as feedback. The agents in turn analyze this and purge those rules that are found to be less effective. This process of removal of the less effective rules is akin to *negative selection*. With more efficient rules circulating within the network, more robots benefit from them leading to a faster *secondary response*. The system as a whole thus endeavours to *learn* and refine the rules so that the *self* is preserved.

In the Figure 2-2 the antigen comprises the three sensor values, viz. Battery Sensor value, Left and Right Light Sensor values, reported by the robot. A rule consists of a range of sensory values for each sensor along with the corresponding code (program) for the action to be performed. The expected sensory ranges after the execution of an action are also appended to it. This helps in assessing the effectiveness of the rule after execution. The action is triggered only when the sensor values within the antigen fall within the respective sensor ranges specified in the rule.

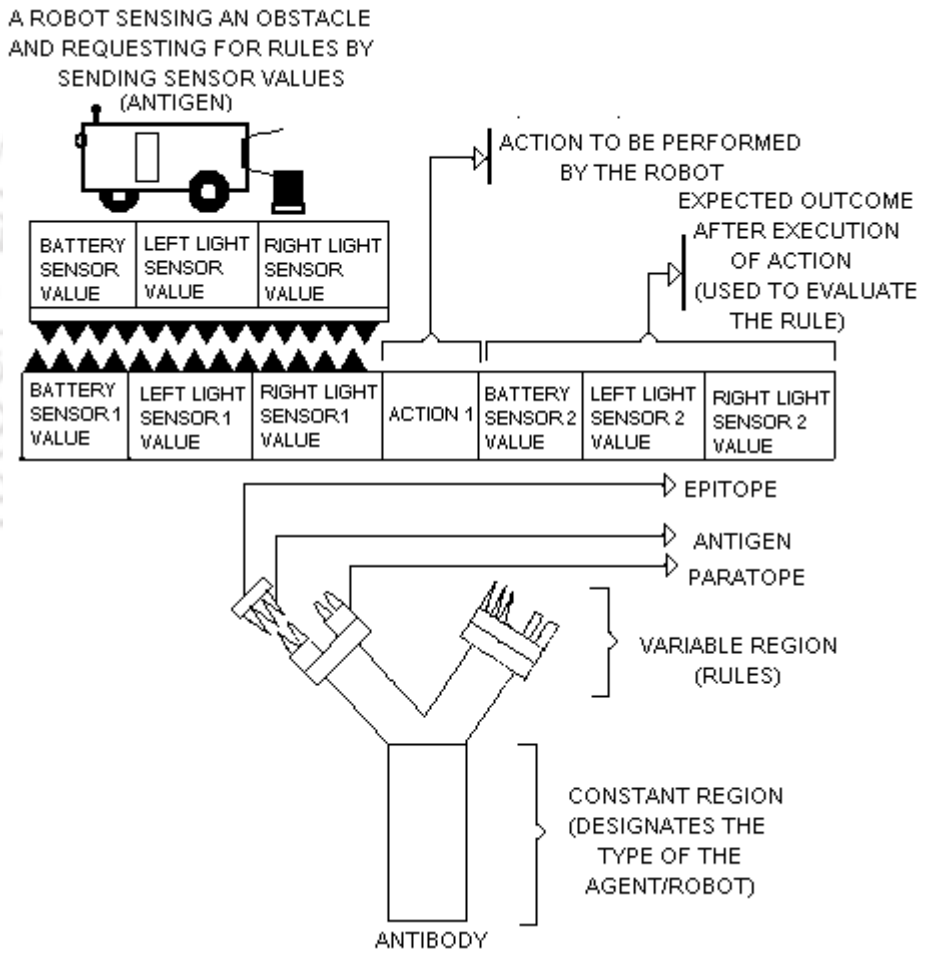


Fig. 2-2 Structure of the Y-shaped Antibody, Antibody-Antigen association and an example rule based on [109]

Thus the entire structure including the robots, the computing nodes and the mobile agents constitute an *Artificial Being* (*AB*) emulating AIS behaviours.

The *AB* is thus capable of movement through the effectors of the robots tethered to it while the internal control and learning mechanisms seem to be governed by AIS based functions.

2.5 Modified *AB* - The Mobile Agent based Networked Robotic System

The rudimentary *AB* architecture described in the previous section comprised a static network wherein the robots tethered to their respective nodes could be of the stationary type. However in a real application scenario one may require a network of mobile robots capable of communicating with each other wirelessly as in a MANET. The modified architecture presented in Figure 2-3 consists of multiple robots connected to each other via wireless links having a specific range. A robot communicates with another only if the latter is positioned within its communication range. Also shown in the figure is a depiction of a robotic node. Unlike its predecessor, this architecture suggests that these mobile robots themselves double as nodes with their on-board processing capabilities, thus being able to facilitate the hosting of static and mobile agents. Static agents act as mediators between the robot and the mobile agents and also perform some node related functions. The set of mobile robots thus constitutes a modified *AB* on the move – the robots forming its actuators while the on-board processors acting as the organs. The mobile agents migrate from one robot to another thus sharing both information and knowledge amongst the various organs of the *being*. They also serve to protect the organs (processors) and hence actuators (robots) from antigens (problems) presented by the environment. Protection or antigen elimination, in this context, refers to the programs that the mobile agents provide to the robots as and when they encounter a situation they are not

programmed to tackle. This modified *AB* can thus be looked upon as a Mobile Ad-hoc network of Robots (MANER) [112].

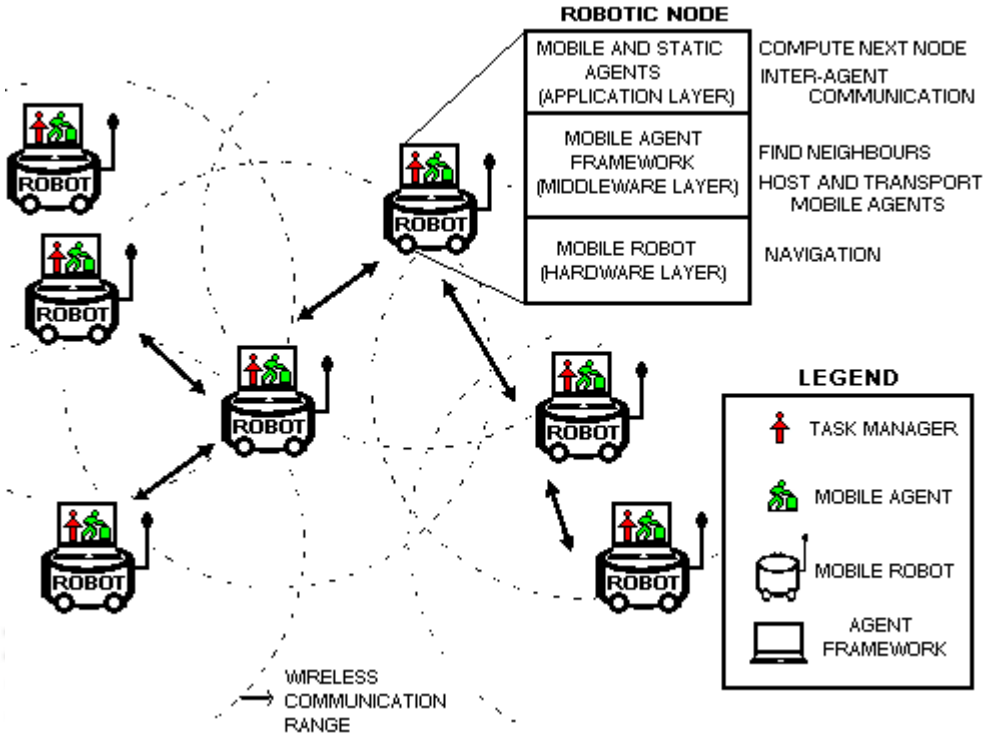


Fig. 2-3 The *AB* - A Mobile Agent based Networked Robotic System

Each robotic node constitutes three layers viz. –

a) The Hardware Layer: This forms the lowest layer and is responsible for the actual control and sensing of the robot. It constitutes all pertinent sensing and control circuitry and is capable of executing the associated programs.

b) Middleware Layer: This layer provides for a platform to host both the mobile and static agents. Using the lower hardware layer, the static agent keeps abreast of the changing topology of the neighbouring robotic nodes and provides all information necessary to the migrating agents. This neighbour information is

essential as the robotic nodes are mobile. Each node thus needs to sense its neighbours constantly and localize itself within the *being*.

c) Application Layer: This layer hosts and runs programs that govern the behaviours of the agents. It also caters to the migration strategies and intra-node computations and inter-node communications.

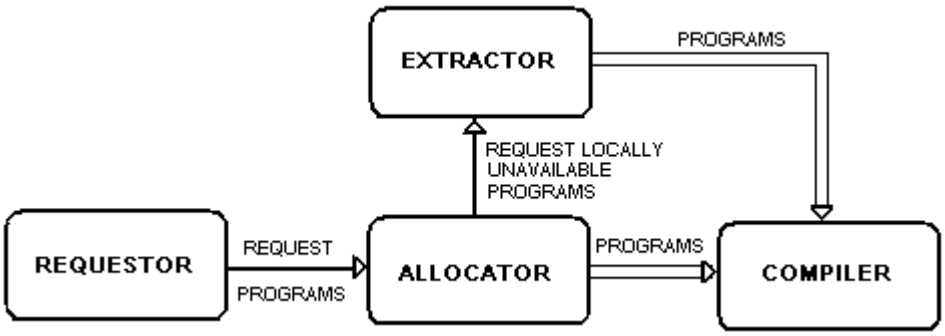


Fig. 2-4 Functional Schematic of the Task Manager

Every robot in this system is capable of hosting multiple mobile agents simultaneously. Mobile agents carry code for executable services and migrate from one robot to another and provide the required service to the requesting robot thus contributing to a flow of intelligence. Each node also hosts a set of four static agents that together form a *Task Manager* as shown in Figure 2-4 whose functions are given below.

1) Requestor: When a robot faces an unforeseen situation that it cannot handle it posts a request for the relevant programs to the task Allocator. It is envisaged that the execution of these programs will enable the robot to overcome its present situation.

2) Allocator: The Allocator passes all programs locally available within the node to the Compiler. A request is sent to the Extractor for those that are not available locally.

3) Extractor: The Extractor scans the payloads of all the incoming mobile agents for these programs. On finding the same it passes them onto the Compiler.

4) Compiler: After all the programs are locally available, the compiler compiles the programs and then passes the executable version onto the robot controller which in turn executes the same.

A robot that needs a service is termed as a *Robot Requesting Service* (RRS).

Each mobile agent carries within it a payload (program for a task) which on demand, is made available to the requesting robot (RRS). Many such mobile agents, each carrying payloads for different robotic problems/tasks, populate and continuously keep migrating within the network of robots. Mobile agents carry payloads that are basically programs which when executed by a robot facilitate the execution of a task allocated to a robot by a human being. For instance, if a robot is given a set of tasks $T = \{T_1, T_2, T_3, \ldots, T_n\}$ for which it does not have solutions (programs), then it has to wait till all the relevant mobile agents carrying the programs for all the n tasks visit it.

Once the programs for all these tasks are discovered, they are ordered in the desired sequence, compiled and executed by the robot.

Hooking robots to a network and programming them for the required tasks is non-trivial. If there were some method by which new robots could autonomously search and procure relevant programs/code for the tasks they need to perform, it would greatly reduce the burden on novice as also amateur programmers. This architecture describes the use of mobile agents that carry such code for tasks and service new and existing robots that populate a network. Code is either released as agent payloads by expert robot programmers or may be gathered from other robotic nodes by the agents during their sojourn within the network. Such a mechanism thus provides for information and intelligence to

be shared amongst both experienced programmers and novices as also other robots capable of learning.

2.6 Conclusions

Efforts to model an *Artificial Being* based on AIS paradigms have been portrayed in this chapter. A set of robotic nodes that can communicate wirelessly with one another and also host mobile agents forms the basic skeleton of the *being*. The architecture however remains incomplete for want of a proper middleware that can cater to the many needs of a robotic node. A few of the prominent challenges are listed below:

1) A generic and scalable framework for facilitating rapid deployment of robots and associated devices: The proposed architecture is scalable to the extent that a new mobile robot could join the set of existing robots even if they do not have the required code to execute tasks which they may be assigned. Such robots could post requests and eventually procure the necessary programs and execute the assigned tasks.

2) Efficient mechanisms for servicing robots: The rudimentary architecture used a round robin mechanism for mobile agent migration. Such a naïve mechanism would degrade the performance of the modified *AB* in terms of mobile agent based service. This is so because the robotic nodes are mobile making the network topology to dynamically change with time. New, faster and more robust mechanisms need to be formulated to cater to quicker RRS services.

3) Management of network resources: An increase in the number of mobile agents would mean a consequent decrease in RRS waiting times. However a large number of agents would eventually lead to cluttering of the network thereby decreasing the available bandwidth. This may affect agent migration times within the network and thus increase RRS waiting times. Thus a mechanism to control the mobile agent population and avoid such clutter needs to be investigated.

4) Realizing an application development environment for writing and deploying adaptive programs: The codes carried by the agents are currently static. In real AIS, the rule-sets or programs need to change based on their respective performances which are rated by the robots executing them. Programs carried as payloads by the mobile agents need to be adaptive so that they can cater to the ever changing needs of the robotic nodes that make up the *AB*.

In subsequent chapters we try to address and solve the second and third challenges using bio-inspired mechanisms while the last remains an area open to future work.

Chapter 3
Existing Mobile Agent Migration Mechanisms

As mentioned in the previous chapter one of the main challenges is to devise an efficient mobile agent migration mechanism. Migration mechanisms play a major role in guiding the agent quickly to the node which requires its service. Patrolling strategies such as those cited in [113] do not prove to be efficient migration mechanisms that can cater to servicing nodes within a network. Patrolling, by agents, attempts to visit and service a set of networked nodes in a uniform manner. The underlying assumption is that the requirements for agents to visit nodes arises in a systematic manner and thus are serviced likewise. In the real world a node *n* could generate a request immediately after a mobile agent has visited and migrated from it. If an agent is to use the conventional patrolling technique, then the node *n* would have to wait for its turn which would come only after the agent has visited all the other nodes. Mere patrolling thus may result in sub-optimal results. Moreover, general patrolling mechanisms support homogeneous mobile agents while real time situations and our experimental setup requires support for different types of mobile agents within the network. This chapter provides a description of some of the prominent mobile agent patrolling mechanisms and also points to their drawbacks.

In Chapter 4 we try to circumvent the problems faced in mere patrolling by augmenting it with mechanisms that will allow nodes like *N* to be serviced immediately.

In distributed applications based on mobile agents [114], the migration mechanism [115] used by such agents play a vital role in determining the

efficiency and resource utilization [116] of the underlying application. Consider a situation where a mobile agent needs to visit all the nodes of a network at regular intervals. A mobile agent using a predetermined migration itinerary fails to adjust itself to topological changes and accidental link or node failures. Hence a reactive mechanism [117] for mobile agent migration is preferable. In addition, to dynamic path adjustment, migration techniques should also guarantee reliability, stability and efficiency.

3.1 Random Migration

The random strategy has been used widely for mobile agent migration. This random walk technique has also been used in mobile agent based routing and network management applications [118], [119], [120], [121]. The agent herein selects one of its one-hop neighbours randomly and migrates to it. The mobile agent does not use any heuristic to decide on its migration strategy and is thus memory-less. This algorithm does not guarantee that the agent will find the node that requires its service in a stipulated time. The algorithm has been used here merely to act as a baseline in the comparative study of migration mechanisms made discussed in this thesis.

The next node position of a mobile agent *a* which is currently residing at node *n* is given by P^n_a.

The random method decides the course of movement of the mobile agent based on the equation (3.1) below –

$$P^n_a = \text{rand}(\psi_n) \qquad (3.1)$$

where ψ_n is the set of one-hop neighbours of node *n* and *rand* is a function that returns one of the neighbours randomly.

The random migration strategy is useful for applications where time and memory are limited. However in a more complex network topology, random walk proves to be naïve thereby increasing the chances of a node being starved of a visit by an agent.

3.2 Conscientious Migration

Conscientious mobile agent migration was first used by Nelson Minar et al. [122] for finding out the topology of a network using cooperative and non-cooperative mobile agents. In Almeida [123] proposes two more conscientious methods of mobile agent migration viz. Conscientious Reactive and Conscientious Cognitive.

In conscientious migration, , each mobile agent maintains a *Visited-Node* list. This list contains the identifiers of all the robots visited along with their respective time-stamps. An agent selects one of its one-hop neighbours which it has not visited so far and migrates towards it. If all nodes in the current one-hop neighbourhood have already been visited, it selects the node which has been least recently visited depending on the time-stamp information available in the *Visited-Node* list and migrates towards it. As described by Minar et al [122], this method results in a considerable decrease in time as compared to the random method.

A mobile agent *a* migrating based on the conscientious method decides its course of movement based on the equation (3.2) below –

$$P^{n}_{a} = j \qquad (3.2)$$

where $j \in \psi_n$ and $j \notin V_a$, V_a is the *Visited-Node* list and ψ_n is the set of one-hop neighbours of node *n*.

3.3 The EVAP based Migration

EVAP [124] is a swarm intelligence based patrolling mechanism which relies on the evaporation process of pheromones laid by the agents. This model only uses the evaporation process as an indicator of the time elapsed since the last visit to a node. The patrolling agent selects the next node to visit depending on the extent of evaporation among the neighbouring nodes.

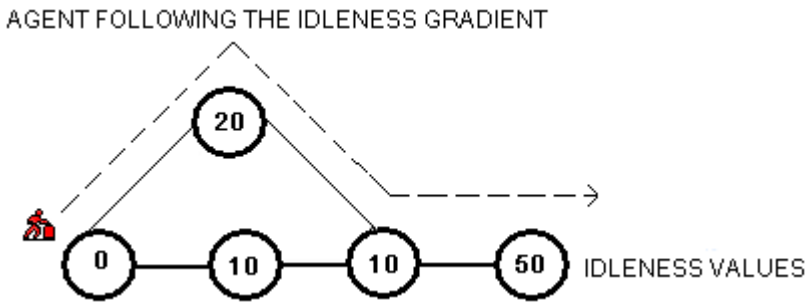


Fig. 3-1 The Depiction of EVAP [124] based migration of a mobile agent

Each node maintains a variable called the *Idleness*. Whenever an agent visits a node, the *Idleness* value is initialized to zero. This value is incremented for every time step till an agent revisits the node. *Idleness* is thus an indicator of the time which has elapsed after the last visit by an agent to that node. An agent resident at a node migrates to that one-hop neighbouring node which has the maximum *Idleness*, thus visiting the node that had not been visited for the longest period.

The next node position of a mobile agent *a* currently at node *n* is given by P^n_a using the equation (3.3),

$$P^n_a = j \quad / \quad r_j' = \max \psi'_n \qquad (3.3)$$

where $\psi'_n = \{r_i'\}$, i=1 to k, $i \neq n$, is the set of idleness values of *k* one-hop neighbours of *n*.

Figure 3-1 shows a depiction of the EVAP based mobile agent migration. The numbers inside each of the nodes denote the *Idleness* values. The mobile agent selects the nodes with the maximum *Idleness* values among its one-hop neighbours and migrates towards it. It thus follows the *Idleness* path - 0, 20, 10 and 50.

EVAP is an impressive mobile agent migration mechanism but is similar to the conscientious mobile agent migration mechanism if only one agent

populates the network. The two mechanisms differ in the fact that the information regarding the visits of the former is stored within the agent while that of the latter is maintained within each node individually.

EVAP is well suited for a homogeneous set of mobile agents. In heterogeneous agent scenarios, each node would have to maintain distinct *Idleness* values for each type of agent. Since conscientious and EVAP mechanisms operate upon the recently visited information and *Idleness* values of the one-hop neighbours respectively, there are times when some nodes need to wait for a longer period of time for a service. Chu et al. [124] have improved on EVAP and proposed another patrolling mechanism named CLInG which is described in the next section.

3.4 CLInG based Migration

In CLInG [124] each node maintains two values viz. an *Idleness* value and a *Propagated Idleness* value. The *Idleness* value is maintained in a similar way as in EVAP and is indicative of how recently a node has been visited by an agent. The *Propagated Idleness* provides an *Idleness* gradient towards the least recently visited node in the vicinity. Every node computes its *Idleness* by incrementing an internal counter at every step. This counter is reset when the agent visits it. The *Propagated Idleness* value computed for each node is equal to the maximum *Idleness* value of all the one-hop neighbours of the node. This value is propagated through the neighbouring nodes to all the nodes in the network. The agent always migrates to that one-hop neighbour node which has the maximum *Propagated Idleness.*

The next node position of a mobile agent *a* currently at node *n* is given by P^n_a using the equation (3.4)

$$P^n_a = j \mid r''_j = \max \psi''_n \qquad (3.4)$$

where $\psi''_n=\{r''_i\}$, i=1 to k and $i \neq n$ is the set of propagated idleness values of *k* one-hop neighbours of *n*.

The propagated idleness r''_n, of node n is given by the equation (3.5)

$$r''_n = \max\{ r'_n, \max \psi''_n\} \quad (3.5)$$

where r'_n is the idleness value of node n.

Figure 3-2 shows a depiction of CLInG based mobile agent migration. The numbers inside each of the nodes denote their *Idleness* values while the numbers outside denote the *Propagated Idleness* values. The maximum of the *Idleness* values is propagated across the neighbouring nodes. The agent in this case follows the *Propagated Idleness* gradient along 20, 30, 40 and 50.

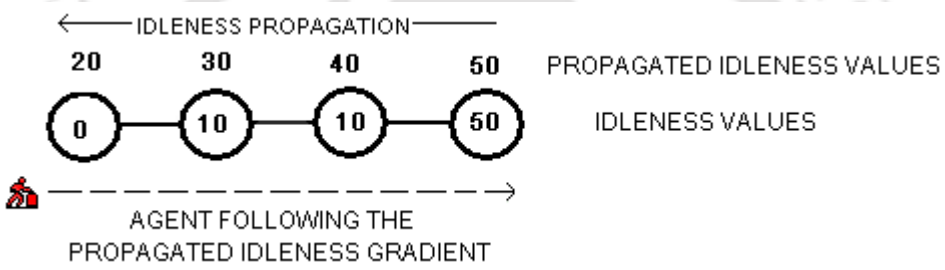


Fig. 3-2 The Depiction of CLInG **[124]** based migration of a mobile agent

While this mechanism successfully overcomes the problem of most idle nodes concealed from the agent visitations it uses two variables as opposed to one in EVAP which needs to be continuously exchanged across the nodes in the network failing which the mobile agent may be misguided and forced to follow other paths. This exchange of idleness values across the network results in a large number of inter-node communications. Further, this patrolling mechanism assumes the use of homogeneous agents. When heterogeneous agents are used, this mechanism would require a consequent increase in the number of variables used as also in the number of inter-node communications. This can drastically slowdown the system. CLInG assumes a synchronous exchange of *Propagated Idleness* values across all the nodes in the network. In a real network, each node behaves asynchronously. This may contribute to delays in procuring *idleness*

values by a node forcing the agent within to use the older values of *idleness* and *propagated idleness* to find the next node. This asynchronous nature can thus force the agent to follow arbitrary paths within the network making the practical implementation of CLInG a futile exercise.

3.5 The Gaber-Bakhouya's Random walk and Cloning based Approach (G-B Algorithm)

The G-B algorithm [125] has been used for discovering resources in a peer to peer network. Resources may be distributed within the nodes of a network. In a peer to peer network, nodes may need to find the locations of the nodes hosting the resources they require. This could be realized using centralized servers [126], [127] hosting information about such nodes or by sophisticated indexing mechanisms [128]. The approach described by Gaber and Bakhouya [63] uses mobile agents for discovering resources in a network. The main goal for the mobile agents is to migrate and locate the required resources within a network for the node that spawned them. Initially the node that requires the resource location spawns a mobile *Request Agent*. As this agent migrates from one node to another it also keeps track of the information on the resources available within the nodes. When it arrives at a node connected to several others, it creates clones and sends one each along these paths and takes the left-over path. These clones keep track of the nodes they visit and the resources they discover as they migrate just as the spawned agent. They also clone as and when they encounter multiple paths. Eventually one of the clones reaches all the required resources through the shortest path connecting all the resource locations and communicates this information to the node that spawned the *Request Agent*. The number of clones increases exponentially as time progresses.

This mechanism incurs a much larger cost in terms of the energy expended during the search process. It may also be noted that an increase in the

number of agents can drastically affect the performance of the network, an aspect that has been studied later in Chapter 5 of this thesis.

3.6 Conclusions

While mobile agent based patrolling mechanisms may seem to be a viable alternative to realize the servicing of nodes, it cannot efficiently cater to randomly generated service requests. In the architecture of the *AB* proposed in Figure 2-3, the robotic nodes could generate requests for a service randomly based on need. Thus if a node generates a request for a service just after an agent has migrated from it, the patrolling mechanisms described in this chapter will force it to wait till the agent has visited all others. In the practical world, it would be best if the agent somehow knows this and retraces its step to the requestor node and then service it before continuing its journey in the network. The mobile agent migration mechanisms discussed in this chapter thus do not seem to be ideally suited for the architecture described earlier. A pro-active mechanism on part of both the node and the agent so that they mutually discover each other at a faster rate is discussed in the next chapter.

Chapter 4
Pheromone based Mobile Agent Migration Mechanisms

Mobile agent migration mechanisms play a vital role in servicing networked robots. In the envisaged architecture of an *AB* using AIS paradigms, the mobile agents constituting the immune cells need to reach the robotic nodes as soon as the latter generates a request (detection of an antigen).With the robotic nodes generating requests for a service in a random and asynchronous manner, mere patrolling of the network by the mobile agents cannot assure a prompt service. This chapter proposes a bio-inspired mobile agent migration mechanism nicknamed *PherCon* which uses the concept of *pheromone diffusion* on part of the robotic node to attract mobile agents carrying the requested service towards it. An inherent disadvantage of this mechanism is also pointed out and rectified using *localized cloning* by the concerned agents. The chapter also provides results obtained by simulating the same using a custom simulator and compares them with those of the other migration mechanisms described in the previous chapter.

4.1 Introduction and Related Work

Natural pheromones were discovered way back in 1959 by Karlson [108]. They proposed the term pheromone to describe chemical signals from organisms of similar species that elicit innate behaviours. Their use as a mode of communication and interaction in the computational world was pointed out only in the early '90s when Deneubourg, Aron, Goss and Duerinck [129] saw the possibility of creating pheromone based agents. Pheromones have been modeled

as physical entities or paths in the environment or as synthetic messages. Kitamura et al. [130] have used pheromone communication for understanding swarm intelligence in terms of complex and adaptive population behaviours for robots. They describe robots that can lay pheromones on a desktop using coloured pens mounted on them. Susnea et al. [131] describe virtual pheromone based control of mobile robots where robots relay their position information to a remote program which in turn uses a pheromone based algorithm to compute the alignment to be effected by the robot. Parunak et al. [132] discuss pheromone based computations by a group of small sensor bots thrown onto a hostile area to check out the maneuverability of vehicles in the region.

Research on various techniques for coordinating a large number of robots has been conducted by Payton et al. [133]. Anies and Russell [134] have discussed pheromone communication between robots using chemical signals. They use a physical chemical for the task of assessing the quorum size of a group of robots equipped with gas sensors. The communication is bi-directional and their investigations have revealed the potential advantages and drawbacks of implementing physical pheromone signaling between robots.

Pheromone based algorithms have also been used for routing in MANETS. Roth and Wicker [135] describe TERMITE, a distributed routing algorithm for mobile wireless ad-hoc networks inspired by the hill-building behaviour of termites. It is designed using a swarm intelligence framework [136] for achieving better adaptivity, low control overhead and low per-node computation. Contributions of both Gunes and Spaniol [137] and Roth and Wicker [135], [138] have shown enhanced performance over traditional approaches used for MANET routing. Gunes and Spaniol use an ant-colony based algorithm for routing in MANETs while Roth and Wicker use *stigmergy* to reduce the amount of control traffic.

In the succeeding sections a pheromone based bi-directional search strategy that can drastically reduce the time taken by the mobile agents in reaching robots that require a service, is described.

4.2 *PherCon*: A Mechanism for *Search and Service* Using Conscientious Mobile Agents and Pheromone diffusing Robots

Mobile agents normally use the conscientious method for migration within the network. In this approach they keep track of the nodes already visited and avoid migration onto such nodes. This section presents an enhanced migration strategy nicknamed *PherCon*, wherein both the mobile agent and the robotic node requiring the former's services, play a pro-active role in discovering one other. The mobile agent uses the conscientious migration strategy while a robotic node requiring its service, referred to as the Robot Requiring Service (RRS), diffuses pheromones to attract the former. An illustration of *AB* which uses *PherCon*, is depicted in Figure 4-1. The mobile agents populating the network carry the information similar to that of the Y-shaped antibodies mentioned in Chapter 2, as their payloads. This includes the name of the service and the program to effect the service. The name of the service resembles the information in the constant region of the Y-shaped antibody while its associated program is analogous to the rule-set. Whenever an robotic node (RRS) within the network requires a program or a service, it diffuses pheromones with the highest concentration possible, along the paths connecting its neighbouring nodes. The neighbours in turn lay the same beyond, with lesser concentrations forming a network of pheromones in the vicinity of the RRS. Thus, if the mobile agent is anywhere in the vicinity or within the neighbours of this RRS it will be guided, from thereon by the pheromone concentration gradient, along the shortest path towards the RRS. If pheromones are diffused many hops away from an RRS, it will proportionately decrease the time required for convergence of the agent to the RRS. As can be seen from Figure 4-1, the RRS diffuses pheromones (red dotted lines) onto its immediate one-hop neighbours R_1, R_2, R_3 and R_4 which in turn re-diffuse them to their respective neighbours viz. R_5, R_6, R_7 and R_8 at lesser pheromone concentrations. An agent resident at R_{10} continues its migration using the conscientious approach to eventually reach R_7 via R_9 to find a pheromone

trail. If the information within this pheromone matches that carried by the agent, then it starts to climb the pheromone concentration gradient to finally reach the RRS via the shortest path henceforth. This matching of the information within the pheromone with that contained by the agent is akin to matching the *type* of the robot and the *type* stored within the constant region of the Y- shaped antibody as discussed in section 2.4. The pheromones diffused by the RRS and the conscientious migration on part of the agents carrying the requested services constitute a bi-directional search. However it is unlike the normal bi-directional search which has no guidance mechanism when it nears the destination. The RRS seems to extend and lay a path for the mobile agent while the latter searches concurrently and conscientiously for such paths thereby facilitating faster service of the former. While the mobile agent uses the conscientious approach till it senses the pheromone, the RRS spreads out a pheromone network around it to trap and guide the agent towards it.

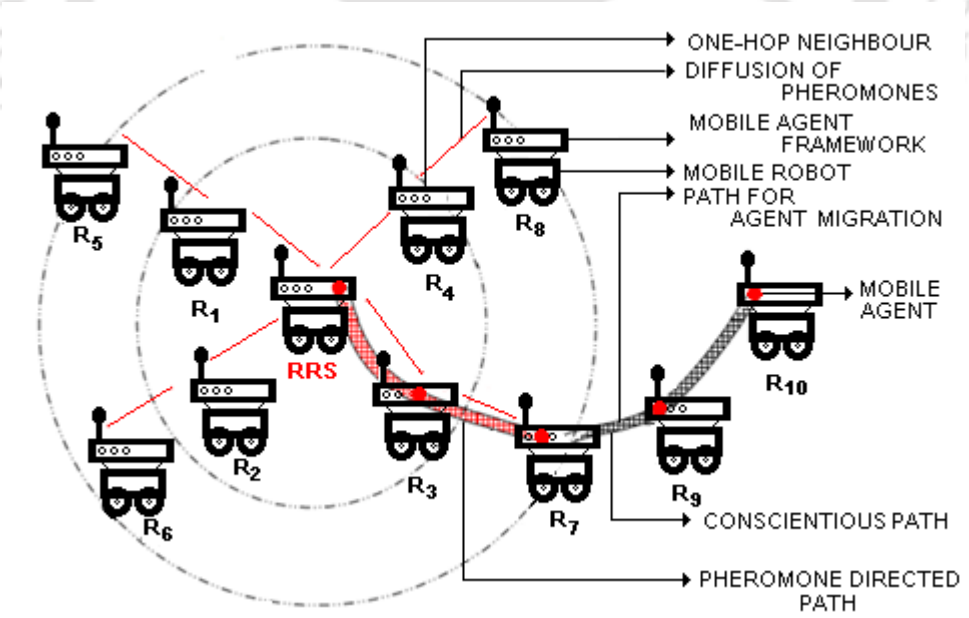


Fig. 4-1 Illustration of the Mobile Agent based Networked Robot System with Pheromoning Robots and Mobile Agents [78]

4.2.1 Pheromones in *PherCon*

When a robot requires a service, it diffuses pheromones onto its immediate neighbours. These pheromones inherently carry the following information.

1) RRS ID: Pheromones carry the robot id of the RRS that originally initiated its diffusion within a field called RRS ID. This RRS ID stored within the pheromone ensures that the mobile agent does not perform a *redundant service,* described later.

2) Service Information: This information allows a mobile agent to identify the type of service required by a robot. It enables the agent to decide whether or not to follow the pheromone gradient to the RRS that diffused it. A service, as mentioned could be a program or code required by the RRS to execute a task.

3) Pheromone Concentration: The RRS lays pheromones with the highest concentration onto its neighbours while the neighbours in turn diffuse it to their neighbours at a lesser concentration. At a robotic node, the mobile agent is thus presented with a pheromone concentration gradient that eventually leads it towards the RRS via the shortest path. The robotic nodes continue the diffusion of pheromones till finally its concentration becomes zero. The pheromone concentration thus decides the extent to which the diffusion occurs in the network.

4) Time-out Information: Natural pheromones tend to evaporate so that the less used paths are discarded while others are reinforced [108]. *Time-out* information is akin to evaporation of the virtual pheromones used herein. Each pheromone is provided with a *Time-out* by the RRS as and when they are diffused. The value of *Time-out* is calculated proportional to the pheromone concentration. Unlike the real evaporation modeled in ant colonies, evaporation of pheromones in this case is facilitated by the removal of pheromones whose life-times have expired. The *Time-out* can be decided by the RRS based on its

urgency to retrieve information from the mobile agent. It thus signifies the time the RRS can afford to wait for a service. In time-critical networked robot scenarios, a robot may wait till the *Time-out* reduces to zero and if not serviced within this period, opt for a change in strategy. It may otherwise diffuse pheromones having higher initial concentrations to make them penetrate deeper into the network.

5) Neighbour ID: Mobile agents move through the neighbouring robotic nodes to reach the RRS. The Neighbour ID information identifies the one-hop neighbour which caused the diffusion onto the current node. The neighbour ID encoded within the pheromone helps the agent to identify this robotic node.

If the mobile agent finds itself in a non-pheromoned node, it uses the conscientious method of migration.

4.2.2 Pheromone Management

While pheromone diffusion entails robotic node to node communication, its management and maintenance is performed by the node that receives the pheromone. An RRS proactively attracts the mobile agent that provides the relevant service using the pheromones. The mobile agents in turn also move in a proactive manner towards the robotic nodes along the pheromone gradient. Both the robots and the mobile agents actively involve themselves in searching one another and represent a partial bi-directional parallel search. Therefore the robot side and mobile agent side functions are discussed separately.

4.2.2.1 Robotic node side functions

The pheromone diffusion and evaporation mechanisms are based on the work reported by Godfrey and Nair [139]. As described earlier, initially the RRS diffuses pheromones onto its one-hop neighbourhood and subsequent diffusions are carried out by the neighbouring robots. Pheromone evaporation is carried out by the robotic nodes that have received the pheromones as also by the RRS that emanated it.

Pheromone diffusion

The RRS diffuses a pheromone trail onto its immediate one-hop neighbours with the maximum concentration (C_{max}) so as to indicate a path from its neighbours to itself. The neighbouring robotic nodes which receive these pheromones, in their turn diffuse them onto their one-hop neighbours but with a reduced concentration thereby forming a pheromone concentration gradient towards the RRS via intermediate robotic nodes. The concentration gradient of the successive diffusions is given by equation (4.1).

$$\Delta C = 100/d \qquad (4.1)$$

where d is the spanning length. The diffused pheromone concentration P_{pc} at the n^{th} hop from the RRS is given by equation (4.2).

$$P_{pc}(n) = C_{max} \quad \text{for } n=1$$
$$= P_{pc}(n-1) - \Delta C \quad \text{for } n>1 \qquad (4.2)$$

where n is the number of hops from the RRS.

Spanning and laying pheromones to a greater depth within the network can result in faster convergence of the mobile agent onto the RRS. However increasing this depth can have a direct impact on the overall energy and resources consumed by the robot nodes within the network. A compromise, between the depth, the energy and resources consumed and the urgency of the request for realizing a service, needs to be made based on the application.

Pheromone Time-out

As mentioned earlier each of the pheromones has a *Time-out* (T) parameter. This is decremented by an amount (ΔT) for every time or step-count. As soon as the *Time-out* variable reaches zero, the pheromone trail is removed from the robotic node resulting in the breaking of the link in the gradient towards the RRS. Thus *Time-out* helps in removing obsolete pheromones from the system.

The *Time-out* gradient is proportional to the concentration gradient and is given by equation (4.3).

$$\Delta T = T_{max} (\Delta C/100)$$
$$= T_{max}/d \qquad (4.3)$$

where T_{max} is the maximum value of *Time-out*. While laying the pheromones, their *Time-out*s are reduced over the hops using equation (4.4).

$$P_{t\text{-}o}(n) = T_{max} \text{ for } n=1$$
$$= P_{t\text{-}o}(n-1) - \Delta T \quad \text{for } n>1 \qquad (4.4)$$

where *n* is the number of hops from the RRS.

Higher the *Time-out*s of the pheromones, the longer the trails last within the network and more the probability of attracting the right agent. Conferring higher *Time-out*s to the pheromones however has its own overheads. An agent services the RRS and waits within it for the pheromones that guided it to die, to avoid a redundant service. If the *Time-out*s were high, this waiting period within the RRS would also be proportionately high. This can delay the servicing of other RRSs by this agent. A solution to the problem of such a redundant service is dealt with in a later section. Conferring lower *Time-out*s to the pheromones will force the RRS to re-diffuse them more frequently, thereby consuming more resources and energy. A judicious compromise needs to be made in deciding the value of this *Time-out*.

Pheromone Re-laying Interval

It may happen that the pheromones expire (i.e. their *Time-out*s becomes zero) even before the RRS is serviced. Under such circumstances the RRS reinitiates the pheromone diffusion process. The pheromone diffusion interval δ, between two such successive diffusion initiations by an RRS, is chosen such that it is greater than the maximum *Time-out* of the pheromones. This ensures that re-diffusion is performed only after all pheromones have evaporated. Thus, δ is equal to a constant K which is greater than T_{max}. However if the value of δ is

made much larger than T_{max}, the waiting times of the unsatisfied RRSs could become higher. δ should thus be selected to be marginally higher than the maximum *Time-out* of the pheromone.

4.2.2.2 Mobile agent side functions

The *PherCon* agent migration mechanism described herein uses pheromones to decide which path an agent is to follow towards the RRS. If pheromones are not detected then the migration decision is based on the conscientious method. This strategy is chosen only when the agent discovers a pheromone trail whose service information matches the service which it carries within its payload. When pheromones are detected the mobile agent selects the pheromone containing the highest concentration and migrates towards that robotic node indicated by the *Neighbour ID* of that pheromone.

Next-Node Calculation

The next node of an agent *A* located in a robotic node *n* carrying a service *x* as its payload, $P^n{}_{a_x}$ in the next time *time-step*, t+1 is given by equation (4.5).

$$P^n{}_{a_x}(t+1) = \pi_i^B(t) \tag{4.5}$$

where $x = \pi_i^S(t)$ and $\pi_i^C(t) = \max_{j=1}^{|\pi^S(t)|} \pi_i^C(t)$ and $\pi_i^T(t) > 0$

$\pi_i^S(t)$ is the Service information contained in the i^{th} pheromone at the node *n* at time *t*

$\pi_i^C(t)$ is the Concentration contained in the i^{th} pheromone at the node *n* at time *t*

$\pi_i^T(t)$ is the *Time-out* contained in the i^{th} pheromone at the node *n* at time *t*

$\pi_i^B(t)$ is the Neighbour ID contained in the i^{th} pheromone at the node *n* at time *t*

If a robotic node *n* does not have pheromones at time *t* then, the agent migration is based on the conscientious approach using equation (3.2).

Avoiding Redundant Services

It may happen that after servicing an RRS, the agent may migrate to the next node before the pheromones that guided it to this RRS have expired. In such a case, the agent may be forced to loop back to this already serviced RRS. In order to avoid such a redundant service the mobile agent is made to *hibernate* within the RRS till the pheromones pertaining to that request have expired. After the associated pheromones of the current service expire, the agent continues its sojourn in the network using the conscientious strategy. This hibernation incurs an unfortunate delay in the process of servicing other RRSs by this agent.

If the initial concentration and hence *Time-out* of the pheromones were to be made high so as to increase greater penetration into the network and realize a possible faster service, then this hibernation period within the RRS could also be proportionately high. Conferring lower life times to the pheromones will force the RRS to re-diffuse them more frequently, thereby consuming more resources and energy.

A better method to avoid this loop-back to an RRS and hence the redundant service would be to make the mobile agent carry the *robotic node ID* of the just serviced RRS and the *Time-out* of its associated pheromone having the highest concentration. In this method, the mobile agent is made to suppress its inherent pheromone tracking behaviour for this specific RRS for a period of time equal to the *Time-out* it carries, thus preventing a redundant service.

4.3 Mobile Agent Dilemma in PherCon

When several RRSs populate the network, it is highly probable that a certain robotic node may have pheromones that subsequently connect it to one or

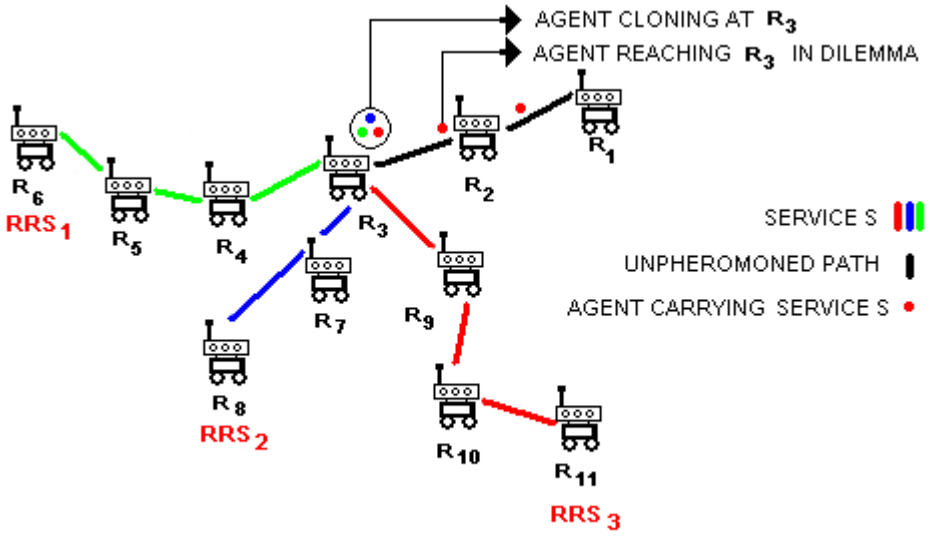


Fig. 4-2 Illustration of the scenario depicting a Mobile Agent in Dilemma

more other RRSs requesting the same service. When a mobile agent capable of servicing these RRSs reaches such a node onto which pheromones from different RRSs requesting the *same* service have been diffused, it finds itself in a dilemma as to which RRS it should service first. A situation of this type is portrayed in Figure 4-2. The robotic nodes R_6, R_8 and R_{11} which form the RRS_1, RRS_2 and RRS_3 respectively have been shown to be diffusing pheromones for the same service (S). These RRSs need not necessarily be triggered at the same time. In the Figure 4-2 it can be seen that by the time the pheromones reach node R_3, an agent carrying the requested service, S, also arrives at this node. The agent thus detects pheromones from the three different RRSs depicted in three different colours. The agent could choose an RRS using one of the two approaches.

1) *Random Approach*

The agent could randomly select one of the RRSs to track and follow its associated pheromone trail. In such a case, there would be no streamlined way of finding the other RRSs requiring its service. It thus may switch back to the

conscientious strategy of discovering RRSs without any memory of the previous dilemma.

2) Conscientious Approach

The agent could choose one of the RRS randomly and remember the path back to this node. This will allow it to retrace its path back to this node (such as R_3) after the service is effected and then choose the next RRS to track. This naturally means the other RRSs would have to wait till this agent finds its way to each one of them sequentially. It also entails overheads in remembering and book-keeping the retrace paths.

Both options thus have their own disadvantages and do not contribute much to the problem at hand. In this work a localized cloning strategy to solve this issue and thus enhance the performance of *PherCon* is proposed.

4.4 Enhancing *PherCon* using Localized Cloning – *PherCon-C*

The pheromoned area with many RRSs is analogous to an area reporting high antigenic activity and would thus call for more number of antibodies (relevant mobile agents) to cope up with. This has been achieved by using localized cloning. When the mobile agent detects multiple serviceable pheromone paths to different RRSs which it can service, it clones and sends each clone along the other paths and opts to go along another. This leads to a parallel service of all the RRSs thereby improving performance. Localized cloning can be beneficial in the sense that while more RRSs can be serviced in parallel, it greatly reduces the repeated diffusion of the pheromone by an RRS and hence the overall energy consumed. *PherCon* was thus augmented with localized cloning to result in an improved version which has been referred to as *PherCon-C*. The clones generated have been referred to as *local-clones*.

4.4.1 Inherent Information within a Local-Clone

Though localized cloning can easily lead to a better performance, there are several issues to be handled. Each clone generated is embedded with the

following information:

Local-Clone Sterilization

If clones contained the same logic as their parent, they would further clone when faced with a similar situation. This could lead to an avalanche of clones that would populate the network and finally lead to choking of the bandwidth especially in the sensitive densely pheromoned area. The clones generated at such dilemma points in the network are rendered *sterile*, in the sense that their cloning logic is inhibited at the time when they are created. This ensures that they do not clone like their parent agents thus ensuring that bandwidth is available.

Local-Clone Life-time

If localized cloning were to occur many times, a large number of clones would populate the network. The use of the localized clones is actually limited to servicing an allocated RRS. In order to facilitate a mechanism for automatic removal of such clones after they service the concerned RRS, these clones are conferred a *life-time* as and when they are created so that they live just about as long as they are needed. The *life-time* of the clone, L_{lc} is fixed at a value which is equal to the time taken to travel the spanning length and is given by, $L_{lc}=T_h*d$, where T_h is the hop-time and d is the spanning length. The parent confers this lifetime to each of the clones.

Local-Clone Service Information

If there are n number unique pheromones diffused by *n* RRSs at a node, for the same service, the mobile agent carrying that service at that node generates (*n-1*) clones and sends one each to the next nodes pointed by the (*n-1*) pheromones and itself opts to travel along the remaining one. From the next node onwards the clones migrate just like the normal mobile agents in *PherCon*. Localized clones, if capable, service all RRSs that they encounter in their paths. For instance in Figure 4-2 the green coloured clone at R_3 tracking the green

pheromone towards RRS_1 could also service R_5 *en route* if R_5 became an RRS requesting the same service as RRS_1.

4.4.2 *PherCon-C* for a Network of Mobile Robotic Nodes

In the real world since the robotic nodes are mobile the network topology changes dynamically. Thus the dynamics of the system need to cater not only to the migration of the mobile agents but also to the relative movement of the robotic nodes. The mobile robotic nodes could move constantly along random directions and speeds. As the robotic nodes start moving, the performance of the strategies that depend on the data received from neighbouring nodes begins to deteriorate. This decrease in performance mainly depends on how fast the topology changes. Since, *PherCon-C* is intended to cater to a network of mobile robots it needs to perform consistently or with minimum degradation in performance as the topology changes.

The performance of a network of mobile robotic nodes depends on the value of T_{max}, which is the maximum *Time-out* of the pheromone. If this value is large and the movement of the robotic nodes is fast, the tendency of breakages in the pheromones diffused will be high resulting in longer waiting times by an RRS. Broken pheromones may continue to populate the network and mislead the mobile agents. If this *Time-out* is made small then they may have to be re-diffused more often (i.e. a lower value of δ), thus resulting in more energy consumption. In such a system, a low value of *Time-out* is advisable since higher values may result in broken pheromone trails to remain for a longer time within the network forcing the mobile agent to go astray. The spanning length d to which the pheromones are diffused into the network should also be kept low in the dynamic scenario since the chances of the mobile agent reaching the RRS via a pheromone trail farther away, is low. Low values of T_{max} and d would mean that the relaying interval δ be high, resulting in high energy consumption. Thus in dynamic scenarios, *PherCon-C* makes robotic nodes diffuse pheromones with

a constant low valued *Time-out* without maintaining a gradient as in the static case.

With a low value of *d* the effective pheromoned area is small. Under such conditions the chances of the relevant agents hitting the periphery of this area is small. Hence it is advisable to ensure that the pheromones in the periphery last for a longer time in case of dynamic scenarios. This is realized by the use of a constant *Time-out* value. In the static scenario the value of *d* is kept high to ensure higher penetration into the network. This means the effective pheromoned area is large thus increasing the chances of several relevant mobile agents hitting the periphery and tracking the pheromones. This results in unnecessarily attracting multiple agents towards same RRS. The use of a non-zero positive value for ΔT ensures that the pheromoned area shrinks from the periphery to the inside towards the RRS thereby reducing the chances of too many similar mobile agents capable of servicing the same RRS from entering this area and tracking it.

Since the *Time-out* is short and constant, it means that, the pheromones at the edge of the spanning area will live longer than the pheromones closer to the RRS contrary to what happens in a static scenario. Thus in order to maintain a constant outflow of pheromones until the RRS is fully serviced, the value of pheromone relaying interval δ for dynamic scenarios is chosen to be equal to the constant *Time-out* value.

4.5 Messages versus Mobile agents and Pheromones

Rather than use mobile agents and pheromones to realize the search for a service, a robot may opt to flood the network with a request message for the concerned service. Theoretically, flooding the entire network with a service request message is bound to quickly search and guide the concerned agent towards the RRS. In the real world this brute force approach consumes a large amount of bandwidth as also energy. Depending on the topology of the network and availability of bandwidth, flooding may at times cause a message to reach its

destination at a slower pace. Messages may continue their journey into the network even after the agent has been identified and the service effected at the RRS with no way to contain them. This naturally results in unnecessary consumption of bandwidth which may tend to retard other tasks within the network. Further, if the number of RRSs is increased, flooding by each of them can cause a near choking of the available bandwidth. In a networked robotic scenario, it is but natural that many robots may concurrently request for one or different services. The return path through which the service has to routed to the RRS will be the optimized inverse forward path of the message. The optimization may not necessarily provide an effective path back to the RRS. This can cause delays in the service being effected at the RRS. The problem becomes worse when the robotic nodes forming the network are mobile and alter the topology of the network with time.

Diffusing pheromones, on the contrary, can decrease the use of bandwidth and at the same time provide for an effective and conservative use of energy and time. Unlike messages, pheromones have a concentration and a life time and thus penetrate and remain within the network only for a finite amount of time. If the RRS emanating them is not serviced in this time, it once again re-diffuses the pheromone in the same manner. Pheromones also provide for the shortest path from the periphery of the pheromoned area to the RRS thus optimizing part of the return path for the agent.

4.6 Results and Discussions

In order to compare the efficacy of *PherCon* and *PherCon-C*, with each of the five migration strategies, - Random, Conscientious, EVAP, CLInG and G-B, simulation runs were carried out for each of the cases viz. (I) Single Agent and Four RRSs and (II) Multiple Agent and Four RRSs and the step-counts required to service each of the RRSs along with the energy consumed in terms of intra-node computations and inter-node communications were found. Energy is

consumed both when a mobile agent migrates, executes on the robot's mobile agent framework and also when the pheromones are diffused and evaporated. Since most of these operations are memory accessions and inter-node communications it is assumed that the energy consumed by each of the robotic node is proportional to the number of such accessions and communications.

A *run* comprises several discrete simulation steps and ends when all the RRSs in the network are serviced by the mobile agents.

The term *step-count* refers to the number of simulation steps taken. The simulator and its features have been described in Chapter 6. In both the cases, the initial positions of the robotic nodes and the agents were kept identical. The simulation times in terms of step-counts for each of the strategies for the two cases, for both static and dynamic robot networks, were separately determined.

4.6.1 Static Scenario

The parameters used for Random, Conscientious, EVAP, CLInG, G-B, *PherCon* and *PherCon-C* in the simulation of static scenarios were as follows:

Total Number of Nodes in the network=200, Total Number of Links= 1624. The values of the additional parameters used for *PherCon* and *PherCon-C* are given below:

d=10, C_{max}=*100,* ΔC=20, T_{max}=20, ΔT=2, δ =20.

Case (I) Single Mobile Agent and Four RRSs:

In this scenario, the network was populated with just one mobile agent and simulation commenced with the existence of four RRSs.

For each of the seven strategies viz. Random, Conscientious, EVAP, CLInG, G-B, *PherCon* and *PherCon-C* simulations were carried out five times. In the G-B strategy this single mobile agent cloned to increase its population. The graph in Figure 4-3 depicts the number of steps taken by each of the seven strategies while those in Figure 4-4 and Figure 4-5 show the energy in terms of intra-node computations and inter-node communications.

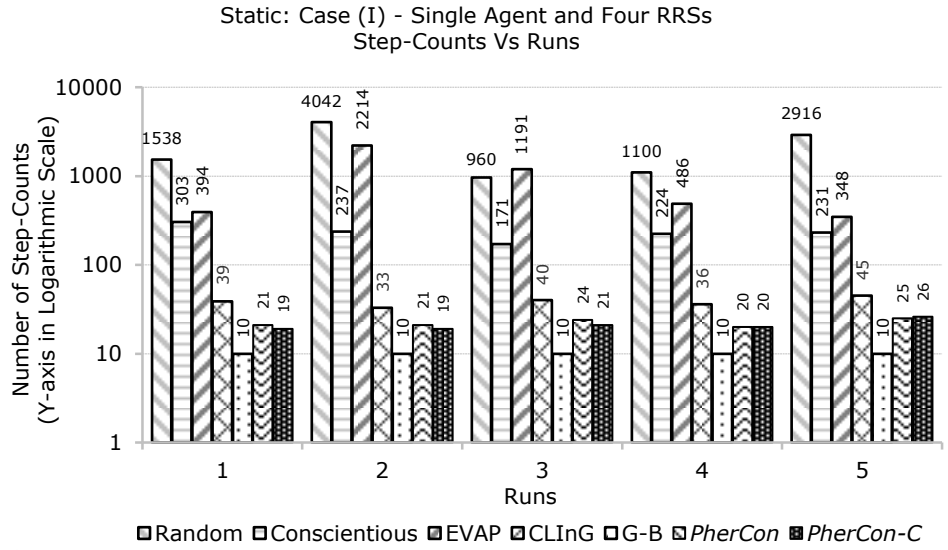


Fig. 4-3 Graph depicting Number of Step-Counts Versus Runs for Static Scenario: Case (I) - Single Agent and Four RRSs obtained for five individual runs

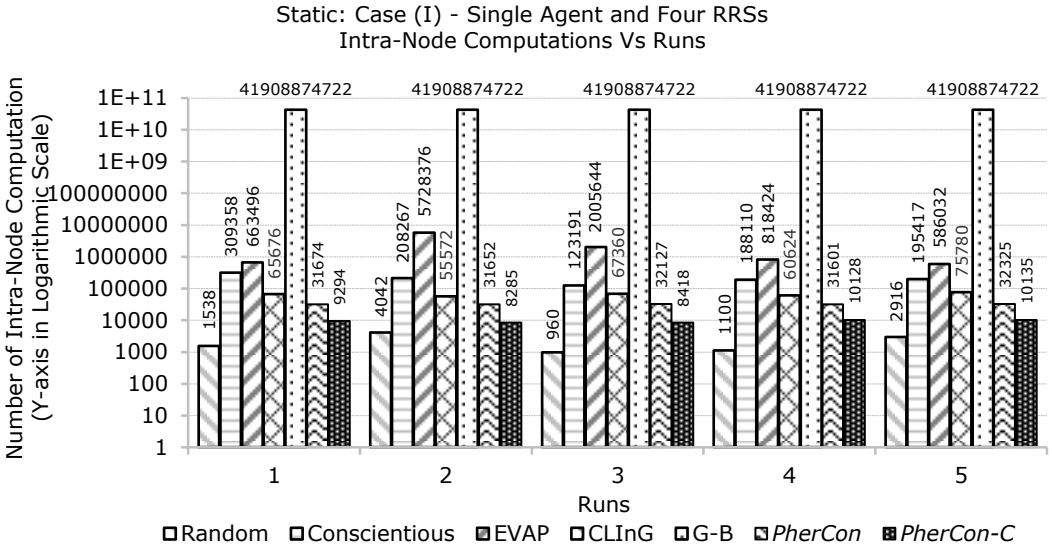


Fig. 4-4 Graph depicting Number of Intra-Node Computations Versus Runs for Static Scenario: Case (I) - Single Agent and Four RRSs obtained for five individual runs

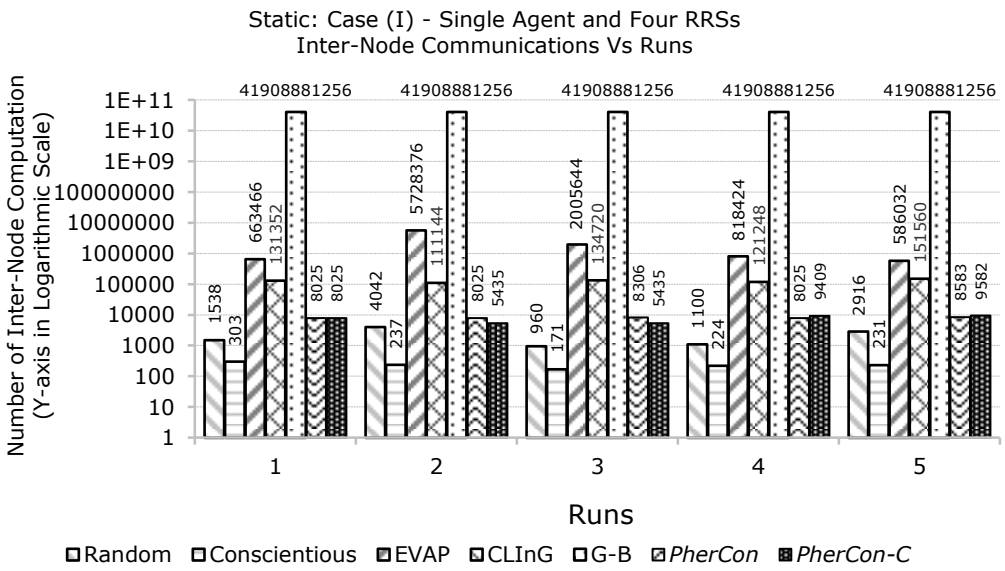


Fig. 4-5 Graph depicting Number of Inter-Node Communications Versus Runs for Static Scenario: Case (I) - Single Agent and Four RRSs obtained for five individual runs

From the graph in Figure 4-3 it can be observed that the Random, Conscientious, EVAP and CLInG strategies consume far more number of steps than the rest. The G-B strategy consistently takes minimum number of steps followed closely by *PherCon* and *PherCon-C*. The maximum difference between the former and the latter two is just about 16 steps which is in contrast with the large differences in case of the other strategies.

The graphs in Figure 4-4 and 4-5 reflect the proportional energy consumed by each of these strategies in finding and servicing the four RRSs within the network. These graphs present a contrary performance to that reflected in the previous graph. As seen from them, the G-B strategy consumes most making it in no way energy efficient. The Conscientious, EVAP and CLInG seem to be energy efficient but do not fare as well as *PherCon* or *PherCon-C* in terms of time. The Random migration strategy is more energy efficient since both intra-node computations and inter-node communications are minimal but performs the

worst in terms of the number of steps (time). When both time in terms of simulation steps and energy in terms of intra-node computations and inter-node communications is taken into consideration, it can be inferred that *PherCon* and *PherCon-C* perform the best.

Case (II) Multiple Mobile Agents and Four RRSs:

In this scenario, the network was populated with four RRSs and multiple mobile agents carrying the requested service. The number of mobile agents was varied from 2 to 12 to study the effect of an increase in agent population. Figure 4-6 through 4-8 depict the graphs of the results obtained. As can be seen, the graphs clearly indicate that an increase in number of agents hastens the service of the RRSs for all the strategies uniformly. A comparison of the performance of each of the strategies however does not provide any extra information since the trend seems to be the same as discussed in Case-(I) both in terms of step-counts

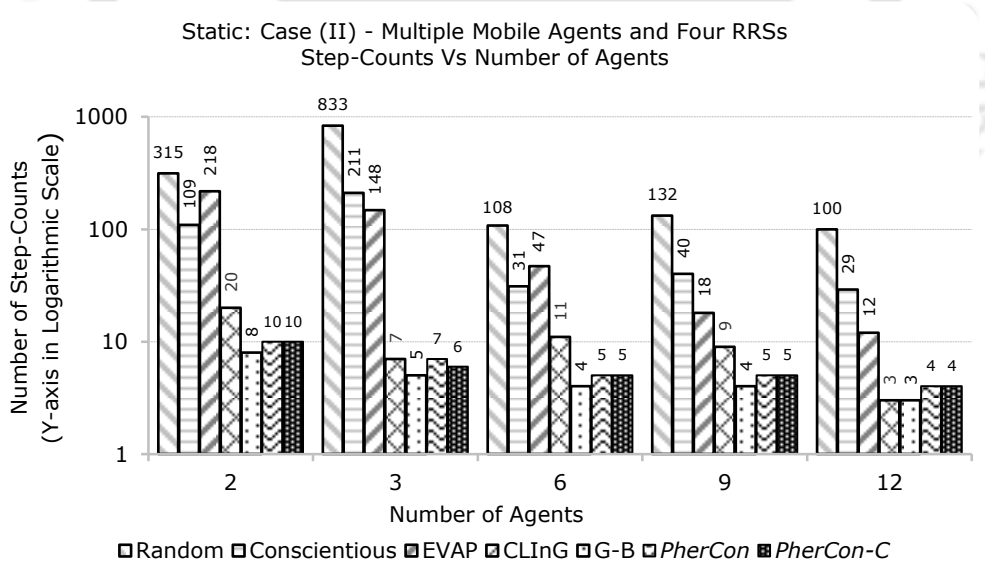


Fig. 4-6 Graph depicting Number of Step-Counts Versus Runs for Static Scenario: Case (II) - Multiple Mobile Agents and Four RRSs obtained for five individual runs with 2, 3, 6, 9 and 12 Agents

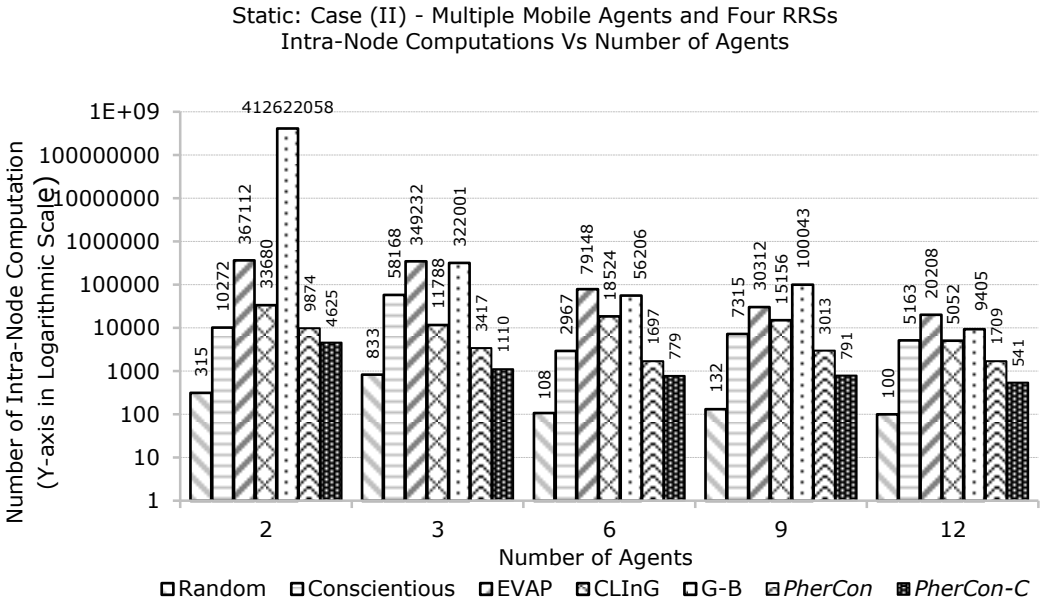


Fig. 4-7 Graph depicting Number of Intra-Node Computations Versus Runs for Static Scenario: Case (II) - Multiple Mobile Agents and Four RRSs obtained for five individual runs with 2, 3, 6, 9 and 12 Agents

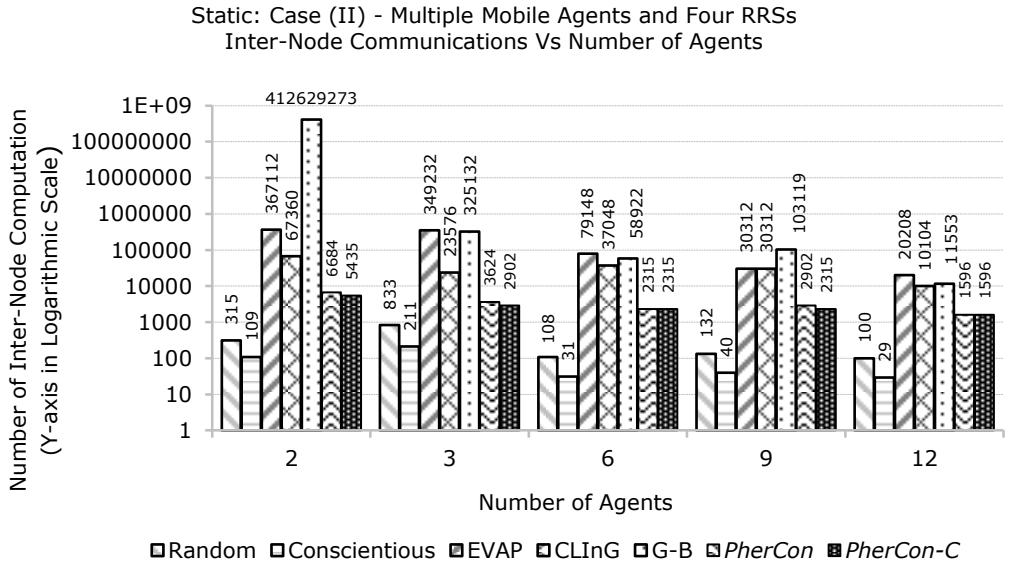


Fig. 4-8 Graph depicting Number of Inter-Node Communications Versus Runs for Static Scenario: Case (II) - Multiple Mobile Agents and Four RRSs obtained for five individual runs with 2, 3, 6, 9 and 12 Agents

and energy. The conclusion inferred earlier in Case-(I) that *PherCon* and *PherCon-C* perform better when both time and energy are taken into consideration, holds here too. However if energy is not a criterion, then the graphs in Figures 4-3 through 4-5 and Figures 4-6 through 4-8 may portray the G-B strategy to be the best. Comparisons between the G-B strategy and the proposed *PherCon and PherCon-C* together with their implementational issues given below, unravels several concerns which the graphs, presented herein, do not portray.

G-B versus *PherCon and PherCon-C*

From the graphs shown in Figure 4-3 and Figure 4-6, it can be seen that *PherCon* and *PherCon-C* take slightly more number of steps than the G-B strategy during simulation. However in the real world, the G-B strategy may take more time than *PherCon* or *PherCon-C* since the time required for one simulation step in G-B is greater than those of *PherCon* and *PherCon-C*. In the G-B strategy, cloning at each of the nodes could be assumed to be in parallel. Cloning in G-B is carried out at the initial part of each step in the simulation. Immediately after cloning, the actual times required for the agent and the clones to migrate to other nodes would be high and difficult to ascertain exactly. This is so because, in practice, during the time when an agent migrates from a certain node N_1 to node N_2, no other agent from any node in the network can migrate to either N_1 or N_2.This can cause agents to wait for their turn to migrate and contribute to the latency. If all the nodes contain agents ready for a migration (after cloning) then many migrations would be performed sequentially making the total time taken for all these agents to migrate to their respective destination nodes to be large. It may also be observed that if the G-B strategy were to be implemented in a real network, the actual number of intra-node computations and inter-node communications will increase exponentially for every subsequent time-step making the initial steps to consume far lesser time than the later ones. This is so because of the explosion in the number of clones at subsequent nodes

having branches. The migration of these clones can cause a severe slow down as the execution progresses.

In a highly connected network topology, most inter-node communications would be sequential in nature, thus consuming more time than intra-node computations. Thus, though G-B takes far less number of steps in simulation, the total time required for their execution in the real world will be much more than the other strategies.

In *PherCon* and *PherCon-C*, the network is not cluttered with clones, which in turn allows for faster migration of agents between the nodes. The situation described for G-B will hardly ever occur in *PherCon* or *PherCon-C* due to the limited number of agents populating the network.

PherCon and *PherCon-C*, thus appears to be the best when a tradeoff between the time consumed and the energy expended in terms of intra-node computations and inter-node communications, is considered.

The problem of cluttering of the network has been further revisited, argued and a solution proposed in Chapter 6.

PherCon* versus *PherCon-C

Comparing *PherCon* and *PherCon-C*, it can be found that, with respect to step-counts, the latter is only slightly better than the former. Whenever an agent reaches a node where multiple pheromone requests for the same service from multiple RRSs are present, a marginal improvement due to *PherCon-C* is observed. If such a situation does not occur the agent behaves just as in *PherCon* migration strategy and performs accordingly with no improvement. With respect to intra-node computations and inter-node communication energy, *PherCon-C* performs similarly marginally better than *PherCon*.

Implementation Issues

Many an algorithm is portrayed to be the best using results obtained from simulations. Implementation issues play a major role in the viability of the use of the strategy or algorithm. Implementing G-B strategy accounts for a huge

number of clones to be generated that can hog the bandwidth of the network and subsequently degrade the performance in a real system.

In such systems, the agents populating the network could be heterogeneous and thus carrying different services as payloads. EVAP and CLInG seem to inherently support only homogeneous agents. In a heterogeneous agent scenario, the use of EVAP and CLInG would mean that each node in the network would have to maintain the *idleness* and *propagated idleness* values of each type of agent. More the number of distinct agents carrying different services more will be the overheads of maintaining these values in each node and propagating them to their immediate neighbours in every time step. An inherent assumption made in the simulation of EVAP and CLInG is that the exchange of the idleness values between all nodes and their respective neighbours is a single step parallel operation. As pointed out earlier this is not true in the real world. The process of exchanging the values also calls for more bandwidth, thus affecting performance. Further the use of EVAP and CLInG would pose difficulties in scaling the system when agents carrying new services are introduced into the network on the run.

4.6.2 Dynamic Scenario

The parameters used for Random, Conscientious, EVAP, CLInG, G-B, *PherCon*, *PherCon-C*, in the simulation of dynamic scenarios were as follows:

Total Number of Nodes in the network=200, Total Number of Links= 1624

The values of the additional parameters used for *PherCon* and *PherCon-C* are given below:

d=10, C_{max}=*100,* ΔC=20, T_{max}=7, ΔT=0, δ =7.

The mobile robotic nodes were made to move relative to each other in a random fashion to induce dynamism into the network.

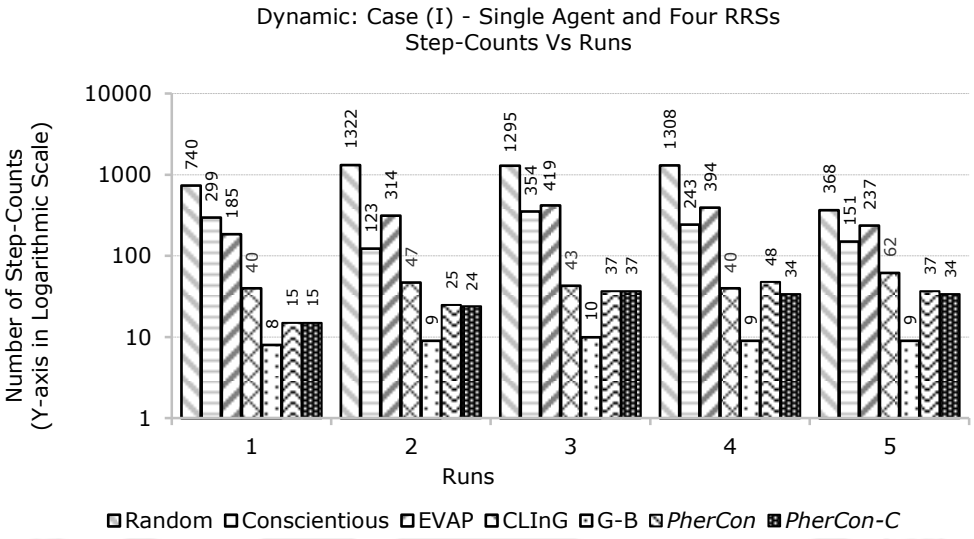


Fig. 4-9 Graph depicting Number of Step-Counts Versus Runs for Dynamic Scenario: Case (I) - Single Agent and Four RRSs obtained for five individual runs

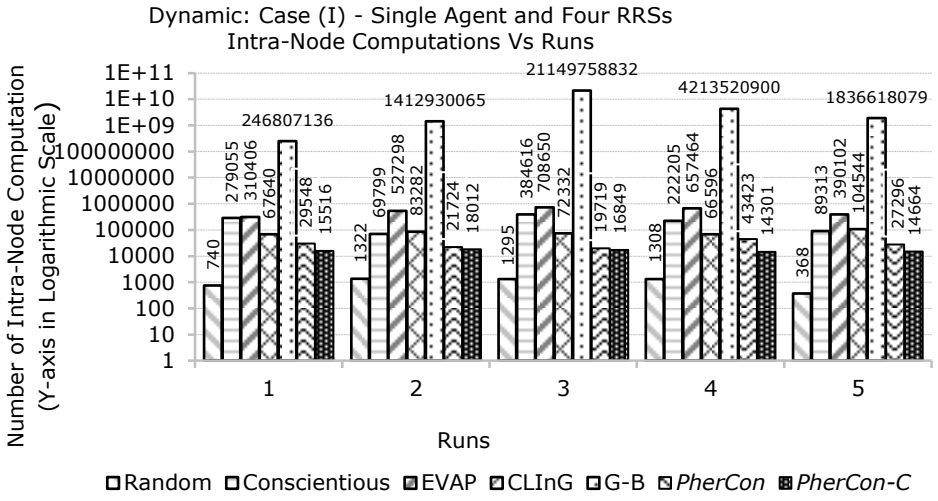


Fig. 4-10 Graph depicting Number of Intra-Node Computations Versus Runs for Dynamic Scenario: Case (I) - Single Agent and Four RRSs obtained for five individual runs

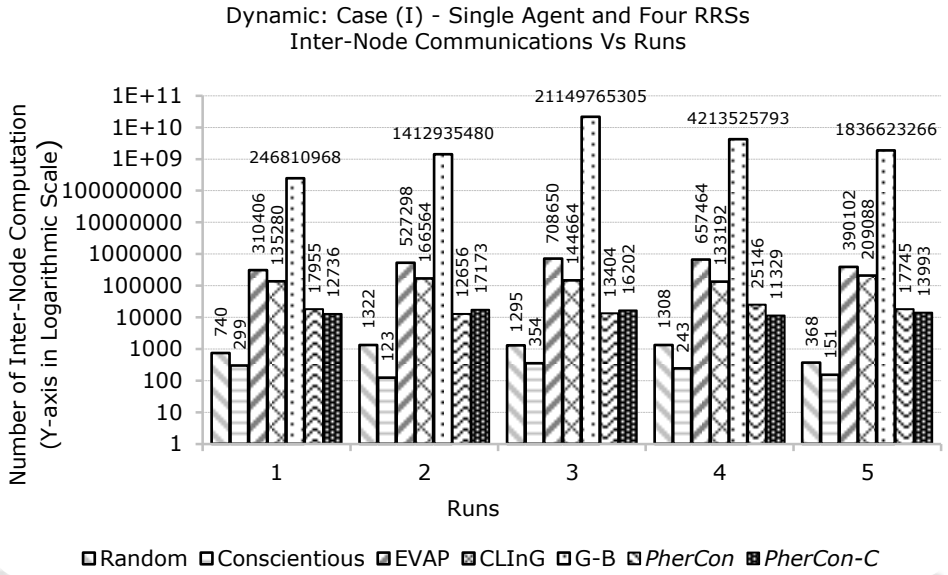


Fig. 4-11 Graph depicting Number of Inter-Node Communications Versus Runs for Dynamic Scenario: Case (I) - Single Agent and Four RRSs obtained for five individual runs

Case (I) Single Mobile Agent and Four RRSs:

The graph in Figure 4-9 depicts the step-counts times taken by an agent to reach the four RRSs respectively by each of the seven migration strategies while those in Figure 4-10 and 4-11 show the energy consumed in terms of intra-node computations and inter-node communications for one agent to service four RRSs.

Comparing the graphs for the static scenario (Figure 4-3 to Figure 4-5) with those of the dynamic scenario (Figure 4-9 to Figure 4-11), it can be observed that there is an increase in the number of step-counts taken by all strategies in the latter scenario. This can be attributed to the dynamic nature of the robotic nodes which force the randomness into the agent migration strategies. Agents may

have been lead astray due to the movement of the nodes. However the relative trends in the performances of all the migration strategies seem to remain the same as in case of the static scenario. The random and conscientious migration strategies consume lesser energy in terms of intra-node computations and inter-node communications but take more *step-counts* to reach the RRS. The G-B migration strategy takes the minimum number of step-counts but is expensive in terms of energy. *PherCon* and *PherCon-C* provide a consistent performance in terms of both time and energy.

Case (II) Multiple Mobile Agents and Four RRSs:

The graphs in Figure 4-12 to 4-14 are similar to the graphs in Figure 4-9 to 4-11 but for the fact that the robotic nodes are dynamic. Here too the relative trends seem to be the same as observed in the static case. As in the static scenario, the difference between the performances of the migration strategies diminishes as the initial number of agents is increased. This is because of the increase in the number of agents carrying the same resource.

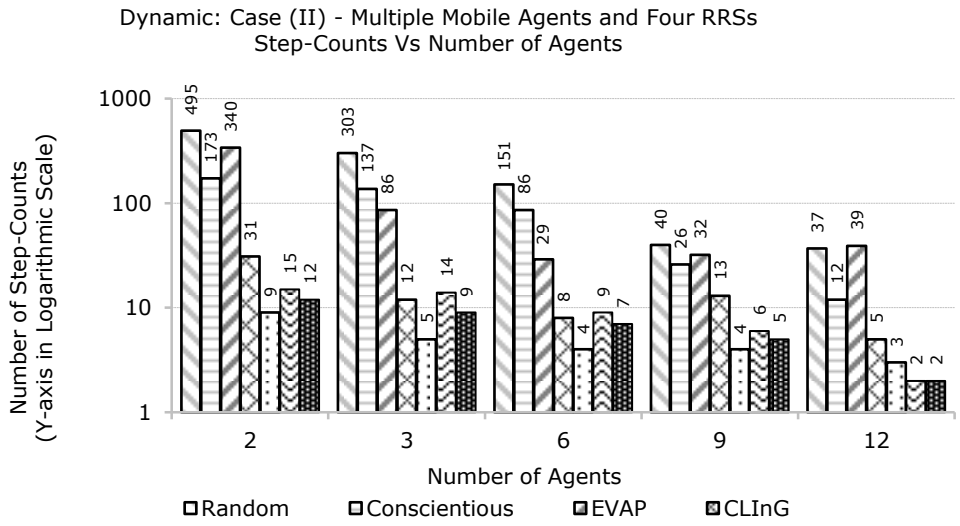


Fig. 4-12 Graph depicting Number of Step-Counts Versus Runs for Dynamic Scenario: Case (II) - Multiple Mobile Agents and Four RRSs obtained for five individual runs with 2, 3, 6, 9 and 12 Agents

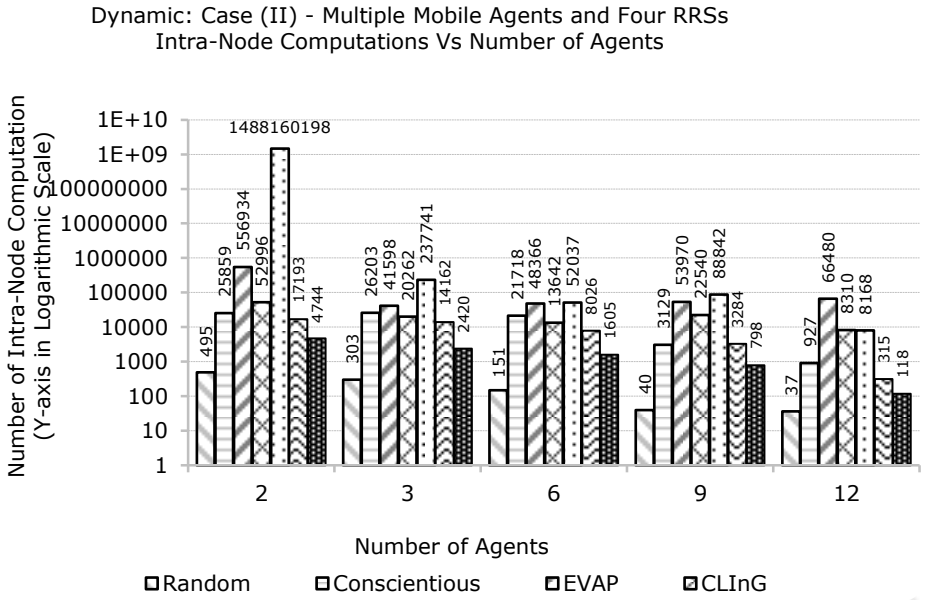


Fig. 4-13 Graph depicting Number of Intra-Node Computations Versus Runs for Dynamic Scenario: Case (II) - Multiple Mobile Agents and Four RRSs obtained for five individual runs with 2, 3, 6, 9 and 12 Agents

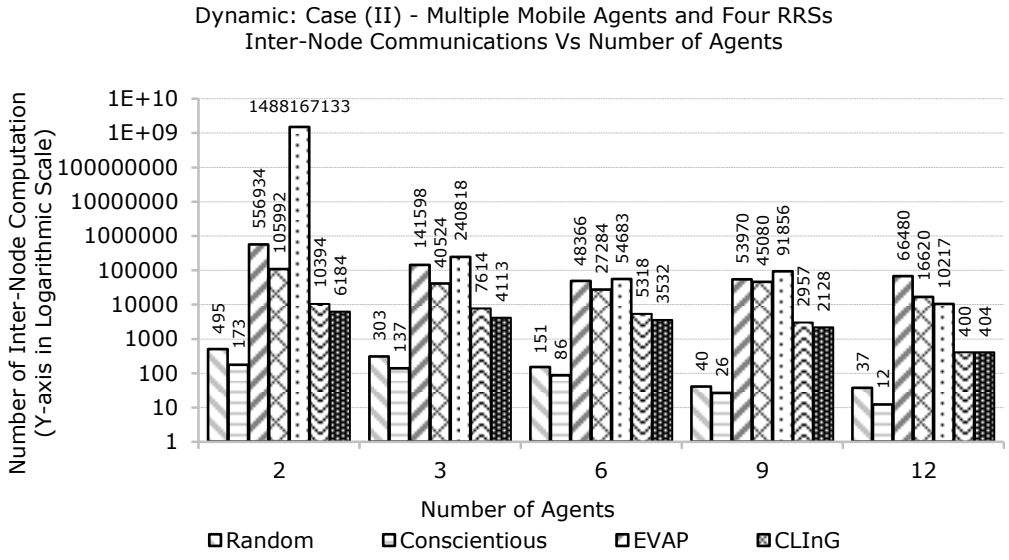


Fig. 4-14 Graph depicting Number of Inter-Node Communications Versus Runs for Dynamic Scenario: Case (II) - Multiple Mobile Agents and Four RRSs obtained for five individual runs with 2, 3, 6, 9 and 12 Agents

4.7 Conclusions

This chapter presents a bio-inspired *search and service* mechanism for use in conjunction with a dynamic network of mobile robots and agents.

A mobile agent based service mechanism for networked robotics nicknamed *PherCon*, using a search strategy that resembles a bi-directional search has been proposed. The mobile agents search for an RRS that in turn diffuses pheromones to attract the former to facilitate a quicker service. While the mobile agent normally opts the conscientious migration strategy, it changes its stance on encountering a pheromone and seeks the RRS based on the pheromone's concentration gradient. The RRS on its part diffuses pheromones with a concentration and a *Time-out* onto its neighbouring nodes that provide a means for the agent to find the shortest route towards itself. Together the RRS and the mobile agent carrying a service reach out to one another in the network to eventually effect the service. An enhancement to this strategy – *PherCon-C* provides to speed up the service process by generating clones at locations where more than one RRSs have pheromoned for the same service. Simulation results show that this migration strategy proves to be far more conservative in terms of the consumption of both time and energy taken together in static as well as dynamic network scenarios. Since a simulation cannot bring out the complete inherent parallelism of the system, it is envisaged that in practical scenarios this migration strategy will perform even better. CLInG seems to show promising results in simulated environments but the same may not be guaranteed in real world heterogeneous agent scenarios.

The improvement in performance exhibited by the use of *PherCon-C* due to its inherent capability to increase the number of agent clones in a controlled manner, reveals a simple premise –

An increase in the overall performance of the system (in servicing the RRSs) can be realized if all the heterogeneous mobile agents populating the network could clone without choking the available bandwidth.

Controlling cloning and the number of clones populating a network need to be performed in a distributed manner without any additional burden incurred on the available bandwidth lest it degrade the performance of the system. A real networked robotic system could be populated with heterogeneous agents – agents that carry different services as payloads. Increasing the number of each of these agents based on their current need within the network will drastically decrease RRS service times. The next chapter discusses a novel yet simple manner of controlling cloning and the population of clones of these heterogeneous mobile agents within the robotic network so as to not only provide faster services to the RRSs but to also ensure that the bandwidth is not hogged by any one homogeneous set of agents or its clones. The chapter also portrays the results of using *PherCon* coupled with such a cloning controller.

Chapter 5
Mobile Agent Population Control

Mobile agents share all characteristics of their static counterparts but stand apart in their capability to migrate and clone autonomously. The concept of cloning featured in mobile agents helps increase their population and indirectly allows for parallel operation and information transfer across a network. It can also aid in the upward scalability of a distributed system. Having multiple copies of an agent can, not only enable parallel execution but can also provide fault-tolerance, thus increasing both robustness and efficiency of the system. The down side is that an increased population of mobile agents may result in higher network resource utilization which in turn may deteriorate the performance of the system. While on one side increasing the population of agents using cloning increases its performance, uncontrolled cloning can flip the same and cause the network to become overcrowded. A controlled, adaptive and demand based cloning can ensure that the performance remains well near the achievable optimum. This chapter describes an attempt to design a bio-inspired adaptive, on-demand cloning controller for controlling a population of a heterogeneous mobile agents in both static and dynamic network of robotic nodes.

5.1 Motivation and Related Work

Results portrayed in Chapter 4 indicate that there is a marked improvement in the performance of *PherCon-C* that uses localized cloning when the RRSs requesting the same service are in close proximity. Such Cloning seems to yield lower RRS service times. In order to effectively patrol and provide on-demand services in a networked robotics scenario using heterogeneous mobile agents, cloning needs to be performed by every agent based on its demand within the network. Heterogeneity in the current context

indicates that the agents populating the network carry different services and are thus capable of servicing different requests generated by the robotic nodes. In the previous chapter, the agents populating the network were assumed to carry the same payload and hence were homogeneous in nature. All RRSs too requested the same service. In a real networked robot scenario, robotic nodes could generate requests for different kinds of services which essentially emphasize the need for such heterogeneous agents.

Agents whose services (payloads) are more often requested should clone proportionately so as to provide a quicker service. With the build-up of more clones within a network, the service times are bound to fall but not for long. A large number of agents and clones can clutter the network increasing migration times and thus deteriorating the performance of the system.

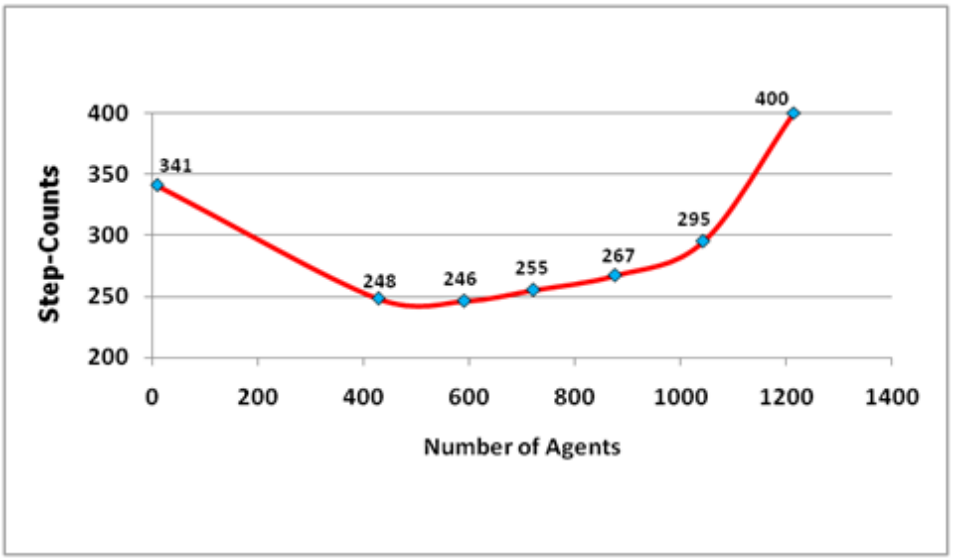


Fig. 5-1 Step-Counts versus number of Mobile Agents required to service 100 RRSs in a 200 node connected network with no control over cloning

In order to study this effect a simulation on a 200-node network with 100 RRSs using *PherCon* mobile agent migration mechanism was carried out. In the simulation, each node was assumed to be able to host a maximum of *N* number

of agents. If an agent in a node finds that there are *M* agents (*M*<*N*) residing within, then it generates (*N*-*M*) clones, thereby filling up the node to its maximum hosting capacity. This facilitated a mechanism to increase the agent population within the network. If *M*=*N* then it would not clone but wait for a possible migration as and when its turn comes up. A migration to the next node was possible only if that node had less than *N* agents. A choking effect would thus occur when most of the nodes were filled to their maximum capacity thereby inhibiting agent migrations. *N* thus determines the maximum number of agents that can populate the network. Figure 5-1 shows a graph depicting the number of simulation steps versus the number of agents (and clones) that populated the network after all the 100 RRSs were serviced generated by varying the value of *N* from 1 to 7.

The number of steps decreases initially with increasing number of agents (and their clones) and later increases due to cluttering within network. The graph endorses the imperative need for controlled cloning so as to ensure that the total number of agents and its clones populating the network remains high while also providing minimal service times. In scenarios where heterogeneous mobile agents populate the network, it is essential that the total number of such agents remain optimal so as to provide low RRS service times and also that these agents clone proportionate to their demand. The mechanism should therefore also facilitate the depletion of mobile agents that are no more in demand thereby giving way to other in-demand agents while also maintaining the net population at an optimal value so as not to cause cluttering or severe bandwidth contention.

Suzuki et al. [119], [140] have addressed the control of a population of mobile agents in dynamic networks. Their mechanism is based on the popular ecological model which assumes that the population of a single species converges to a number in proportion to the amount of *food* available in its environment. Every node in their network generates *food* at regular intervals in proportion to the number of its links to other nodes. An agent migrates to a node, consumes the *food* within the node and clones in proportion to the excess *food* it

cannot consume. An agent which does not get *food* starves and thus is eliminated. They have suggested two algorithms – one in which an agent is provided with the actual number of links in a vertex and the other wherein the agent makes an estimate of the link density. The latter algorithm proves to be a useful aid in calculating the *food* to be generated in case of dynamic networks where the nodes are mobile causing a change in topology and hence the links. Ma et al. [141] and Luczak et al. [142] describe algorithms that use a mobile agent population control protocol wherein each node keeps at most one copy of an agent. If there is a single agent in a node then a new agent is born with a certain probability *p*, computed based on the fraction of target nodes. In this case, the agents make use of a random migration policy. Another mechanism for population control of agents within the Internet described in [143] consists of nodes, mobile executor agents, static controller agents and blackboard agents among others. Each of the executor agents is conferred some energy which is consumed as and when it performs a task. When the energy level falls below a certain threshold, the agent requests for more energy from its controller, via the blackboard agent. If the controller does not respond immediately, the agent becomes an *orphan* and is thus removed from the network. This protocol is complex in terms of inter-entity communications and the control is not fully decentralized. There is a heavy dependency on the controller agent whose failure would mean the same for the entire system. A biologically inspired mechanism for a homogeneous mobile agent patrolling system has been suggested by Amin and Mikler [144] using pheromones. An agent that visits a node lays pheromone which is volatile in nature. Based on the amount of pheromone an inter-arrival time of an agent at a node is calculated by the agent that has just reached this node. This inter-arrival time is used to estimate the frequency of visits made by an agent to this node. If this time is higher than a certain threshold, the agent assumes that the there are lesser number of agents in the network and thus clones to cope up with the situation at hand. On the contrary if it finds this value below another threshold the agent kills itself assuming it to be redundant. If the number

of visits is in between these two thresholds the agent merely migrates to a neighbouring node. Gaber and Bakhouya [145] focus on the dynamic regulation of the mobile agent population in a distributed system inspired by concepts from the biological immune system. They embed three basic behaviours onto the mobile agents viz. *cloning*, *moving* and *killing* and attribute them to three different antibodies, each suppressing or stimulating the other. The behaviour to be chosen depends on the inter-arrival time of the agents at a particular node, similar to that suggested by Amin and Mikler [144], which in turn controls the agent population.

All the mechanisms cited so far support population control of a homogeneous set of mobile agents and are inherently suited for a regular patrolling problem where the inter-arrival times between visits made by the same type of agent are to be kept a constant or a minimum. In all these mechanisms the maximum number of agents is a factor of the total number of nodes and needs to be known *a priori*. These mechanisms are neither scalable for different types of agents (heterogeneous) nor adaptive in terms of their manner of controlling the agent population. Instead they merely try to ensure that the existence of a certain number of agents of the same type (homogeneous) will not clutter the network and thus avoid bandwidth contention.

These mechanisms can be extended and envisaged to support a heterogeneous set of mobile agents by adding the necessary book-keeping methods within both the nodes and the agents. These include, for instance, pheromone management at every node for each type of agent as in [144] or the maintenance of inter-arrival-times for every type of agent in each node [145], [144]. This naturally will not only increase computational overheads and slow down the performance of the system, but will also increase the net number of agents populating the network to around M x N where M is the number of distinct types of agents, presence of which contributes to the heterogeneity of the network and N is the optimal number of agents that would populate the network in the homogeneous case ($M=1$) assuming that the inter-arrival time for all types

of agents is the same. Since *N* is itself the limit to the number of agents populating the network beyond which a choke-up or cluttering can occur, positive integer multiples of *N* could make the network come to a stand-still. This happens because the number *N* is optimized independent of *M* making most of the work cited so far unsuitable for applications that need to be scaled in terms of the number of heterogeneous agents.

In scenarios as described in the previous chapter where the nodes (RRSs) asynchronously generate requests for a service, the waiting times can be further decreased if the mobile agents populating the network are made to clone proportionate to their demand within the network. Cloning a mobile agent carrying a certain service as its payload, in large numbers based on its utility may theoretically decrease waiting times of those nodes that request this specific service. However in practice, this may not be a viable alternative since the network may have several nodes requesting different services. If all the pertinent agents were to clone in large numbers at the same time, they would quickly degrade the available bandwidth and choke up the system. Cloning thus needs to be carried out judiciously based on the network conditions (available bandwidth) and also demand (number of RRSs). Mobile agents need to also back-off and die in case of choke-ups (low bandwidth). Decisions on whether to or not to clone or to back-off have to be made autonomously by the agent and not in consultation with others agents or robotic nodes. As in [146] if each agent were to take opinions (such as stimulations or suppressions) from other agent peers, this additional communication would cause further overheads on the bandwidth. With a large number of such agents communicating and transacting stimulations and suppressions to one another, such as that suggested by Farmer et al. [147], progress of their movement towards the respective RRSs could be greatly retarded. Mobile agents in real-world application scenarios such as the robotic network described in [109] and the one explained in Figure 2-3 in Chapter 2 will need to make a decision on whether to or not to clone by sensing the active medium they migrate through, in a *stigmergic* manner. Co-ordination or

communication among agents performed in an indirect manner is referred to as *stigmergy*. This bio-inspired mechanism finds its origin in the manner in which insects such as ants and termites indirectly communicate with one-another using their common environment as the medium of communication to achieve complex tasks. Agents sense the left-over traces of their previous actions in the environment and conclude on the action they need to take next, thus avoiding any direct agent to agent communication. In this chapter, we describe a cloning controller which facilitates mobile agents within a network to use *stigmergic* means to decide the extent of cloning and realize faster servicing of the RRSs. The work augments the *PherCon* migration strategy to achieve low RRS waiting times in this network populated by a heterogeneous set of agents.

5.2 Architecture of the Cloning Controller

We describe herein, a novel bio-inspired cloning control mechanism [148] suited for active patrolling and servicing within a network populated by heterogeneous agents. The mechanism is both scalable and adaptive. Cloning is controlled egocentrically by each mobile agent without consuming any additional bandwidth for communicating with other mobile agents across the network, thus making the mechanism well suited for distributed systems. There is no centralized mechanism to serve information of any kind to these mobile agents. Each node is embedded with a mechanism to individually approximate the condition of the bandwidth of the network.

Each node has a queue, called the *intra-node queue* into which all the mobile agents hosted by it are lined up before migrating to another node. Thus mobile agents migrate from the *intra-node queue* of one node to that of its one-hop neighbour. If the node is an RRS and the incoming agent has the requested service then the same is downloaded onto the node before the agent enters the queue for onward migration to a neighbouring node. Apart from the service or code for a task, the mobile agents carry within themselves a record of their

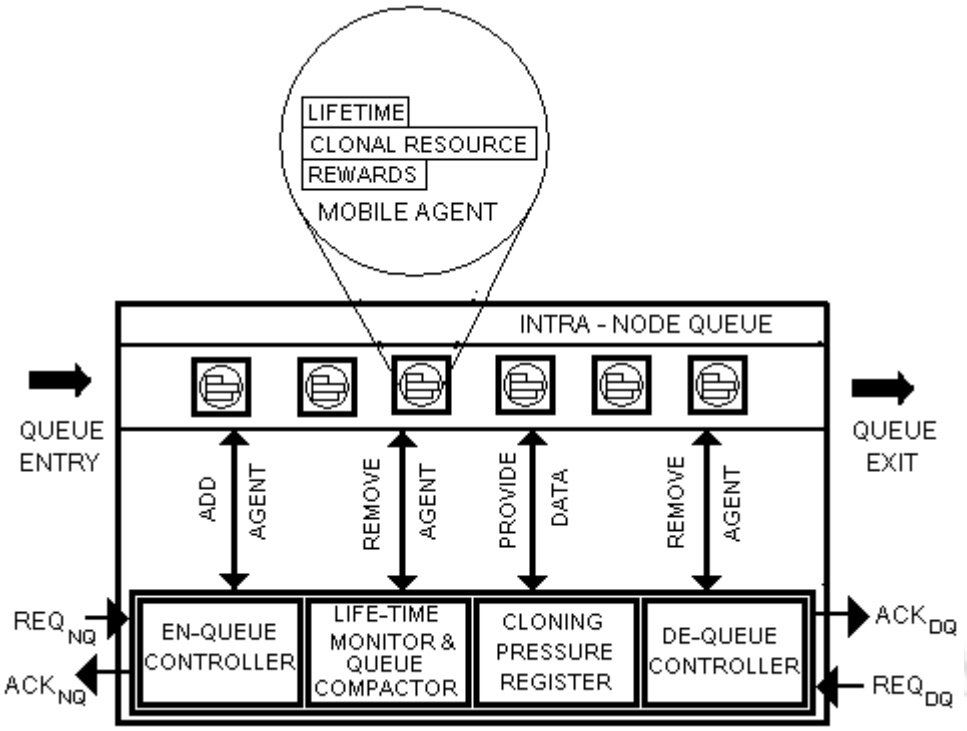


Fig. 5-2 Architecture of the Cloning Controller

current *life-time*, *cloning resource* and the number of rewards received. These concepts have been dealt with in subsequent sections.

Figure 5-2 depicts the composition of the cloning controller residing within each node while Figure 5-3 shows the mechanism behind the cloning control. A *Q-Manager* controls the intra-node queue. The *De-Queue* (DQ) *Controller* within this manager performs the job of handshaking with a node in its immediate neighbourhood by sending a request for the migration of an agent within its associated intra-node queue to the next node via REQ_{DQ}. It receives the acknowledgement from its peer in the other node via ACK_{DQ}. The *En-Queue* (NQ) *Controller* performs the task complementary to the *De-Queue* Controller and caters to requests for migration of an agent residing in another node into its intra-node queue using REQ_{NQ} and ACK_{NQ} in a similar fashion. Migrations are performed only if the queue in the next node has a vacant slot. The *Life-time*

Monitor cum Q-Compactor unit ensures that the life-times of each of the agents within the queue are decremented in each (time) step and the queue is compacted as and when an agent dies within the queue. The agent residing in the queue, reads the *Cloning Pressure*, ρ, from the *Cloning Pressure Register* resident within the *Q-Manager*. The *cloning pressure* ρ is calculated based on the number of agents populating the *intra-node queue* and has been dealt with in a subsequent section on dynamics of the controller.

```
The following operations occur in every step at each node:
At NodeQueue
   1. If an Agent is at the top of the Intra-node Queue
         a. CloneifNecessary()
         b. Compute "NextNode" using Pheromone-Conscientious Algorithm
         c. Check if movement to the next node is possible. If true goto (d) else goto(3)
         d. Execute OnDeparture() Method
   2.If an Agent is permitted into the Intra-Node Queue
         a. Execute OnArrival() Method.
   3. Decrement the Life-time of every agent in the queue
OnArrival()
   1. Execute the service if this is the RRS that requested for its service. If true goto (2) else
      goto (3)
   2. Charge the Cloning Resource and the Life-time based on rewards
   3. Enter into the Intra-Node Queue of the entering Node
OnDeparture()
   1. Remove from the Intra-Node Queue of the exiting Node
CloneifNecessary()
   1. Find Resource needed for Cloning
   2. Find the Number of Clones
   3. Decrement the Resource based on the Number of clones
   4. Create clones
   5. Recharge the Cloning Resource
```

Fig. 5-3 The Cloning Control Mechanism

A *queue threshold* (Q_{Th}) determines the maximum number of agents allowed in the intra-node queue. The basic objective is to ensure that this queue is moderately filled so as to also achieve quicker migrations and hence faster service for all the RRSs. Agent migrations can be realized only if there is a

vacancy in the intra-node queue of the destination node; else the agent needs to wait within the queue of the current node while also using up its life-time. If we assume that cloning by agents causes their numbers to increase to a level that there are no empty slots within any of the intra-node queues in the network, then no agent will be in a position to migrate. This will mean that the network is choked-up. Only when some of the mobile agents within the intra-node queues die out due to their decreasing life-times, will other agents with higher-life times be able to migrate to vacancies created by their dead counterparts in other intra-node queues. Thus, the agents need to clone in such a contained manner as to increase their numbers to result in a faster service of the RRSs but at the same time ensure that such a choke-up of the network is never reached.

On reaching a node, an agent provides its services to the node if the latter is an RRS. The agent is then queued in an intra-node queue present at this node and made to wait for its turn to migrate to the next robotic node. A higher number of agents in this queue essentially means more waiting times within it and also that node to node migrations have been taking more time. Such increased migration times convey a reduction in available bandwidth within the network. Likewise, a lesser number of agents within the queue means the availability of more bandwidth. The number of agents within the queue thus seems to indirectly convey the status of the available bandwidth within the network. A mobile agent therefore *stigmergically* senses the available bandwidth using the number of agents in the queue within the node in which resides and clones proportionately.

5.3 Cloning Resource – The Underlying Rationale and Functions

The concept of resource [149], [150] can greatly alter an otherwise linear cloning mechanism. Biological glands [151] cannot secrete in large amounts continuously as they are limited by an inherent *resource*. This is much like squeezing a completely wet piece of cloth to extract a certain amount of water.

The squeezing force required to extract a certain quantity of water initially is far less than the force required to extract the same amount of water a second time. Water, in this case, is the *resource* being extracted. In biological systems, the resource is recharged by various factors which include the nutrients supplied to the gland over periods of time. We embed a similar mechanism to portray a contained form of cloning based on some externally sensed parameters and also on a *cloning resource* that is replenished by rewards which a mobile agent gains after servicing an RRS. With the cloning resource charged using such rewards, its chances of generating more clones also increases. Apart from rewards, the cloning resource of a mobile agent is also boosted periodically based on its current value and certain ambient conditions. The dynamics of cloning and the manner of resource charging have been discussed in later sections.

The cloning controller works based on a reactive mechanism to maintain the population of mobile agents within the networked system. This mechanism, embedded within each agent, senses the number of existing mobile agents in the intra-node queue waiting for their turn to migrate to the next node. The number of agents within this queue potentially gives a *feeling* of the available bandwidth based on which the agent decides whether or not to clone. The extent of cloning depends on the following parameters:

(i) The *Cloning Resource* available within the agent,

(ii) The *Rewards* it has gained by servicing the RRSs and

(iii) The *Cloning Pressure* which is proportional to the number of vacant slots in the queue.

Cloning resource is charged partly by rewards and partly by an inherent charging mechanism embedded within the agent. Apart from a cloning resource, agents also have a life-time stamped on them which increases with the rewards they acquire as they service the RRSs.

5.4 Dynamics of Mobile Agent Cloning

Initially at time *t=0*, the network comprises agents each of which carries a distinct service as its payload. These are referred to as *parents*. The distinct services carried by the agents contribute to their heterogeneity. The system initially consists of only parent agents – at least one per type of agent which always exists and never dies. Parent agents are required since agents carrying a service should not die lest the service (payload) too dies with it. As has been mentioned earlier the services carried by such agents are basically programs which the robots can execute to achieve their allotted tasks. These programs are written either by the manufacturer of the robots or by experts robot programmers, packed into the agent as its payload and then released into the network. They are thus akin to the initial antibodies which the biological being has when it is just born. Parent agents continue to populate the network all through ensuring that at least one copy of their service (payload) exists within the network. In subsequent discussions we will use the term *agent* to refer to either the parent agent or the clone unless otherwise specified. An agent makes the decision to clone based on several factors and the total number of clones generated affects the overall performance of the system in terms of both the utilization of the network resources and also RRS service times. The decision as to whether or not an agent, resident in node *n* at time *t*, should clone is made based on the *Cloning Pressure*, $\rho_c^n(t)$ given by equation (5.1) .

$$\rho_c^n(t) = Q_{Th} - Q_n(t) \quad \text{for} \quad \rho_c(t) > 0 \qquad (5.1)$$
$$= 0 \quad \text{otherwise}$$

where Q_{Th} is the *Queue Threshold* and $Q_n(t)$ is the number of mobile agents populating the queue at time *t* in node *n*. The number of clones generated by a mobile agent *a* residing at node *n* at time *t*, $C^a(t)$ is given by equation (5.2)

$$C^a(t) = \rho_c^n(t)\{R_{av}^a(t)/R_{max}\} \qquad (5.2)$$

where R_{av}^a is the available *cloning resource* within the agent *a* and R_{max} is the maximum cloning resource an agent can possess. The number of clones

generated, $C^a(t)$ is rounded off to the next lowest integer. Initially all parent agents have the maximum cloning resource to their credit i.e. $R_{av} = R_{max}$. The cloning resource is depleted from an agent *a* for every cloning session based on the equation (5.3).

$$R_{av}^{a}(t+1) = R_{av}^{a}(t) - C^{a}(t)\, R_{min} \qquad (5.3)$$

where R_{min} is the minimum value of the resource to be conferred onto a clone. This value is extracted from the cloning resource of the agent that created the clones. If this agent cannot afford to confer this minimum amount of resource then its cloning is inhibited.

The cloning resource is charged based on the equation (5.4).

$$\begin{aligned} R_{av}^{a}(t+1) &= R_{av}^{a}(t) + \tau_c\, e^{-1/R_{av}^{a}(t)} + \tau_r\, Rew(t) && \text{for } R_{av}^{a}(t) >= 1,\ \rho_c^{\,n}(t) <= 1 \\ &= R_{av}^{a}(t) + \tau_c + \tau_r\, Rew(t) && \text{for } R_{av}^{a}(t) < 1,\ \rho_c^{\,n}(t) < 1 \\ &= R_{av}^{a}(t) + \tau_c\, e^{(1-1/x)} + \tau_r\, Rew(t) && \text{for } \rho_c^{\,n}(t) > 1 \text{ where } x = \rho_c^{\,n}(t) \\ &= R_{max} && \text{if } R_{av}^{a}(t+1) > R_{max} \end{aligned} \qquad (5.4)$$

where *Rew(t)* is taken to be 1 if the agent is able to service an RRS; else it is zero at time *t*. *Rew(t)* thus acts as a reward conferred on the agent interms of an increase in Cloning Resource as and when it services an RRS. τ_c and τ_r are non-zero positive constants. Care is taken to ensure that the cloning resource, $R_{av,}$ never exceeds its maximum value R_{max} and if so it is brought down to the latter. Every mobile agent or clone has, in addition to the cloning resource, a life-time conferred on it at the time it is created. The life-time $L(t)$ is decremented in every time step according to equation (5.5).

$$L(t+1) = L(t)-1 \qquad (5.5)$$

Life-times of the agents are increased by a factor as and when these agents earn rewards as given by equation (5.6).

$$L(t+1) = L(t)+ \sigma\, Rew(t) \qquad (5.6)$$

where σ is a non-zero positive integer constant.

5.5 Results and Discussions

The custom simulator developed for *PherCon* was modified and embedded with the clonal controller described herein. Accordingly a network of robotic nodes with respective intra-node queues and queue managers capable of hosting the mobile agents was simulated. The agent migration mechanism used was *PherCon* rather than *PherCon-C* since the cloning mechanism described herein makes localized cloning redundant.

Further though in *PherCon-C* the approach of creating clones and destroying them once the service is accomplished is a flexible approach, this idea does not adapt to sudden changes in the network dynamics and availability of resources.

In addition, this simulator ensures that simultaneous multiple mobile agent migrations do not occur in the same step-count just as in real networks [152]. Thus all the simulation results portrayed herein closely match the real world wherein when two entities (robotic nodes) transact with one another, a third entity desirous of a transaction with any of the former two, needs to wait for the current transaction to end as mentioned in section 4.6.1.

The simulations and results are presented in three scenarios using heterogeneous mobile agents (agents carrying distinct services) in the subsequent sections.

5.5.1 Scenario-1: Heterogeneous Mobile Agents with several RRSs requesting the same service [148]

The simulation was carried out using 200 nodes with an initial population of 10 mobile agents out of which one agent of type α carried the service requested by all the RRSs. The remaining nine agents designated of type β carried another service which was never requested for by any RRSs. Every run of the simulation comprised 400 step-counts. The values of the various constants used in the simulations were: $Q_{Th}=5$, $\tau_c=0.1$, $\tau_r=100$ and $\sigma=7$. For the parent agents, the initial values were taken to be $R_{av}= R_{max}=100$. The values of the

parameters pertaining to the clones were taken to be, $R_{av}=R_{min}=10$ and $L=10$. RRSs requesting for a service carried by the agent of type α were generated at various simulation steps and made to diffuse pheromones as per *PherCon*. Initially during the first 20 step-counts, twenty new RRSs requesting the service carried by the agent of type α were generated at the rate of almost one per step. Later, after a lull period, another burst of 20 RRSs from step-count 211 requesting the same service, was generated in a similar manner.

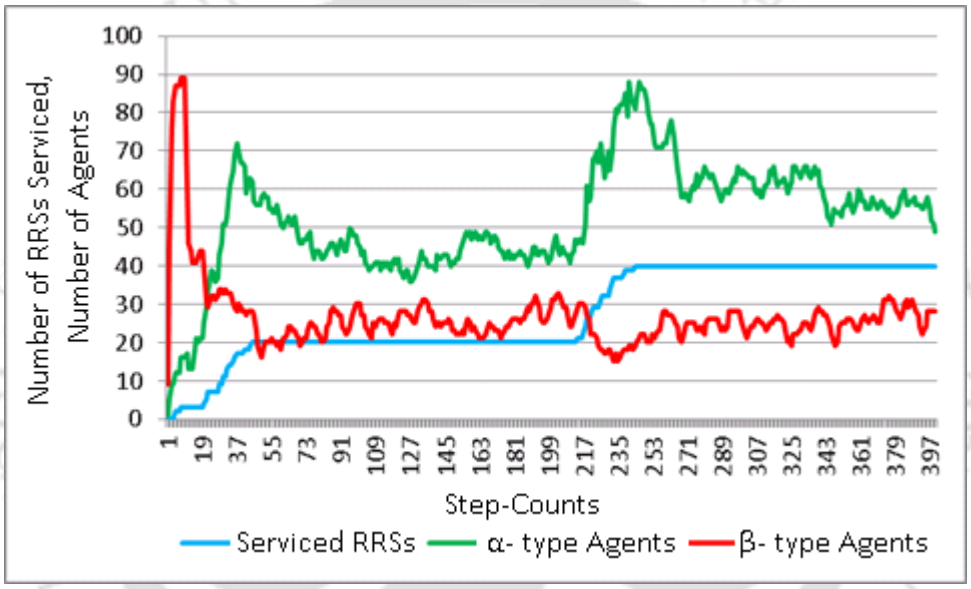


Fig. 5-4 Number of RRSs serviced (Cumulative), Number of α-type and β- type agents and their clones versus Step-Count – With Cloning Resource

Figure 5-4 depicts the graph showing the variation of the number of agents of type α and the number of agents of type β where a cloning resource is used to modulate cloning. Initially since all agents have the same (maximum) amount of cloning resource and most intra-node queues are almost empty, the clonal pressure on each of these agents is high causing an increased generation of clones. This explains the sudden initial surge in the number of agents of types α and β. However with the onset of RRS generation, the cloning resources of

agents of type α are charged by rewards attained in servicing these RRSs. This causes the agents of type α to clone further while the other agents of type β remain restricted from doing so as their clonal resources have depleted largely. With more RRSs being satiated, there is a corresponding increase in the rewards and hence clonal resource and life-times of the agents of type α which could provide the service. This increases the number of clones of Agents of type α thereby drastically reducing the waiting times of subsequently generated RRSs. The concept of a life-time conferred on the agents and clones helps in depleting them when they receive no rewards. Clones of type β thus die quickly making way for the increased population of the clones of type α. This avoids choke-up and further reduces RRS waiting times.

At step-count 243 all the RRSs are satisfied and beyond this since no rewards are obtained, the clones of type α too, dwindle and die out. The number of clones seem to oscillate mainly because agents/clones having large cloning resource, initially generate a proportionately large number of clones in a short spell of time and populate the queues thereby reducing clonal pressure proportionately. Since all these clones are born almost at the same time and are conferred the same amount of life-time, their deaths too take place almost concurrently. The number of these clones thus decreases rapidly, effectively increasing clonal pressure and causing further oscillations. It may also be noted that 44 step-counts were consumed to satisfy the first burst of 20 RRSs while only 32 step-counts were required to satisfy the latter burst. This exhibits an adaptive behaviour of selective increase in the rate of cloning on part of the agents of type α. This behaviour is due to the fact that the decreased number of agents of type β during and after the second burst facilitated more vacant intra-node queues thereby allowing agents of type α and its clones to multiply at a higher rate. Both clonal pressure and cloning resource thus play a vital role in embedding this adaptive behaviour onto the agents. By merely sensing the intra-node queue and the cloning resource boosted by rewards, the agents of type α

proliferate and spread across the network while agents of type β back off from the network to give more bandwidth and mobility to the former agents of type α.

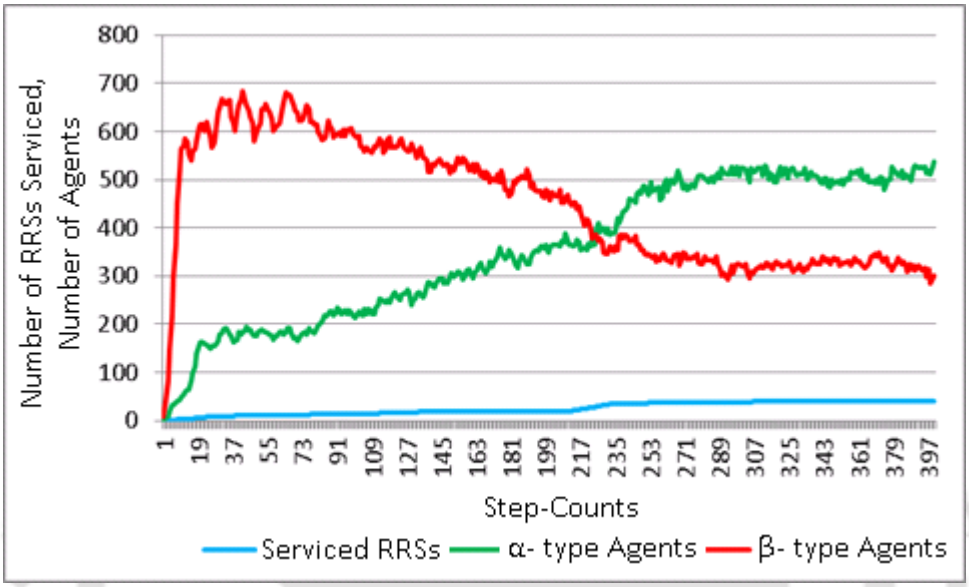


Fig. 5-5 Number of RRSs serviced (Cumulative), Number of α- type and β- type agents and their clones versus Step-Count – Without Cloning Resource

In order to understand whether the concept of resource actually plays a significant role we portray similar results when the resource is not taken into account. Here we assume that the factor (R_{av}/R_{max}) in equation (5.2) is unity. This means that the agents have a never-depleting or constant supply of resource and can thus clone irrespective of the resource and fill the queue. Figure 5-5 depicts the corresponding graph for such a case. It can be seen that the agents of type α take more time to service the RRSs. It is also difficult to exactly demarcate the end of servicing of the first burst of RRSs and that of the second. The first bursts of 20 RRSs were satisfied by the 136th step-count while the second burst was satisfied only in the 305th step-count. This is in contrast to the earlier step-counts namely 44 and 32 consumed in servicing the first and second bursts of the RRSs respectively, using the cloning resource. A comparison of the graphs depicted in

Figure 5-5 with Figure 5-4, reveals that the use of a cloning resource helps reduce RRS waiting times by freeing the bandwidth for the *in-demand* agents and their clones. It can also be observed that the total number of agents populating the network in the *without-cloning-resource* case is consistently far higher than that in Figure 5-4 where a cloning resource is used to modulate the cloning rate. Figure 5-5 also shows only a gradual increase in the population of the *more-in-demand* clones of type α and a proportional and gradual decrease in the population of clones of type β. This is in contrast to what is observed Figure 5-4 wherein the number of *in-demand* agents of type α increases sharply while that of type β decreases in a similar manner.

Initially in the *without-cloning-resource* case most of the intra-node queues are mainly occupied by the nine β type parent agents and their clones. This filling up of intra-node queues by such not-on-demand agents of type β inhibits the cloning and migration of the in-demand agents of type α. As the clones of type β gradually die they give way to agents of type α and clones which gain additional lifetimes from rewards obtained by servicing RRSs. This accounts for the gradual increase of α type agents and the gradual decrease of the β type agents.

5.5.2 Scenario-2: Heterogeneous Agents with RRSs requesting different services

This simulation was carried out using 200 nodes with an initial population of eight heterogeneous parent agents designated *Agent-0* through *Agent-7* where *Agent-0* carries *Service-0*, *Agent-1* carries *Service-1* and so on. The values of the various constants used in the simulations were: $Q_{Th}=4$, $\tau_c=0.1$, $\tau_r=100$ and $\sigma=7$. For the parent agents, the initial values were taken to be $R_{av}= R_{max}=100$. The values of the parameters pertaining to the clones were taken to be , $R_{av}=R_{min}=10$ and $L=10$. A set of 194 RRSs requesting for *Service-0* through *Service-7,* were generated at discrete simulation steps in a random manner. The RRSs diffused pheromones for the requested service across their neighbours as in *PherCon.*

One simulation run, each comprising 1000 step-counts was carried out separately in the absence and presence of the cloning resource. Results were obtained by observing the maximum number of mobile agents generated and the total number of step-counts required to satisfy all the 194 RRSs.

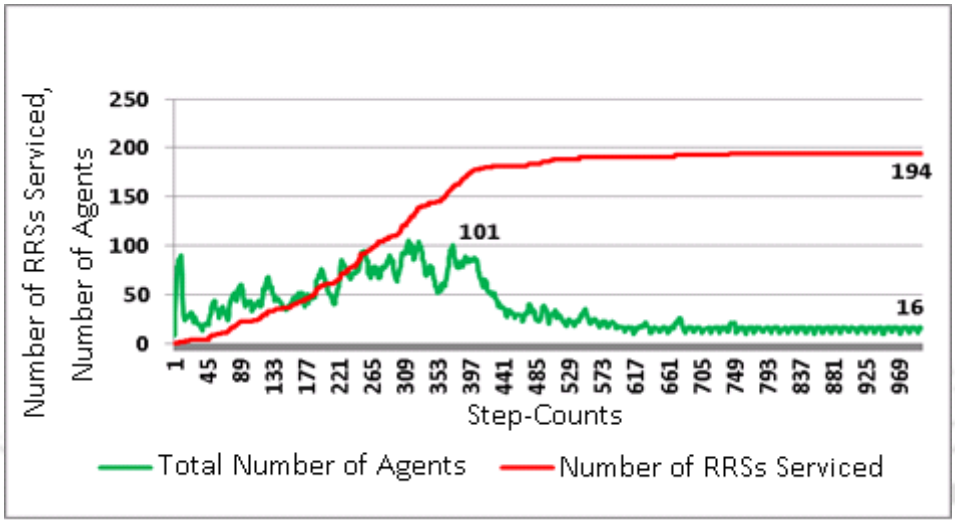


Fig. 5-6 The variation of the total Number of Agents and the RRS service times – With Cloning Resource

The graph in Figure 5-6 depicts the performance of the cloning controller using the concept of the cloning resource. It can be seen from this graph that by using just 101 agents, all 194 RRSs were satisfied in 743 step-counts. This graph also shows a sharp increase in the number agents proportional to the RRSs generated within the network followed by its decrease to a very low value after most of the RRSs are serviced. The increase and decrease conform to the rate at which the RRSs are generated and serviced thus exhibiting an adaptive behaviour on part of the cloning controller.

The graph in Figure 5-7 shows a plot of the variation of the total number of agents and the number of RRSs serviced with the simulation step-counts when the cloning resource feature was disabled. It can be observed from this graph that

only 191 out of 194 RRSs were satisfied in 1000 step-counts using a total agent population of around 700 agents in the network. The population of mobile agents seems to continue to hover around 700 even though no more RRSs were generated and a few were still waiting to be serviced, thus needlessly consuming precious bandwidth and resources.

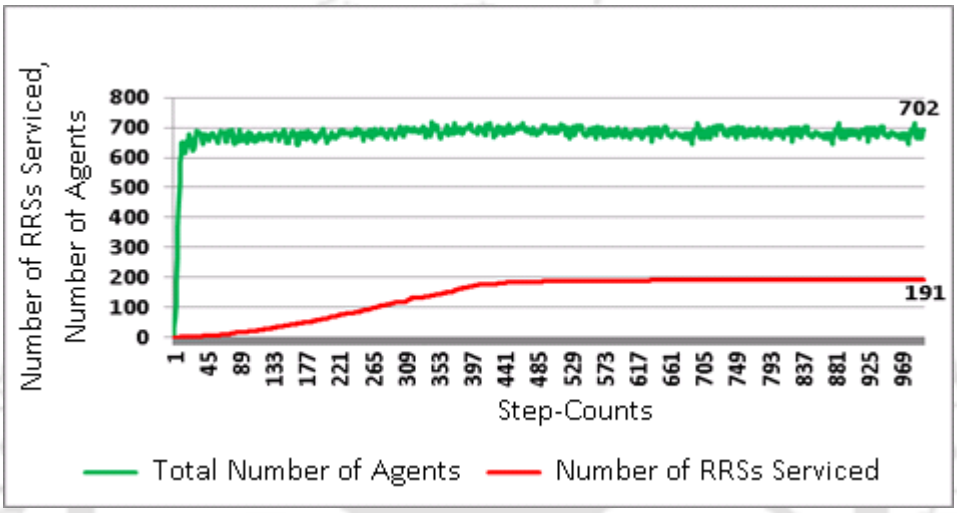


Fig. 5-7 The variation of the total Number of Agents and the RRS service times – Without Cloning Resource

Further the number of agents populating the network when almost no RRSs exist is about 700 in contrast to just 15 when the cloning resource was used. Thus the *with-cloning-controller* case exhibits resource sensitive behaviour. Such behaviour saves both precious computational resource and bandwidth which could be utilized for other purposes or by entities within the network. The heterogeneous mobile agents thus clone based on their own decision but also ensure that they do not grow too large in number so as to curb the migrations of the other types of agents.

5.5.3 Scenario-3: Primary and Secondary Responses

In order to test the adaptiveness of the controller, simulations were also

conducted with a total of 292 RRSs generated over 1000 step-counts. Other parameters were kept the same as scenario 2. On an average around 30 RRSs requested for the same type of service. RRSs requesting a certain type of service were generated in an interleaved fashion every 100 step-counts. In addition 27 RRSs once again requesting *Service-0* carried by *Agent-0* were generated starting from step-count 250 onwards.

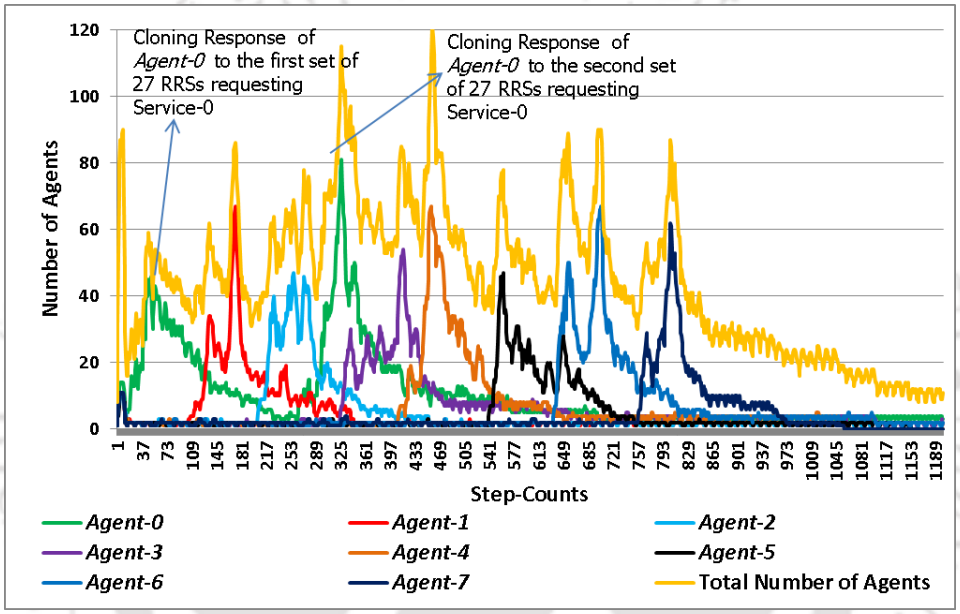


Fig. 5-8 Graph showing the increase and decrease in the number of each type of Agent – *Agent-0* through *Agent-7*

Figure 5-8 shows the nature by which the number of agents of each type increases to a maximum and then go down to a minimum after the corresponding RRSs are satisfied. It can also be seen that the rate of generation of the clones of *Agent-0* in the second phase after the 250th step-count is much faster than that in the initial stage when the first set of RRSs requesting services of *Agent-0* were activated. The initial increase in number of *Agent-0* to around 45 is analogous to a *primary response* exhibited by an immune system [78]. As an agent services

more RRSs it gains both cloning resource and life-time due to the rewards it accrues in doing so. Increased life-times make some of these agents to live longer. When the second burst of RRSs occurred, the population of *Agent-0* and its clones had not died down to the minimum. This facilitated the generation of a larger number of clones in a shorter period of time causing the steeper peak. This is similar to a *secondary* and *adaptive* response exhibited by a typical immune system [153]. In a biological immune system, the initial attack of an antigen causes the body to generate antibodies whose concentration increases till finally the antigens are vanquished. A few of the best antibodies turn into memory cells and patrol the body for the rest of the being's life. On being attacked by the same antigen later, these memory cells replicate to quickly mobilize an attack that is far faster than the primary one. Secondary responses are faster as the body already possesses the antibodies required to tackle a particular antigen. The clonal controller thus seems to exhibit such responses.

The graph also shows how the overall population of heterogeneous agents (M x N) varies. It can be seen that this value is contained below 120 and goes down rapidly when no RRSs populate the network indicating clearly that the heterogeneous mobile agent population controls itself selectively and on-demand. All unnecessary clones are automatically flushed once all the RRSs have been serviced. The existing parent agents (*Agent-0* through *Agent-7*) then continue to patrol the network in a conscientious manner thereby freeing most of the bandwidth for other network related activity.

5.5.4 Effect of Queue Length

The intra-node queue length Q_{Th} has a direct impact on the number of agents and clones generated within the network as also the amount of time taken to service the RRSs.

Figure 5-9 shows the effect of the variation of the intra-node queue length, Q_{Th}, on the number of agents and clones generated and the step-counts

taken to service a constant number of RRSs (100 in this case) using the cloning resource. It can be observed from the graph that at low Q_{Th} values the total number of agents and clones generated is also low thereby taking more steps to service all the 100 RRSs. As the Q_{Th} value increases the performance increases due to an increase in number of agents and clones generated. However further increase in Q_{Th} beyond 4, does not portray any drastic change in performance. The number of steps consumed in each of the cases beyond Q_{Th}>=5 seems to be

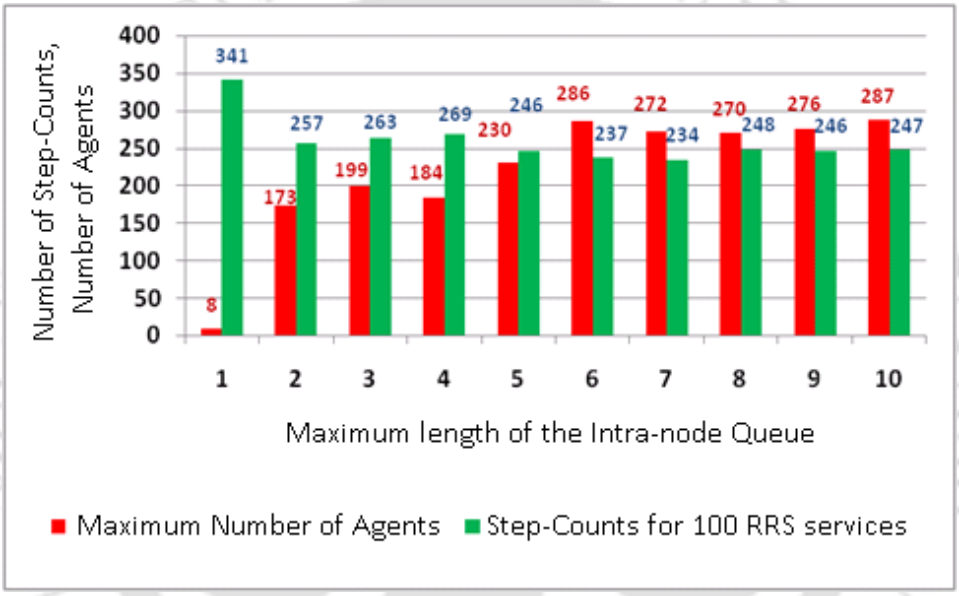


Fig. 5-9 Variation in the number of Mobile Agents and Clones and Step-Counts to service 100 RRSs for different values of Q_{Th}

restricted to a narrow band just below 250. It may thus be inferred that even though with higher values of Q_{Th} the agents or their clones find more empty spaces in the queue they refrain from needless over-cloning. Such a judicious cloning coupled with minimal service times of the RRSs, can provide more bandwidth to other entities that populate the network. In real networks, inter-node communications and message passing will also require bandwidth. Since the cloning controller consumes only a part of the bandwidth and that too only

on-demand it ensures that a fair amount of bandwidth is always available for the other communications. Thus choke-ups will rarely occur in such networks where this cloning controller is used.

5.6 Performance in Network of Mobile Robotic Nodes

Since the underlying application was aimed towards controlling the population of clones in networked robot systems as cited in section 2.5 of Chapter 2 [139], it is important to study the performance of the same when the robotic nodes hosting the mobile agents move relative to one another. Simulations were carried out for such a dynamic scenario where the robotic nodes moved in a random fashion. The results were compared with those obtained from a static scenario. The parameters used in the underlying *PherCon* mobile agent migration mechanism were the same as that mentioned in section 4.6.2 of Chapter 4. Seven simulation runs each comprising 400 step-counts were carried out and the maximum number of agents and clones generated and the number of step-counts required to satisfy 100 RRSs were found.

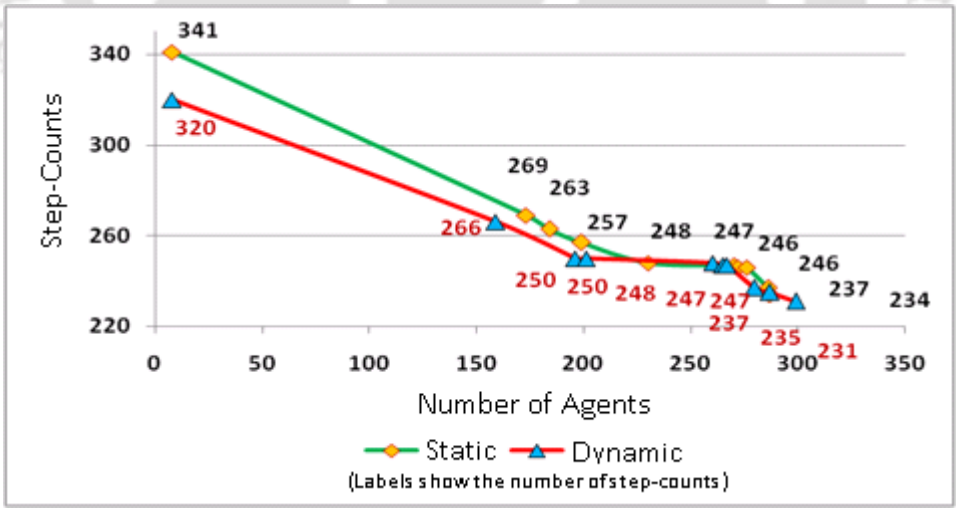


Fig. 5-10 Comparison of the performance of the Cloning Controller when used in Static and Dynamic Networks

Figure 5-10 shows the variation in the performances for servicing 100 RRSs for both the static and dynamic cases. Both the graphs seem to follow the same trend. For lower number of agents and clones, the step-counts required for the service to be completed are high. As the number of these agents and clones increases the step-counts decrease and hover around 230-250, similar to the static case thus indicating clearly that the cloning controller works almost the same way in both static and dynamic scenarios.

5.7 Conclusions

Most of the state of the art population control mechanisms are directed towards controlling the cloning of homogenous agents. Controlling a clone population in heterogeneous mobile agent based scenarios is non-trivial. Described herein is a novel demand based adaptive mechanism for controlling the population of heterogeneous mobile agents using *stigmergic* sensing of the available bandwidth within a network. *Stigmergic* sensing becomes essential in the real world since it is not possible for each mobile agent to communicate with all the others to compute the true global information on its population. This bio-inspired sensing mechanism, based on the number of mobile agents queued up for migration, coupled with a cloning resource charged by rewards accrued by servicing RRSs, facilitates effective utilization of the available network resources and bandwidth. Cloning resource and clone life-times contribute greatly to a faster and adaptive response so that only the on-demand agents and their clones form the majority of agents that populate the network. The proposed cloning controller mechanism also exhibits a faster secondary response which is akin to that in the realm of immune systems. This contributes to a faster service for all repeated requests (antigens) generated for the same type of service.

In a real system, networked robots or nodes may require extra bandwidth for communicating a variety of information to other robots or servers by means of message passing, flooding, etc. By not cloning to the limit of the available

bandwidth which could result in an eventual cluttering of the network, the mechanism described herein, uses a *stigmergic* technique to allow both mobile agents and other information to flow efficiently within the network. This work also explains the robustness of the mechanism in dynamic scenarios.

The faster secondary response exhibited indicates that the cloning control mechanism can be further enhanced using concepts from the Biological Immune System so that a highly rewarded agent could become an *Immune memory* [78] with life-time comparable with that of its original parent. This will ensure that further responses to the requests made for the service carried by such an agent will be much faster.

In the succeeding chapter we provide details of the design of the simulator that was used for studying the mobile agent migration mechanisms and the mobile agent population control mechanism discussed so far.

Chapter 6
Design of Simulators for the Mobile Agent based Networked Robotics System

In the initial testing phase of any system, simulations offer an edge over real experimentation. They are low-cost, visual, fast, scalable and can be reconfigured to suit new experiments. Simulations play an important role in academia and industry as a necessary research tool to analyze and justify proposed theoretical models. There has not been any remarkable attempt to simulate mobile agents for networked robots. Although Webots© [154] and CARMEN [155] are powerful tools for simulation of multi-robot system applications, mobile agent middleware for multi-robot or networked applications such as described in [73] and [74] have not been transferred to the simulation environment.

In order to study the performance of agent migration and cloning mechanisms developing a simulator for simulating mobile agent based networked robotic scenarios thus became imperative. This chapter reports the efforts taken in developing a simulator that allows mobile agents to move around a network of nodes, service them and clone on demand. The nodes can generate requests for service and hence are synonymous to robots. They can also host mobile agents and provide for all related computations thus simulating the quintessential middleware. The simulator thus supports the architecture depicted in Figure 2-3 of Chapter 2. Simulations carried out could thus be viewed to be those for a mobile agent based networked robotic system. The simulator provides the number of step-counts taken to reach or achieve the desired goal, the number of hops taken as also the intra-node computations and inter-node

communications – all of which can be used to judge the performance of the mechanisms being simulated. The performance measurements of the existing and proposed mobile agent related mechanisms described in earlier chapters of this thesis were found using this simulator.

6.1 Salient Features of the Simulator

The simulator developed is basically a Java application that can simulate mobile agents and mobile robotic nodes. It can simulate all the seven migration mechanisms (viz. Random, Conscientious, EVAP, CLInG, G-B, *PherCon* and *PherCon-C*) and also the cloning controller described in earlier chapters. Agent behaviours and its mobility along with robotic node movement relative to one another can be simulated through this application. The salient features of this simulator are described below.

1. Robotic Nodes: Each node in the simulated network performs all actions pertaining to a robot. These include requesting for services (acting as an RRS), pheromone management and discovering its one-hop neighbours in dynamic environments. The robotic node also doubles to act as the middleware (explained in Section 2.4 of Chapter 2) and hosts the mobile agents and performs the relevant agent related tasks.

2. Mobile Agent Support: The simulator allows for mobile agent to migrate and clone. These agents and clones may also be removed based on specified conditions.

3. Inter-entity communications: The simulator supports inter-entity communications that include – Agent-Agent, Agent-Robotic node and Robotic node-Robotic node interactions.

4. Network Topology: Simulations can be carried out for custom network topologies. The simulator need only be given the coordinates of the robotic nodes through an input file. This facility comes handy when one needs to test a

variety of topologies and situations. Other inputs specifying the exact step-count when a robotic node should become an RRS can also be provided. Each node has a communication radius which is used to decide which nodes can form the links and communicate. Two nodes are linked only if they fall within each other's communication radius.

The simulation environment supports both static and dynamic scenarios. The topology of the network in a static scenario is fixed and is based on the initial positions of the robotic nodes. In dynamic scenarios, the underlying topology can be made to vary in a random manner during the simulation run. The nodes act as mobile robots thus simulating a Mobile Ad hoc Network of Robots (MANER) [112]. Their interconnecting links with other nodes are formed and broken as they move.

5. Logging: All results available on the simulation console as also the intermediate states and data pertaining to the entities of the network can be recorded and saved onto log files for off-line analysis.

6. Rendering: A visual presentation of the simulation is provided by the simulator. The visualization helps in quickly observing the behaviour and migration of the mobile agents and robotic nodes. In single-stepping mode the simulator allows the user to trace the manner in which both the agents and the nodes perform. A screenshot of the rendered simulation window is shown in Figure 6-1 has a total of 200 nodes. The white squares within denote robotic nodes that can host the mobile agents while the red squares denote the RRSs. The yellow spots on the white squares denote the mobile agents which migrate to their neighbours using one of the chosen migration strategies. Faint circles around each node denote the wireless communication range of that node. The green segments between the nodes indicate that they can communicate with each other. The yellow colored links indicate the pheromones diffused across he neighbours. In a dynamic scenario the nodes can be seen to move in random directions.

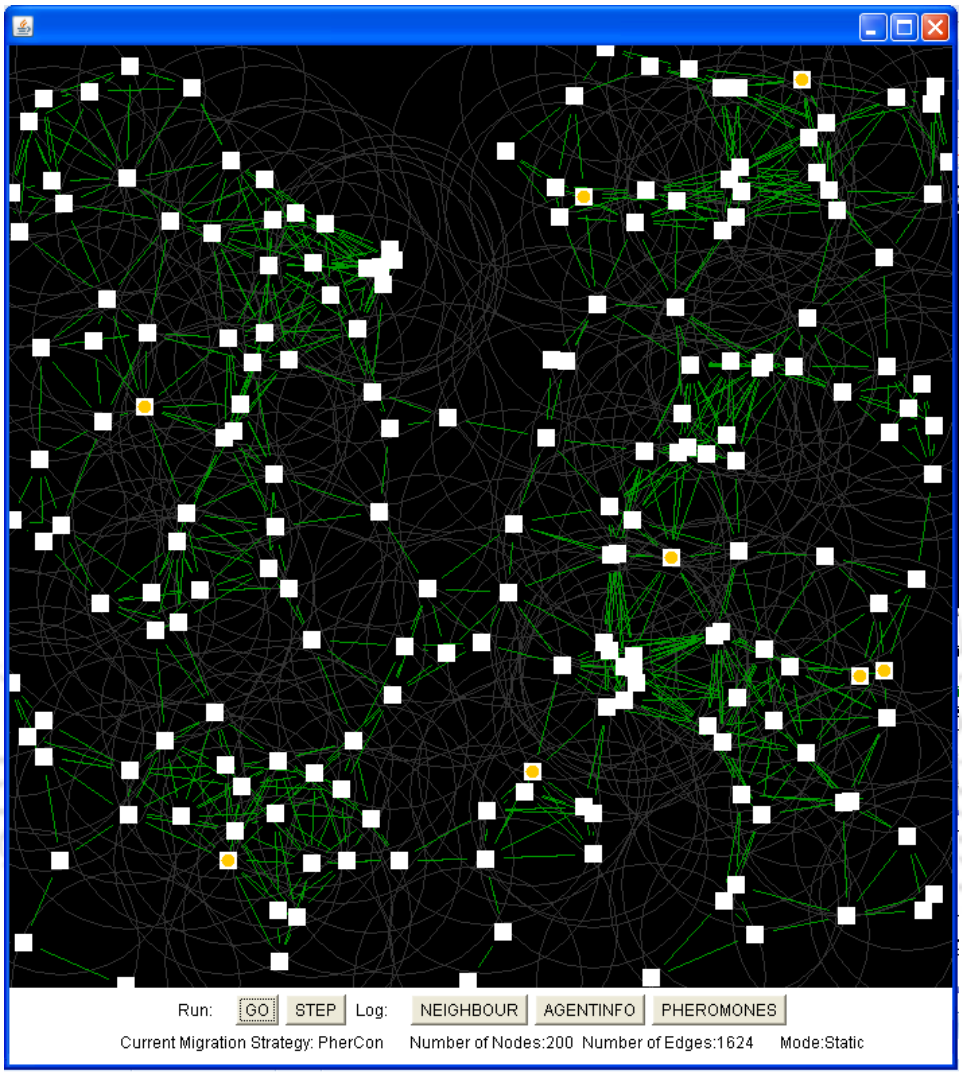


Fig. 6-1 Screenshot of the window rendered by the simulator

The GO button starts the simulation and stops only when the termination condition, such as servicing a pre-determined number of RRSs, is reached. The STEP button allows for stepwise execution. The buttons designated NEIGHBOUR, AGENTINFO and PHEROMONES facilitate the storing of the pertinent information in specific files. The simulator also offers status

information such as the current migration strategy of the mobile agents, Number of Nodes and Edges, and the Mode of simulation.

6.2 System Requirements

The simulator can be run on either Windows or Linux based computing systems provided Java Development Kit (JDK) version 1.7.x and JAVA 2D Graphics visualization tool are installed.

6.3 Conclusions

This chapter provided a brief description on the simulator developed for the purpose of testing the proposed bio-inspired mechanisms. It also highlighted the salient features of the simulator. It was observed that the simulator could be used to test algorithms used in many other domains of networks. This simulator could prove to be a useful tool in the study of algorithms related to a myriad of areas, with minimal modifications. These areas include –Resource discovery in Ad-hoc networks, Sensor network protocols and Network management using mobile agents. Though simulations play a vital role in proving the efficacy of algorithms and mechanisms, the underlying assumptions that form the basis of simulation can make it portray results that need not hold good in the real world. This is especially true in the realm of robotics where closed world simulated solutions cannot be directly thought of to be true. It was thus necessary to carry out real world implementations of the proposed algorithm to prove its efficacy and viability. The succeeding chapters portray the attempts made towards the realization of a physical working model of a robotic network that uses the proposed *PherCon-C* mechanism. It also describes how the very same mechanism could provide and support the existence of an *Internet of Things*.

Chapter 7

Implementations of Mobile Agent based Service Mechanisms for Networked Robotics

Though simulations carried out, do prove the efficacy of the proposed mechanisms, it was thought that implementing the same would bring to light fallacies in any of the possible assumptions commonly associated with simulation. For instance, Satoh [156] describes a test-bed for a bio-inspired distributed system and points out that the results obtained using simulation and actual implementation of ant-based routing mechanisms were different. Further, actual implementations always pose challenges in the real world and also provide a fresh set of practical ideas and innovations.

This chapter describes an implementation of the mobile agent based networked robotics system using a LAN and Lego NXT® [157] robots and associated sensors. The agent migration mechanisms used in the same were – Random, Conscientious and *PherCon-C*. The other mechanisms viz. EVAP, CLInG and G-B were not implemented for reasons cited in Chapter 4. As mentioned therein, their use in conjunction with a heterogeneous set of mobile agents may use up excessive bandwidth forcing the system to come to a standstill. The cloning controller proposed in Chapter 5 could not be implemented due to lack of infrastructure. The controller's real forte is exhibited only under high bandwidth utilization scenarios when the number of nodes, agents and their clones is high. A full-fledged stand-alone and dedicated network whose parameters including the bandwidth could be easily altered on demand would be required to do the same.

7.1 Mobile Agent based Networked Robots

This network was realized on a LAN that had several desktop computing systems that formed the nodes of the network. Each node had a mandatory and an optional software platform installed on it.

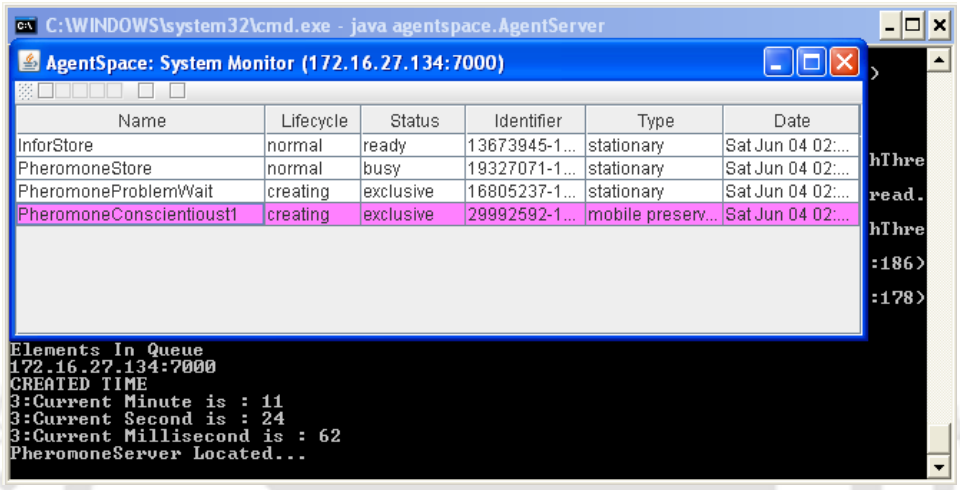


Fig. 7-1 The AgentSpace [158], [159] Mobile Agent framework running on a host displaying the names of Static agents and a Mobile Agent (Latter is highlighted)

The mandatory part forms the middleware that supports all mobile agent activities and functionalities of the network. In this implementation the AgentSpace [158], [159] mobile agent platform was used to cater to this middleware. AgentSpace is a mobile agent framework that supports creation of active networks. This facilitates implementing protocols within mobile agents which can be dynamically deployed at remote nodes. It allows for programming mobile agents and supports functionalities such as cloning, caching in addition to mobility. Figure 7-1 shows the snapshot of the AgentSpace mobile agent framework with the static and mobile agents running on a desktop computer acting as a node. The optional part of the software constitutes the robot controller which is responsible for all node-robot communications and

interactions. This software installed in the node hosting the robot is mainly used to download a program (code or service) from the node onto the robot so as to enable its execution. The robot side has only a program that continuously listens to the node, accepts the code and executes it. The robots did not have any other program for executing any task initially. Not all nodes had robots connected to them making this software installation optional. The robots used were Lego NXT® running LeJOS [160] on-board the robot. Communication between the nodes and the robots was achieved using Bluetooth®. The robots could be assigned a set of tasks via the consoles of their respective nodes. The two cases implemented have been described below.

Case 1: Robots requesting services using *PherCon-C*

In order to check whether *PherCon-C* is actually implementable in a real network a simple experiment was carried out. In this case, a total of six nodes hosting the mobile agent platforms are deployed forming a topology as shown in Figure 7-2. The topology was enforced in *soft* manner by maintaining a neighbor list at each node. A snapshot of the figure is shown in Figure 7-3. Two robots, *R-1* and *R-2* with two onboard sensors namely light, ultrasonic were attached to the nodes, *Node-1* and *Node–6*. If the value reported by the light sensor is greater than 40 and that of the ultrasonic sensor is less than 20, then the robot is assumed to be in a stable position. The program within the robot is capable of merely sensing whether it is in the stable state. The robot therefore does not know how to get back to its stable state when there is a change in the values reported by these sensors. Thus when the robot is in an unstable state, it sends a request to its associated node along with the current state (sensor values) of the robot. The node in turn starts pheromoning to attract those agents with programs which when executed by the robot will guide it to a stable state. Three different programs *x*, *y* and *z* for the robot to recover from an unstable position were written and injected into the payloads of three heterogeneous agents, designated as A_x, B_y and C_z respectively. These agents were released from *Node-4* as shown in the Figure 7-2.

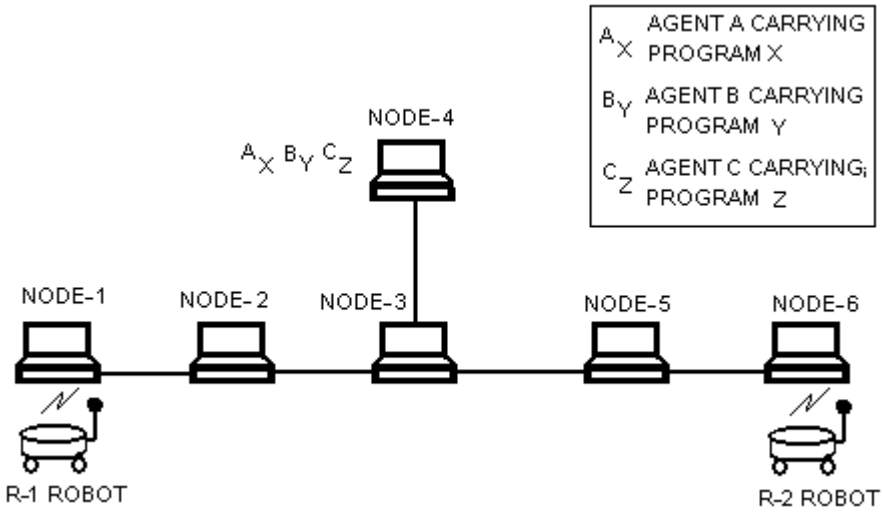


Fig. 7-2 Topology of the Network with Robots, *R-1* and *R-2* connected to *Node-1* and *Node-6* and Mobile agents A_x, B_y and C_z carrying programs *x*, *y* and z respectively starting from *Node-4*

The mobile agents and the pheromones function as per *PherCon-C* to attract the concerned mobile agent towards the robotic node. A robot can be in any one of the four states, 00, 01, 10 or 11 as shown in Table 7.1, based on the values reported by its two sensors. Table 7.2 lists the state changes that occurred for both *R-1* and *R-2* together with the programs that were brought by the agents to the respective nodes and executed.

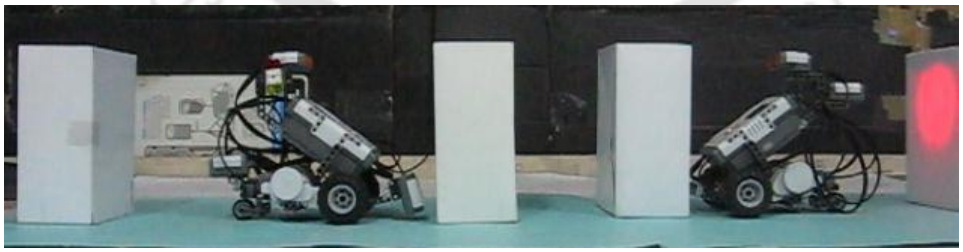


Fig. 7-3 Snapshot showing the side view of the experimental setup with robots requesting for corresponding programs based on sensor values (Case-I)

Table 7.1 Robot states indicated by their sensor values with the corresponding program requested by the robot

State (Stable-S Unstable-U)	Light Sensor Range	Ultrasonic sensor Range	Program-(Actions)
00 (S)	41-100	0-20	No Action
01(U)	41-100	21-100	*x*-(Move 10 steps Forward)
10(U)	0-40	0-20	*y*-(Turn Right)
11(U)	0-40	21-100	*z*-(Turn Right and Move 5 steps Forward)

Table 7.2 Sequence of state changes for robots, *R-1* and *R-2* along with the corresponding outcomes

Sl. No	Robot R-1	Robot R-2	Remarks
1	00	00	No pheromoning
2	01	10	Pheromone for programs *x* and *y*
3	11	11	Cloning of agents carrying program *z*
4	10	01	Pheromone for programs *y* and *z*
5	00	00	No pheromoning, stable state
6	01	10	No pheromoning, programs already available

It can be seen that the initial transitions to new unstable states such as in sl.no 2, 3 and 4 for robot *R-1* force it to become an RRS. Since the relevant programs for recovery are not locally available, the concerned node therefore pheromones for the relevant programs - *x*, *z* and *y* in that order which are eventually served by the agents A_x, C_z and B_y to *R-1* robot viz. *Node-1* to lead it

back to the stable state. Further transitions made by *R-1* to unstable states (serial numbers 5 and 6) do not call for pheromoning since the concerned recovery programs are available locally.

Localised cloning was also observed when both the robots *R-1* and *R-2* were in the same unstable state initially as in serial number 3 and requested for the same program *z*.

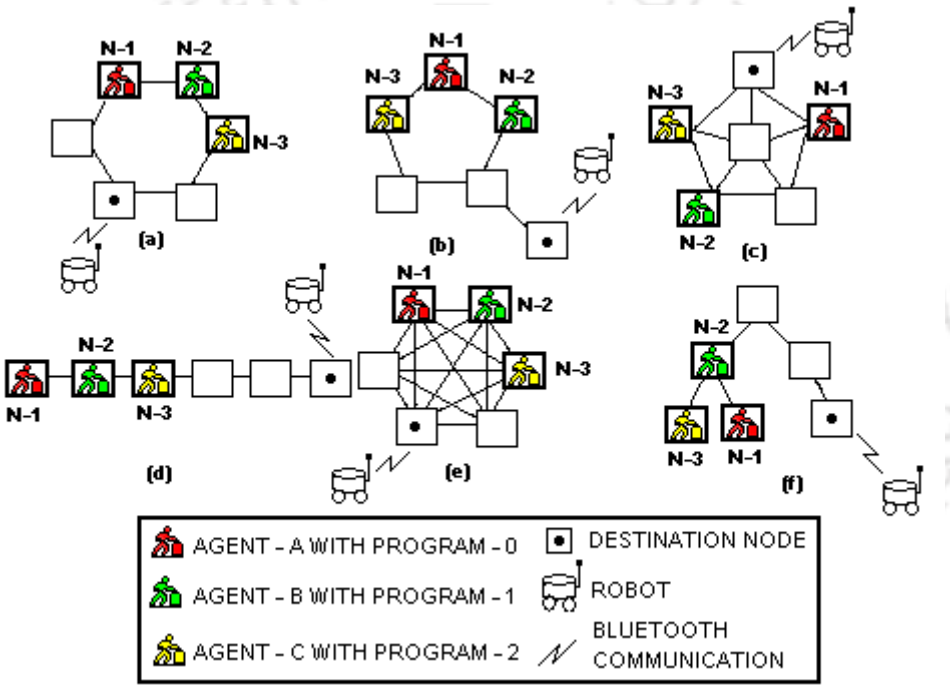


Fig. 7-4 Initial placement of agents *Agent-A*, *Agent-B*, and *Agent-C* carrying programs *Program-0*, *Program-1*, and *Program-2* respectively residing at Node Locations, *N-1*, *N-2* and *N-3* along with the destination robot with the nodes forming (a) Ring Topology (b) Mesh Topology (c) Star Topology (d) Line Topology (e) Connected Topology and (f) Tree Topology

Case-II: Comparing the Implementable Migration Algorithms

In section 4.6.1 of Chapter 4, the problems faced in the implementation of EVAP, CLInG and G-B mechanisms were discussed. Thus in this

implementation only the remaining migration mechanisms viz. Random, Conscientious and *PherCon-C* were used.

Three heterogeneous mobile agents carrying three different programs (services) designated *Program-0: Search for a black line in the vicinity*, *Program-1: Follow the black line* and *Program-2: Stop and beep* needed to reach a node hosting a robot (RRS) were initially positioned at specific nodes, *N-1*, *N-2* and *N-3* in the topologies depicted in Figure 7-4 (a) through (f). The neighbour lists at the various nodes were altered to reflect these topologies. A snapshot of the robotic setup is shown in Figure 7-5.

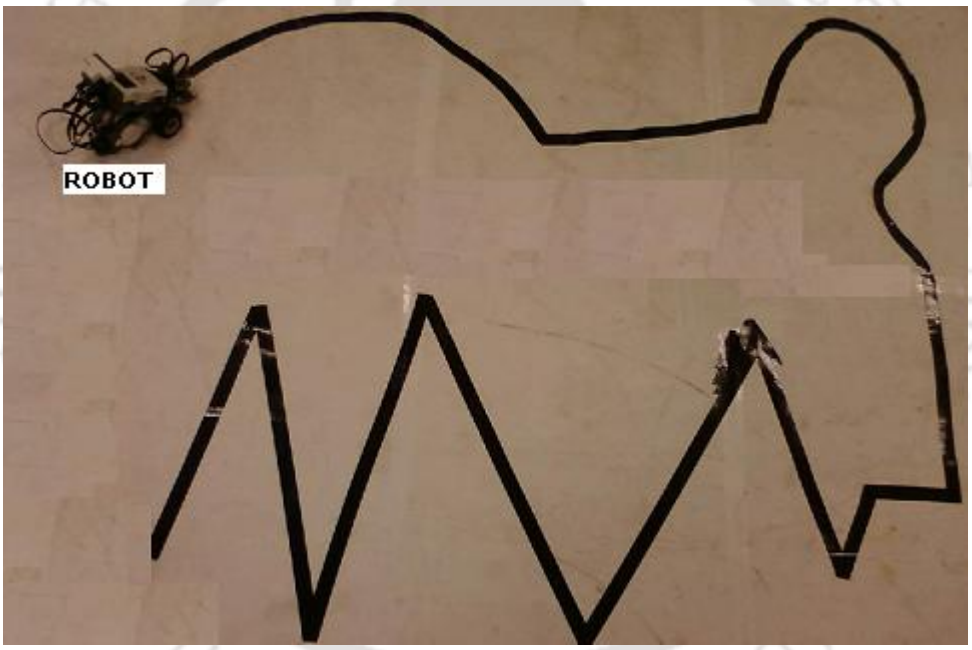


Fig. 7-5 Aerial View of the Experimental Setup (Case-II)

The robot was made to pheromone for *Program-0*, *Program-1* and *Program-2* sequentially and the number of hops required by the three agents carrying these programs to reach this RRS was found using each of the three migration mechanisms (viz. random, conscientious and *PherCon-C*). The graph

shown in Figure 7-6 indicates the hopcounts taken by the agents using each of these three migration mechanisms for the six topologies depicted in Figure 7-4 (a)-(f). It is clear from the graph that *PherCon-C* outperforms the other two mechanisms.

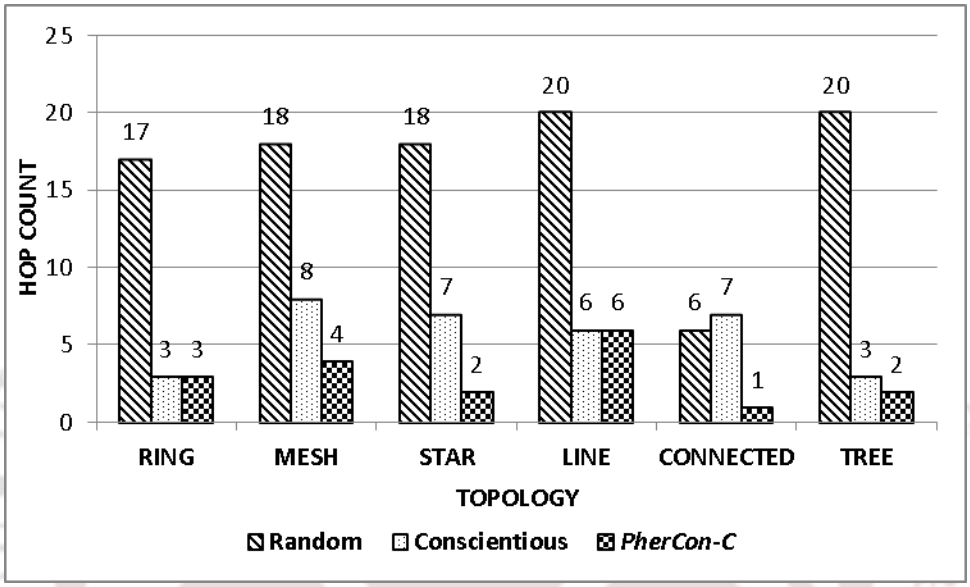


Fig. 7-6 Graph depicting the maximum of the number of hops taken by three agents carrying Program-0, Program-1 and Program-2 to reach the destination robot

In the ring and line topology, both the conscientious and *PherCon-C* perform similarly. It is so in the ring topology because both seem to use a round-robin mechanism. In the ring topology, if the robot connected to destination node were to make it pheromone (using *PherCon-C*) to attract a certain agent which has just migrated from destination to one of its neighbours, then it could make the agent backtrack to node-6 and service it. Such a scenario could occur even in the line topology. This would not happen in case the conscientious mechanism was used. Since such a situation possibly did not occur in the ring and line topologies, both Conscientious and *PherCon-C* seem to be at par in performance.

These results substantiate the work reported in [139], [161] and this [162].

7.2 Conclusions

Real implementations of *PherCon-C* on a network comprising two robots and three heterogeneous agents provides for an effective and easy way of deploying robots. As has been observed, the initial deployment of the robot did not require much of programming effort. The programs required by the robot were supplied to it on-demand in an autonomous manner without human intervention. It is assumed here that the programs carried as payloads by the agents are written by other robot programmers. The mobile agents carrying these varied programs constitute an effective way of sharing programs and hence expertise. From the experimentations carried out, it is clear that *PherCon-C* outperforms other algorithms in terms of the time required to effect a service. Appendix-A provides a link to download the video of *PherCon-C* in action.

PherCon-C can also act as a mechanism for migration of mobile agents in a network of things comprising both robots and devices. The term *things* could refer to any device that could be hooked on to a node in the network or by itself act as one. High-end mobile phones, network printers, sensor nodes, network-ready appliances including dispensing machines, etc. could be taken as examples of *things*. Going by this argument realizing an *Internet of Things* (*IoT*) could also be envisaged using the architecture suggested in this work. The *things* being network compatible could act as RRSs and diffuse pheromones when they require a service. Using the *PherCon-C* mechanism, agents populating the network could be attracted towards these *things* and effect the required service. Further these *things* could also communicate and pass information amongst one another and co-operate to accomplish tasks across the network. *PherCon-C* implemented on a network of *Thing*s thus opens a plethora of applications. To emphasize its versatility a simple *Internet of Things* was realized. This forms the basis of the next chapter.

Chapter 8
A Model for the *Internet of Things*

The actual implementation of *PherCon-C* threw more insights into the possible scenarios where the mechanism could be used. It was found that a network of robots could also be populated with other devices or *things* that could work in conjunction with one another as also the robots. With robots acting as actuators and the other *things* that could include mobile phones, network compatible appliances, printers, sensor nodes, etc., a heterogeneous network of *entities* could co-operate and form an intelligent system. The *NB* described in Chapter 2 could now be empowered with more capabilities. This chapter provides a brief introduction to the *Internet of Things*, challenges and related issues and proposes an architecture for both the *thing* and the *Internet of Things*. It also describes an implementation of a simple yet profoundly useful *Internet of Things* application scenario. The chapter thus throws more light on the versatility of *PherCon-C* which in turn could open up a plethora of *Internet of Things* based applications.

8.1 Introduction

There has been an exponential growth in the development of embedded gadgets and devices most of which have both networking and high processing capabilities. These include the vast gamut of mobile phones, tablets PCs and networked appliances. All such gadgets have been mostly configured as a simple network [163] or used as singular entities which try and connect to a server as and when needed. These devices have been rarely used even as a simple network and if so require custom software. The hardware involved too needs to be compatible in many ways. The need of the day is to provide a standardization

that will help extract the best out of such numerous processing units that could form a highly dynamic ad-hoc network. A mechanism for the same can aid in the realization of an ever increasing number of such devices forming nodes of a distributed yet co-operative processing system. With many such devices forming a network with changing topologies, each empowered by a means to share and gain information from one another, it is easy to visualize the formation of an *Internet of Things* [79]. In recent times, research communities have made efforts to create such a network of communicating and uniquely identifiable devices so as to develop sophisticated applications [164], [165], [166].

Further the embedding of information sharing and learning capabilities can realize an Intelligent *Internet of Things*. Of late new concepts in networking have provided greatly towards ubiquitous environments [167]. The ubiquitous cell phones can now be used for controlling appliances such as air-conditioners. There is yet a need to standardize such utilities to allow the same to be extended to a range of appliances. Imagine a multi-hop network wherein the nodes are mobile devices such as mobile phones. In addition, as its nodes designated. In addition the network could also be populated by various networked appliances such as smart refrigerators, vacuum cleaners, garage doors or even robots and printers. In a co-operative and intelligent set-up each of these devices will need to not only communicate but also service the other. If M_1, M_2,…,M_m designate the mobile phones and A_1, A_2,…,A_n are the network appliances, one may imagine a scenario where a robot A_1 may require the services of a vacuum cleaner A_5. Normally if the two devices are out of each other's wireless communication range, such a service will never be realized. The network of nodes M_m can now act as intermediary nodes to facilitate a path and allow the request messages as also the services to migrate from the networked robot A_1 to the networked vacuum cleaner A_5. With navigation and position information provided by the robot to the vacuum cleaner, the two can soon come within their respective wireless range and effect the required service.

The major challenge in making such scenarios into reality is the heterogeneity of the devices forming the network, the interfaces and protocols used, the lack of an architectural framework for devices to perform cooperative or coordinated tasks and the absence of a competent middleware to support device to device integration. Scalability and communication are other areas of concern in the vision of developing an *Internet of Things*.

Intel has been a forerunner in identifying the potential of this area and has heavily invested in its research. With sensory devices on the verge of becoming ubiquitous and with a view to take on research in *Internet of Things*, Intel has released its low-power Atom processor for mobile devices [168]. This processor supports basic computing as well as network access and comes in both single and dual core packages with the best in-class capabilities, support for Windows and Linux and excellent energy saving features resulting in a longer battery life. Intel has proposed the use of this processor in treadmills, home based appliances and many gadgets for medical use. It has identified the need for device to device communication in a variety of areas such as factories involving complex manufacturing stages, sensor networks on railroads, smart roads, smart lights, smart shopping, etc. and in many ways suggested the imminent era of the *Internet of Things*. Current techniques used for device to device communication are based on RFID technologies [169] which have been mostly designed for passive functioning. An *Internet of Things*, on the contrary demands a design capable of performing real-time monitoring and active and on-demand flow of information and services. IBM has invested on its vision of creating a smarter planet for mankind and claims that all *things* in the future may have computational power and intelligence. These *things* could generate huge amounts of data which may need to be accumulated and processed to extract useful information [170]. Their research is focused on creating numerous sensors and RFID technology solutions for real world problems.

In succeeding sections, we present a device to device communication architecture using the mobile agents based networked robotic system discussed

earlier as a base with a view to cater to a heterogeneous set of devices or *things*. Since mobile agents can carry a payload, the *Internet of Things* portrayed herein also features servicing of the devices in addition to active communication. We also provide the validation of the architecture by means of a real *Internet of Things* implemented using heterogeneous devices.

8.2 Current Scenarios

The current state of the *Internet of Things* deals more on the identification of different devices using RFID techniques [171], Object Naming Services [172], Electronic Product Code discovery [173], to name a few. Research on the *Internet of Things* also deals with defining architectures to support various real time scenarios which include supply chain management [174], [175], shop-floor integration [176], facility management [177], [178] and logistics [179]. Rodríguez and Favela [180] describe the use of autonomous agents that proxy for users and provide the required services in their absence. A prototype that employs mobile agents to be used in conjunction with mobile devices has been cited in [181].

Drakonkas [182] has suggested the use of mobile agents to achieve communication and non-disruptive interaction between devices constituting an *Internet of Things*.

This work endorses the use of mobile intelligent agents that can adapt based on the knowledge gained from the environment which in turn alters the state of the system. Drakonkos [182] discusses the use of mobile agents on mobile ad-hoc networks and suggests possible solutions for problems faced within, such as security, congestion and prioritizing information flow. In addition to the above advantages, mobile agents also solve the problem of heterogeneity among the communicating devices and also act as building blocks for the design and development of an asynchronous application on top of the participating devices.

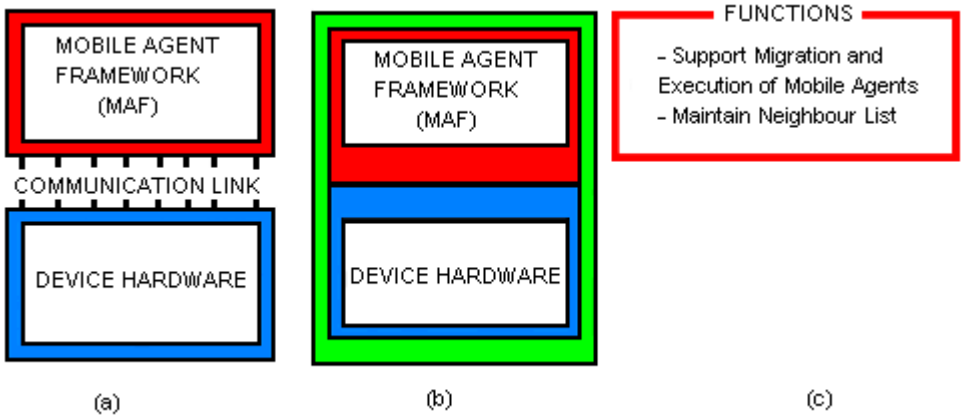


Fig. 8-1 Proposed architectures of mobile agent based devices that can populate the *Internet of Things*. (a) The components of the off-board Mobile Agent Framework (MAF) running on a dedicated node or a computer; the device communicates with it via a link (b) The on-board version of the MAF embedded on a device/appliance (c) The respective functions of the components.

The next section discusses the manner in which a mobile agent based networked robotic system can be transformed and incorporated in an *Internet of Things*. A real application scenario which involves communication between heterogeneous devices culminating in the service of a device is also described.

8.3 Proposed Architecture for the *Internet of Things*

Any proposal for the *Internet of Things* should also take into consideration means for existing devices to be participating nodes in this network. Figure 8-1 shows the architectures of the mobile agent based devices that can populate an *Internet of Things*. In Figure 8-1 (a), the upper block comprises the Mobile Agent Framework (MAF) which runs on a dedicated node or computer connected to the network. Existing devices with communication capabilities can connect to such nodes to gain the services of the mobile agents. Figure 8-1 (b) shows a similar architecture for proposed (future) devices wherein the MAF is shown to be embedded within the device itself. This aids such devices to directly

connect to the *Internet of Things* so as to gain the services of the mobile agents populating it. Figure 8-1 (c) lists the functions of each of the blocks within. The MAF supports all mobile agent based functions such as migration, execution, cloning, etc. and allows the programmer to develop applications using these mobile agents.

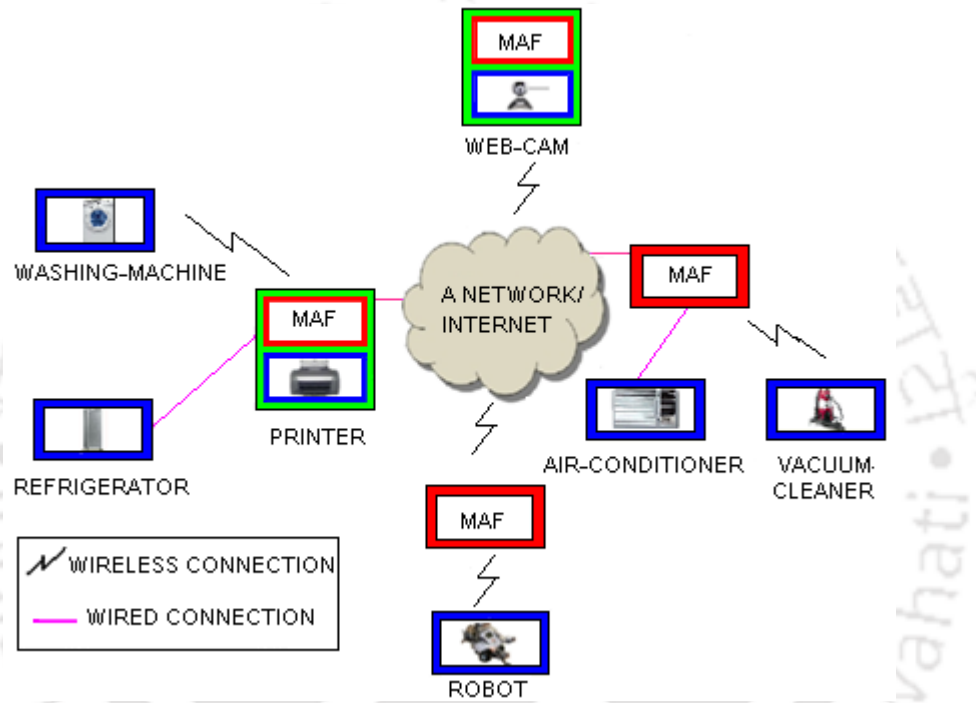


Fig. 8-2 The envisaged *Internet of Things* (The air-conditioner, vacuum-cleaner and the robot use the MAFs running on dedicated nodes while the printer and the web-cam have the same embedded within themselves)

Figure 8-2 depicts the manner in which these devices connect to the network that forms the *Internet of Things*. It can be seen that the air-conditioner, the vacuum cleaner and the robot connect via the nodes running the MAF while the printer and the webcam have the MAF embedded within and hence are capable of hosting the mobile agents and consuming the services offered by them, directly. Devices without a MAF (like the washing-machine and the

refrigerator in Figure 8-2) can also use the services of these agents via the devices with an embedded MAF provided they have the necessary communication capabilities.

8.4 The Mobile Agents in the *Internet of Things*

Mobile agents use *PherCon-C* and work in the same manner as described in earlier chapters of this thesis. They carry the services they can provide as their payloads. This migration mechanism makes them patrol the entire network in a pro-active manner. They may also clone based on need to effect a parallel search within the network. Agents carrying services that may be prominently required by the devices constituting the network are released by third parties or manufacturers of these devices. In addition these agents also allow the devices to pass information to others, retrieve information and also signal the requirement of a service available in some other device. The context of a service may change based on the type of device or appliance requiring or assisting in the service. In a certain scenario, a device may require a service that can be rendered only by another device. Such a service could be treated as a resource and the process could be termed Search-for-Resource (*SfR*). In yet another scenario, a device may be capable of physical actuation (as in case of a robot) but may not possess the correct program or service to actually perform an assigned task. Thus such a device needs to be provided a service. This act is termed as Provide-a-Service (*PaS*). This is similar to the one described in the previous chapter. Mobile agents in this architecture facilitate a variety of such scenarios to aid in searching as also providing services. Two such scenarios described below depict the use of mobile agents.

In the *Internet of Things*, there could be a need for devices to cooperate and coordinate among themselves. For instance a parcel which is being couriered from its source to the destination may pass through several intermediate zones. A web service could facilitate the tracking of the same by the receiver/sender.

When a tracking request is received by the node hosting the web service, it could spawn a mobile agent that could search and discover a networked camera and actuate it so as to get a snapshot of the parcel. With the snapshot as its payload the agent can search for another node that could enable it to email the same as an attachment to the intended receiver/sender so as to provide him/her with a view of the physical condition of the parcel along with other requested details. Such Cameras could be mounted on robots capable of wireless communication in which case the agent could provide the information about the parcel being tracked (tracking number, destination address, etc.) so that the robot can locate it and take the snapshot. Cloning could occur when multiple paths (links) between the nodes exist so as to hasten the search process.

In real-world situations such mobile agents on an *Internet of Things* can help provide emergency services too. Assume that all people possess a mobile phone (device) each of which contains the pertinent information about its owner. In case of a medical emergency the patient can force his/her mobile phone to spawn an agent and travel to other nodes (mobile phones) in the network and almost non-intrusively find a doctor who can provide relief and then direct him/her to the patient. It may also search for a node inside a hospital and inform its staff about this contingency. Here the owner of the device (mobile phone) receives the services on demand. The mobile agents can thus be used and deployed to cater to several different scenarios.

The subsequent section describes an implementation of *PherCon-C* used to realize an *Internet of Things* for both Search-for-Resource (*SfR*) and Provide-a-Service (*PaS*) tasks performed by mobile agents and the heterogeneous devices forming a network.

8.5 Implementation

To validate the use of mobile agents and the *PherCon-C* mechanism, a network of five nodes was set up using a LAN. Each node constituted a

computer running the AgentSpace [183] mobile agent platform. To demonstrate the working of a heterogeneous network, we used an ultrasonic sensor wirelessly connected to one of the nodes and a Lego NXT® [157] mobile robot connected to another. The latter used a Bluetooth® link to communicate with the node it was connected to. The programs were coded using the Java library for Lego NXT® robots running on LeJOS [160] just as described in the previous chapter. Mobile agents populating the network were capable of moving across the nodes of the network and also communicating with both the sensor and the robot as and when they landed up at the respective nodes connected to them. *PherCon-C* was used as the underlying mobile agent migration mechanism in this implementation.

8.5.1 *SfR* and *PaS* Scenarios

We implemented a problem that required both *SfR* and *PaS* to be carried out by the different agents populating the network. Two types of agents were released – one that was capable of *SfR* and the other of *PaS*. The former was capable of finding a resource. In this application the *SfR* could search the network and find a robot (resource) that is currently free and could perform a given task. Agents that carry the source code for task related to a robot, as its payload, constituted the *PaS* type. The *Internet of Things* based scenario is described below.

8.5.2 The *Internet of Things* Scenario

Figure 8-3 shows an aerial view of the experimental set-up used to demonstrate the working of the *Internet of Things*. The MAF based nodes are of the type depicted in Figure 8-1 and their topology has been superimposed on the snapshot for a better understanding.

The analogy of the set-up could be described as follows. A sensor is mounted on a scare-crow positioned in a field. As and when a stubborn bird or intruder is detected in its vicinity it requires the services of an actuating device to scare it away. Intruders like the bird can thus be driven away if a robot reaches the vicinity of the sensor.

In the implementation the ultrasonic sensor placed in the field was programmed to trigger as and when it detected the presence of an obstacle that intruded its sensing range. Since the sensor is immobile, it needs a *resource* that can remove the obstacle away to a safe distance from it.

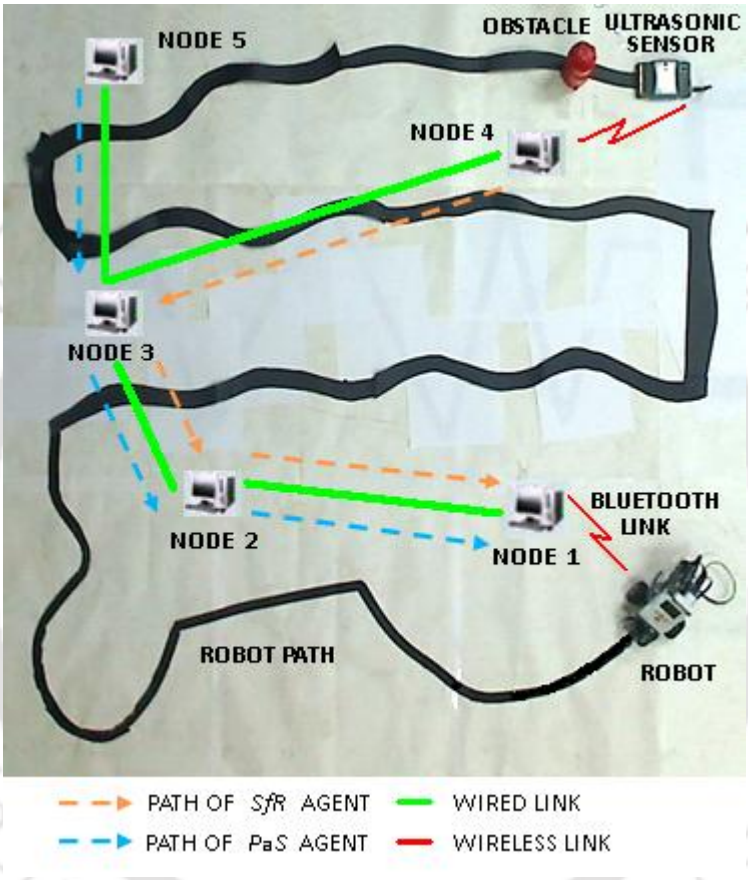


Fig. 8-3 Snapshot of the aerial view of the actual experimental setup with the network topology superimposed on it. The green lines indicate the connection between the nodes while the orange and blue lines depict the path traced by the *SfR* and *PaS* type agents respectively

When the ultrasonic sensor connected wirelessly to node 4 detects such an intruder or obstacle within a certain distance, it sends a request for a robot (*resource*) which is free and capable of moving the obstacle away. This request

signal from the sensor to node 4 forces the MAF in the latter to spawn an *SfR* based mobile agent. This agent migrates across the network in a conscientious manner to search for a robot within the network that can service the problem at the sensor's location. Idle robots within the network force the MAFs within the nodes they are tethered to, to diffuse pheromones, across their neighbours so as to indicate their willingness to act as a resource.

In Figure 8-3, the robot connected to node 1 forces the MAF within the node to pheromone in such a manner. These pheromones tend to attract the *SfR* type agents. The robot and the *SfR* mobile agent together use the *PherCon-C* strategy to discover one another. When the *SfR* agent hits the pheromone trail it travels to the node where the robot is tethered and provides information regarding the sensor's bearings and nature of request. The robot checks if it has the relevant program required to service the sensor locally. If it does not then it forces the MAF within its associated node (node 1) to diffuse pheromones that can attract those *PaS* type agents that carry the source code for the programs required to service the condition reported by the sensor. Once again the *PherCon-C* mechanism brings the relevant *PaS* agent populating the network to node 1. The node extracts the source code, compiles and downloads the code onto the robot which in turn executes it.

In the actual implementation we used a simple line following algorithm (service) to force the robot (resource) to move towards the vicinity of the sensor and then push the obstacle away. This path is depicted in Figure 8-3 in the form of a dark black line. The superimposed portions of the figure also depict the paths taken by the *SfR* and *PaS* agents. In the process if the robot is wirelessly delinked from its original node (Node 1) then it searches for an alternate node within its wireless range, connects to it and forces its MAF to commence pheromoning once again so as to indicate its availability as a free resource. All operations to recover from a contingency reported by the sensor are thus performed autonomously and actively by the *SfR* and *PaS* mobile agents and the devices that populate this network of *things*.

8.6 Conclusions

It may be noted that though the implementation described seems naïve it opens up a plethora of application scenarios to realize a robust *Internet of Things*. With several *SfR* and *PaS* agents released into the network to cater to either specific or generic devices, development of new services can be performed in a distributed manner. A human user need not go through the arduous task of programming the concerned device or robot since the same may be available within one of the agents populating the network. Instead the device can pro-actively pheromone to attract *PaS* agents that carry the required program and make use of the same. Companies that provide the device can release such *PaS* agents with programs that their respective devices can execute, into the network. Devices can then pheromone and attract the desired agents to avail of these services. A device incapable of performing a task, such as the sensor in this case, can call upon the services of another, autonomously. The overall set-up thus forms a congregation of heterogeneous devices that can function in a distributed and autonomous manner with minimal user intervention. Appendix-A provides a link to download the video of the *IoT*.

Though several proposals have been made to formalize the manner and working of an *Internet of Things*, a generic mechanism for communication, service and distributed control still remains a far cry. The *Internet of Things* envisages a huge increase in the number of interconnected devices which in turn increases the demand for network addresses. The IPv4 [184] protocol used currently cannot scale to meet the needs of such an *Internet of Things*. IPv6 [185], the next generation protocol, will provide a larger address space to accommodate this increase in number of devices. In this work, we demonstrate the use of mobile agents as a means for both communicating and servicing a network of heterogeneous devices constituting an *Internet of Things* and try to portray how the much needed generic approach is satisfied. Sophisticated applications can be designed and deployed in a non-intrusive manner by

embedding a Mobile Agent Framework (MAF) in each of the devices and then using mobile agents to perform the various *SfR* and *PaS* tasks. Security and authenticity of mobile agents still remain to be addressed. The approach described herein is however sustainable, scalable and non-intrusive and allows both device-device and human-device interactions. The deployment of mobile agents coupled with the *PherCon-C* migration strategy over such a network of *things* can have far reaching implications. The strategy doubles for both resource and service discovery making the agents converge at devices where they are required. The only requirement is the embedding of an MAF that can host these agents, on every device. This paradigm shift in the embedded system design can give rise to a new era of applications on the *Internet of Things*.

PherCon-C based migration strategy implemented on an *Internet of Things* can assist in the development of the *AB* proposed in Chapter 2. Such a network provides most of the capabilities required for an AIS based paradigm to be actually implemented. *SfR* and *PaS* agents can now be looked upon as immune cells that have different purposes. The *SfR* agents could be synonymous to Antigen Presenting Cells (APC) [78] which processes an antigen (a problem at the sensor) while the *PaS* agents could be looked upon as the antibodies that carry the valuable programs. The nodes and the robots by themselves form part of the *complement system* that facilitate the removal of the antigen which is analogous to effecting the service at the sensor. The payloads carried by the *SfR* and *PaS* agents could be honed so as to generate more effective programs using distributed learning algorithms as in AIS.

Chapter 9
Conclusions and Future work

9.1 Work done

The main objective of the work described in this thesis was to formulate an *Artificial Being* (*AB*) comprising a network of robots and devices populated with mobile agents all of which co-exist and aid one another using bio-inspired paradigms. The AIS was considered to be well suited for inter-entity co-operation and interactions and accordingly an architecture for the *being* was suggested. However this architecture called for mechanisms for faster and effective ways for migration of the agents towards robots or devices that need their services. Normal patrolling algorithms are non-proactive and suited only when the requests from the nodes are sequential or synchronous. Their performance could be adversely affected when the agents are heterogeneous in nature. Since such algorithms cannot cater to the asynchronous nature of generation of requests for services and the inherent heterogeneity within the architecture of the *AB*, a bio-inspired pro-active mechanism for mobile agent migration mechanism termed *PherCon-C* was proposed, simulated and implemented. Results prove that this mechanism outperforms the others when one considers a trade-off between the time and energy consumed. The mechanism was also found to be implementable. Cloning of agents can obviously ameliorate the performance of the system in terms of service times but their population has to be controlled lest they clutter the network and bring the system to a stand-still. Conventional algorithms to control cloning cater to only homogeneous agents and use some part of the available bandwidth in doing so. The use of *stigmergy* can greatly reduce agent-agent or node-node interactions and hence bandwidth. A mechanism for controlling the population of heterogeneous agents was thus devised using both *stigmergy* and a novel yet

simple concept of *cloning resource*. While the former saved on entity-entity interactions, the latter inhibited the spontaneous cloning of out-of-demand agents, thus providing more bandwidth for the in-demand agents. Simulations of the same using *PherCon* as the agent migration mechanism showed positive results. The lack of a dedicated network infrastructure prevented the actual implementation of the cloning controller.

Implementations of *PherCon-C* on a simple network of robots also opened up possibilities to extend the network to an *Internet of Things*. A naïve network of such *things* was also implemented to portray the manner of working of two distinct types of heterogeneous agents that function to *search for a service* and to *provide for a service*. *PherCon-C* coupled with such agents thus empowers the devices forming the nodes of the network to communicate, interact and co-operate with one another and autonomously accomplish tasks.

9.2 The Outcomes and Application Scenarios

The *AB* can be viewed as a collection of connected entities (organs and effectors) which allow the flow of mobile agents (cells). The mobiles agents form the *cells*, the nodes in the network the processing *organs*, the robots the *effectors* while the network provides for the much needed *plasma* to provide mobility to the agents. Agents provide for both *SfR* and *PaS* functionalities. The formulation of an *AB* as such can open up new avenues in the study of networked systems, swarms and also multi-cellular beings. While the simulator developed allows for testing new mechanisms, the real implementation of the network can provide as a test bed to study their performances in the practical world.

The *AB* needed to be scalable both in terms of robots and mobile agents. Tethering and programming robots and devices has always been an arduous exercise. If expert programmers could release, *SfR* agents with efficient robot/device programs, to realize a variety of services or tasks, as their payloads,

an automatic on-demand serving of such programs at the node where a new robot or device is deployed could be effected. The user need only connect the robot/device onto the network and issue commands. Programming these commands is not mandatory if there is an agent(s) within the network which carries the program required to execute these commands. The migration mechanism will eventually attract the concerned agent towards the robot and provide the service (program). Since users need not be aware of the intricacies of such programming they could greatly benefit from such a facility. It would also encourage new and novice users to try their hand at programming robots and devices.

PherCon-C and the cloning controller mechanisms can find use in a range of applications. With ubiquitous computing almost a reality, it is not far when all devices become accessible in some way via a network. If this is so mobile agents will be in a position to migrate non-intrusively from one device to another and perform both *SfR* and *PaS* activities. For instance, consider highly populated roads where traffic jams obstructing the movement of ambulances are known to cause many an inconvenience to the patients. If all vehicles on the road could act as a mobile ad hoc network and host an MAF each, an ambulance on standby could pheromone to indicate its availability. At an accident site, a vehicle could automatically spawn an *SfR* agent with its current co-ordinates within, to search for the ambulance. When it reaches the ambulance through the network of dynamic vehicles, the ambulance could take off to the accident site. While doing so it could spawn a *PaS* agent which will eventually clone and move forward to inform all vehicles and traffic *en route* to clear the path towards the site. Vehicle drivers can be on guard and ensure that there are no traffic jams in the path of the ambulance. It may seem that the initial spawning of the *SfR* agents could be replaced by a mere phone call to the ambulance and thus be a redundant exercise. Finding and activating the nearest ambulance, however may require all ambulances to receive this call. The *SfR* agent on the contrary could travel non-intrusively to the MAF of the ambulance and provide the exact co-ordinates to

the GPS within and allow for faster and reliable communication. On its journey to the accident site and then to the hospital the MAF within the ambulance could spawn a short-time-span agents and then send them to the MAFs within the vehicles ahead of it and inform them to steer clear of the ambulance. Such agents could clone if they land up in the MAF of vehicles in the forward path not yet covered by the ambulance. Others would naturally die if they reach MAFs of vehicles not in the designated path of the ambulance. Further *SfR*-like agents could also inform the hospitals in the vicinity to standby for this eventuality. Many such non-intrusive applications can be envisaged in the social world.

9.3 Future directions

One of the major outcomes of this metaphorisation was to adapt theories from the realm of biology and also immunology and use them to enhance the working of real world systems. Concurrent to the work reported in this thesis, Banerjee et al [186] have described an immune system based methodology for scaling modular multi-agent systems on the basis of the architecture proposed in Chapter 2 [109] as the basis. The same authors [187] have also created an immune inspired *search and rescue* strategy.

Since fast migration of the agents, to the point where the service is required, is crucial, the proposed migration mechanisms need still be improved. A typical speed-up of service could be performed if nodes receiving pheromones have within themselves the requested service. In such a case the pheromoned node could pro-actively spawn an agent and send it over to the RRS along with the requested service. This will not just hasten the service process but also save energy which would otherwise be consumed in re-diffusing pheromones. A full implementation of such a mechanism coupled with the cloning controller needs to be tried and tested.

With agents migrating from one node to another freely, security is a matter of great concern. The thesis does not address this issue. It assumes trust within

the closed network. There have been several attempts to ensure a secure manner of deploying and using agents in a network [188], [189] [190]. It will be worthwhile to try the efficacy of such proposals in the real implementation.

The proposed *Internet of Things* also has broad and long term impact on the development of future mobile devices. A device with an on-board MAF can act as an ideal node to receive requests for a service from other non-MAF devices and pheromone on their behalf. Further, embedding a learning mechanism into the mobile agents would mean the realization of an Intelligent Internet of Things.

The *AIB* described in this thesis is only partially ready. Only the first of the three main challenges listed out in Section 2.6 of Chapter 2 have been addressed. An adaptive bio-inspired learning mechanism will be required to embed the much needed intelligence within it. Since the *AIB* comprises both agents and a network, a hybrid of the Clonal Selection [78] and the Idiotypic network theories [89] could be envisaged to ensure that new and more efficient agents continuously evolve and populate the network. Somatic hypermutation [78] to edit the rules that form the variable region of the Y-shaped antibody could further aid robots to fine tune the rules or programs they execute, eventually helping the system as a whole to learn new and enhanced rules.

With one *AIB* in place, the possibility of several *AIB*s and hence an artificial society could be envisioned. Trust and emotions [150] could then be embedded on each to make them behave akin to their natural counterparts.

Response to Reviewers' Comments

I would like to take this opportunity to thank the examiners for their detailed analysis and critical comments on the research work presented in this thesis. A number of suggestions were given and many interesting issues, questions have been raised through the comments. In this section, their suggestions, queries followed by my response are given. The concerns of the Indian examiner are addressed first through items 1 to 10 and then those of the foreign examiner are discussed through items 11 to 16. Once more I wish to express my appreciation for the insightful comments, which have helped me substantially during the revision of the thesis.

Indian Examiner:

1. ***In order to understand the concepts, it would have been better if a couple of examples on mobile agents, along with their role, agent technology, agent creation, agent migration, agent communication etc. were provided.***
   Action Taken: Some more on mobile agents, related technologies, how they are created, made to migrate, etc. have been inserted beginning from Page number 9 to page number 10 under the Section 1.7.

2. ***It is not clear what the modifications are in the AIS architecture and functioning and why they need to be incorporated in the existing system.***
   Action Taken: The motivation for this work was to create a multi-robot system comprising of multiple robots capable of co-operation and co-

ordination and possessing characteristics that include scalability, ability to adapt based on network resources and evolving and catering to the resource and communication needs of robots efficiently. The biological immune system offers a biological framework with the aforementioned properties whose metaphorisation definitely suits the multi-robot system under investigation. Thus immune system provides a motivation in the creation of a framework where learning agents co-exist together to aid the multi-robot system.

The robots form the various organs of a multi-robot being. The mobile agents form the immune cells or entities that learn and provide information thereby destroying the antigen. A robot which requires a service (RRS) forms the antigen. Tracking an RRS within the robotic network and servicing the request can be looked upon as the process of annihilating the antigen. Thus the robotic network or the being is protected by the mobile agents that act as immune cells.

In order to implement and test the viability of such a system there is a need for devising efficient mobile agent migration and RRS discovery mechanisms within the robotic network to quickly weed out the antigens. The thesis thus portrays efficient mechanisms to achieve this. The task of learning by the agents is left for future work.

3. ***In Section 3:All techniques such as Random…CLInG would have supported by good examples on immunization systems.***

Action Taken: Efficient migration of the antibody to the location of the antigen is a vital requirement in an immune system. In our scenario, the mobile agents which constituted the immune cells/antibodies required to quickly migrate, search and service the RRSs. The AIS model discussed in Chapter 2 proposed a round robin strategy of migration on art of the mobile agents. This is not a demand based strategy and thus not efficient enough. CLInG, EVAP and G-B algorithms that suggest faster ways of patrolling or

searching for resource within a network by mobile agents. They could thus replace the round robin mechanism. We chose to compare our migration strategies for servicing RRSs with these algorithms because they could also be used to realize the migration mechanisms required in the realization of the AIS portion of our Being. Apart from the baseline algorithms viz. Random and Conscientious, the EVAP, CLInG and G-B strategies are demand-based algorithms and provide faster services, thus making them ideal choices for comparing our mechanisms.

4. ***In Section 3.5: It is not clear why G-B algorithm is placed in this chapter which deals with only migration algorithms.***

   Action Taken: While techniques such as random, Conscientious, EVAP, CLInG define how the agent migrates, these strategies are utilized here primarily aimed at servicing the robotic nodes. G-B algorithm provides for a different angle and defines how the mobile agents can reach the nodes using cloning to service them. Since *PherCon-C* described later in the thesis also uses cloning we felt it best to include the same along with the other algorithms that were used in the subsequent comparisons. Hence it was put together in the same chapter.

5. ***In Chapter 6: It will be appropriate if MANER and the modifications incorporated in the simulation to suite this work is presented.***

   Action Taken: MANER refers to a Mobile Ad-hoc Network of Robots where each robot acts as a router capable of forwarding the messages across to other robots. The main focus of work done with respect to MANER [112] is in developing routing protocols [191], [192], for robots facilitating transfer of messages from source to destination considering intermediate robot nodes as routers. Zhigang et al. [191] and Saumitra et al. [192] describe protocols, which are useful in applications where co-ordination and sequencing of actions are essential. But in most of the real world scenarios the topology of

the robots cannot be the same due to the mobility of mobile robots. Normally MANER protocols deal with designing routing protocols that suit a mobile ad-hoc network of robots to route the communication messages from source to destination. The work dealt with in this thesis is different from this scenario in that it proposes how mobile agents reach the destination robots quickly while overcoming the challenges en route.

6. ***In Chapter 7: The testing of the implementation has not been enclosed in the thesis.***

   Action Taken: Since network infrastructure to rigorously test the experimental set-up in real-time was not available, evaluation was performed based on the number of hops taken by the different mechanisms in different network topologies. These results have been depicted in Fig.7-6. The graph herein clearly shows that the proposed *PherCon-C* outperforms the others for various topologies.

7. ***In Chapter 8: The significance is not clear. May be future internet. [REF 44]- As per this reference the implementation of PherCon-C using IoT would have given more clarity.***

   Action Taken: Chapter 8 discusses a novel proposal of using a mobile agent middleware for devices attached to a backbone internet. When devices are attached, the importance of communication across devices becomes paramount. In this case the performance of this system in terms of service times can be obviously improved by use of the proposed *PherCon-C* algorithm for migration of mobile agents. In this chapter, we have not shown a comparison of various strategies of mobile agent migration since these methods have already been compared with others earlier. But we incorporated the *PherCon-C* strategy for mobile agent migration directly to the experimental *IoT* scenario.

8. ***In Section 8: More Experimental results or simulation results to evaluate the system of Internet of Things are required.***

Action Taken: The main thrust of this work is the proposal to use mobile agents for the Internet of Things application. So far the Internet of Things applications have used static infrastructures such as RFID for device to device communication. But with the recent advent of technological developments, embedding processors into devices is becoming a reality which calls for flexible software framework for device deployment and application installation. We propose in this work that this can be made possible by the use of mobile agents. We have metaphorized a set of heterogeneous mobile networked robots to an Internet of Things. Thus the same robotic network being heterogeneous could populated with sensors (such as WSNs), robots and devices to form an IoT. Since the proposed mobile agent migration mechanisms have been shown to work efficiently for a network of robots, we assume the same to be true for a modest IoT too.

9. ***In Page 5, Section 1.4:The title here is middleware design approaches for sensor networks. But it neither talks about the importance of middleware in networked robotic systems nor sensor networks. He needs to specify the middleware for the purpose and some middleware approaches may be given later.***

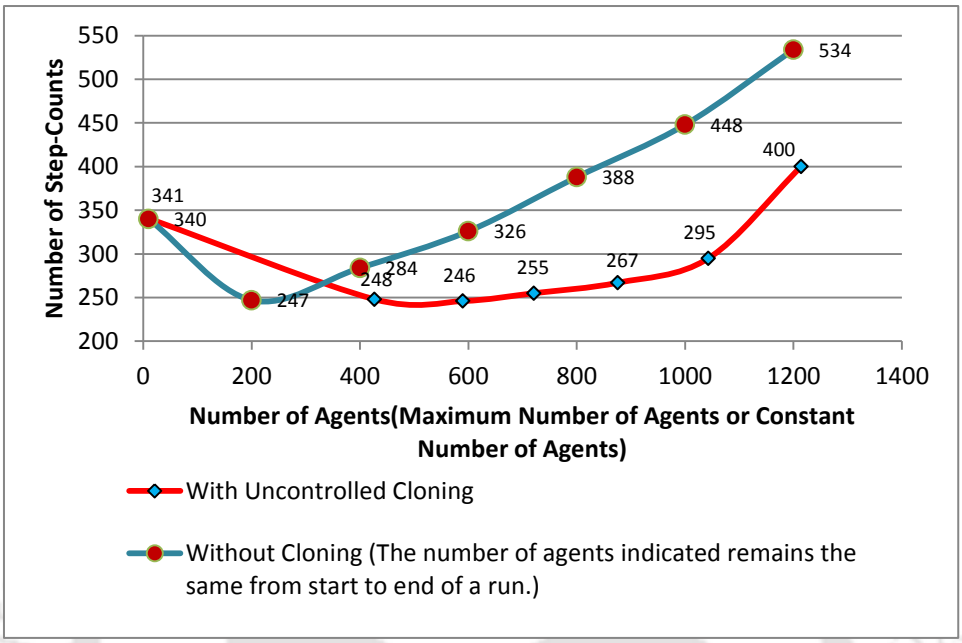
Action Taken: Some more on middleware design approaches and the importance of middleware in networked robotic systems is included from Page number 8 to page number 9 under the Section 1.7.

10. ***In Page 65, Figure 5.1: A single curve has been plotted to study the step-count pattern as the number of agents or clones increases. The circumstances where there are no cluttering within the network could have been plotted for better comparison.***

Action Taken: When we use mobile agents for servicing mobile robots, we intend to decrease the service times. In order to decrease the service times, PherCon strategy for migration of mobile agents was modeled. This work was later extended by PherCon-C since presence of multiple mobile agents naturally reduces the service times due to their parallel behaviour. But this posed a problem since hosting numerous mobile agents beats the purpose due to cluttering and consumes network resources which need to be reduced. Hence a mechanism to control cloning was proposed to contain the cloning was devised.

Cluttering here is assumed to be caused by the increasing number of agents and clones. The graphs plotted in Chapter 4 by the *PherCon* and *PherCon-C* methods explain the improvement obtained by cloning of agents in servicing the robots.

The single line plotted in graph 5.1 depicts the effect of cluttering which occurs due to unbounded cloning by observing the number of step counts to service 100 RRSs. Similar results on controlled cloning was not included since the motivation for the plot was to show the need for controlled cloning. However a graph showing the comparison between the "without cloning" case and "with cloning" cases is included below. The blue line in the graph shown below has a similar nature as the one shown in Figure 5.4. However cluttering due to excessive agents (not clones) seems to commence earlier beyond 247 after which an increase in number of agents tends to increase the number of steps required to service the RRSs rapidly, thus deteriorating the performance.



Foreign Examiner:

11. ***Literature survey is narrowly focused on just bio-inspired approaches (very limited). A lot of successful robot building was indeed done based on ecological psychology, insect behaviors, often termed as simple intelligence by Rodney Brooks at MIT.***

    Action Taken: A brief literature survey including the mentioned references has been inserted beginning from Page number 2 to page number 5 under the Sections 1.2 and 1.3.

12. ***Team Robotics should have been included in the literature survey. Lessons learned from robot-soccer competitions should be included. Advances which are significant in the context of networked robots considered in this thesis and vice versa.***

    Action Taken: The same have been highlighted in the response to Question 1.

13. ***The thesis was focused on trying something new based on biological immune system. However, the purpose seems to be missing. Why did you want to do it and what was the major scientific challenge you wanted to solve here?***

    Action Taken: Originally it was envisaged to create a multi-robot system comprising of multiple robots capable of co-operation and co-ordination which in turn would form, work and behave like a real AIS. However it was found that to realize the same it was imperative to develop a middleware framework which is generic and scalable, can adapt to network resources and capable of evolving and catering to the resource and communication needs of robots in an efficient manner. Thus this aspect became the major focus of the thesis. Using the efficient mechanisms for mobility and service of the individual robots comprising the network it is now possible to realize the immune model proposed in Chapter 2.

    A base immune system has however been demonstrated using mobile agents as the immune cells and the RRSs as the antigens. By servicing the RRSs, the mobile agents perform the job of annihilating the antigens autonomously and efficiently. The realization of the proposed AIS based learning mechanism suggested in Chapter 2 was left for future work.

14. ***Presentation could be improved. The logic of the presentation/coherence could be structured better by putting together 3 and 4,6 and 7.***

    Action Taken: The same will be done if advised in the final report.

15. ***Not convinced that the characterization of connected network of agents/robots as AB (Artificial Being) is different from standard multi-agent systems/team robots where in these systems each agent/robot is autonomous and is able to communicate and coordinate activities with other agents.***

Action Taken: The thesis does not claim that the system under study is different from a multi-agent system. Even though there are multi-robot systems incorporating mobile agents, associating mobile agent based multi-robot systems with immune system of a being has opened up new avenues to view the context of using mobile agents. Hence we use the acronym AB to denote the multi-robot system using mobile agents under study. Moreover the use of mobile agents in robots allows for these mobile agents to share and evolve according to local knowledge and hence acquire information in parallel, just as in evolution. Also the robots remain grouped as a loosely coupled system akin to the cells that constitute the basic building blocks of a massively multi-cellular being.

In brief, the point being emphasized is that a group of robots can be looked upon as organs comprising a body being glued by a (wireless) communication link. Knowledge learnt by one could now be passed on to the others in the group proactively by the mobile agents. The movement of the mobile agents is akin to that of the immune cells/antibodies that destroy the antigens (service requests/information needed).

16. ***Metaphors need to be used in a meaningful way; otherwise they may become distractive with respect to the real scientific questions.***

Action Taken: Artificial immune system offers inspiration to solve some real issues with respect to multi-robot middleware by metaphorising entities of the robotic system as organs which coexist together and also maintain system stability against antigens using antibodies. Antibodies are entities who sense the approach of antigens and quickly reach the site of antigen invasion to thwart any danger to the organism. Antibodies have the ability to sense the approach of antigens through chemical signals and they respond by reaching the antigen site through the bloodstream. The entities on use in our multi-robot system namely Mobile agents are to sense their need and reach the needing robots requesting for service immediately so that system functions

with stability. Hence we study a few mobile agent migration algorithms such as Random, EVAP, CLInG and G-B and propose a better migration algorithm viz. *PherCon* and *PherCon-C* and experimentally analyse these algorithms.

A table of metaphors is shown below.

Immune system	**Robot's world**
Adaptive immune system	Multi-robot system & the Mobile Agent system
B-Cells	Mobile Robots
T-Cells	Mobile Agents
Genome	Initial Set of rules
Genes	Rules
Antibody	Robot Action rule-set
Self-preservation-surviving fear or pain or discomfort	Selection and Execution of rules from rule-set
Antigens (Stimulus)	Condition for Rule execution
Flow of Cells	Agent Mobility
Lifetime of the Cell	Battery Charge

Bibliography

[1] Bruno Siciliano and Oussama Khatib, Eds., *Springer Handbook of Robotics*.: Springer, 2008.

[2] R.H. Taylor and D. Stoianovici, "Medical robotics in computer-integrated surgery," *IEEE Transactions on Robotics and Automation*, vol. 19, pp. 765–781, 2003.

[3] B.M. Yamauchi, "PackBot: A versatile platform for military robotics," in *Proceedings of SPIE*, 2004, pp. 228–237.

[4] P. S. Schenker, T. L. Huntsberger, and P. Pirjanian, "Robotic autonomy for space: closely coordinated control of networked robots," in *Proceedings of the 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Montreal, Canada, 2001.

[5] Matthew Dunbabin, Peter Corke, Iuliu Vasilescu, and Daniela Rus, "Experiments with Cooperative Control of Underwater Robots," *International Journal of Robotics*, vol. 28, no. 6, June 2009.

[6] J. Rashid, M. Broxvall, and A. Saffiotti, "Digital Representation of Everyday Objects in a Robot Ecology via Proxies.," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nice, France, 2008.

[7] J. Rashid, "Extending a Networked Robot System to Include Humans, Tiny Devices and Everyday Objects," Orebro Universitet, , Sweden, Dissertation 2011.

[8] N. Mohamed and J. Al-Jaroodi, "Middleware Framework for Resource Discovery in Mobile Environments ," in *Proceedings of the International Symposium on Collaborative Technologies and Systems*, 2008, pp. 524 - 531.

[9] A. Makarenko, A. Brooks, and T. Kaupp, "Orca: Components for Robotics," in *Proceedings of the International Conference on Intelligent Robots and Systems*, 2006, pp. 163-168.

[10] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, and W. Yoon, "RT-Middleware: Distributed component Middleware for RT (Robot Technology)," in *Proceedings of the International Conference on Intelligent Robots and Systems*, 2006, pp. 3555-3560.

[11] F. Heckel, T. Blakely, M. Dixon, C. Wilson, and W. D. Smart, "The WURDE Robotics middleware and RIDE Multi-Robot Tele-Operation Interface," in *AAAI Mobile Robotics Workshop*, 2006.

[12] S. Ahn et al., "UPnP Robot Middleware forUbiquitous Robot Control," in *Proceedings of the 3rd International Conference on Ubiquitous Robots and Ambient Intelligence*, 2006.

[13] V. Ng-Thow-Hing, T. List, K.R. Thrisson, J. Lim, and J. Wormer, "Design and Evaluation of Communication Middleware in a Distributed Humanoid Robot Architecture," in *Workshop Measures and Procedures for the Evaluation of Robot Architectures and Middleware*, 2007.

[14] M. Broxvall, B.S. Seo, and W.Y. Kwon, "The PEIS Kernel: A Middleware for Ubiquitous Robotics," in *Workshop on Ubiquitous Robotic Space Design and Applications*, 2007.

[15] M. Mizukawa et al., "ORiN: Open robot interface for the network," in *Proceedings of the Annual Conference of the Society of Instrument and Control Engineers of Japan*, 2002, pp. 925-928.

[16] P. Gil, I. Maza, A. Ollero, and P. Marrn, "Data Centric Middleware for the Integration of Wireless Sensor Networks and Mobile Robots," in *Proceedings of the 7th Conference on Mobile Robots and Competitions*, 2007.

[17] E. Takcuchi and T. Tsubouchi, "Sensory Data Processing Middlewares for Service Mobile Robot Applications," in *Proceedings of the International Joint Conference*, 2006.

[18] M. Kranz, R. Rusu, A. Maldonado, M. Beetz, and A. Schmidth, "A Player/Stage System for Context-Aware Intelligent Environments," in *System Support for Ubiquitous Computing Workshop*, 2006.

[19] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "Middleware for Robotics: A Survey," in *Proceedings of the IEEE Conference on Robotics, Automation and Mechatronics*, Chengdu, 2000, pp. 736 - 742.

[20] Pedro Jose Marron, "Middleware Approaches for Sensor Networks," *Summer School on WSNs and Smart Objects*, 2005.

[21] P. Bellavista, A. Corradi, and C.Stefanelli, "Mobile Agent Middleware for Mobile Computing," *IEEE Computer Journal*, vol. 34, no. 3, pp. 73- 81, 2001.

[22] M. Chen, T. Kwon, Y. Yuan, and V.C.M. Leung, "Mobile Agent Based Wireless Sensor Networks," *Journal of Computers*, vol. 1, no. 1, 2006.

[23] H. Qi, S.S. Iyengar, and K. and Chakrabarty, "Multi-resolution data integration using mobile agents in distributed sensor networks,"

IEEE Transactions on Systems Man Cybernetics – Part C, vol. 31, no. 3, pp. 383–391, 2001.

[24] H. Qi, X. Wang, S.S. Iyengar, and K. and Chakrabarty, "Multi-sensor data fusion in dis-tributed sensor networks using mobile agents," in *Proceedings of 5th International Conference on Information Fusion*, 2001, pp. 11–16.

[25] H. Qi, Y. Xu, and X. Wang, "Mobile-agent-based collaborative signal and information pro-cessing in sensor networks," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1172–1183, 2003.

[26] Y.C. Tseng, S.P. Kuo, Lee H.-W., and C.-F. Huang, "Location tracking in a wireless sensor network by mobile agents and its data fusion strategies," *The Computer Journal*, vol. 47, no. 4, pp. 448–460, 2004.

[27] D. Marsh, D. O'Kane, and G. M. P. O'Hare, "Agents for wireless sensor network power management," in *Proceedings of the International Conference on Parallel Processing Workshops* , 2005, pp. 413–418.

[28] L. Tong, Q. Zhao, and S. Adireddy, "Sensor networks with mobile agents," in *Proceedings of the Military Communications International Symposium*, 2003, pp. 688–693.

[29] C.L.Fok, G-C. Roman, and C.Lu, "Agilla: A Mobile Agent Middleware for Self-Adaptive Wireless Sensor Networks," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 4, no. 3, July 2009.

[30] A. Outtagarts, "Mobile Agent Based Applications: A Survey," *International Journal of Computer Science and Network Security*, vol. 9, no. 11, September 2009.

[31] L. Cragg and H. Hu, "Application of Mobile agents to robust teleoperation of internet robots in nuclear decommissioning," in *IEEE conference on Industrial technology*, 2003.

[32] L. Cragg and H. Hu, "A multi-agent system for distributed control of networked mobile robots," , 2005.

[33] L. Cragg and H. Hu, "Mobile Agent approach to networked robots," *The International Journal of Advanced Manufacturing Technology*, vol. 30, no. 9-10, pp. 979–987, 2006.

[34] Lynne E. Parker, "ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation," *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 220--240, 1998.

[35] Y. Kambayashi, T. Yasuhiro, Y. Hidemi., T. Munehiro, and Y. Hisashi, "Design of a Multi-Robot System Using Mobile Agents with Ant Colony Clustering," in *Proceedings of 42nd Hawaii International Conference on Systems Science*, 2009, pp. 1-10.

[36] Y. Kambayashi, Y. Harada, O. Sato, and M. Takimoto, "Design of an intelligent cart system for common airports," in *Proceedings of the IEEE 13th International Symposium on Consumer Electronics*, 2009, pp. 523-526.

[37] L.N. de Castro and J. Timmis, *Artificial Immune Systems: A new Computational Intelligence Approach*. London : Springer, 2002.

[38] C. Jonker and J. Treur, "Agent Oriented modeling of the dynamics of biological organisms," *Journal of Applied intelligence*, pp. 1–20, 2007.

[39] A. Ishiguro, Y. Watanabe, T. Kondo, Y. Shirai, and Y. Uchikawa, "A Robot with a Decentralized Consensus-making Mechanism Based on the Immune System," in *Proceedings of ISADS 1997*, 1997, pp. 231–237.

[40] A. Ishiguro, Y. Watanabe, T. Kondo, and Y. Uchikawa, "Immunoid: A Robot with a Decentralized Behavior Arbitration Mechanisms Based on the Immune System," in *Proceedings of the Fourth International Conference on Control, Automation, Robotics and Vision*, 1996, pp. 1600–1605.

[41] A. Ishiguro, S. Ichikawa, and Y. Uchikawa, "A gait acquisition of a 6-legged robot using immune network," in *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, vol. 2, 1994, pp. 1034–1041.

[42] S. Sung-Oog, L. Jug-Ooo, and B.D. Kwon, "A mobile agent based multi-robot design method for high-assurance," in *Proceedings of the 10th IEEE High Assurance Systems Engineering Symposium*, 2007, pp. 389-390.

[43] S. Banerjee and Moses. M., "Modular RADAR: An Immune System Inspired Search and Response Strategy for Distributed Systems," in *Proceedings of the 9th International Conference on Artificial Immune Systems*, 2010, pp. 116-129.

[44] S. Banerjee and Moses. M., "Scale Invariance of Immune System Response Rates and Times: Perspectives on Immune System Architecture and Implications for Artificial Immune Systems," *Swarm Intelligence*, vol. 4, no. 4, pp. 301-318, 2010.

[45] N. Minar, K.H. Kramer, and P. Maes, "Cooperating Mobile Agents for Mapping Networks," in *Software Agents for Future Communications Systems*. New York: Springer-Verlag, 1999, pp. 287-304.

[46] A. Almeida, *Patrulhamento Multiagente em Grafos com Pesos (in Portuguese)*. Recife, Brasil: M.Sc. Thesis, Centro de Informática, Univ. Federal de Pernambuco, 2003.

[47] H. N. Chu et al., "Swarm approaches for the patrolling problem information propagation vs pheromone evaporation," in *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence*, 2007, pp. 442-449.

[48] J.Gaber and M.Bakhouya, "Mobile Agent-based Approach for Resource Discovery in Peer-to-Peer Networks," in *Proceedings of the Fifth International Workshop on Agents and Peer-to-Peer Computing at AAMAS*, 2006, pp. 1-9.

[49] A.Libsman, "Global napster usage plummets, but new file-sharing alternatives gaining ground," *Jupiter Media Metrix*, July 2001.

[50] J. Li et al., "On the Feasibility of peer to peer web indexing and search," in *Proceedings of the 2nd International Workshop on peer-to-peer systems*, 2003, pp. 207-215.

[51] P. Karlson and M. Lüscher, "Pheromones: A new term for a class of biologically active substances," *Nature*, vol. 183, pp. 55-56, 1959.

[52] J. Deneubourg, S. Aron, S. Goss, J. M. Pasteels, and G. Duerinck, "Random behaviour, amplification processes and number of participants : How they contribute to the foraging properties of ants," *Physica*, vol. 22, no. D, pp. 176–186, 1986.

[53] Naoyuki. Kitamura, Yoshiyuki. Nakamichi, and Koji. Fukuda, "Development of a desktop swarm robot system based on pheromone communication," *Artificial Life Robotics*, vol. 14, no. 3, pp. 329–331, 2009.

[54] H. Van Dyke Parunak, Sven Brueckner, and John A. Sauter, "Digital Pheromones for Coordination of Unmanned Vehicles," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, 2002, pp. 246-263.

[55] M. Roth and S. Wicker, "Termite: Emergent Ad-Hoc Networking," in *Proceedings of the the Second Mediterranean Workshop on Ad-Hoc Networks*, 2003.

[56] E. Bonabeau, M. Dorigo, and Theraulaz. G., *Swarm Intelligence:*

From Natural to Artificial Systems.: Oxford University Press, 1999.

[57] M. Roth and S. Wicker, "Asymptotic Pheromone Behavior in Swarm Intelligent MANETs: An Analytical Analysis of Routing," in *Proceedings of Sixth IFIP IEEE International Conference on Mobile and Wireless Communications Networks Behavior*, 2004.

[58] W.W. Godfrey and S.B. Nair, "Mobile Agent Cloning for Servicing Networked Robots," in *Proceedings of the 13th International Conference on Principles and Practice of Multi-Agent Systems*, 2010.

[59] T. Suzuki and T. Izumi, "Biologically Inspired Self-Adaptation of Mobile Agent Population," in *Proceedings of the 16th International Workshop on Database and Expert Systems and Applications*, 2005, pp. 170-174.

[60] J. Ma, G.M. Voelker, and S. Savage, "Self-stopping worms," in *Proceedings of the 2005 ACM workshop on Rapid malcode*, 2005, pp. 12-21.

[61] T. Łuczak, M. Kutyłowski, and F. Zagórski, "Self-stabilizing population of mobile agents," in *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing*, 2008, pp. 1-8.

[62] K. Amin and A Mikler, "Dynamic agent population in agent-based distance vector routing," in *Proceedings of Second International Workshop on Intelligent Sytem Design and Applications*, 2002, pp. 195-200.

[63] J.D. Farmer, N.H. Packard, and A.S. Perelson, "The immune system, adaptation, and machine learning," *Physica D*, vol. 2, no. 1-3, pp. 187-204, 1986.

[64] S.B. Nair, W.W. Godfrey, and D.H. Kim, "On Realizing a Multi-Agent Emotion Engine," *International Journal of Synthetic Emotions*, vol. 2, no. 2, 2011.

[65] W.W. Godfrey, S.B. Nair, and D.H Kim, "Towards a Dynamic Emotional Model," in *Proceedings of the IEEE International Symposium on Industrial Electronics*, 2009, pp. 1932-1936.

[66] R.K Mishra, A. Srivastava, K. Bhaumik, and S.S. Chaudary, "Acth and Regulation of Adrenocortical secretion: A mathematical model," *Indian Journal of Pure Applied Mathematics*, vol. 13, no. 12, pp. 1503-1512, 1982.

[67] L.N.D Castro, *Fundamentals of Natural Computing: Basic*

Concepts, Algorithms, and Applications, 1st ed.: Chapmann & Hall Publishers, 2006.

[68] W.W. Godfrey and S.B. Nair, "A Pheromone based Mobile Agent Migration Strategy for Servicing Networked Robots," in *Proceedings of the 5th International ICST Conference on Bio-Inspired Models of Network, Information, and Computing Systems*, 2010.

[69] (2011, July) Webots: Robot Simulator. [Online]. http://www.cyberbotics.com/overview

[70] (2011, July) CARMEN. [Online]. http://carmen.sourceforge.net/

[71] LEGO.com MINDSTORMS: Home. [Online]. http://mindstorms.lego.com

[72] A. E. Martinez, R. Cabello, F. J. Gbmez, and l. Martinez, "INTERACT-DDM A Solution For The Integration Of Domestic Devices On Network Management Platforms," in *Proceedings of IFIP/IEEE International Symposium on Integrated Network Management*, 2003, pp. 485-488.

[73] M. Rob, Xu Wenyuan, K. Pandurang, and T. Wade, "Service Discovery and Device Identification inCognitive Radio Networks," in *Proceedings of the 2nd IEEE Workshop onDigital Object Identifier*, 2007, pp. 40 – 47.

[74] Intel atom processsors - Enabling internet access everywhere. [Online]. http://www.intel.com/technology/atom/

[75] IBM-Smarter Planet - United States. [Online]. http://www.ibm.com/smarterplanet

[76] R. Jedermann and W. Lang, "The Benefits of Embedded Intelligence- Tasks and Application s for Ubiquitous Computing in Logistics," in *Proceedings of the First International Conference on The Internet of Things*, vol. 4952, 2008, pp. 1-18.

[77] C. Decker et al, "Cost-benefit Model for Smart Items in the Supply Chain," in *Proceedings of the First International Conference on The Internet of Things*, vol. 4952, 2008, pp. 1-18.

[78] M. Rodríguez and J. Favela, "A Framework for Supporting Autonomous Agents in Ubiquitous Computing Environments,".

[79] A.Wang, C.Sorensen, and E.Indal, "A Mobile Agent Architecture for Heterogeneous Devices," in *Proceedings of the International Conference on Wireless and Optical Communications*, 2003, pp. 14-16.

[80] Drakontas LLC, "Wireless Collaboration for First Responders," A White Paper on Wireless Networking and Intelligent Mobile Agents 2005.

[81] Ichiro Satoh Homepage. [Online]. http://research.nii.ac.jp/~ichiro/agent/agentspace_v3

[82] RFC 791- Internet Protocol.

[83] RFC 2460 – Internet Protocol, Version 6 (IPv6) Specification.

[84] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "A review of middleware for networked robots," *International Journal of Computer Science and Network Security*, vol. 9, no. 5, pp. 139-148, 2009.

[85] T. Izumi, F. Ooshita, H. Kakugawa, and T. Masuzawa, "A Biologically Inspired Self-Adaptation of Replica Density Control," *IEICE Transactions on Information and Systems*, vol. 92-D, no. 5, pp. 1125-1136, 2009.

[86] K. Murphy, P. Travers, and M. Walport, *Janeway's Immunobiology*, 7th ed.: Garland Science, 2008.

[87] M. Bakhouya and J. Gaber, "Adaptive Approach for the Regulation of a Mobile Agent Population in a Distributed Network," in *Proceedings of the 5th International Symposium on Parallel and Distributed Computing*, 2006, pp. 360-366.

[88] Ruoting Yang, Sharon Bewick, Mingjun Zhang, R. Hamel William, and Tzyh-Jong Tarn, "Adaptive immune system inspired perimeter patrol control strategy," in *Proceedings of the IEEE/RSJ international conference on Intelligent robots and systems*, 2009, pp. 371 - 376.

[89] J. Kim and P. Bentley, "Immune memory and gene library evolution in the dynamical clonal selection algorithm," *Journal of Genetic Programming and Evolvable Machines*, vol. 5, no. 4, pp. 361-391, September 2004.

[90] D. Dasgupta, "Artificial immune systems: Modeling and simulation," in *Artificial Immune Systems and Their Applications*., 1999.

[91] F. Varela and A. Coutinho, "Second generation immune networks," *Immunology Today*, vol. 12, no. 159, 1991.

[92] M. Ayara, J. Timmis, R. de Lemos, L. N. de Castro, and R. Duncan, "Negative selection: How to generate detectors," in *Proceedings of the 1st International Conference on Artificial Immune Systems*,

Cantebury, UK, 2002, pp. 89-98.

[93] N. K. Jerne, "Towards a network theory of the immune system," *Annals of Immunology*, 1974.

[94] S. Sathyanath and F. Sahin, "Application of Artificial Immune System based Intelligent Multi Agent Model to a Mine Detection Problem," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, vol. 3, Tunisia, 2002.

[95] Y. Watanabe, A. Ishiguro, and Y. Uchkawa, "Decentralized behavior arbitration mechanism for autonomous mobile robot using immune system," in *Artificial Immune Systems Applications*., 1999.

[96] N. Toppo and S.B. Nair, "A Framework for Sharing Intelligence among Mobile Robots on a Network," in *Proceedings of the 2nd IASTED International Multi-Conference on Auto-mation, Control and Information Technology*, 2005, pp. 93-98.

[97] P. Soni, R.K. Rathore, and S.B. Nair, "A Framework for the Realization of Networked Robotics and Human Beings," in *Proceedings of the International Symposium on Automation and Robotics in Constructio*, Kochi, India., 2007, pp. 331-338.

[98] J. Byong kug and Chi. Young geng, "Design and Implementation an efficient migration policy for mobile agent," *Korea Information Processing Science Journal*, vol. 6, no. 7, pp. 1770-1776, 1999.

[99] P.H. Jin, "Design of Agent Migration Information System for Providing Optimal Achievement Order," *Korea Information Processing Science Journal*, vol. 9, no. 4, pp. 345-357, 2002.

[100] W.W. Godfrey and S.B Nair, "A Mobile Agent Cloning Controller for Servicing Networked Robots (Accepted)," *Special Issue of Advanced Materials Research Journal*, 2011.

[101] M. Saravanan, K.V.D. Pradeep Kumar, and S.B. Nair, "A Communication Protocol for a Mobile Adhoc Network of Robots," in *Proceedings of the The International Conference on Emerging Applications of IT*, Science City, Kolkata , 2006, pp. 223-226.

[102] I. Satoh, "A Test-bed Framework for Self-organizing Approaches over Distributed Systems," in *Proceedings of 2nd International Workshop on Nonlinear Dynamics and Synchronization*, Klagenfurt , 2009, pp. 128 - 134.

[103] I. Satoh. (2009) AgentSpace. [Online]. http://islab.is.ocha.ac.jp/agent/index.html

[104] I. Satoh, "A Mobile Agent-Based Framework for Active Networks,"

in *Proceedings of IEEE Systems, Man, and Cybernetics Conference*, 1999, pp. 71-76.

[105] Lejos. (2009) Java for Lego Mindstorms. [Online]. http://lejos.sourceforge.net/

[106] S. Evdokimov, B. Fabian, and O. Günther, "Multipolarity for the Naming Service (reading RFID data)," in *Proceedings of the First International Conference on The Internet of Things*, vol. 4952, 2008, pp. 1-18.

[107] A.M Whitbrook, U. Aickelin, and J.M. Garibaldi, "Idiotypic Immune Networks in Mobile Robot Control," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 37, no. 6, pp. 1581-1598, 2007.

[108] D.J. Toncich, "Local Area Networks - Fundamentals," in *Data Communications and Networking for Manufacturing Industries*.: Chrystobel Engineering, 1993 , ch. 7, pp. 211-246.

[109] S. Poslad, *Ubiquitous Computing Smart Devices, Smart Environments and Smart Interaction*.: Wiley, 2009.

[110] R.R Hightower, S. Forrest, and A.S. Perelson, "The Evolution of Emergent Organization in Immune System Gene Libraries," in *Proceedings of the 6th International Conference on Genetic Algorithms*, 1995, pp. 344--350.

[111] A. Aloulou, "Formal Approach for Specifying, Verifying and Imposing Dynamic Security Policies in Mobile Agent Systems," Faculty of Economic Sciences and Management of Sfax, Tunisia , Dissertation 2010.

[112] O.M. Paracha, "A Security Framework for Mobile Agent Systems," Mohammad Ali Jinnah University, Islamabad, Dissertation 2006.

[113] N.M. Karnik and A.R. Tripathi, "A security architecture for mobile agents in Ajanta," in *Proceedings of the 20th International Conference on Distributed Computing Systems* , Taipei , 2000, pp. 402 - 409.

[114] R.R. Choudhury, K. Paul, and S. Bandyopadhyay, "Marp: A multi-agent routing protocol for mobile wireless adhoc networks," *Autonomous Agents and Multi-Agent System*, vol. 8, no. 1, pp. 47-68, 2004.

[115] Zhang, De-yu, and Z. Liu, "Study on Improved Algorithm for Mobile Agent Migration Path," , 2010, pp. 564-567.

[116] Alberto Sanfeliu, Norihiro Hagita, and Alessandro Saffiotti,

"Network robot systems," *Journal of Robotics and Autonomous Systems*, pp. 793-797, 2008.

[117] Stefan Enderle et al., "Miro: Middleware for autonomous mobile robots," in *Proceedings of the IFAC Conference on Telematics Applications in Automation and Robotics*, 2001.

[118] Carle Côté, Yannick Brosseau, Dominic Létourneau, Clément Raïevsky, and François Michaud, "Robotic Software Integration Using MARIE," *International Journal of Advanced Robotic Systems*, vol. 3, no. 1, pp. 55-60, March 2006.

[119] *Programming for the Series 60 Platform and Symbian OS (Symbian Press).*: John Wiley& Sons, 2002.

[120] Yasushi Kambayashi, Takimoto Munehiro, and Yasushi Kodama, "Controlling biped walking robots using genetic algorithms in mobile agent environment," in *Proceedings of Third IEEE International Conference on Computational Cybernetics*, 2005, pp. 29-34.

[121] A. Silverstein and Paul Ehrlich, "Archives and the history of immunology," *Nature Immunology*, vol. 6, no. 7, pp. 639–639, 2005.

[122] S. Forrest, S. Hofmeyr, and A. Somayaji, "Computer Immunology.," *Communications of the ACM*, vol. 40, no. 10, pp. 88–96, 1997.

[123] Lu Tan and Neng Wang, "Future internet: The Internet of Things," in *Proceedings of the 2010 3rd International Conference on Advanced Computer Theory and Engineering*, vol. 5, 2010, pp. 376-380.

[124] D. R. Carvalho and A. A. Freitas, "An Immunological Algorithm for discovering small-disjunct rules in data-mining," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2001.

[125] L. N. de Castro and F. J. Zuben, "aiNet: An Artificial Immune Network for Data Analysis," in *Data Mining: A heuristic Approach, Chapter 12.*: Idea group Publishing, pp. 231–259.

[126] P. A. Vargas, L.N. de Castro, R. Michelan, and F. J. Von Zuben, "An Immune Learning Classifier Network for Autonomous Navigation," in *Proceedings of the Second international conference on Artificial Immune System*, 2003.

[127] A. Ishiguro, S. Kuboshiki, S. Ichikawa, and Y. Uchikawa, "Gait coordination of hexapod walking robots using mutual coupled

immune networks," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, vol. 2, 1995, pp. 672–677.

[128] A. Ishiguro, S. Ichikawa, T. Shibata, and Y. Uchikawa, "Moderationism in the immune system: Gait acquisition of a legged robot using the metadynamics function," in *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, vol. 4, 1998, pp. 3827–3832.

[129] A. Ishiguro, Y. Shirai, T. Kondo, and Y. Uchikawa, "Immunoid: An architecture for behavior arbitration based on the immune networks," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, 1996, pp. 1730 - 1738.

[130] Emma Hart, Peter Ross, Andrew Webb, and Alistair Lawson, "A role for immunology in "Next Generation" robot controllers," in *Proceedings of the Second international conference on Artificial Immune System*, 2003, pp. 46-56.

[131] W. Lee Dong and K. Sim, "Artificial Immune network-based cooperative control in collective autonomous mobile robots," in *Proceedings of the 6th IEEE International Workshop on Robot and Human Communication*, 1997, pp. 58 - 63.

[132] L. Henry and V. Wong, "Immunologic control framework for automated material handling," in *Proceedings of the Second international conference on Artificial Immune System*, 2003, pp. 57-68.

[133] S. Sathyanath and F. Sahin, "AISIMAM - An Artificial Immune System Based Intelligent Multi Agent Model and its Application to a Mine Detection Problem," in *Proceedings of the 1st International Conference on Artificial Immune Systems*, Canterbury, U.K., 2002.

[134] C.T. Singh and S.B. Nair, "An Artificial Immune System for a MultiAgent Robotics System," *World Academy of Science, Engineering and Technology*, pp. 6-9, 2005.

[135] W. W. Godfrey and S. B. Nair, "Multi Mobile Agent Multi-Robot Network," in *Proceedings of the International Conference on Emerging Trends in Robotic and Communication Technologies*, 2010, pp. 464-467.

[136] W. W. Godfrey and S. B. Nair, "An Immune System Based Multi-robot Mobile Agent Network," in *Proceedings of the 7th international conference on Artificial Immune Systems*, vol. 4128,

2008, pp. 424-433.

[137] D. Portugal and R. Rocha, "A Survey on Multi-robot Patrolling Algorithms," in *Proceedings of the 2nd Doctoral Conference on Computing, Electrical and Industrial Systems*, 2011, pp. 139–146.

[138] K. A. Bharat and L. Cardelli, "Migratory Applications," in *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology*, 1995, pp. 132-142.

[139] Y. Lee and K.J Kim, "Optimal Migration Path Searching using Path Adjustment and Reassignment for Mobile Agent," in *Proceedings of the Fourth International Conference on Networked Computing and Advanced Information Management*, 2008, pp. 564-569.

[140] Tony White, Pagurek Bernard, and Dwight Deugo, "Management of Mobile Agent Systems using Social Insect Metaphors," in *Proceedings of the 21st IEEE Symposium on Reliable Distributed Systems*, 410 - 415, p. 2002.

[141] Napster Website. [Online]. http://www.napster.com

[142] Ioan Susnea, Grigore Vasiliu, Adrian Filipescu, Adriana Serbencu, and Adrian Radaschin, "Virtual Pheromones to Control Mobile Robots, A Neural Network Approach," in *Proceedings of the IEEE International Conference on Automation and Logistics*, 2009, pp. 1962 - 1967.

[143] David Payton, Mike Daily, Regina Estowski, Mike Howard, and Craig Lee, "Pheromone Robotics," *Autonomous Robots*, vol. 11, no. 3, pp. 319-324, 2001.

[144] Anies Hannawati Purnamadjaja and Andrew R. Russell, "Bi-directional pheromone communication between robots," *Robotica*, vol. 28, pp. 69–79, 2010.

[145] Mesut Gunes and Otto Spaniol, "Ant-routing-algorithm for mobile multi-hop ad-hoc networks," in *Network control and engineering for Qos, security and mobility II.*: Kluwer Academic Publishers, pp. 120-138.

[146] M. Flores-Badillo, A. Padilla-Duarte, and E. Lopez-Mellado, "A Population Control Protocol for Mobile Agent Based Workflow Automation," in *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, 2009, p. 4059—4064.

[147] C. de Paula Bianchini, "Intelligent management of network devices aided by a strategy and a tool," in *Proceedings of IFIP/ACM Latin America conference on Towards a Latin American agenda for*

network research, 2003.

[148] L. Zhong, M. Sinclair, and N. K. Jha, "A personal-area network of low-power wireless interfacing devicesfor handhelds: system and hardware design," in *Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, 2005, pp. 251-254.

[149] P. Vuorinen, "Applying the RFID technology for field force solution," in *Proceedings of joint conference on Smart objects and ambient intelligence*, 2005, pp. 39-41.

[150] C. Kürschner, C. Condea, O. Kasten, and F. Thiesse, "Discovery Service Design in the EPC global Network (Supply chain visibility)," in *Proceedings of the First International Conference on The Internet of Things*, vol. 4952, 2008, pp. 19-34.

[151] E. Grummt and M. Müller, "Fine – Grained Access Control for EPC Information Services," in *Proceedings of the First International Conference on The Internet of Things*, vol. 4952, 2008, pp. 35-49.

[152] A. Dada and F. Thiesse, "Sensor Applications in the supply Chain," in *Proceedings of the First International Conference on The Internet of Things*, vol. 4952, 2008, pp. 140-154.

[153] L. M. S. de Souza et al., "SOCRADES:A Web Service Based Shop Floor Integration Infrastructure," in *Proceedings of the First International Conference on The Internet of Things*, vol. 4952, 2008, pp. 50-67.

[154] Sudha Krishnamurthy et al., "Automation of facility Management Process Using Machine-to –Machine Technologies," in *Proceedings of the First International Conference on The Internet of Things*, vol. 4952, 2008, pp. 68-86.

[155] M. Rothensee, "User Acceptance of the Intelligent Fridge," in *Proceedings of the First International Conference on The Internet of Things*, vol. 4952, 2008, pp. 123-139.

[156] R. A. Brooks and T.Lozano-Pérez, *Solid Modeling by Computers*, Pickett and Boyse, Ed. New York: Plenum Press, 1985.

[157] R. A. Brooks, *Guest Editorial in Manufacturing Engineering*., March 1988, vol. 148.

[158] R. A. Brooks, "The Role of Learning in Autonomous Robots," in *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, Santa Cruz, CA, August 1991, pp. 5–10.

[159] R. A.Brooks and A. M.Flynn, "Robot Beings," in *Proceedings of*

the International Conference on Intelligent Robots and Systems, Tskuba, Japan, September 1989, pp. 2–10.

[160] R.A. Brooks, "Steps Towards Living Machines," in *Proceedings of the Evolutionary Robotics 2001 Symposium*, October, 2001, pp. 72–93.

[161] R.A. Brooks, "A Layered Intelligent Control System for a Mobile Robot," in *Proceedings of the Third International Symposium of Robotics Research*, Gouvieux, France, October 1985, pp. 1–8.

[162] M.J. Mataric, "Issues and approaches in the design of collective autonomous agents," *Robotics and Autonomous Systems*, vol. 16, pp. 321-331, 1995.

[163] S.Singh,D.Hershberger,J.Ramos, and T. Smith R.Simmons, "First results in the coordination of heter-ogeneous robots for large-scale assembly," in *Proceeding of the Seventh International Symposium on Experimental Robotics*, New York, 2000.

[164] O. Holland, and J. Deneubourg R. Beckers, "From local actions to global tasks: Stigmergy and collective robotics," in *Proceedings of the 14th International Workshop on the Synthesis and Simulation of Living Systems*, Cambridge, 1994, pp. 181–189.

[165] M.J. Huber and E. Durfee, "Deciding when to commit to action during observation-based coordination," in *Proceedings of the 1st International Conference on Multi-Agent Systems*, 1995, pp. 163–170.

[166] J. McLurkin and J. Smith, "Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots," in *Symposium Distributed Autonomous Robotics Systems*, 2004.

[167] J. F. Montgomery, and R. T. Vaughan G. Sukhatme, "Experiments with Cooperative Aerial-Ground Robots," in *Robot Teams: From Diversity to Polymorphism*, L. E. Parker T. Balch, Ed., 2002, pp. 345–368.

[168] L.E. Parker and F. Tang, "Building multi-robot coalitions through automated task solution synthesis," *IEEE special issue on Multi-Robot Systems*, vol. 94, no. 7, pp. 1289–1305, 2006.

[169] F. Fernandez and L.E. Parker, "A reinforcement learn-ing algorithm in cooperative multi-robot domains," *Journal of Intelligent Robotic Systems*, vol. 43, pp. 161–174, 2005.

[170] M.J. Mataric B. Gerkey, "A formal analysis and taxonomy of task allocation in multi-robot systems," *International Journal of*

Robotics Research, vol. 23, no. 9, pp. 939–954, 2004.

[171] C.F. Touzet, "Robot awareness in cooperative mobile robot learning," *Autonomous Robotics*, vol. 2, pp. 1-13, 2000.

[172] S. Cammarata, F. Hayes-Roth, P. Thorndyke and R. Wesson R. Steeb, "Distributed Intelligence for Air Fleet Control," Rand Corporations, R-2728 SFPA , 1981.

[173] M. Tin, L. Li, N. Gu and S. Wang Z. Cao, "Cooperative hunting by distributed mobile robots based on local interaction," *IEEE Transaction on Robotics*, vol. 22, no. 2, pp. 403–407, 2006.

[174] V. Jagannathan, R. Dodhiawalla M. Benda, "On optimal cooperation of knowledge sources," Boeing AI Center Technical Report, 1985.

[175] T. Haynes and S. Sen, "Evolving Behavioral Strategies in Predators and Prey," in *Adaptation and Learning in Multi-Agent Systems*, S. Sen G. Weiss, Ed. Berlin, Heidelberg: Springer, 1986, pp. 113–126.

[176] S. Mahadevan and J. Connell, "Automatic programming of behavior-based robots using reinforcement learning," in *Proceedings of the American Association for Artificial Intelligence Conference*, 1991, pp. 8-14.

[177] M. Mataric, "Behavior-based control: Examples from navigation, learning, and group behavior," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 19, no. 2-3, pp. 323–336, 1997.

[178] P. Stone and M. Veloso, "A layered approach to learning client behaviors in the RoboCup soccer server," vol. Applications of artificial intelligence, no. 12, pp. 165–188, 1998.

[179] J. Adibi, Y. Al-Onaizan, G. Kaminka, I. Muslea and M. Tambe S. Marsella, "On being a teammate: Experiences acquired in the design of RoboCup teams," in *Proceedings of the 3rd Annual Conference on Autonomous Agents*, 1999, pp. 221–227.

[180] S. Fleury, M. Herrb, F. Ingrand and F. Robert R. Alami, "Multi-robot cooperation in the MARTHA project," *IEEE Robotics & Automation Magazine, ,* vol. 5, no. 1, pp. 36–47, 1998.

[181] A. Okon, M. Robinson, T. Huntsberger, H. Aghazarian, and E. Baumgartner A. Stroupe, "Sustainable cooperative robotic technologies for human and robotic outpost infrastructure construction and maintenance," *Autonomous Robots*, vol. 20, no. 2, pp. 113–123, 2006.

[182] R.R. Murphy, "Marsupial robots for urban search and rescue," *IEEE Intelligent Systems Magazine*, vol. 15, no. 2, pp. 14–19, 2000.

[183] S.Singh, D.Hershberger, J.Ramos and T. Smith R.Simmons, "First results in the coordination of heterogeneous robots for large-scale assembly," in *Proceedings of the Seventh International Symposium on Experimental Robotics*, New York, 2000.

[184] L.E. Parker and R. Madhavan Y. Guo, "Towards collaborative robots for infrastructure security applications," in *Proceedings of the International Symposium on Collaborative Technologies and Systems*, 2004, pp. 235–240.

[185] P.R. Wurman and R. D'Andrea C. Hazard, "Alphabet Soup: A testbed for studying resource allocation in multi-vehicle systems," in *Proceedings of the Association for the Advancement of Artificial Intelligence Workshop on Auction Mechanisms for Robot Coordination*, Boston, 2006, pp. 23–30.

[186] M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa and H. Matasubara H. Kitano, "RoboCup: A challenge problem of AI," *AI Magazine*, vol. 18, no. 1, pp. 73-86, 1997.

[187] H. Kitano and S. Tadokoro, "RoboCup rescue: A grand challenge for multiagent and intelligent systems," *AI Magazine*, vol. 22, no. 1, pp. 39-52, 2001.

[188] M. Zhou and N. Ansari Z. Wang, "Ad-hoc Robot Wireless Communication," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 2003.

[189] Y. Charlie Hu, C.S. George Lee, and Yung-Hsiang Lu Saumitra Das, "Efficient Unicast Messaging for Mobile Robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2005.

[190] R. A. Brooks, "How To Build Complete Creatures Rather Than Isolated Cognitive Simulators," in *Architectures for Intelligence*, K. VanLehn, Ed. Erlbaum, Hillsdale, NJ, Fall 1989, pp. 225–239.

[191] R. A. Brooks, "Intelligence Without Representation," in *Preprints of the Workshop in Foundations of Artificial Intelligence*, Dedham, MA, June, 1987.

[192] R. T. Laird, D. M. Carroll, G. A. Gilbreath, T. A. Heath-Pastore, R. S. Inderieden, T. Tran, K. J. Grant and D. M. Jaffee H. R. Everett, "Multiple Resource Host Architecture (MRHA) for the Mobile Detection Assessment Response System," SPAWAR Systems,

3026, 2000.

[193] R. A. Brooks, "AI: Great Expectations," *Guest Editorial in Manufacturing Engineering*, vol. 148, March 1988.

[194] P. Maes and R.A.Brooks, "Robot Insect Societies," *Data Manager Magazine*, May, 1990.