On Mobile Agents for Learning & Coordination in a Networked Robotics Milieu

Thesis submitted in partial fulfilment of the requirements for the award of the degree of

Doctor of Philosophy

in

Computer Science and Engineering

by

Shashi Shekhar Jha

Under the supervision of

Prof. Shivashankar B. Nair

Department of Computer Science and Engineering

Indian Institute of Technology Guwahati

Guwahati - 781039 Assam India

October, 2016

Dedicated to

My beloved Parents,

My sweet Sister

and

My charming Brother

It is from their love, care and support, I derive my strengths...

Acknowledgements

I would like to express my heartfelt gratitude to my supervisor *Prof. Shivashankar B. Nair* who nurtured me from a mere novice to a mature researcher. I am indebted to his consistent support and care, expert guidance, years of experience and wisdom. His encouragement to always complete that extra mile, to have a strong stand on ethics, and to be impeccable in every respect will reverberate throughout my life.

I am highly grateful to *Prof. Pradip Kr. Das* for his invaluable support and encouragement throughout my PhD. His discussions always helped me to see beyond the boundaries of my work. I would like to acknowledge the other members of my Doctoral Committee - *Dr. Pinaki Mitra*, *Dr. V. Vijaya Saradhi* and *Prof. Harshal B. Nemade* whose timely suggestions and constructive feedbacks helped me to improve on my work and clarity on my subject. I am also thankful to the anonymous reviewers of my research works in various forums for their critical comments which helped me to add quality to my work.

I would also like to acknowledge the heads of the department of Computer Science and Engineering at IITG during my PhD work - Prof. Purandar Bhaduri, My own supervisor Prof. Shivashankar B. Nair and Prof. Diganta Goswami for their support with respect to the departmental resources and facilities. I would like to thank the staff of the department of Computer Science and Engineering - Nanu Alan Kachari, Bhriguraj Borah, Hemanta Kr. Nath, Nava Kumar Boro, Raktajit Pathak, Pranjit Talukdar and Prabin Bharali who were always approachable and welcoming whenever I sought their help. I am also thankful to the entire Computer Science and Engineering Department, security persons and other staff members for their help and support.

I would like to gratefully acknowledge Tata Consultancy Services (*TCS*) and *MHRD*, *Govt. of India* for their financial support rendered throughout my years of PhD without which this research could not have taken shape. I would also like to acknowledge TCS for the travel grants which helped me to present my research work at the national and international levels. I am thankful to Mr. Sachin Parkhi, Program Manager, TCS Research Scholar Program who was always prompt to reply my emails and resolved any issues related to TCS grant in a timely manner. I am grateful to the FIST Project of Department of Science and Technology, India for facilitating infrastructure support for the Robotics Lab. where I carried out my research work.

I would like to acknowledge the Academic Registrar, IITG, Mr. Sunil Kumar Barua for ratifying any issues at his earliest. Mr. Barua always personally looked into the issues until the very end that they get settled. I would also like to thank the satff at the Academic Affairs office who were supportive to process my applications and grant requests. I am grateful to Mr. Sunil Sarma and Ms. Kajallata from Finance and Accounts office of IITG for fecilitating their support regarding all my claims and related issues.

I am thankful to my lab-mates at the Robotics Lab. - Godfrey, Vibhor, Surjeet, Kunal, Sourabh singh, Saurabh prajapati, Akash, Shyam, Akshay, Jatin, Tonmoy, Mohit, Abhay, Manoj, Nikhil, and Tushar. It was a great experience to work with them in the lab. Their companionship always encouraged me to sharpen my skills and move forward in my research. I would also like to thank the research scholars of the department of Computer Science and Engineering for creating a warm atmosphere of mutual support and encouragement.

I am deeply indebted to my friends at IIT Guwahati whose love and care transcends the years of ordeal of a PhD. I find myself fortunate to have found such friends. I would like to mention - *Shrinivas* sir, *Pravati* and *Mamata* di who were my seniors, my friends and my guardians at IIT Guwahati and who always took care of me as a family. Their compassion and love filled my stay at IIT Guwahati with loads of joy and resounding happiness. I am thankful to God for giving me buddies cum batchmates like *Nilkanta*, *Shirshendu*, *Mayank* and *Shilpa* who added colours to my PhD life and made this journey a memorable one. I feel grateful to have met *Debanjan*, *Basant*, *Satish*, *Shivaji*, *Tushar* and *Achyut* with whom I shared a good companionship. I would also like to acknowledge my friends at dibang hostel of IIT Guwahati - *Mridul* and *Jitendra* whose persistent interactions made my stay pleasurable.

I am grateful to my friends from school - *Chander*, *Ranjeet*, *Ravi*, *Kush*, *Swati*, *Sharda* and *Ramjee* who did not loose touch with me and always extended their support and encouragement through all these years. I also like to acknowledge my friend - *Naval* who always comes up with a technical challenge and keeps on motivating me to achieve great heights.

It goes without saying that this journey would not have been possible without the persistent support, the unconditional love and profound encouragement of my mother, father, sister and younger brother. They never doubted my intentions and whole-heartedly supported me in all my endeavours. I fall short of words to express my gratitude to them.

October 16, 2016 Shashi Shekhar Jha

Declaration

I certify that

- The work contained in this thesis is original and has been done by myself and under the general supervision of my supervisor(s).
- The work reported herein has not been submitted to any other Institute for any degree or diploma.
- Whenever I have used materials (concepts, ideas, text, expressions, data, graphs, diagrams, theoretical analysis, results, etc.) from other sources, I have given due credit by citing them in the text of the thesis and giving their details in the references. 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 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 in a position to check for any possible instance of plagiarism within this submitted work.

October 16, 2016 Shashi Shekhar Jha

Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
Guwahati - 781039 Assam India

Dr. Shivashankar B. Nair
Professor
Email : sbnair@iitg.ernet.in
Phone : +91-361-258-2356

Certificate

This is to certify that this thesis entitled "**On Mobile Agents for Learning & Coordination in a Networked Robotics Milieu**" submitted by **Shashi Shekhar Jha**, in partial fulfilment of the requirements for the award of the degree of Doctor of Philosophy, to the Indian Institute of Technology 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, Assam, India. 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.

Date: October 16, 2016
Place: Guwahati

Prof. Shivashankar B. Nair
(Thesis Supervisor)

Abstract

Networked robots have a wide variety of applications ranging from industry automation, manufacturing and construction, autonomous port and warehouse management to planetary exploration and military applications. The advancement in computing and communication capabilities together with the availability of enhanced sensors have propelled the development of networked robotic systems for real-world applications. Coordination of activities along with learning and sharing in these systems is a notable domain of research. Although several mechanisms have been proposed in the literature for coordination and control among the entities comprising networked robots, very few attempts have been made to realize solutions using mobile software agents or simply mobile agents. It has been shown that the mobile agents provide various distinguishing features in terms of localized decision making and limited communication that stands out against the traditional approaches. As both these domains viz. networked robotics and mobile agents, have various intrinsic similarities, such as distributed and asynchronous behaviours, mobile agent based mechanisms can be well exploited in networked robotic scenarios.

This thesis describes a paradigm that amalgamates mobile agents with networked robotic systems and provides for mechanisms that can be used in various applications. Although researchers have proposed mobile agents based mechanisms for providing distributed services to the entities of networked robotic systems, the aspects of exploiting mobile agents for sharing of information, distributed learning and coordination seem to be grossly missing. These agents carry solutions or programs for various service requirements of the robots in networked robotic systems and deliver them to the robots as and when required. However, it is possible that there could be multiple solutions available for similar problems or requirements of the robots. In such a case, it is always beneficial to choose or deliver the solution which provides the best outcome.

The first contribution of this thesis discusses a mechanism to evolve the best of a given set of solutions for the networked robotic system using mobile agents. The proposed mechanism is inspired from models based on the Artificial Immune Systems specifically the Idiotypic network. Mobile agents are viewed synonymous to antibodies while the service requirements occurring in the network of robots are modelled as antigens. The mutual interactions among the mobile agents carrying solutions for the exchange of stimulations and suppressions signals are used for selecting the best performing mobile agent. The whole mechanism was emulated over a real network test-bed. The results obtained from the experiments depict not only the convergence on to the best solution in the network of robots but more importantly the proliferation of the more generic solutions which can cater to more than one service requirement.

While learning to find generic solutions from a given set is an essential requirement, sharing of information gained by mobile agents during their migration within the network can greatly accelerate the learning gradient. The thesis thus presents a framework to achieve distributed and asynchronous sharing of intelligence and consequent learning using mobile agents within a networked robotic system. The framework allows the users of the system to define an objective and the manner in which the gathered information should be used for learning. The framework further facilitates cooperative learning using the sharing mechanisms embedded within the mobile agents. This framework couples localized agent-agent communication with the available mobile agent based techniques to realize asynchronous intelligence-sharing and learning. The framework

was emulated for both static as well as dynamic networks of robots and experiments performed substantiate the applicability of this framework in real distributed mobile computing environments comprising networked robots.

Synchronized execution of a sequence of tasks is a major issue in multi-robot systems especially since robots do not work using a common clock and thus act asynchronously. Such synchronized execution of a sequence of tasks in a network of robots is thus addressed next in the thesis. A set of mobile agents regulates the synchronized execution of the tasks in the network of robots. These agents are equipped with the mechanisms to execute the tasks in the sequence along with the order of the execution of tasks. The synchronization ensures that the execution of the next task in sequence does not ensue until the executions of the previous tasks are completed by all the robots in the network. The proposed mechanism uses the concept of stigmergy to facilitate communication among the mobile agents populating the network. The mechanism also inherently facilitates self-healing in case of failures. Emulation experiments on both static and dynamic networks were carried out. In addition this mechanism was also implemented using real robots. The results obtained by varying the different parameters used, such as number of robots in the network, number of tasks and agents, etc., validate the feasibility of the proposed mechanism in real systems. The mechanism of synchronized task execution has been extended further to augment the same with a population control strategy for mobile agents. In this extended mechanism, the mobile agents are provided with partial information about the sequence of tasks and are allowed to clone to increase their population as per the current demand. A stigmergy-based cloning controller governs the cloning of on-demand mobile agents to facilitate parallel executions. This alleviates the problem of finding the number of mobile agents required to regulate the executions of tasks. The extended mechanism too has been emulated on both static and dynamic networks. The results obtained were found to be in accordance with those from the earlier approach.

With learning, synchronization and cloning control in place, the next obvious step would be to provide a solution to task allocation in such robotic networks. Though several algorithms for task allocation have been reported, they all assume that only one set of tasks is issued to the robotic network at any point of time. In the real world simultaneous multiple task allocations could occur in such networks. The last contribution of this thesis thus focuses on the case of multiple instances of the allocation of tasks to the robots in a large heterogeneous networked robotic system. The proposed mechanism for task allocation using mobile agents uses an intentional market based strategy to allocate tasks to the heterogeneous robots in the network. The contentions among the competing instances of task allocation are resolved using voluntary and asynchronous back-offs on part of the mobile agents. The proposed mechanism was emulated along with two other prominent mechanisms reported in the literature. The results obtained from the experiments show that the mobile agent based mechanisms outperform the existing ones in terms of both number of inter-node communications and task completion times.

All experiments performed herein, were *emulated* using *Typhon*, a mobile agent based framework. The thesis concludes by summing up the contributions made and discusses the future avenues of research and their possible applications.

Contents

List of Figures

List of Algorithms

List of Tables

List of Symbols

Symbols	Description
ψ_{p_i}	Affinity function for problem p_i
ξ_{p_i}	Feedback function for problem p_i
Γ_{p_i}	Affinity threshold for problem p_i
γ_{m_i}	Endurance potential for mobile agent m_i
γ_{max}	Maximum value of Endurance potential
γ_{min}	Minimum value of Endurance potential
η_H	Hamming Distance
$S^t_{m_i}$	Stimunaltions accumulated by mobile agent m_i
$S^u_{m_i}$	Suppressions accumulaated by mobile agent m_i
E_{Ag}	Epitope of an Antigen
P_{An}	Paratope of an Antibody
W	Network of nodes
N	Set of nodes
E	Set of links or edges
A	Set of static agents
M	Set of mobile agents
I	Set of Integers
P	User defined learning problem
G	Learning goal of the system

L	Learning algorithm provided by the user
φ	Set of actions executed by static agents
Q_{n_i}	Queue of a node n_i
Γ	Length of the queue Q_{n_i}
S^{m_i}	Sharable intelligence of mobile agent m_i
B^{m_i}	Bag of collected information carried by mobile agent m_i
ρ_{m_i}	Migration resource of mobile agent m_i
ξ_{m_i}	Execution potential of mobile agent m_i
ϕ_w	Function which returns the weight of a piece-wise information
Π_{m_i}	Weight of the Bag B^{m_i}
SV	Sensor Vector within the *Maze-World*
T	Set of tasks
μ	Number of mobile agents
τ	Number of distinct tasks to be serviced
$P_{T_i}^k$	Potential of task T_i for k^{th} mobile agent
P_{max}	Maximum value of potential
$\theta_{T_i}^k$	Reinforcement of task T_i for k^{th} mobile agent
H	History Window
ω	Length of H
δ	Penalty bias
A^T	Set of heterogenous mobile agents
C_R	Cloning Resource
C_R^{max}	Maximum cloning Resource
C_R^{min}	Minimum cloning Resource

L_t	Lifetime of cloned agent
R_d	Reward for population growth
η_c	Number of clones
q_{th}	Queue Threshold
Ψ	Action potential
R	Set of heterogenous robots
χ	Mobility Resource
μ^r	Frequency of rewards
μ^p	Frequency of penalties
N^r	Number of robots required
N^q	Number of robots reserved
U_R	Number of distinct robots discovered
V_R	Total number of robots visited
BI^i	Back-off Interval for the agent i
BI_{max}	Maximum value of back-off interval
BI_{min}	Minimum value of back-off interval
Δ^R	Current hop-time at the robotic node R

List of Abbreviations

Terms	Abbreviations
SRS	Single Robot System
MRS	Multi-Robot System
IoT	Internet-of-Things
CPS	Cyber-Physical System
MAS	Multi-Agent System
DAI	Distributed Artificial Intelligence
RDE	Robotic Development Environments
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
RPC	Remote Procedure Call
FIPA	Foundation for Intelligent Physical Agents
LAN	Local Area Network
CAN	Campus Area Network
TCP/IP	TCP/IP protocol suite
BIS	Biological Immune System
AIS	Aritificial Immune System
PC	Personal Computer
CNP	Contract Net Protocol

Glossary of Terms

Terms	Description
Artificial Immune System (AIS)	Computational Systems having models and algorithms inspired from Biological Immune Systems.
Immune Network Theory (Idiotypic Network)	This theory postulates that the antibodies are in constant interaction with one-another using stimulations and suppressions signals
Antigens	Foreign substance or pathogens responsible for infection or disease in an organism which triggers an immune response.
Antibody	The Y shaped proteins in the immune system which bind to infectious antigens leading to their destruction.
T-Cells	The immune cells generated in the thymus of the body responsible of recognizing the antigens.
B-Cells	The immune cells generated in the bone-marrow of the body responsible for secondary immune response.
Epitopes	The part of an antigen which is recognized by antibody receptors.
Paratopes	The part of an antibody which binds with the antigens.
Idiotopes	The part of an antibody which binds with the paratopes of other antibodies to create an Idiotypic network.
Internet-of-Things (IoT)	An interconnection of various devices to exchange information.
Cyber-Physical Systems (CPS)	Integration of computational, networking and physical processes as a unified entity of a system.

Mobile Agents	Autonomous computer programs or chunks of code with the capability to migrate a network of nodes.
Idle or non-productive periods	The period during which the mobile agents moves in the network without carrying out any execution.
Typhon	A mobile agent framework which supports mobile agent migration, cloning, execution, etc. over a LAN.
Maze-World	A grid environment with each cell of the grid having a set of sensor values.
Stigmergy	An indirect form of communication through the sensing of the stimuli available within the environment.
Back-off	An event which suspends all the activities of a mobile agent and makes it dormant.
Murdoch	An auction based multi-robot task allocation strategy using a publish-subscribe model.
ANTA	Adhoc Network based Task Allocation for MRS using a tree based method for dissemination of trade messages and elicitation of bids.

"You have to dream before your dreams can come true."

A.P.J. Abdul Kalam (1931 - 2015)
Indian scientist and leader

1

Introduction

Robots are increasingly gaining ground into the human space [2] as an offshoot of the technological developments in on-board computing, affordable communication and multi-modal sensing. Robots, which are primarily designed to facilitate and support humans for finishing industrious (sometimes inaccessible) tasks in order to increase the overall productivity, are being employed in a variety of applications including planetary exploration [3], manufacturing and construction [4, 5], agriculture [6], military operations [7], medical assistance [8], transportation [9], entertainment [10], service and personal use [11], elderly care [12], search and rescue [13], and warehouse automation [14]. The research in the field of robotics have thus fanned out into various sub-fields such as robotic manipulators [15], humanoid robots [16], surgical robots [17], swarm robots [18], tele-robotics [19], rehabilitation robots [20], human-robot interaction [21], etc. The advent of other emerging technologies viz. biotechnology and nanotechnology, have further contributed to the development of nano-robotics [22] as an emerging field of study wherein minuscule robots would become capable of providing targeted treatment to tumours without any surgery. These nano-robots may also work at the cellular level for research in genetics and allied areas.

The proliferation of the networks of sensors and devices and the availability of cheap embedded technology have aided in diversifying the capacities of robotic systems with the growth of smart environments such as smart buildings [23], smart cities [24], etc. Since automation of services is one of the key issues in these smart environments, it inherently makes robots an important constituent of such systems. These smart environments have also given rise to newer domains of research such as Internet-of-Things (IoT) [25] and Cyber-Physical Systems (CPS) [26]. While IoT emphasizes on the inter-communication amongst a gamut of devices (including robots) over the Internet throughout the world, the thrust in CPS is to integrate the computational (cyber) entities and physical processes under the umbrella of a communication framework. The International Telecommunication Union (ITU) Internet Report [27] on IoT lists robotics as one of the enabling technologies in this field. Robots also form a crucial aspect of CPS which provides distributed control for the robots to function as a collective system [28]. Thus the applications of robotic systems in the context of IoT and CPS are abound. Recently, there has been some endeavours to create cloud robotic systems [29, 30, 31, 32]. The basic concept of cloud robotics [33] is to use the cloud infrastructure such as cloud computing, cloud storage, remote data centres, etc. for off-loading tasks and information, from the robots, which are not required in real-time. Whenever, the robots need these information or requires a high level and powerful computation (such as 3D map generation), it communicates with the cloud systems and acquires the much needed knowledge. Hence, these remotely located robotic brains enhances the utilities of robots with very minimum

complexity in its design. Cloud robotics also encourages mutual sharing among remotely located robots, a phenomenon being referred as the social network of robots [34].

Hence, the realm of robotics, that had started to facilitate industries with automated factory floor workers, has evolved as an extremely dynamic field with diversified interests and applications. As the world is progressing towards a digital era, various novel approaches are being innovated to create simpler yet robust robotic solutions. The primary focus now is to bring robots out of the structured environments of the laboratories and deploy them in real environments so that they can assist and work with humans. The spinoff of autonomous car or intelligent automobile vehicle [35] is one such example. Canonically, the research in robotics can be segregated into two broader categories viz. Single Robot Systems and Multi-Robot Systems.

1.1 Single Robot Systems (SRS)

SRSs focus on modelling the interactions of individual robots with their environments. These involve integration of multiple sensors on to a single robot along with complex control mechanisms and algorithms (executing on-board) to analyse the perceived sensors values for performing physical actuations. Hence, SRSs are designed for targeted applications wherein the completion of tasks solely depends on the capabilities of individual robots. The contemporary examples of SRSs include ASIMO [36], BigDog [37], NAO [38], Roomba [39], Baxter [40], KUKA [41] and PR2 [42].

Traditionally, SRSs are modelled using the deliberative paradigm of Sense-Plan-Act (SPA) [43]. A robot initially gathers information from the environment using its sensor. This sensory information is used to create a world model for updating the robots policy to choose the next appropriate action. Finally, the robot performs the chosen action using its actuators. Another paradigm to model SRSs is the reactive paradigm [43] which focuses on Sense-Act based behaviours. The robot is supplied with a pool of such behaviours. The final action of the robot is an aggregated effect of multiple behaviours triggered based on environmental stimuli. Within the reactive paradigms, the pioneering work done by Rodney A. Brooks needs a special mention for propounding the concept of model free Subsumption architecture for SRSs [44, 45]. Brooks emphasized that instead of having a centralized control unit which makes all the decisions, the robots should have segregated functional units which concentrate on specific behaviours. He based his layered control architecture on the dogmatic principle that

> "Complex (and useful) behavior need not necessarily be a product of an extremely complex control system. Rather, complex behavior may simply be the reflection of a complex environment. It may be an observer who ascribes complexity to an organism-not necessarily its designer." [46]

Other than these two paradigms, SRSs are also modelled using the hybrid paradigm which includes the elements of both the deliberative and reactive paradigms [43]. Such systems make use of deliberative paradigm for planning and task decomposition while the reactive paradigm is used to suitably assign/choose behaviours to accomplish various subtasks. The deliberative planners take advantage of the world model to create appropriate behavioural sets. These sets are then used for interacting with the environment. In a hybrid SRS model, the sensor data are available to the individual behavioural sets as well as the deliberative planner. Akrin and Makenzie describe the advantages of a hybrid paradigm as:

> "....This creates greater flexibility for a reactive robot by allowing a high-level deliberative planner to configure the robot's behaviors in accordance with the task at hand, known or anticipated environmental conditions, and available robotic resources." [47]

Although SRSs have higher efficiencies for applications requiring much localized manipulations, they are limited in their capabilities for spatial coverage. Hence, the tasks which are distributed in the environment and require coordination over larger areas become too complex or even impossible for SRS to perform. SRSs also possess a high risk of failure because of the single point of control. In addition, for SRS, it is very difficult to overcome the problems posed by sensory noise, and uncertain and dynamic environments due to the lack of the complete global view of the surroundings.

1.2 Multi-Robot Systems (MRS)

MRSs involve multiple robots dispersed in the environment which can be homogeneous [48] or heterogeneous [49] depending upon the application under consideration. This allows to blend in the diverse capabilities of various robots within an MRS in order to achieve complex tasks and goals [50]. MRSs provide various distinguishing advantages over SRSs such as system reliability, scalability, inherent parallelism, flexibility and versatility [51]. They are capable of accomplishing those spatially segregated tasks which cannot be executed using SRSs. MRSs are not only more efficient in performing tasks but are also more resilient and robust towards task failures (due to the presence of redundancy among robots) and flexible towards task executions. MRSs also provide the advantage of distributed sensing and control which enhances the task completion times and optimizes the use of resources.

Coordination and organization of activities is a crucial aspect of MRS research. A well-coordinated MRS can enhance the productivity of all the robots as a whole with optimal or better quality solutions, faster problem solving and increased robustness [51]. Sharing of information, mutual learning, allocation of tasks, synchronization of behaviours and interaction with other objects (sensors and devices) embedded within the environment constitute other facets of research within the MRS domain. In order for the robots within an MRS to work efficiently, one needs to overcome various formidable challenges. The robots within the MRS 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 an MRS perform in a complex and distributed manner. Hence, this calls for efficient and simple frameworks to interconnect the high level behaviours with the low level physical functionalities of various robots and devices so as to ease the development of real world MRS applications. Such frameworks called as *Robotic Middleware* aid in the development of robotic systems with richer and versatile functionalities.

1.2.1 Middleware

Formally, a robotic middleware is defined as an abstraction layer sandwiched in between the operating system or the firmware of the robotic device and the user-defined software applications [52]. Various robotic middleware platforms exists viz. RT-Middleware [53], Player [54], CLARty by NASA [55], Miro [56], OpenRDK [57], Marie [58], Orca [59], Webots [60], RoS [61], Orocos [62], iRobot Aware [63], RSCA [64], etc. The central design goals of a robotic middleware is to integrate the heterogeneity of the set of robots and devices, support for rapid application development and prototyping, dynamic access control, behaviour coordination, multi-modal interfaces, security, resource management, fault detection and recovery, and ease of application development [52, 65]. An efficient robotic middleware facilitates the developer of the robotic applications to write application programs which are easily deployable. In addition, the middleware should also support replacements, extensions, removals and additions of various components from/within the MRS application.

The authors in [66] describe the major benefits of having a robotic middleware for an MRS as - modularity in the design of application software wherein different components can work in a plug and play manner, hardware abstraction so that the developer can simply concentrate on writing the application software without caring about the device specific details of the robotic hardware, standardized APIs to provide easily accessible functionalities, platform independence to cater to a variety of hardware and software within an MRS, and application portability so that the robotic applications can be ported to a variety of platforms with minimal changes to the system's configuration and attributes. Hence, the major challenges in the robotic middleware design [52, 66] include - simplifying the developmental process, support communication and interoperability, providing efficient utilization of available resources, providing heterogeneity with other systems, offering often-needed robot services, providing automatic resource discovery and configurations and supporting embedded components and low level devices. The middleware should have a wide coverage in terms of its operability and should not be dependent on the underlying hardware or software. Scalability, reusability of components, robustness, ease of use and maintenance, support for parallel and concurrent tasks along with synchronous and asynchronous executions are other desirable features of a robotic middleware.

1.2.2 MRS Characterizations

MRSs are best suited for tasks which require complex coordination for their execution such as localization [67], foraging [68], patrolling [67], target tracking [69], object transportation [70], mapping and exploration [71], formation control [72], etc. The MRS team architectures may be designed based on one of several existing philosophies which include centralized, hierarchical, decentralized and hybrid [43]. A completely decentralized and distributed coordination among the robots within MRSs allows for scaling up to large number of robots and objects. On the other hand, a centralized coordination model for MRS provides a single point of control with tightly coupled elements which creates computational and communication overheads. The authors in [51] argue that the issues within an MRS should not be considered as a special case of Multi-agent systems (MAS) or Distributed Artificial Intelligence (DAI). MAS and DAI refer to the issues within the set of situated stationary agents distributed in a computational environment whereas MRS deals with the physical interactions with the real-world environment along with the uncertainties, noise and dynamicity within. Farinelli et al. [73] provide a taxonomy for MRS and classify MRS into two dimensions viz. coordination dimensions and system dimensions. The coordination dimension, which has been further classified into four levels viz. Cooperation, Knowledge, Coordination and Organization, characterizes the form of coordination used within the MRS. The system dimension which focus on team or group development has been further grouped according to communication, team composition, system architecture and team size.

In [51], the authors categorize MRS environment as cooperative and competitive. In cooperative environments, multiple robots operate in tandem to achieve a common global objective. Hence in cooperative MRS environments, all the robots have a top level single objective whereas there can be multiple sub goals at the bottom level. Multi-robot localization [67], Multi-robot exploration [71], Multi-robot search and rescue [13], and Multi-robot transportation [70] are some of the examples of cooperative MRS environment. In competitive environments, the robots within the MRS compete against each other to fulfil their egoistic goals or objectives. The most common example of competitive environments are those having resource conflicts [74] such as Space sharing for Multi-robot movement [75] or Single Object manipulation [76], etc. A prominent example of competitive as well as cooperative MRS environment is multi-robot soccer games RoboCup [77] wherein two MRS teams compete while cooperation is required among the robots of each MRS

team.

Communication among the robots within an MRS forms the essential fabric that keeps the whole system intact. Communication, as a means of interaction and information-sharing, is an essential ingredient for the success of an MRS in order to perform coordinated tasks. The authors in [51] describe that communication emerges as a rational behaviour among the robots within an MRS as a consequence of effective coordination. In [78], the authors classify communication within MRS according to the mode of interaction viz. interaction via environment, interaction via sensing and interaction via explicit communication. Farinelli et al. [73] categorizes communication within MRS as direct and indirect. The authors in [51] discuss two forms of communication for MRSs viz. implicit and explicit communications based on information transfer modes.

Implicit communication is the form of communication wherein the information is transferred indirectly either by the modifications in the environment or by observing and sensing the behaviours and actions of other robots. This form of communication is also referred as *Stigmergy*, a term coined by Grassé [79] in the context of social insects. In stigmergy based communication, the robots continuously keep the track of environmental cues and stimuli to comprehend the information about their peers and act accordingly. The domain of emergent and distributed coordination [80] and dynamic task allocation within MRS usually employ this implicit form of communication wherein the behaviours of individual robots are modulated based on the stigmergic information received by them. These coordination models based on indirect communication are inspired by the distributed self-organized activities in large swarm of insect colonies such as ants [81], bees [82], wasps [83], etc. where simple decentralized behaviours of individual entities congregate to exhibit an apparently intelligent behaviour in an autonomous fashion without any central authority. Swarm robotics [18] which has emerged as a sub-field of MRS specifically focuses on deriving models of interactions prevalent in natural swarms so as to replicate the same in large swarms of robots for distributed control and cooperation. Şahin [84] provides a description of the inspirations for swarm robotics and its applications in realizing real-world MRS solutions.

Explicit communication, on the other hand, uses a point-to-point communication [85] model among the robots within an MRS. The robots are equipped with an on-board communication interface such as WiFi to carry out direct message based communication with their sub-ordinates within the MRS. The communication among the robots can be a unicast, multicast or broadcast. The intentional multi-robot coordination approaches [86, 87] employ the explicit form of communication to carry out task distribution, task planning and allocation, and task decomposition among the robots to achieve cooperation. Networked robots [88] is a stream of robotic research that concentrates on the explicit form of communication within the MRS for real-world applications. Although, the information transmitted using the explicit communication is more accurate and reliable than the implicit communication, the communication load on the system can be a catastrophe [89]. Heavy and unrestricted communication can lead to diminished system performance or even system failure.

This thesis focuses on autonomous networked robots constituted of multiple mobile robots in a networked milieu capable of the explicit form of communication. Instead of using simple messages for the exchange of information amongst the robots, the thesis explores the applicability of the agent based middleware design for networked systems. The research reported in this thesis uses intelligent mobile software agents (mobile agents) [90] to sense and analyze information available at different robots within the MRS. The agents hence provide the much needed services as per the situation and requirement of individual robots. The robots then act on the environment based on the information received from the agent in a coordinated manner to achieve the system's objectives.

1.3 Networked Robots

The concept of networked robots was first proposed by a Japanese study group which described networked robots as systems wherein robots, environment sensors and humans communicate and cooperate through network [91]. They proposed three types of elements within networked robots viz. visible such as physical robots, virtual such as computational entities or agents, and unconscious such a camera or a sensor of which the users are usually unaware. Kumar et al. [88] define networked robots as a set of multiple robots operating together in coordination or cooperatively with sensors, embedded computers and human users. They argue that networked robots have the capability to perform those kinds of tasks with the supporting components from the surrounding that are far away from the proficiencies of either single robot or multiple uncoordinated robots. Networked robots extends the concept of MRSs towards collaboration among different types of robots. The availability of a reliable communication interface among the robots, alleviates the necessity of sensing, actuation and computation to be collocated singularly. Hence, networked robots can collect information from a gamut of sensors and other devices spread across the environment to make informed decisions on their tasks and actions thus improving on the overall efficiency. Typical examples include - military unmanned vehicle collaborations with UAVs and UGVs [92], deployment of a satellite in space which requires coordination among the shuttle robot arm and various related components [93], healthcare surgical robots[17], telemedicine devices [94], etc.

According to IEEE Society of Robotics and Automation's Technical Committee on Networked Robots [95], a networked robot is

> ".....a robotic device connected to a communications network...
> ....based on any of a variety of protocols....."

They classify networked robot into two classes viz. *Tele-operated* and *Autonomous*. In the former, a human being controls the robot by sending and receiving control signals over the communication channel such as telepresence robots [96], remotely controlled UAVs [97], etc. while in the latter, robots and sensors exchange data autonomously via the network e.g. MANERs [98].

Kumar et al. [88] explain that multiple robots tethered to each other in a communication network allows fault tolerance in the system. Robots could thus be plugged-in and played or swapped automatically contributing to a robust operating environment among the components having complementary benefits. Alberto et al. [91] list five elements of a networked robotic system:

1. Physical Embodiment: All kinds of networked robots should have a tangible robot along with on-board software and hardware.

2. Autonomous capabilities: To be a part of networked robots, a physical robot should ensure autonomy with the system.

3. Network-based cooperation: A communication network should string together all the components of the networked robots viz. robots, sensors and human.

4. Environment sensors and actuators: There should be sensors present in the environment except those of the robots.

5. Human-robot interaction: Human and robot should collaborate in some way to be a part of networked robots.

Hence, the field of networked robots involves various domains viz. perception, sensors, ubiquitous computing, artificial intelligence, and network communication, in order to realize all its

behaviours and capabilities. The emerging areas of IoT [99] and CPS [28] further expand the horizon of networked robots in a broader perspective. These newer domains of research have propelled the development of robust low-level interfaces and light-weight protocols which could all aid in creating a milieu of networked robotic systems. In addition, the proliferation of Robotic Development Environments (RDEs) with integrated sensor networks [100] is facilitating richer functionalities for networked robots and thus such systems are becoming easier and faster to develop. Researchers have already started putting emphasis on using the IoT and CPS infrastructure to use networked robot in shopping malls [101], museums [102], elderly homes [12], motels [103] and hospitals [104]. The Dustbot project [105] and Kiva systems for warehouses automation [14] makes two very prominent examples.

The authors in [106] propose cloud networked robots designed to support ordinary activities of daily lives with a concentrated effort for the elderly and disabled. In the prescribed cloud networked robotics, the authors put equal emphasis on physical as well as computation elements comprising various equipment and devices such as stand-alone robots, sensors, and even smartphones which are capable of either sensing the environment or interacting with the people. Hence, all kinds of devices are referred as robots which are integrated and networked to form a cloud in order to provide various robotic services on-demand depending upon the availability of the required resources. The authors describe the robotic services as the systems, devices and robots having three functionalities viz. sensing, actuation and control. The authors present a case study of a shopping mall touring service provided to the elderly using the Ubiquitous Network Robot Platform (UNR-PF) [107] from the Life Support Robot Technology project of Japan.

Networked robots are most suited to provide solutions requiring physical support in real-world. With the technology becoming a part of the human life and society, these networked robots may soon find their way in among humans to supplement their daily activities as a next wave of technological advancements. However, the general purpose robotic system is still a far cry. There are a gamut of issues and formidable challenges that one needs to overcome before these robots can become mainstream.

1.4 Research Challenges in Networked Robots

The domain of networked robots transcends its conventional counterpart in the sense that it includes not only robots that provide services and execute tasks but also various other entities such as sensors and devices embedded within the environment which are woven seamlessly into the network fabric. Such an array of heterogeneous elements puts forth a variety of challenges some of which are listed below.

1. Coordinating multiple autonomous units: Coordination among the entities within such a network is one of the major design goals in realizing real-world applications. A framework which could allow seamless and universal coordination of activities and cooperation among different functions and behaviours poses to be a perpetual challenge for the researchers.

2. Interaction among heterogeneous components: Since networked robots involve a range of heterogeneous robots, environment sensors and devices, the modes and methods of mutual interactions among these entities poses to be yet another challenge. One needs to take care of the inherent complexities of the individual entities along with their communication and computational limitations while devising any interaction protocol.

3. Decomposition and allocation of tasks: The execution of a task may require the participation of various components within the network. Hence, a task needs to be divided into sub-tasks

and a judicious allocation of these sub-tasks must be performed in order to facilitate each entity with its required goals and sub-goals.

4. On-demand service: Networked robots must be embedded with mechanisms to service different entities based on the demand and the availability of the resources. These services could be a program or code or a map required by a robot or a threshold value required by some sensors in the environment. The networked robotic system must ensure that these requests must be satiated with a fixed time-frame so as to support the smooth execution of various tasks.

5. Dynamic Network: Unlike sensor networks [100], a network of multiple mobile robots is dynamic. The dynamicity of the network includes not only the change in the neighbours of the individual entities but also the topology. Hence, the mechanism designed for networked robots must be able to cope up with these changes occurring at run-time without compromising the performance of the system.

6. Distributed sensing and group perception: Networked robots focus on a wide area of coverage for the execution of complex coordinated tasks such as multi-robot exploration or multi-robot object transportation. To effectively perform such tasks, the local information available within the vicinity of a robot may not be enough. Hence, the networked robot systems must be empowered with functionalities that support group perception and distributed sensing so that individual robots can ascertain the world model of their environment.

7. Distributed and decentralized control: Networked robots are naturally distributed systems with multiple autonomous units interacting with one-another. However, controlling such a diverse system in a distributed and decentralized manner is a great challenge. Such controls should support flexible task execution, fault tolerance and robust decision making.

8. Scalability: The scale of a system can substantially affect its performance. Hence, the mechanisms designed to build networked robotic system must ensure scalability as one of the primary objectives. This will allow the tethering of new devices and robots to the system with ease.

9. Management of networked resources: In networked robotic systems, there can be multiple components that may be dependent on each other for accessing required information or delivering services from one end of the system to the other. The effective management of such networked resources ensures the survivability of the overall system. Managing such resources in a heterogeneous environment with autonomous components forms another challenging task.

10. Ease in deployment and *on-the-fly* (run-time) modification: The system of networked robots must be easy to deploy in the real-world. An easy and rapid deployment enhances the acceptability of the system by its users and makes it comprehensible to manage and control. In addition, the system should allow for *on-the-fly* modifications of various components of the network without bringing down the whole system.

11. Interfaces for human intervention and interaction: Humans are the ultimate users of these systems. Such systems are designed in order to support humans in their daily activities, be it in industry or in personal use. Hence, a system of networked robots must provide multi-modal interfaces to enable humans to interact with the system so as to read or set crucial parameters and properties of the system.

12. Sharing of information and learning: Learning and sharing are two important functions in dynamic environments for continuously optimizing the performance of a system over time. Networked robots provide a good test-bed to test the efficiency of various learning algorithms designed for multiple agents in distributed systems by evaluating their feedback from the real-world. A mechanism which can facilitate seamless sharing of important information such as action feedback and learning is still a challenging task in the domain of networked robots.

13. Self-adaptation and reconfiguration: The system of networked robots may need to adapt by either acquiring new functionalities or purging the ones not required by the system so as to counter the changing dynamics of its environment. The changes in the environment could include - addition or removal of devices or robots, availability of new services or a change in communication interfaces. Hence, networked robots must be capable of reconfiguring themselves and adapt as per the requirements.

14. Security of networked components: Since the system of networked robots comprise various networked components, making everything secure from malicious attacks becomes crucial. Hence, new mechanisms and algorithms need to be innovated in order to provide adequate security with functions like authentication, authorization and digital signatures.

15. Support for inherent autonomy: Autonomy can enable a system to take intelligent decisions based on the analysis of the information available. This could mean scheduling of parallel or sequential tasks, search for services within the system, or monitoring the performance of the system. Embedding such autonomy within the system of networked robots can aid in dealing with various sporadic problems without external interventions.

This thesis endeavours to provide a framework for developing mechanisms for the above mentioned challenges. In the thesis, the focus is on a hybrid approach by amalgamating networked robots with the paradigm of mobile software agents or simply mobile agents. By making use of a mobile agent based middleware design for networked robots, this thesis explores the mechanisms for evolving better and on-demand services, sharing and learning along with the execution and allocation of a set of tasks.

1.5 Mobile Agents

A mobile agent [108] is defined as a computational entity capable of reasoning, making decisions, interacting with other entities, collecting results and feedback, representing a user and migration into a network infrastructure along with its state and data. Mobile agents can be thought of individual chunks of information processing units that can mobilize in a network and collect and process information available at remote sites. Outtagarts [109] has listed applications of mobile agents in several areas including network management, electronic commerce, energy efficiency and metering, wireless multimedia sensors, grid computing and grid services, distributed data mining, multimedia, human tracking, security, affective computing, climate environment and weather, e-learning, location, recommendation and semantic web services. The author also compares mobile agent technology with the prevailing client-server architecture, message passing paradigms, remote procedure calls (RPC) [110], CORBA [111], etc. and postulates that the mobility of such agents gives them an edge over other conventional communication techniques. The distinctive features of mobile agents are efficiency, persistence, peer-to-peer communication, and fault tolerance. Lang [112] describes agents from an end-user perspective and compares stationary and mobile agents. He gives seven pertinent reasons in favour of using mobile agents:

1. Reduced network load: In a distributed system, a large amount of communication take place due to multiple interactions among the entities within the network which in turn clutters the network. Mobile agents can be packaged with the information that needs to be communicated and can be dispatched to the destination host. Thus interactions can happen locally.

2. Reduced network latency: Robots deployed in real time environments such as manufacturing units need to respond quickly. Hence the latency offered by the intermediate network is not acceptable. Mobile agents dispatched from a central controller can furnish the actions locally without any delay.

3. Protocols can be encapsulated: In a distributed environment, each host has to maintain a variety of protocols to converse with other hosts which is a cumbersome task. Mobile agents can encapsulate protocols and migrate to remote hosts making this communication easier.

4. Asynchronous and autonomous execution: Mobile agents can be embedded with the task and dispatched through the network. They can independently reach their respective destinations, execute asynchronously and even return autonomously.

5. Dynamically adaptable: Mobile agents can distribute themselves over the network and can adapt to their environment based on the change in the environmental parameters.

6. Natural heterogeneity: Since mobile agents only depend on their execution environment, they offer natural heterogeneity in terms of their structure and behaviours.

7. Robustness and fault-tolerance: Mobile agents can react dynamically and like their static counterparts take decisions, both of which make them robust and fault-tolerant.

Lang [112] also compares client-server, code on-demand and the mobile agent based network paradigms and provides a range of applications suitable for mobile agent. The author envisages the future of distributed computing with mobile agent technology. In [113], the authors describe the advantages of using mobile agents along with their classification and application areas. The authors present the missing links in mobile agent technology which has made them difficult to evolve and use. Among these the are mechanisms for agent security, control structures and transactional support.

In relation to networked robots, Cragg and Hu [114] initiated the use of mobile agents to control tele-operated robots. They describe an architecture [115, 116] to use mobile agents for the distributed control of networked robots. In [117], these authors systematically examine the applicability and advantages of mobile agents in the domain of networked robotics. They argue that to control networked robots four control software architectures can be adopted viz.

1. Client-Server, where robots contain the knowledge and resources to perform the task and can be controlled from a simple remote client.

2. Remote invocation, wherein robots contain the resources but need to obtain the knowledge from a client to perform the task.

3. Code on-demand, where robots contain resources but act as clients to access knowledge at a server to perform a task.

4. Mobile agents, where robots contain the resources and a task can be performed by moving a mobile agent containing knowledge and results, between robots.

They describe the functional benefits of mobile agent as adaptability, fault tolerance and dynamic control and learning. They also describe some of the benefits of using mobile agents from the software engineering perspective courtesy of their modularity, ease in understanding, loosely coupled functionalities and extensibility from the point of view of a system developer.

Various researchers have experimented with the mobile agent framework in the context of networked or multiple robots. Kambayashi et al. [118, 119, 120] used mobile agents in multi robot environments to design an intelligent cart system and implemented an evolutionary ant colony clustering algorithm. Abe et al. [121] used mobile agent to search targets in multi-robot environments. Gaber and Bakhouya [122] used mobile agents for discovering resources in peer to peer network. In [123], the authors presented a mathematical model to discuss the macro dynamics mainly affected by the migration behaviour of the mobile agents using an eco-system inspired architecture. Godfrey and Nair [124, 125, 126, 127] clubbed the mobile agent technology with bio-inspired algorithms for servicing networked robots. They proposed a mobile agent migration strategy nick-named *PherCon* and compared it with the contemporary strategies. They showed that their strategy to be better suited to networked robots. They also proposed a cloning control mechanism [128] to regulate the population of mobile agents in the robotic network. Further, in [129], the authors exhibited a practical application of their proposed algorithm in realizing an Internet of Things (IoT).

Thus the applicability and appropriateness of mobile agent technology as a middleware design for networked robots has already been portrayed in a substantial amount of research papers. This thesis builds on the work of Godfrey and Nair [127, 130] and attempts to extend the use of most of the salient features of the mobile agent based framework to aid in learning and coordination within a networked robotics milieu. The sections that follow describe the major contributions of the work carried out followed by a chapter-wise summary of the thesis.

1.6 Contributions of the Thesis

The work reported in this thesis attempts to portray mobile agents based mechanisms for networked robots based applications. The major contributions made in this thesis are listed herein -

1. **Selection of the best performing solutions using an emulated idiotypic network :** A novel architecture for emulating Jerne's [131] Idiotypic Network model over networked robots that functions in a distributed and asynchronous manner has been elucidated herein. Though the Idiotypic network has been used for in a variety of applications, the computations within have been mostly sequential. This is unlike what happens in the biological world where the concentrations of the antibodies that constitute the network evolve in a parallel and concurrent manner. The work reported herein emulates the manner in which the best performing solutions can be autonomously selected by the networked robots using an emulated Idiotypic network. The novelty of this approach is that the intelligence is scattered in the environment comprising networked robots in the form of mobile agents that are synonymous to antibodies. These agents selectively mitigate the problems arising at different nodes within the networked robots concurrently along with an inherent competition among themselves (agents/antibodies) to evolve the optimal solutions (i.e. the best set of antibodies).

2. **Distributed learning using mobile agent based localized sharing :** This contribution presents a framework for sharing intelligence and mutual learning among a set of location-unaware and spatially segregated networked entities or robots within a distributed environment; all of which have the same objective. The framework uses static agents

which are resident within each of the entities (robots) while the mobile ones facilitate the exchange of information and localized sharing over a network. The framework focuses on the generic mechanisms required to be embedded within individual mobile agents so as to result in the overall convergence of the entire agent population, towards their common goal. The proposed framework is fully distributed in the sense that neither the agents (both static and mobile) possess any knowledge about the overall number of such agents present in the network nor do they possess any location information about other agents. Further the sharing of information among the agents is completely asynchronous and local. The major contributions of this work include:

(a) A multi-agent framework for distributed intelligence-sharing and learning

(b) Modalities for local sharing and exchange of information

(c) Dynamics for facilitating agent migration, inter-agent interactions and asynchronous executions

3. **Stigmergy based synchronized execution of a sequence of tasks by networked robots :** This work presents a novel technique, inspired by the self-organized synchronous behaviour of insect colonies, for near synchronous execution of a sequence of tasks by networked robots. The technique uses mobile agents populating a network of robots acting as insects which exploit the concept of indirect stigmergy based communication wherein a sequence of tasks is to be executed within each of the robots synchronously. The synchronization activity is carried out by a set of mobile agents operating autonomously and without any direct mutual interaction. In the proposed technique, the mobile agents remain oblivious to their total number within the network as also to the total number of robots. Further the technique also exhibits self-healing in case of task failures. The technique has been emulated of real networks along with an implementation using real robots, both of which validates its practical viability.

4. **On-demand population control of mobile agents for task synchronization :** This work extends the previous contribution which used a fixed population of homogeneous mobile agents and proposes a technique for near synchronous execution of tasks by a heterogeneous set of mobile agents. Heterogeneity is referred in the sense that there are different types of mobile agents in the environment; each type capable of performing a distinct task on a robot. Each mobile agent perceives a stimulus from the environment comprising networked robots and performs its assigned task. This technique also manages the change in population of these heterogeneous set of mobile agents so as to suit their current demand. The population of agents required to complete an activity is generated dynamically using the population control model proposed by Godfrey et al. [128] without any supervised control.

5. **Strategy for multiple concurrent task allocations with voluntary back-offs based conflict resolution :** This forms the last contribution of the thesis and discusses a novel strategy for the allocation of task(s) within a large network of heterogeneous robots in the presence of competing task allocation instances. An instance of a task allocation implies a process to allocate a task to a set of unoccupied robots within the system. The goal of such allocation is to maximize utility (minimize cost) in terms of completion time and resource use. The mechanism uses mobile agents to handle concurrent and multiple task allocation instances. Using a novel concept termed *mobility resource* to adaptively explore the robotic network, this mechanism ensures multiple task allocations by forcing voluntary and adaptive back-offs on part of the mobile agents. Results portrayed therein have shown

that this mechanism outperformed two prominent multi-robot task allocation approaches cited in the literature. Experiments conducted in the real-world using robots further fortified the practical viability of this approach.

1.7 Outline of the Thesis

The thesis comprises eight chapters. The chapter wise organization of the thesis is given below:

Chapter 1: This chapter discusses the motivation behind the research, followed by a survey of the state-of-art in robotics and multi-robot systems. It also briefly describes the contributions made in this thesis.

Chapter 2: In this chapter, a survey and the state-of-art on the use of mobile agents in the domain of networked control has been presented. This chapter also discusses the important concepts, features, platforms and applications of mobile agents which forms the motivation behind the selection of this technology for the work carried out herein.

Chapter 3: This chapter presents the contributions made in emulating an Immune-inspired Idiotypic Network on a real network of robots for evolving a set of the best/optimally performing solutions while purging off the others. It also provides the description of the mobile agent emulation platform named *Typhon* [132] which has been used throughout this thesis for carrying out the emulation experiments. This chapter uses the material from the reference [133] of the author.

Chapter 4: Addressing the challenges faced in sharing of information and learning forms the crux of this chapter. It presents a framework for information sharing and learning among a set of nodes within the system of networked robots using mobile agents. This chapter uses the material present in the reference [134] of the author.

Chapter 5: This chapter elaborates on a technique for near synchronous execution of a sequence of tasks by a set of robots using homogeneous mobile agents. The chapter also discusses the emulation experiments and the implementation on real robots. This chapter is based on reference [135] co-authored with W. W. Godfrey and S. B. Nair.

Chapter 6: This chapter elucidates how the synchronization technique proposed in the previous chapter is augmented with a mechanism for population control of mobile agents. The work reported in this chapter uses the material from the reference [136] of the author.

Chapter 7: This chapter touches upon the problem of allocation of tasks among networked robots. It presents an improved mechanism to handle multiple task allocations within a network of heterogeneous robots using mobile agents.

Chapter 8: This final chapter highlights the conclusions arrived at and also summarizes the contributions made. New avenues for future research have also been described.

"Research is to see what everybody else has seen, and to think what nobody else has thought."

Albert Szent-Gyorgyi (1893 - 1986)
Hungarian physiologist

2

Mobile Agents: Concepts, Features and Applications

Mobile agent technology is still in its infancy in terms of its understanding and use by both technology developers and practitioners. Although the initial researchers viewed mobile agents as the technology of future [137] in the age of Internet, the concerns related to their security aspects and open-ended use refrained the industry from adapting them in practical applications. Yet the mobile agent paradigm is still thriving due to its inherent advantages in realizing applications involving large scale distributed, decentralized and dynamic systems. Another crucial aspect which encourages researchers to revisit this technology with a new perspective is the emergence of the newer application domains such as the Internet of Things [25] and Cyber-Physical Systems [26] both of which are basically large scale, distributed and dynamic in nature. In addition, the currently available mobile agent technology is far more mature than at the time of its genesis. This has made it possible to host and use mechanisms exploiting such agents at granular levels. It is thus important to understand the basic idea of a mobile agent along with its distinguishing characteristics and limitations. This chapter attempts to present the core concepts of mobile agents, environments for which they are suited, mobile agents based mechanisms together with the areas of applications wherein one can reap their advantages. The chapter also provides a comparison to other technologies such as those in networked robots while also discussing their inherent problems and challenges.

2.1 Basic Concepts

Mobile agents are autonomous computational entities capable of dynamically changing their execution environments based on their acquired intelligence within a network [90]. The concept of mobile agents evolved from the research in distributed operating systems for migrating processes along with their current state, register values and data in order to balance processor load [138]. Using such a concept the client machines can migrate processes having heavy computational overheads to server machines after which the processes return with the outcome of their computations. The emergence of the network as a medium of large scale computing along with the growth in the domain of multi-agent systems and distributed operating systems resulted in the development of mobile agent technology as a whole. Mobile agents are not mere processes but a complete package of software usually termed as agents that can seamlessly be transported in a network.

As the mobile agents can migrate from one execution environment to another, it may happen

that some of the functions or resources, that the agents require, are not available in their current location or operating system. These agents are thus not bound to the functions of the operating systems they inhabit; instead they are bound to the functions available within their execution environments [108]. This not only provides flexibility for developing and deploying heterogeneous services within a distributed environment but also makes a mobile agent a dynamic entity. Mobile agents are thus preferred to be run in an interpreted language than in machine code [139]. Further, as claimed by Chess et al. [139], it is much easier to secure the interpreted languages as it is the language developer who explicitly controls the system resources.

Mobile agents are thus suited as a middleware for large-scale distributed and dynamic environments [140]. As mentioned by the authors in [139], these agents can provide a generalized farmworker for pervasive environments in order to develop personalized network services. The significant advantage of mobile agents is that it brings computations close to the entities which require them. Instead of dispersing loads of raw data to a server over a network, mobile agents can process the raw data at its source *par se* and transfer the valuable information back to the server. Thus, the mobile agents form ideal candidates for performing real-time control over a distributed network without choking the network bandwidth with a series of control-command sequences resulting in unavoidable latencies for real-time operations [141]. This also brings robustness to the system and facilitates *on-the-fly* programming of the entities in the network. For instance, in an automated warehouse with a large number of networked robots, one may require to change/update the manner in which the items are picked or placed. Conventionally, all the robots need to be stalled and updated for such a change in their respective configurations. The use of mobile agents alleviates such a problem by updating the robots *on-the-fly* without external intervention and without bringing the entire system to a halt.

2.2 Features

Mobile agents inherit all the characteristics defined for an agent with the additional feature of mobility within the network. The desired features of a mobile agent are as follows:

1. Autonomy: A mobile agent should be autonomous to have a direct control over its actions and states.

2. Social ability: A mobile agent should be social in order to interact with other agents (both static and mobile) for soliciting information and achieving its task.

3. Reactivity: A mobile agent should be reactive for providing real-time responses to the changes it perceives in the environment.

4. Pro-activeness: It should be proactive to know its environment and take premeditated actions for achieving its goal(s).

5. Mobility: The ability to migrate with their code, data and internal states is a crucial characteristic for mobile agents.

6. Adaptability: A mobile agent should have the capability to learn and adapt to the changing dynamics of the environment both spatially and temporally.

7. Rational: A mobile agent should be rational to direct its actions in a goal directed manner instead of selecting those tasks which may prevent it from advancing towards its goal.

8. Integrity: A mobile agent should be immune to any attempts by any host or other agents to tamper with its code or data. The structure and code of a mobile agent should also remain intact in case of any corruption in the host environment or data.

Such features of a mobile agent makes it an intelligent entity which has awareness about its internal states, interests and environment and which can make informed decisions to cope up with various situations encountered while progressing towards achieving its objectives.

The mobile agent technology provides various advantages as compared to the conventional client-server or Remote Procedure Calls (RPC) paradigms for realizing distributed applications [142, 143, 109]. Chess et al. [139] list eight technical advantages for using mobile agent technology viz. high bandwidth remote interaction, support for disconnected operation, support for weak clients, ease of distributing individual service clients, semantic routing, scalability, lower overhead for secure transactions and robust remote interactions. Outtagrats [109] points out that the mobility of mobile agents is important as it brings - efficiency by reducing network traffic and latency, persistence as the mobile agents are not tied to the system that creates them, peer-to-peer communication which allows the mobile agents to act either as client or server as per the requirement and fault-tolerance as the mobile agents are able to process the request by moving locally to a node. Poggi and Tomaiuolo [140] describe that there are three key benefits of using mobile agents which are exploited by all kinds of applications that use them. These are - reduction in network traffic for remote operations, ease in customizing services and applications with seamless adaptation to the new requirements, and robustness and fault-tolerance while carrying out operations in distributed systems. These agents provide a uniform approach to manage asynchronous operations in a distributed systems. In addition, they offer to encapsulate protocols [141], for heterogeneous interactions with various entities and can work remotely in an asynchronous manner even when disconnected from a network.

2.3 Limitations and Scope

One of the major concerns for openly using mobile agent based applications on Internet is security [139, 144]. The security concerns in the context of mobile agent technology are two fold viz. agent security and host security. Host is a platform running on the network nodes that facilitates agent activities such as migration, execution, etc. In open systems running over the Internet, it is extremely difficult to establish mutual trust among various entities using the system. Hence, malicious agents as well as malicious hosts can take advantage of the system by stealing information or denying services.

A malicious agent can [145, 146]:

1. impersonate the identity of another agent in order to gain access to certain resources or information available within the host,

2. create denial-of-service kind of attacks by overloading the host with enormous spurious computational load,

3. spy on the activities of the host and other agents hosted within that host to pilfer crucial information,

4. provide false information to the host regarding a transaction or

5. repudiate some actions that can cause damage to the system.

On the other hand, a malicious host can [146]:

1. masquerade as another host in order to gain access to the information with the agent or fool the agent to react to some situation for its egoistic benefits,
2. deny services to the agent to delay its activity or even terminate the agent in order to stall some process,
3. forcefully decipher the agent's code and data to obtain some proprietary information,
4. monitor an agent's actions and strategies to understand some of its behaviour for which new counter strategies can be generated to create a negative effect or
5. induce some malicious code within the agent by modifying either its code or data.

Various efforts have been made to create secure environments for using the advantages of mobile agents in open applications over the Internet. Researchers have also proposed sandboxing techniques [147] in order to bar mobile agents from an unrestricted access to the resources available at the host. Other mechanisms include agent and host authorization and authentication [148], cryptographic mechanisms for securing privacy and integrity [149] of agents and hosts, Access Control List (ACL) to force restrictions on the access of resources [144], etc. Security is an everlasting endeavour that has to keep evolving with the advancements of the available technology. The currently available security techniques provide much room for using mobile agents within autonomous systems in a fairly secure manner [150, 146].

It is thus prudent to use mobile agent technology within the boundaries of a closed local area network or within a virtual private network (VPN) owned by proprietary organizations. In such systems, there is a high degree of confidence among the various entities within, which can be exploited to use mobile agent based mechanisms for distributed computing and applications such as network management [151], multi-party event scheduling [152], etc. Various other closed-world application environments include warehouse automation, military infrastructure, networked robots [116], sensor networks and load-balancing in manufacturing industries [153], among others.

As can be noted the security concerns in the use of mobile agent technology lies in its open exploitation over the Internet. Although the early researchers assumed mobile agent technology as an enabling technology for future of the Internet for boosting e-commerce [109, 141], they could not provide strong methods to create a secure framework for the same. The concurrent development of various other large-scale distributed systems such as grid computing [154], cloud computing [155], data centres, distributed file system such as Hadoop [156], etc. have opened new avenues and scope for using mobile agent technology and reaping the benefits it can deliver. There is still a strong need to rediscover this technology with a new perspective for its applicability in other emerging domains of computing.

2.4 Mobile Agent Platforms

The mobile agent technology can serve as a middleware solution to realize distributed applications, if only, there exists such platforms which can provide and support all its functionalities along with providing scope for extensions and supervisory control. A number of such platforms have been developed in the last two decades exploiting the low-level features that different programming and scripting languages had to offer. This section discusses some of the well-known platforms, from the very old to the recent ones, and highlights their merits and demerits.

Telescript [157] was developed in 1994 by General Magic, Inc. as a commercial interpreted object-oriented programming language designed to develop mobile agent based applications. The Telescript platform used two key concepts viz. places and agents. Places are the hosts which can be inhabited by the agents. The Telescript platform had four components viz. Telescript language, Engine, protocol set and software tools. The Telescript language was designed to facilitate complex tasks such as navigation, transportation, authentication, access control, etc. while the engine performed the responsibilities of an agent host which included language interpretation, execution environment for the agents, communications amongst the agents and an interface to interact with other applications. The protocol set dealt with encapsulation of agents to be transported among different places while the software tools facilitated a development framework for developing applications. Four levels of security were ensured viz. during object execution, process state, system and network. However, due to the proprietary language and concurrent development of various Java based platforms, Telescript lost out.

Agent Tcl [158] was another mobile agent platform designed using TCL scripts. The safety of TCL language based agents was ensured using sandboxing technique [147]. An access control list is managed to provide permissions to the agent for the resources available at the host. The platform supported the notion of strong mobility by allowing the access to the threads' status. Cryptography was also incorporated to provide security. The support for other languages such as Java and Scheme were added and the platform was renamed as D'agents [159]. However, the platform was not maintained after 2003.

Aglets [160] was developed by IBM following the basic concepts of Java Applets. Hence, agents in Aglets are serializable Java objects which can be ported from one JVM to another. The use of JVM accounted for the platform independence of an Aglet. There are six key abstractions available viz. aglet, proxy, context, message, future reply and identifier. Aglet was developed as an event driven programming model which emphasized the delegation and reuse of certain components such as event listeners. The project is available as an open source project [161] distributed under the IBM Public License. However, there seems to be no development on the same after 2001.

Ara (Agents for Remote Action) [162] was designed as a mobile agent platform with two specific components viz. the core and the interpreter. While the issues related with the interpreted programming language of the mobile agents were handled by the interpreter, the core was used to handle all the language independent issues of the system. The basic idea was to design a generic core on the top of which various language interpreters can be added to handler agents written in different languages. The initial interpreter support was for TCL and C/C++ while the support for Java was added later. Agent security was ensured by attaching each agent with a passport that contained all the relevant information such as identity, owner, author, accessible service, accessible resource, etc. This passport was created during the instantiation of an agent and made immutable throughout.

Voyager [163] was developed as a commercial product by ObjectSpace (now Recursion Software, Inc.) in order to develop distributed Java based applications. Voyager is based on Java RMI (Remote Method Invocation). It ended the CORBA Object Request Broker (ORB) with object mobility to provide the functionalities such as creation and calling objects remotely and obtaining references to remote objects. Silva et al. [164] claimed that Voyager was a better alternative for RMI but does not focus on the functionalities of an agent. Hence, they proposed AgentSpace, a mobile agent system developed over the Voyager system to develop mobile agent based applications in a seamless manner. Interestingly, there exists another Java based AgentSpace mobile agent platform [165] developed by Ichiro Satoh from Japan.

JADE (Java Agent DEvelopment framework) [166] was developed over Java by the Telecom Italia labs. and the University of Parma. JADE is FIPA [167] complaint and hence provides

standard protocols for agent to agent communication. JADE also provides a set of graphical tools for the development and deployment of JADE based applications. JADE supports the BDI architecture for developing agents and provides simplistic task executions and composition models. Communication is possible both ways i.e. across agent hosts (containers) using FIPA-ACL and within a container using JADE communication primitives. JADE also provides a directory service to the agents for supporting publish-subscribe kind of discovery mechanism. Various other plugins are available for authentication, authorization, security, mining, etc. JADE has a dynamic transport mechanism as it adapts to the requirement by choosing the appropriate protocol. JADE uses a variety of transport protocol such as Java RMI, event-notification, HTTP, and IIOP while their community page claims that more protocols can be easily added via the MTP and IMTP JADE interfaces. JADE has been released as an open source software under LGPL and has a vibrant and large community of developers to its advantage. The latest version of JADE was 4.4.3 released on October 2014.

Tracy [168] is a Java based mobile agent platform with a plug-in oriented architecture developed by the University of Jena, Germany. Tracy agent hosts or agencies thus remain lightweight i.e. only essential services are loaded initially and in case a new service is required then that is loaded by adding the plug-in. Moreover, users can develop new plug-ins to extend services and functionalities. However, one of the limitations of this mobile agent platform is that the communication among the agents is possible only within an agency. For an agent to communicate with a remote agent hosted in another agency, the agent has to migrate to the remote agency. The latest version of this platform was released in 2005.

SPRINGS (Scalable PlatfoRm for movINg Software) [169] was developed at the University of Zaragoza, Spain. The development of this platform was focused on scalability and reliability aspects and used dynamic proxies to provide location transparency for migrations and remote calls. The system is organized in a hierarchical structure based on regions. The authors of SPRINGS claim that it has support to minimize the livelock problem that arises when agents migrate across platforms. SPRINGS is not FIPA compliant for communication and interaction among the agents and security is only loosely implemented. The platform is available for free, however, post 2008 no activity or update seems to be available.

Mobile-C [170] is a C/C++ based mobile agent platform developed by the University of California, Davis, USA. The development of Mobile-C focussed on developing mobile agents for control and operation of embedded systems. This platform [171] uses a C/C++ based interpreter known as Ch interpreter to facilitate portable and secure execution of mobile agents written in C/C++ scripts. Ch interpreter is embedded within the agent hosts of a Mobile-C platform which provides cross-platform scripting and object-based programming. Mobile-C uses FIPA-ACL for communication among the agents. In addition, the complete platform is FIPA compliant in terms of agent management services, directory services, security, etc. The Mobile-C platform is designed as a set of library functions which can be easily be integrated with any application as a service. The platform is currently known to work on Linux (32-bit and 64-bit), Mac OS X, FreeBSD 6.0, Windows, and Solaris 2.7. The latest update v2.1.5 was released in August 2014 and the complete project is hosted at github.com.

Typhon [132] is a LPA WIN-Prolog [172] based mobile agent platform developed recently in 2011 by the Robotics Lab. of Department of Computer Science and Engineering, IIT Guwahati, India. This platform has been developed above the Chimera agent system provided by the LPA. The main focus for developing this platform was to exploit the features of Prolog such as its recursive engine, dynamic database management, *on-the-fly* modification of rules, etc. for agents with mobility. Typhon also provides some extended features for mobile agents such as payloads. Payloads are the data or code (rules) that can be attached to an agent. The agent can carry these

payloads to various nodes in the network. The platform was also extended to support FIPA-ACL for communication with agents from other platforms. The major restriction to use this platform is the requirement of LPA WIN-Prolog which is a proprietary software.

Tartarus [173] is an SWI-Prolog based mobile agent platform developed by the same lab. which created Typhon and was released in 2015. The use of SWI-Prolog which is an open source Prolog environment alleviated the problem of having to use a proprietary software. Further, the agents and services within Tartarus have been implemented as threads. This was not the case in Typhon. Thus, the performance of Tartarus as a mobile agent platform is found to be better than that of Typhon. Tartarus has been used with embedded systems such as the Raspberry Pi and Intel's Galileo boards. Further development of Tartarus is still on-going. Features such as a GUI, deployment and monitoring tools, etc. are being added.

Various other mobile agent platforms which were developed by various communities include - TACOMA [174] developed jointly by the universities of Tromso and Cornell, Grasshopper [175], Ajanta [176] developed by the University of Minnesota and MAF [177] developed over the Python. Jini [178] combined Java and Prolog for facilitating mobile agent based functions while ALBA [179] and IMAGO [180] are other Prolog based platforms. Agilla [179] was developed as mobile agent based middleware for sensor networks.

2.5 Applications

As discussed earlier, mobile agent technology is suited for applications that are distributed, large-scale and involve dynamic, complex and remote interactions. Some researchers have also used mobile agents to emulate ant-like foraging behaviour in real networked environments [120] in order to implement distributed mechanisms. These include the works of Parunak [181, 182] on Polyagents and works by Weyne and Hovolet [183, 184, 185] who introduced the concept of delegate Multiagent Systems or delegate MASs.

Mobile agent based applications have been reported by a large community of researchers working in a variety of fields. In [109, 140], the authors provide a survey on the gamut of applications proposed using mobile agent technology. As the initial focus of developing mobile agents was to use them commercially for business transactions and e-commerce, one can find numerous papers devoted to realizing such applications [186, 187, 188]. Distributed data-mining and information retrieval is another domain where the task is to gather information based on certain criteria from a set of information servers or nodes dispersed within the environment. Mobile agent are found to be the best suited entities for tasks that require them to migrate within a network and discover a specific node or resource [189]. Data gathering and information dissemination in sensor networks [190], network management [151], energy efficiency and metering [191], e-learning [192], network security and intrusion detection systems [193, 194, 195], automatic software updating in large networks [196, 197], semantic web-services [198, 199, 200] and Grid computing [154] are other domains where one can find the applications of mobile agent technology. Mobile agent based applications have also been proposed for system management and accounting of telecommunication services [201]. Tennenhouse and Wetherall [202] describe a mobile agent architecture for an active network wherein the configuration in the programmable switches can be changed dynamically based on the current demand and requirement of the application.

Of late, mobile agent based mechanisms have also been proposed for cloud computing [155, 203], Internet-of-Things (IoT) [129, 204] and Cyber-Physical Systems (CPS) [205]. Aversa et al. [203] have used a mobile agent platform for integrating the Cloud technology over the GRID architecture. The authors have used virtual clusters controlled by the end user. The mobile agents

herein are used to extend and add services to these clusters dynamically. In [129], the authors propose a generic middleware architecture for realizing an IoT using two kinds of mobile agents viz. Search-for-Resource (SfR) and Provide-a-Service (PaS). The authors in [205] exploit the features of mobile agents using a three level architecture viz. node level, task level and combined-task level to minimize the overheads on communications and redundancy of information within a CPS for intelligent transportation system. Bode et al. [206, 207] have proposed mobile agent based traffic management for the movement of emergency vehicles in urban traffic. The authors in [184] have used a mechanism based on delegate MASs for anticipatory vehicle routing. Wu et al. [208] describe a Service Oriented Architecture (SOA) for smart home environments using Open Services Gateway initiative (OSGi) and mobile agents technology. They have used a peer-to-peer model with distributed OSGi platforms using SOA based interactions. Mobile agents are used to deal with dynamic situations and to distribute the work load over multiple service providers available in the smart environment.

Manzoor and Nefti [196, 209] have proposed mobile agent based mechanisms for software deployment in Campus Area Network (CAN) and content based monitoring in complex networks. There mechanisms work autonomously without any human intervention. For automatic unattended installation/un-installation of software packages, they have used a hierarchical agent architecture with a master controller agent at the top followed by various controller agents for different subnets and the mobile agents. File transfer agents and installer/un-installer agents are used at the bottom level. Similarly, for monitoring the user activities across a network, they have again used a hierarchical agent based system with four agents - iDetect Controller Agent (CA) at the top followed by iDetect SubController Agent (SCA) and two mobile agents at the bottom viz. Monitor Agent (MA) and Analyzer Agent (AA). They have tested their mechanisms in a university campus across various laboratories equipped with 60 to 100 PCs. Brahmi et al. [210] present a distributed Intrusion Detection System by combining the mobile agent technology with data mining techniques. Urra et al. [211] have presented a comparative study of the applicability of mobile agents in Vehicular Networks (VANETs). They have discussed the benefits of using mobile agents in VANETs and presented an interesting application use case of monitoring the environment of a region using vehicle passing in that region. Li et al. [212] discuss an architecture for using mobile agents in a cloud infrastructure for managing urban traffic. They argue that the use of mobile agent technology for urban-traffic management is both feasible and effective. They emphasize on a layered architecture to use the mobile agent for large-scale computations. In [213], the authors present an architecture for management and security of multimedia traffic for IoT domains. They present the applicability of mobile agents for processing computational multimedia traffic over the IoT infrastructure.

Gonzlez-Valenzuela et al. [214] present a system nicknamed Wiseman (WIreless SEnsors Employing Mobile AgeNts) as a mobile agent based middleware solution for Wireless Sensor Networks (WSNs). They provide support for various mobile agent migration strategies and flexible execution. They also provide a proof-of-concept of their model in a real hardware platform. Chen et al. [215] provide a scheme based on minimum spanning tree for multi-mobile agent itinerary problem in WSNs. Su and Chiang [216] present a platform for elderly care using sensor network and mobile agents. They facilitate caregivers to locate, access and share critical data related to the treatment and wellness of the people in care. They present their platform as a pervasive and context-aware care service. In [217], the authors present crowd sensing by using role-based mobile agents for collecting, analysing and sharing data within a campaign. They provide mobile agent based solutions for execution and monitoring of campaigns with a concentrated effort to address issues related to data collection and campaign participants. Shen et al. [218] present a load balancing technique for unstructured peer-to-peer networks using mobile agent technology and clustering of resources.

They use mobile agents for balancing load within the nodes of each cluster by identifying congested node and taking quick remedial actions.

Otani and Kobayashi [219] present a mobile agent based SCADA (Supervisory-Control-And-Data-Acquisition) system for supporting flexibility in fulfilling real-time tasks for an autonomous demand-area power system (ADAPS). The mobile agents are equipped with QoS methods to avoid any congestion in communication. To ensure robust and real-time operation, they have provided two types of migration protocols one for an emergency agent while the other is for best-effort delivery. Nestinger et al. [220] present an automation system based on mobile agent technology. They describe how the robots and mechatronic devices can be easily reconfigured and dynamically allocated with different tasks and control algorithms using mobile agents. They provide a proof-of-concept experiment using two robots across a conveyor belt in an automation work cell environment. Abe et al. [121, 120] present a framework for controlling multi-robot system using mobile agents. They use the ant colony clustering algorithm with ant and pheromone based mobile agents to perform control operations on the robots. The mobile agents carry the core logic to execute tasks while the robots act as resources. These mobile agents migrate into the network of robots to find the best suited ones to execute its tasks. They claim that the mobile agent based approach reduces energy consumption for multi-robot task execution.

Godfrey and Nair [127] describe a bio-inspired mobile agent migration strategy for servicing networked robots. They assume a network of heterogeneous robots wherein the robots can request for a service in form of a program or code. Mobile agents equipped with services (programs/codes/algorithms/rules) migrate into the network of robots in order to deliver these services. To reduce the waiting time for a robot to receive its required service, these authors use a bi-directional search mechanism wherein the robots diffuse virtual pheromones in the form of messages to their neighbours. The mobile agents usually migrate using the Conscientious strategy [221], however, once they sense the pheromone which they can service, they follow the pheromone gradient to its source to service the robot. The authors further extended their mechanisms by using cloning of mobile agent [125] to further hasten the process of the delivery of such services. The same authors have also proposed a stigmergy based distributed mechanism for controlling the population of heterogeneous mobile agents in multi-robot systems [128]. Their mechanism controls the selective rise and fall in the population of an agent, based on the demand of its service in the network. This mechanism has been emulated on real static as well as dynamic networks to showcase the practical viability of its use.

2.6 Chapter Summary

This chapter presented the core concepts of a mobile agent along with its basic characteristics, features, limitations and the scope of its use of real-world applications. The prominent mobile agent platforms were also discussed herein which spanned from the very old to the recent ones. The chapter further provided an overview of the application areas using mobile agents. As can be observed, there is a growing interest in the mobile agent technology for developing applications for large-scale distributed systems. The current upward trend in mobile agent technology and applications seem to act as a catalyst forcing many a researcher to revisit this technology and to reap its distinguishing features for use in complex, remote and real-time operations.

"Nature holds the key to our aesthetic, intellectual, cognitive and even spiritual satisfaction."

Edward Osborne Wilson (1929)
American biologist

3

An Immune Inspired Sieve for Selecting the Best Performing Solutions

Delivery of services, on-demand, in the form of code, program or data forms a crucial aspect for a middleware solution for networked robots. Mobile agents based mechanisms have been found to be effective in delivering such services to networked robotic systems [127, 124]. However, there may exist multiple solutions to a particular problem or service requirement. In such cases, it is always beneficial to find the best or an optimal solution among the solutions available in the system so as to improve overall efficiency of the system. Finding such solutions autonomously, depending on the current demand in a mobile agent based distributed middleware can aid in making the networked robotic system adaptive to dynamic changes in the environment.

This chapter presents a mobile agent based distributed mechanism acting as a sieve to arrive at a set of optimally performing solutions in a real system of asynchronously operating networked robots. In this context, the solutions mean a program to service a requirement/request at a robotic node. Henceforth, the words - robots and nodes are used interchangeably. These requirements could be of different types and can occur concurrently across a network. The novelty of the approach presented herein is that the environment comprising a network of robots acts as an active entity (rather than a passive facilitator) wherein the solutions in the form of payloads carried by mobile agents lie scattered in the network. These agents selectively mitigate the problems (service requirements) arising at different nodes in a decentralized and distributed manner. While the best performing solutions eventually dominate the network, the least performing ones are automatically purged from the system by the Sieve that is intrinsically embedded in the network. The mechanism is inspired by the Idiotypic network model proposed by Jerne [222]. This chapter thus presents the manner in which a real-world implementation of the Idiotypic network model can be conceived within a real networked system for selecting the optimally performing solutions. The chapter initially presents an overview of the Idiotypic network model and the motivation behind the emulation.

3.1 Immune Network Theory (Idiotypic Network)

The Immune network theory was proposed by Niels K. Jerne [131] in 1974 and was one of his crucial contributions for which he was awarded Nobel Prize in Medicine or Physiology in 1984. The Biological Immune System (BIS) comprises lymphocytes called T-cells (generated in the thymus) and B-cells (generated in the bone marrow) which are responsible for recognizing the antigens

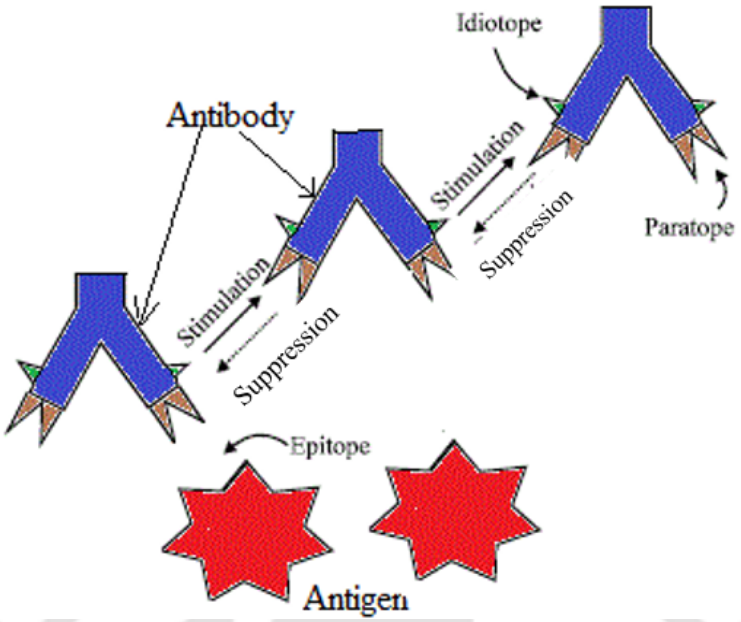


Figure 3.1: An Immune Network

through their receptors. These receptors are polypeptide chains present on the surface of these cells. To recognize an antigen within the body, these receptors have a Y-shaped structure called paratopes having a complimentary shape to that of the epitopes present on the surface of antigens. The degree of matching between the paratopes of these receptors and epitopes of antigens decide the affinity of an antibody towards an antigen.

According to Jerne, the immune cells within the BIS also recognize each other in the absence of antigens and create a network of cells within the body by interacting with one-another. Such recognitions happen as a result of *idiotopes* present on the body of the antibodies or cell receptors. These idiotopes are similar to the epitopes of the antigens which are recognizable by the paratopes of antibodies. Hence the antibodies which are either present as free molecules or as B-cell receptors form a network connected by the paratope-idiotope pairs. Whenever, an antibody detects an antigen by the matching of a paratope-epitope pair, it gets activated. Activated antibodies stimulate those which are connected to their idiotopes, while the ones recognized by their paratopes are suppressed. Figure 3.1 depicts the immune network with the idiotopes present on the surface of the antibodies together with the exchange of stimulation and suppression signals. Thus, in the event of an antigen recognition, a stimulatory signal propagates through the network of antibodies while a suppressive signal propagates otherwise. The interactions among the antibodies using such signals change their respective concentrations resulting in some elements being purged from the network while the others growing in concentration. This leads to the creation of a dynamic steady state. Thus the response to an antigenic attack is not limited to a single antibody but rather the reaction is created as a composite effect of a network of various antibodies.

3.2 Motivation

In the last decade, concepts derived from the BIS have drawn significant attention and have proved to be a potential source of inspiration for novel approaches to solve complex computational problems [223]. The theories and models proposed under the ambit of Artificial Immune Systems (AIS) [222]

try to capture the aforementioned features in order to use them for solving engineering problems. The ability of the immune system to distinguish between the self and the non-self along with other significant features such as specificity, learning, memory, scalability and distributed nature, have opened several new avenues of research in this field. The immune inspired models have been employed in various research areas including computer security, pattern recognition, fault detection, autonomous navigation in robots [224], natural language processing [225, 226] and a variety of other applications [222]. Specifically, the Idiotypic network model [131] is inherently autonomous due to the continuous interaction among the antibodies along with the ability to tune itself. In addition, the coupling of the paratopes and idiotopes modulate the responses of antibodies against any antigenic attack.

Farmer *et al.* [227] proposed a computational model for the change in concentrations of antibodies based on the Idiotypic network model. This computational model calculates the rate of change of concentrations of an antibody as a cumulative sum of the amount of stimulations and suppressions accumulated by it from other antibodies along with the feedback (stimulation) received from neutralizing the antigen. Death of antibodies due to ageing also affects this concentration by decrementing its value. The continuous interactions among the antibodies make the Idiotypic network model a dynamic one. In most AIS literature [228], these concentrations are always presumed to be a mere numeric single valued parameter. In addition, most of the implementations available for the Idiotypic network model are in the form of *simulations* [229] thus providing less room for their practical viability. This calls for an architecture or a framework which can emulate the inherent distributed and decentralized characteristics of the Idiotypic network model for use in real systems. The interactions among the antibodies, which can act as solutions serving nodes in a network within a pervasive environment [230], can be exploited to embed and evolve distributed intelligence. The same can also be used in the embedding intelligence into the applications in the domain of the Internet-of-Things (IoT) [25] and Cyber-Physical Systems (CPS)[26] for selecting a set of better solutions based on the needs and locally available parameters of the system.

As mentioned in the previous chapter (Chapter 2), the mobile agents provide for a fitting middleware platform to realize various population-based computational models in real distributed networked systems. Dasgupta *et al.* [231] describe a system for intrusion/anomaly detection and subsequent responses in networked computers using mobile agents. In their approach, the mobile agents roam around the computer nodes and routers and monitor the situation of the network. Inspired by the Clonal-Selection theory [232], the mobile immune agents used herein interact freely and dynamically with the environment and also with one another. Harmer *et al.* [233] have proposed a distributed static agent-based defence immune system to protect systems from intrusion and computer virus attacks. In their model the agents communicate with each other to protect the systems from intrusions. They have used the AgentMOM [234] communications framework developed by the AFIT Agent Research Group for implementing agent conversations. Castro *et al.* [235], have proposed an artificial immune network model, for data clustering and filtering redundant data. They have used a Euclidean shape-space model in which the network units correspond to the antibodies. The input patterns were treated as the antigens to be recognized and clustered. This network model was successfully applied to several clustering problems, including non-linearly separable tasks. Whitbrook *et al.* [236] have tried merging a Genetic Algorithm (GA) with an Idiotypic AIS to solve a mobile-robot target-finding problem. The GA was used to evolve the initial set of antibodies. They have shown that significantly higher antibody diversity can be achieved when a number of autonomous populations of antibodies are used. Yuan *et al.* [237] have also used the Idiotypic network concepts for solving the path planning problem for robots in complicated environments. Godfrey and Nair [126] describe an architecture of a multi-robot system that uses the AIS and mobile agents to service robots. Based on pain, nodes that control the robot are triggered

to indicate an antigenic attack. Mobile agents moving in a round-robin manner within the network carry the programs (antibodies) to decrease the pain levels of the robot. Bakhouya *et al*. [238] use the Idiotypic network model to regulate the population of mobile agents in a network. They assign three behaviours to each mobile agent viz. *clone*, *kill* or *move*. The mobile agents choose the behaviour having highest concentration while the inter-arrival times at the nodes constitute the antigen.

Although researchers have used mobile agents to realize AIS models, the mechanism for *mutual interactions* among these agents to evolve better solutions have not been addressed. The mobile agents can be used to collect feedback on the effectiveness of the solutions from a node in a robotic network which they eventually carry. This feedback can be translated into interactions viz. stimulations and suppressions among agents to evolve a better set of solutions manifested in the form of real-world distributed intelligence much like what happens in a real BIS.

3.3 Conceptualizing an Idiotypic Network in a Real Distributed Environment

Though quite a few researchers have used Farmer's [227] computational model to realize Idiotypic networks, in most of their implementations, the dynamically changing concentrations of antibodies relate to mere shared or global variables which are manipulated inherently in a sequential manner in the simulations. On the contrary in the biological Idiotypic network, there is no central entity that keeps track of such concentrations (shared global variables). The antibodies herein stimulate or suppress one another and are generated based on their affinities with the concerned antigen. A stimulation causes the concerned antibody to increase its concentration i.e. its population increases since it has proved itself to be more effective in curtailing the antigenic attack. The opposite happens in case of a suppression whereby its concentration reduces. Successive suppressions may eventually lead to the removal of such antibodies from the network. There is thus no actual physical link between all the individual members of the different antibody populations. If a specific antibody population seems to be more efficient in containing an antigenic attack, its population (concentration) increases since the other less effective antibody populations stimulate it to grow in number. Likewise, the more effective population also suppresses the growth of the other less effective ones.

In the computational realm, an Idiotypic network can be emulated just as in [239] wherein a number of nodes form a network. An attack on a node is synonymous to a problem faced by a robot in a network. The solutions to such problems, which could be a set of parameters values, are maintained a priori within the system based on past events. Such solutions are carried by a set of heterogeneous mobile agents as their individual payloads; each agent carrying one solution each. These agents act as antibody carriers. In a real world scenario, the Idiotypic network evolves not by forming physical links between each and every agent or by manipulating global shared variables signifying the concentrations of the individual antibodies, but by increasing (stimulating) or decreasing (suppressing) the actual populations (concentrations) of the various types of antibodies in the network in a distributed and decentralized manner. The interactions (stimulations and suppressions) among the antibodies cause dynamic changes in their respective populations thus contributing to a dynamic network. Antibodies (mobile agents) by themselves remain oblivious of the concentration of their individual populations (number of agents) within the network.

Figure 3.2 depicts a portion of the real network on which the *Idiotypic Sieve* (described later in Section 3.4) is embedded. As can be seen the portion contains a number of networked nodes

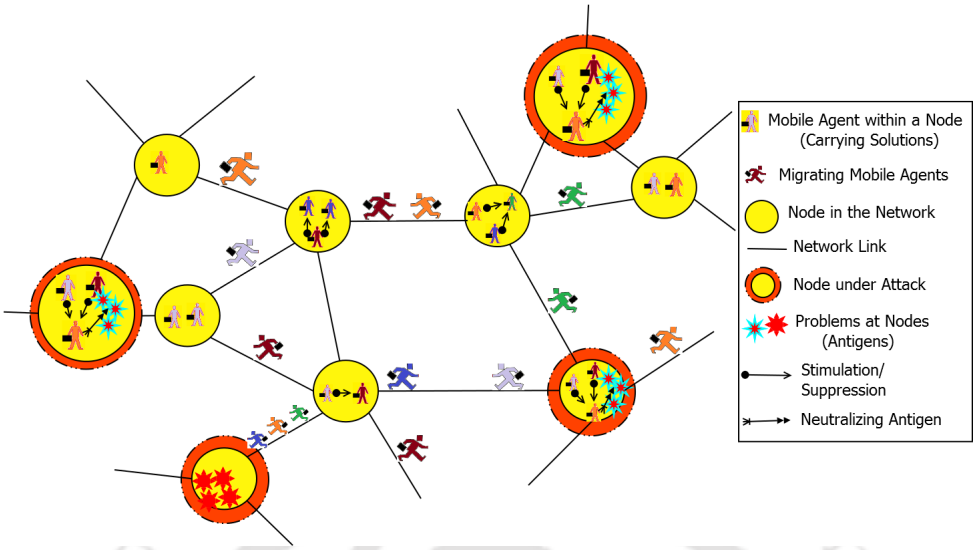


Figure 3.2: A portion of a network of nodes (depicting the components within) on which Idiotypic Sieve was implemented.

which constitute computers, devices and/or robots. Since the nodes are heterogeneous in nature the network could be either static (stationary) or dynamic (mobile). Mobile agents that carry the antibodies (solutions to problems) for a range of antigens (problems or service requirements) as their payload migrate constantly within the network. A node-under-attack is shown with a red periphery indicating that the node is in need of a solution carried by mobile agents inhabiting the network. For instance, a node-under-attack can be a mobile robot requiring an obstacle avoidance program. There could be several mobile agents carrying different solutions/programs for obstacle avoidance. In the portion shown in Figure 3.2, two nodes are attacked by a blue antigen while another is attacked by a red one indicating concurrency and heterogeneity in the nature of attacks. These nodes flash danger signals, described later, by virtue of which they attract the right set of agents that contain solutions to the problem at hand. The solution carried by one of the agents that arrive at the node-under-attack is then selected and used to solve the issue. Based on the feedback received after this, the set of agents stimulate and/or suppress one another before migrating to other nodes. Stimulations accumulated over time at various nodes cause the agent to clone thus increasing its population, while suppressions tend to decrease its life-time which eventually results in its death and consequent removal from the network. The mechanism is thus akin to that of the Idiotypic network [131] and works on a real, asynchronous, distributed and decentralized environment where concurrent and heterogeneous service requirements can occur at many nodes in the network.

The next section describes the *Idiotypic Sieve* to evolve the set of optimally performing solutions using local stimulations and suppressions within the nodes.

3.4 The Idiotypic Sieve

The Idiotypic Sieve is the underlying mechanism that is embedded within a real network of nodes such as the one shown in Figure 3.2. The entire system including the Idiotypic Sieve consists of a physical network of nodes (such as networked mobile robots with devices and sensors). In addition, there exist sets of mobile agents, each of which carries a distinct solution to various problems (service requirement) that may occur at various nodes in the network. These problems

constitute the antigenic attack at a node. These could be the requirement of a program by a robotic node to accomplish a task, a response to an alarm raised by a sensor in the network or a procedure to be executed at a node. Each node in the network is equipped with an agent platform to facilitate all the functionalities of mobile agents. The problems and the solutions have their own descriptors. Descriptors need to be fashioned *a priori* based on the application under consideration. The problem descriptors form the epitopes of an antigen while the solution descriptors act as the paratopes of the antibodies. An affinity function, ψ_{p_i} for each problem p_i, defines the degree of interaction between the epitopes of an antigen and the paratopes of the antibody i.e. the affinity of a solution carried by a mobile agent to the corresponding problem p_i. A feedback function, ξ_{p_i} for each problem p_i, provides a performance measure for the solution applied to circumvent the problem p_i. This feedback ξ_{p_i} can be attributed to antigenic stimulation in the Idiotypic Sieve discussed herein. The performance measure can be viewed as the qualitative feedback of the application under consideration. It could be the total time required to neutralize the attack or the amount of resources such as energy expended or the state of the node-under-attack after applying the solution, etc. It may be noted that the functions ψ_{p_i} and ξ_{p_i} are dependent on the application under consideration and must be modelled based on the requirements or the objectives of the same.

The occurrence of a problem or service requirement at a node initiates an antigenic attack. These attacks can be sparse, for instance a robot requesting an obstacle avoidance routine, or dense such as the detection of fire by a set of fire sensors (spread across the network) raising an alarm at all the respective nodes. Different types of antigens (problems) may also attack several nodes concurrently in which case the respective services (solutions) may have to be provided accordingly in a similar (concurrent) fashion.

3.4.1 Circumventing an Attack

The mobile agent based network service model used herein is similar to the one proposed in [129]. The mobile agents that carry the solutions continuously migrate within the network of nodes using the *PherCon* migration strategy, as described in [124]. As soon as a node detects a problem or requirement of a service, it starts emanating danger signals to its immediate neighbours which in turn diffuse the same onto their neighbours at a lesser intensity than that received. As shown in Figure 3.2, these danger signals penetrate the immediate neighbourhood of the node-under-attack similar to the pheromone diffusion model proposed by Godfrey and Nair [124]. Danger signals contain the network identifier of the node-under-attack, the previous node from where the signal emanated and the problem descriptor which essentially forms the epitope of the antigen. It also contains a danger signal strength whose intensity wanes as it moves away from the node-under-attack. The propagation of the danger signal continues till its strength dies down to zero at nodes in the neighbourhood of the node-under-attack, thus forming a *danger zone* around it, as shown in Figure 3.2.

Under normal conditions, the mobile agents migrate using a conscientious mechanism by which they tend to avoid visits to recently visited nodes thereby ensuring a uniform distribution of their visits across the network. Whenever, an agent detects a danger signal at a node, it ascertains whether it can cater to the attack by calculating the affinity ψ_{p_i} between the solution it is carrying (antibody) and the problem descriptor of the antigen within the danger signal. This process is akin to the paratope-epitope matching. If the affinity ψ_{p_i} is less than a threshold Γ_{p_i}, the affinity threshold, the mobile agents ignore the danger signals and continue to migrate to other nodes. However, if this affinity ψ_{p_i} is found to be greater than Γ_{p_i} then the mobile agent assumes that it can cater to the problem at the node-under-attack. It then moves on to the node-under-attack by following the danger signal strength gradient. This gradient aids the mobile agent which

has detected the danger signal to reach the node-under-attack via the shortest path [124]. This mechanism of attracting the relevant mobile agents could lead to many of such agents reaching the node-under-attack. However, those mobile agents which are redundant i.e. carry the same solutions at the node-under-attack, continue their migration into the network. The node-under-attack thus gets populated with several mobile agents having distinct solutions to the same problem.

One of the solutions carried by the mobile agents within the node-under-attack is selected to neutralize the antigen using the roulette wheel approach [240] over the respective affinities of the solutions carried by these agents. After neutralizing the attack, the mobile agent carrying this solution receives the antigenic descriptor and the antigenic stimulation using the feedback function ξ_{p_i}, both of which are stored within its payload.

3.4.2 Conceiving the Idiotypic Sieve

Once the selected mobile agent receives the antigenic feedback, all agents within the node-under-attack interact with each other after which they continue their migration to other nodes in the network. Such interactions could take place at all nodes in the network concurrently. The mobile agents interact with only those agents which carry solutions to similar problems. This means that their respective antigenic descriptors are similar. This similarity can be attributed to the paratope-idiotope matching within the Idiotypic network model. A simple method to find the related agents could be based on the affinity function ψ_{p_i} as discussed in the previous section. However, more complex matching functions can be derived depending upon the requirements and the nature of the system or application under consideration.

Once a mobile agent has identified the set of related agents with whom it can interact within a node, it exchanges the stimulation and suppression signals. The exchange of the stimulations or suppressions is based on the past performance ξ_{p_i} of the mobile agents in servicing the problem p_i. Mobile agents having higher ξ_{p_i} values are stimulated by the rest while these higher ones suppress those with lower values. These stimulations or suppressions are accumulated within a parameter called an *endurance potential*, γ_{m_i} of a mobile agent m_i which is defined as the difference between the accumulated value of the stimulations and the suppressions stored within each agent. The dynamics governing the value of γ_{m_i} are discussed later. If the value of γ_{m_i} crosses a maximum threshold (γ_{max}), the mobile agent m_i clones itself whereas if the same goes below a minimum threshold (γ_{min}) the agent m_i terminates itself (*apoptosis*). Cloning increases the populations of those agents whose solutions are more effective which in turn forms the ones that dominate the network. Those agents whose solutions are ineffective are purged out of the system. The mechanism described herein thus acts as an Idiotypic Sieve spread across the network which filters out ineffective solutions while retaining the more effective ones. It may be noted that the Idotypic interactions are local and asynchronous in nature, across the network. The mechanism is distributed and decentralized, can act concurrently and does not use any globally shared parameters. As can be observed, the Idiotypic Sieve is not dependent on the number of nodes in the network thus making the overall mechanism scalable.

3.4.3 The Underlying Dynamics

Farmer's computational model [227] for Jerne's Idiotypic network [131] can be expressed in a general form as:

Change in Concentration of an antibody = Antigenic Stimulation (ξ_{p_i}) + Stimulations received from other antibodies ($S^t_{m_i}$) - Suppressions from the selected antibodies ($S^u_{m_i}$) - Deletions due to disuse or lapse of lifetime (Ω_{m_i})

In the Idiotypic Sieve, the change in the concentration of antibodies emerges as the cumulative effect of the stimulations or suppressions received by the individual antibodies. The equations that govern the dynamics of the formation of the Idiotypic Sieve and the consequent changes in the populations of various mobile agents carrying their respective solutions within the network are given below.

The endurance potential γ_{m_i} for a mobile agent m_i accumulates all the stimulations and suppressions including the antigenic ones. The value of γ_{m_i} when a solution is selected to neutralize a problem p_j at the node-under-attack is given by:

$$\gamma_{m_i}(n+1) = \gamma_{m_i}(n) + \xi'_{p_j} \tag{3.1}$$

where

$$\xi'_{p_j} = \frac{1}{1 + e^{(\alpha_{Ag} - \beta_{Ag}\xi_{p_j})}} \tag{3.2}$$

$\gamma_{m_i}(n)$ denotes the value of γ_{m_i} at the n^{th} instant. ξ'_{p_j} is used to squash the value of ξ_{p_j} between 0 and 1.

The change in the value of γ_{m_i} due to interactions within a node is given by Equation 3.3.

$$\gamma_{m_i}(n+1) = \gamma_{m_i}(n) + \sum_{j \in S_T} S^t_{m_j} - \sum_{k \in S_U} S^u_{m_k} \tag{3.3}$$

where $S^t_{m_j}$ denotes the stimulations received from the mobile agent m_j and $S^u_{m_k}$ denotes the suppressions received from the mobile agent m_k within the node. S_T and S_U are the set of related mobile agents within the node that impart the stimulations and suppressions respectively to the antibody carried by the mobile agent m_i. The values of $S^t_{m_i}$ and $S^u_{m_i}$ is calculated as:

$$S^t_{m_j} = \frac{k_t \gamma_{m_j}(n)}{N_{m_j}} \tag{3.4}$$

$$S^u_{m_k} = \frac{k_u \gamma_{m_k}(n)}{N_{m_k}} \tag{3.5}$$

where N_{m_i} for a mobile agent m_i denotes the difference between the total number of unique antigenic descriptors that the mobile agent m_i is carrying and the number of problem descriptors matching with related agents. k_t and k_u are the non-zero positive constants.

The ageing due to the lifetime of the antibodies as mentioned in Farmer's model [227] is implemented by conferring a fixed number of hops (Ω_{m_i}) to every *cloned mobile agent* (antibody) in the network. The value of Ω_{m_i} is reduced by unity whenever a mobile agent lands on a node in the network. Once the value of Ω_{m_i} becomes zero, the mobile agents are terminated and hence removed from the system.

3.5 Experimental Setup

The above mechanism was implemented on a real networked environment using the *Typhon* mobile agent framework [132]. This section provides a brief description of Typhon framework before discussing the details of the implementation of the Idiotypic Sieve.

3.5.1 Typhon - Mobile Agent Framework

Typhon is a WIN-Prolog based mobile agent framework built over the Chimera agent platform shipped along with the LPA WIN-Prolog [172]. The basic core of *Typhon* as developed by Matani and Nair [132] provides all agent based functionalities such as agent creation, migration, cloning, attaching/removing payload to/from an agent, etc. However, for the purpose of experimentation, the core of *Typhon* was augmented with additional features in order to facilitate seamless creation and management of large size networks running on the *Typhon* mobile agent framework. The extensions made to the basic *Typhon* framework are as follows:

- Log server: A log server was developed and integrated to the *Typhon* framework for logging the activities of various events taking places at different nodes of the network. This includes creation of agents and clones, migration of agents, the states of agents and nodes, etc.
- LPA WIN-Prolog to LEGO NXT Interface: In order to integrate the networked robots with *Typhon*, an interface nicknamed LPA-PRO-NXT [241] was developed using which the mobile agents within *Typhon* could communicate with Lego® Mindstroms® NXT [1] robots.
- GUI: The core of *Typhon* uses a command-line based interface for all kinds of user interaction with the system. Since managing such an interface is difficult and cumbersome, a Graphical User Interface (GUI) was developed for *Typhon* to facilitate the automated generation and termination of various networks, deployment of agents, etc.
- Network Creation: The support for generating different network configuration by choosing number of nodes, type of topology and number of physical PCs were added for generating distributed networks.
- Dynamic Networks: The support of dynamic network topologies was also integrated into the framework for realizing the emulation of real mobile computing environments. In the emulated dynamic network, a node drops the connection with any of its current neighbours or creates a new link to another node in the network with a given probability.

The extended version of *Typhon* has been used for performing all the emulation experiments presented through-out in this thesis.

3.5.2 Implementation

The support for the Idiotypic Sieve was integrated into the *Typhon* framework using WIN-Prolog. Hence, an instantiation of the *Typhon* framework acted as a node in the network. Multiple instantiations of *Typhon* nodes were used to create the entire network using more than 10 PCs connected to each other via TCP/IP connections over a LAN. Though the testing of the Sieve was initially performed using static networks, it was felt that the study of its performance on dynamic networks would throw more light on its efficacy in mobile computing environments. The dynamic networks were generated by continuously altering the neighbours of the nodes in the *Typhon* based network with a given probability.

In the implementation, binary (5-bit) sequences were considered to form antigens which were presented at several nodes (nodes-under-attack) to emulate multiple heterogeneous and concurrent antigenic attacks within the network. This binary sequence forms a generalized representation of a multi-dimensional vector manifested as a problem at a node/robot. The sequence could be synonymous to a set of sensor vectors provided by a robotic node or the state information at a node in the network (problem descriptor) for which an action (solution) is to be performed.

The corresponding best antibody is considered to be a 5-bit complemented sequence capable of neutralizing the antigen. In the presented Idiotypic Sieve, every mobile agent carries with it one 5-bit neutralizing sequence constituting arbitrary solutions (candidate antibodies). The Hamming distance [242] (η_H) was used for deriving the affinity function (ψ_{p_i}) between the binary sequences constituting the antigens and the antibodies. The decimal equivalent of the binary operation ***XoR*** between the binary sequences of antigens and antibodies was used as the reward function (ξ_{p_i}). The value of ψ_{p_i} in the implementation is given by:

$$\psi_{p_i}(E_{Ag}, P_{An}) = \eta_{H_{max}} - \eta_H(E_{Ag}, P_{An}) \tag{3.6}$$

where E and P denote the bit sequences of the epitopes of the antigen and the paratopes of the antibody respectively. $\eta_{H_{max}}$ is the maximum possible Hamming distance between E_{Ag} and P_{An}.

Initially since there are no antibodies in the network, the nodes were allowed to generate random antibodies (5-bit binary sequences) pro-actively after the danger signals had died down. The nodes generate a random 5-bit pattern and ascertain its ψ_{p_i} value. If the same is greater than Γ_{p_i} (Section 3.4.1) then the node treats this pattern as the antibody or solution to neutralize the antigen. This new antibody or solution is then encapsulated within a mobile agent and released into the network. Each experiment consisted of multiple rounds of concurrent and heterogeneous antigenic attacks. The antibodies generated in each round were retained for use in the subsequent ones.

3.6 Results and Discussions

Experiments were performed on dynamic networks comprising 100 nodes by forcing distinct antigens viz. 5-bit sequences to attack separate nodes chosen at random from the network. The number and types of antibodies (mobile agents equipped with 5-bit binary sequences) generated/terminated in each round were recorded to plot the graphs. The following values of the parameters were used in the experiments: $\Gamma_{p_i} = 4, \alpha_{Ag} = 10, \beta_{Ag} = 0.5, k_t = 0.5, k_u = 0.5, \gamma_{max} = 1.5, \gamma_{min} = 0.5, \Omega_{m_i} = 1000$

The graph in Figure 3.3(a) shows the change in the population of antibodies in each round of antigenic attack within the dynamic network of 100 nodes. The antigens used for the attack were: [00000], [11111], [10101]. As can be observed, a total of 14 unique antibodies were generated by the nodes in the 15 rounds of attacks. However, only three antibodies viz. [11111], [00000], [01010] which are the more optimal solutions, can be observed to be actively growing in their respective populations. The population of the rest, which constitute the non-optimal solutions, remain limited and eventually dwindle within the network indicating clearly the effectiveness of the Idiotypic Sieve to retard the generation and proliferation of the less performing antibodies while allowing the dominant ones to grow in number. It can be noted that the three dominating antibodies are the complementary bit sequences of the antigens used for the attacks. Also, the antibodies [11110] and [01000] have maintained a small population 3 and 1 respectively in the network. The reason behind this is that the left over population of both these antibodies [11110] and [01000] have not encountered their related dominating antibodies within the nodes of the network so that they can be suppressed and completely removed from the system.

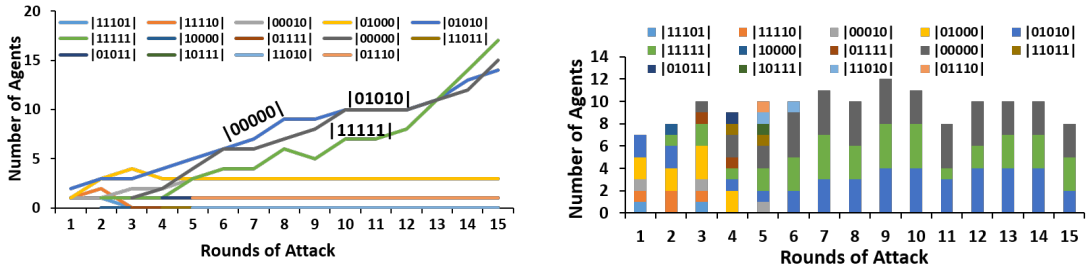


Figure 3.3: Antigens attacking in each round: [00000], [11111], [10101] (a) The change in the population of different Antibodies (b) The number of Antibodies generated

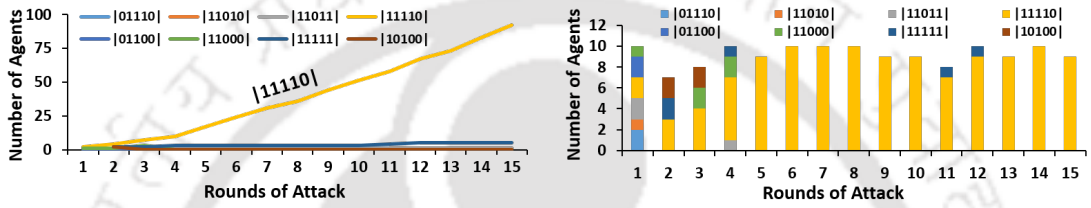


Figure 3.4: Antigens attacking in each round: [00000], [00001], [00011] (a) The increase in the population of generic Antibody along with other Antibody populations (b) The number of Antibodies generated

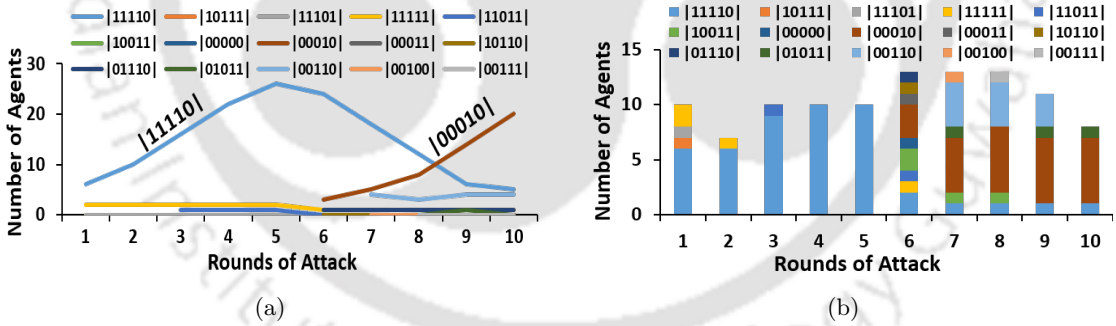


Figure 3.5: (a) The change in the population of different Antibodies (b) The number of Antibodies generated, when Antigens [00000], [00001], [00011] attacked the nodes in the initial five rounds while [11100],[11101],[11001] attacked in later five.

The graph in Figure 3.3(b) depicts the number of antibodies of each type generated in each round of attack. As mentioned earlier (Section 3.5), the antibodies can be generated either by the nodes-under-attack (randomly, if they do not receive any antibody) or by cloning. It can be observed in Figure 3.3(b) that different types of antibodies are generated till round 5. Beyond this only the three distinct antibodies dominate within the network. This shows the capability of the Idiotypic Sieve to selectively maintain the population of best performing solutions within the network while weaning off the remaining ones.

To verify the ability of the Idiotypic Sieve to retain not only the optimal solutions but also

generic ones, another set of three antigens viz. [00000], [00001], [00011] having very small Hamming distances amongst each other were used to attack the network. The graphs in Figures 3.4(a) and 3.4(b) show the results gathered through 15 rounds of their attack. It can be clearly seen from the graph in Figure 3.4(a) that the antibody [11110] solely dominates the network. The computed ξ_{p_i} values for the dominant antibody against the three antigens [00000], [00001] and [00011] were found to be 30, 31 and 29 respectively using the method of calculation mentioned in Section 3.5. These constitute the three highest ξ_{p_i} values achievable within the network which makes this antibody receive maximum stimulations. It also means that it suppresses all other antibodies. It may also be noted that the bit sequence [11111] can cater to the antigens [00000] and [00001] but not to [00011] since the ψ_{p_i} value in case of [00011] is less than 4 i.e. $\psi_{p_i} < \Gamma_{p_i}$ (Section 3.4.1). Hence the population of [11111] can be observed to be thriving within the network. Similar observations can be drawn from the graph shown in Figure 3.4(b) wherein the antibody [11110] generates clones from round 5 to 15. Traces of antibody [11111] can also be observed in round 11 and 12.

The graph in Figure 3.5(a) shows the variations in the populations of different antibodies when the set of antigens - [00000], [00001] and [00011] were used to attack in the first five rounds while the set of antigens - [11100], [11101] and [11001] were used in the rest of the rounds. As can observed, the population of the antibody [11110] grows initially till round 5 and then starts to wane due to death of its clones which have a fixed value of Ω_{m_i}. On the contrary the new antibody [00010] grows within the network since its requirement at the various nodes is higher. This emphasizes the adaptive nature of the Idiotypic Sieve which endeavours to retain and proliferate only those antibodies that are required or are in demand in the network. The graph shown in Figure 3.5(b) also depicts the waxing and waning of the population of the antibodies [11110] and [00010] respectively, in the network. From the results it is clear that the Idiotypic Sieve can effectively empower a network of nodes to churn and retain the populations of the optimal solution(s) required, on-demand while suppressing the proliferation of the others. Further, since all experiments were performed using dynamic networks, one may note that this sieve can cope up with the rigours encountered in mobile computing environments.

3.7 Chapter Summary

This chapter portrayed an Idiotypic Sieve that can be used in conjunction with a real distributed network of robots. The Idiotypic Sieve is conceived by emulating an Idiotypic network model within a computational networked environment. The performance of the Idiotypic Sieve was tested in dynamic network of 100 nodes. The results portrayed the robustness and usefulness of the Idiotypic Sieve for networked robots. Embedded within a distributed network of nodes, the Idiotypic Sieve selectively evolves the best performing solutions based on the current demand (problems) in the network and purges off the others. The results further show that the Idiotypic sieve also generates generic solutions that can cater to a range of problems with high degrees of effectiveness. It can also cope up with multiple, concurrent, asynchronous and heterogeneous attacks at the various nodes in the network. Scalability of the system is also inherent as the sieve functions oblivious of the number of nodes in the network.

The Idiotypic Sieve presented in this chapter functions in a manner similar to the Idiotypic network theory wherein the number (concentration) of those mobile agents that can neutralize the problem (antibodies) increases while others are purged. The mobile agents of each sub-population carries the same solution within themselves. The increase in number of mobile agents of a particular type allows to quickly service the whole network in order to provide solutions to the robots currently facing the requirement of the same. In addition, the Idiotypic Sieve has been emulated on real

network settings which emphasizes that the migration of mobile agents carrying solution and their mutual interactions can be applied in real systems. Further, the notion of Idiotypic Sieve is not limited to the networked robots and can be extended to any other networked system such as sensor networks, Internet-of-Things (IoT) or Cyber-Physical Systems (CPS). Since the Idiotypic Sieve does not assume any specific parameter exclusive to the networked robots, it opens up a wide avenue of networked systems which can use the proposed model catering to their specific requirements.

In this chapter, the mutual interactions among the mobile agents within the presented Sieve was only limited to the exchange of stimulation and suppression signals. While the Idiotypic Sieve contributes to filtering off non-optimal solutions and retaining other solutions within the network using agent-agent interactions (stimulations and suppressions), it does not seem to make any further use of such communications (interactions) amongst participating agents. If the mobile agents were to exchange the information (solutions for instance) they carry in an appropriate manner, it may so happen that they enrich one another based on their experience (feedback or affinity) and facilitate the growth of better solutions thereby aiding in learning as they migrate in the network. Such a distributed and decentralized learning mechanism can be realized only if agents meet at a node and exchange their information in an orderly manner giving rise to a learning framework within a network. The next chapter thus presents a distributed framework for intelligence-sharing and learning using a set of mobile agents within a networked robotic environment.

"Intelligence is based on how efficient a species became at doing the things they need to survive."

Charles Darwin (1809 – 1882)
English naturalist

4

Mobility and Sharing - Catalysts for Learning

Sharing of information, learning and adaptation in a distributed system of networked robots can greatly enhance performance and utility. However, achieving the same in the presence of asynchronous entities is a complex affair. Multi-agent system paradigms possess intrinsic similarities with such distributed systems and thus provide a fitting platform to solve the problems within. Traditional approaches to efficient information sharing and learning among autonomous agents in distributed environments incur high communication overheads [89]. Non-conventional tactics based on social insect colonies provide natural solutions for transfer of social information in highly distributed and dense populations.

This chapter presents a framework for sharing intelligence and mutual learning among a set of location-unaware spatially segregated networked entities such as robots in a distributed environment; all of which have the same learning objective. The framework couples localized communication with the available multi-agent technologies to realize asynchronous intelligence-sharing and learning. The mechanisms used within the framework are inspired by those exhibited by social insects. Static agents, that remain resident within the entities (viz. robots) in the network, and their mobile counterparts both aid in facilitating the exchange of information and localized sharing in the robotic network. The framework focuses on *generic* mechanisms required to be embedded within individual mobile agents so as to result in the overall convergence of the entire agent population towards a common goal. The proposed framework is fully distributed in the sense that neither of the agents (both static and mobile) possess any knowledge about the overall number of such agents present in the network nor do they possess any location information about other agents. In addition, the sharing of information among the agents is completely asynchronous and local. The major contributions of this work include:

- A multi-agent framework for distributed intelligence-sharing and learning
- Modalities for local sharing and exchange of information
- Dynamics for facilitating agent migration, inter-agent interactions and asynchronous executions

4.1 Motivation

Social insect colonies provide the most natural examples of large scale multi-agent systems. These complex and self-organizing societies function based on very simple processes of information transfer

between the individuals, thus providing an ideal perspective to understand the mechanisms of social learning [243]. Social insects invest considerable effort in passing on learned information to others in their group or swarm. The value of information obtained from others depends on the context [244]. Such social learning systems are often flexible enough to ensure that individuals rely on social information only when individual learning does not suffice. In an insect colony, information learned by an individual is not actually broadcasted to its peers or mediated through a supervised or centralized control. On the contrary, the information flows through local interactions among individuals e.g. *Trophallaxis* in bees [245] and *Tandem running* in ants [246]. It has been demonstrated that bees learn associations between floral scent and nectar rewards during trophallatic interactions, just as they would if they were to sample the flowers themselves [244]. These local interactions among the individuals within an insect colony add up to eventually alter the behaviour of the entire colony and converge their searches to the location having maximum availability of nectar.

These complex yet versatile systems of insect colonies have been a source of inspiration for many a researcher in various fields of engineering and science [247]. In the context of learning in multi-agent systems, social insect-colony based models have been of notable interest [248, 249]. In this chapter, the focus is to exploit the use of social interactions among *nomadic* agents so as to facilitate asynchronous sharing and co-operative learning among the entities of a distributed environment. Exchange of information among these *mobile* agents, populating a network of nodes, takes place locally within a node, as and when they meet other such agents within. These local interactions tend to increase the quantum of knowledge they possess. The mobile agents use this extra knowledge gained through local interactions that are spread spatially across the network, for enhancing their self-centric learning thus reducing their individual as also overall search spaces. This learned information is thus provided to the static agents resident within the entities of the distributed environment, which evaluate the new information and provide valuable feedback for further enhancement.

For illustration, imagine a scenario wherein multiple mobile robots are trying to learn the solution to the same problem such as how to draw a circle, how to tighten a nut and bolt, assembling a chair, etc. As can be observed, these tasks require a specific sequence of actions to be performed in a specified manner to achieve the desired objective. Since there are multiple learning robots in this scenario, sharing of their individual experiences can enhance the performance of the whole system and also reduce the time the robots take to achieve the goal. However, sharing information in such a setting wherein the robots are dynamic entities is not trivial. Drawing inspiration from the social insects, one of the possible ways to share information could be by making a robot move to the vicinity of another and share information locally. The robots can gather such locally shared information over time and then use sophisticated learning tools to create a new plan of action. The robots can evaluate this new plan by executing them individually and consequently find the amount of progress they have made in achieving the goal. Repeating such a process would eventually lead all the robots towards learning the requisite sequence of actions in order to achieve a specific objective. The point to be noted here is that the *mobility* of the robots contributes to the spreading of the information in the environment. However, mobility in robots is a costly affair in terms of both energy and actuation. The problem may become more critical if we think of a large network of robots wherein a few of them are only trying to learn and share a common objective.

In another scenario, assume a large Campus Area Network (CAN) populated by different kinds of devices as its nodes forming an IoT. These devices may include several PCs, projectors, air conditioners, sensors, Wi-Fi routers, printers, etc. In such a setting, imagine that a set of devices is required to find/learn to use a specific set of parameters (such as resolution (dpi), mode, paper-type, toner density, etc. in case of a set of printers) so as to optimize their life-time and

utility. It is also possible that the devices need to adapt their settings based on their make and model. The simplest or naïve solution will be to package each device with an algorithm or program which always tries to figure out the optimized settings based on the user-feedback it receives on its own current settings. Assume that if the parameters do not result in good quality printing, the user changes these settings to suit her/his needs of better quality printing by adjusting the dpi, toner density, etc. This could be used as a feedback for the algorithm embedded within the devices. The problem with this approach is that each device (say printer) would try to solve the same problem repeatedly and hence there would be wastage of power, paper and cartridges. The life-time of the printer would also go down due to wear and tear during this laborious learning phase. A smarter way would be to share the locally learned information among the printers as in the case with mobile robots. This will not only reduce the wastage but also improve the time to learn the parameter values since multiple printers would be collaborating to achieve the same goal. However, it is not essential that these printers know the location of other such printers of their kind on the network. Further with no physical mobility and sophisticated programs on-board to handle communication complexities, learning optimal settings by a heterogeneous set of networked location-unaware printers, autonomously in the CAN, becomes a challenging task.

In the multi-agent based approach portrayed herein, the framework tries to leverage the mobility based local sharing model of the mobile robots to alleviate the challenges discussed above. While a static agent manages local tasks at the physical learning entity i.e. a robot or a printer, its mobile counterparts (mobile agents) provide the much needed mobility of all the learned information. Imagine the network of robots supports a framework with all agent based functionalities as proposed in [129]. The authors in [129] describe the methodologies how a mobile agent based framework for a network of heterogeneous devices can be conceived. Let us further assume that the static agents reside within each of the networked robots. These agents manage local information and configuration of a robot such as preserving feedback, executing an action, etc. A set of mobile agents embedded with a learning algorithm suited for such an application could be released into the network of robots. These mobile agents then forage for the robots which are trying to learn within the network and facilitate the exchange of information locally with static agents hosted within each robot. Further, these agents can search the network to share their information with other such mobile agents asynchronously. When a mobile agent encounters other such agents at the various nodes in the network (not essentially the targeted robots) it exchanges information on the newly learned aspects of the solution to the problem. As a result of sharing each mobile agent comes up with a new set of actions/plan as per its learning algorithm. This new learned information is then provided to the static agent hosted within the robots which in turn executes the new plan or solution and provides its feedback to the mobile agent. The latter continues its sojourn in the network of robots after receiving the feedback. The process of learning and sharing continues till eventually all such mobile agents agree on the same set of actions indicating convergence. It may be noted that the job of collaboratively learning the optimal set of actions is achieved by the set of robots using local communications of migrating mobile agents. Similar applications such as learning optimal printer settings, finding unique genome sequences among different databases distributed across a large network, etc., can be envisaged using such multi-agent based approaches.

Although the above mentioned approach may seem trivial, it puts forward many interesting challenges that are crucial in the realization of an asynchronous and distributed intelligence-sharing and learning framework. These include:

1. The parameters both Input and Output such as number of mobile agents, learning algorithm, etc. that are essential to regulate the functioning of the framework.

2. The dynamics associated with the different parameters involved within such as agent migra-

tion, on-node execution, etc.

3. The duration for which a mobile agent should search for other mobile agents so as to share or receive information.

4. The mechanisms for the movement and exchange of learned information.

5. The formulation of conditions that will subsequently trigger the execution of the learning algorithm using the newly collected information.

The succeeding section presents the proposed multi-agent based approach to model the above mentioned challenges followed by the framework in detail.

4.2 Proposed Framework

This section discusses the parameters, the inputs and the mechanisms required to realize the proposed multi-agent framework for distributed and asynchronous sharing of intelligence along with the formalism to model all the processes and interactions used within.

4.2.1 System Model

The system under consideration is modelled based on the following:

- W is an undirected connected network, where $W = (N, E)$ wherein nodes are location-unaware.
- N is a set of nodes such that $N = \{n_i | i \leq C_1\}$, where C_1 is the total number of nodes in the network W, $i, C_1 \in I$ where I is a set of positive integers.
- E is a set of links such that $E = \{e_i | i \leq C_2\}$, where C_2 is the total number of links in the network W, $i, C_2 \in I$.
- A is a set of static agents resident on each of the nodes such that $A = \{a_i | i \leq |N|\}$, $i \in I$.
- M is a finite set of autonomous mobile agents such that $M = \{m_i | i \leq K\}$, where K is the total number of mobile agents in the system and $i, K \in I$.
- P is an application dependent user-defined learning problem such as finding a sequence of actions to execute a particular task.
- G, the goal, is the required outcome or the desired response as prescribed by the user of the system upon solving P.
- L is a learning algorithm provided by the user based on the underlying application which is carried by each mobile agent as its payload.
- φ is a set of actions that a static agent can execute in any order with or without repetitions to achieve the goal G.

A mobile agent $m_i \in M$, which is by itself an autonomous program, is capable of migrating from a node n_i to another node n_j if there exists a link e_i between (n_i , n_j), where n_i, $n_j \in N$ and $e_i \in E$ within W. Each node n_i hosts an agent framework such as [132] that is capable of managing all agent related functionalities. Each node n_i also maintains a queue (Qn_i) of mobile

agents $m_i \in M$ present within node n_i. Each queue Qn_i at node n_i has a fixed length $\Gamma < K$, $\Gamma \in I$, i.e. Qn_i can host a maximum of Γ mobile agents within node n_i.

It is apparent that if $|Qn_i| = \Gamma$, no mobile agent can enter the node n_i. Each node is a uniprocessing entity thus all the operations within a node are executed sequentially. Figure 4.1 shows a typical network along with the mobile and static agents, the agent framework, the learning problem and the respective queues within.

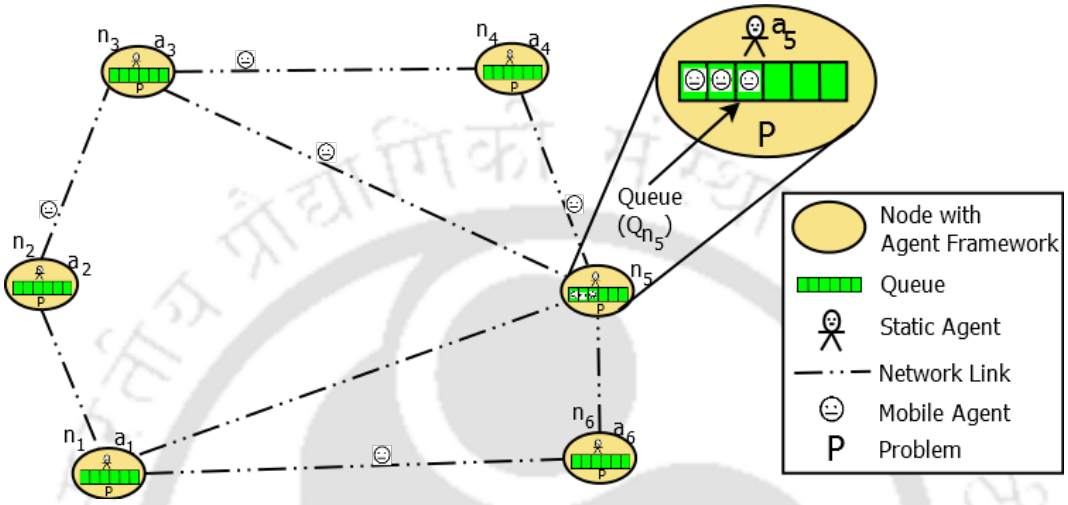


Figure 4.1: A network of nodes having the agent framework, the static agent, the learning problem and the queue within. The mobile agents are shown to be either migrating from one node to another or resident within the queue of a node.

In the proposed framework, the mobile agents carry the learned information within themselves, share it with other such agents during their sojourn in the network. They also assimilate the newly gathered information to discover newer paths towards the goal G and execute the same at a node using the help of the static agents within. They collect the feedback and enrich their learned information and once again set out to collect more information from other such agents. Both, the mobile and static agents, co-exist in the network and share and execute to eventually achieve their objective.

4.2.2 Inter-Agent Interactions

The interaction among the agents forms a crucial component of the proposed framework. As can be observed, there can be two kinds of possible agent interactions within the framework namely mobile-to-static agent interaction and mobile-to-mobile agent interaction. Since the network under consideration is distributed having location-unaware nodes, the static-to-static agent interaction is not possible under this framework.

Mobile-to-Static Agent Interaction

Mobile and static agents interact with each other in the following ways:

(a) En-queue: A mobile agent m_j can request a static agent a_i on the node n_i to execute an *en-queue* operation so as to enter Qn_i to effect its migration from another node n_k to n_i.

(b) De-queue: A mobile agent m_j can request the static agent a_i on the node n_i to execute a *de-queue* operation to enable its exit from Qn_i resulting in its migration to another node n_k from n_i.

(c) Execution of L: A mobile agent m_j at a node n_i, can execute the user-defined learning algorithm L using the computing environment provided by the static agent a_i resident at the node n_i.

(d) Execution of φ: A mobile agent m_j can request the static agent a_i on the node n_i to execute a set of actions and provide feedback. As φ leads to the goal G (Section 4.2.1) which is dependent on the application under consideration, the executions mentioned here could imply either a set of movements for a navigating mobile robot, a set of rules for mining a large database, a set of input parameters for a control algorithm, etc.

Mobile-to-Mobile Agent Interaction

A mobile agent m_i *interacts-with* ($\otimes$) another mobile agent m_j resulting in sharing of intelligence if both the agents are present within the queue of a node n_k i.e.

$$m_i \otimes m_j \quad \textbf{iff} \quad m_i, m_j \in Qn_k, i \neq j$$

Hence, all the interactions among the mobile agents are always local and take place only *within the queue* of a node where the agents reside after migrating to a node.

4.2.3 Mobile Agent Migration Strategy

The mobile agents migrate within the network W using the ϵ-*Conscientious* migration strategy which is a combination of the *Random* and *Conscientious* migration strategies [221]. In the *Random* migration, a mobile agent chooses one of the neighbours of the current node at random and migrates to that node. In *Conscientious* migration strategy the mobile agent maintains a list of previously visited nodes (say V) and migrates to one that it has not visited so far. If it has visited all, it moves to the node which it has visited least recently. It can be noted that due to this migration strategy, the mobile agents tend to evenly distribute their frequency of visits at each node within the network. The *Conscientious* migration may intuitively seem better from the perspective of a single mobile agent. However, in a multiple mobile agent scenario, this strategy may lead the mobile agents to follow one another along a fixed path within the network.

In ϵ-*Conscientious* migration, a mobile agent employs the *Random* migration with a probability ϵ while it follows the *Conscientious* migration with probability $(1 - \epsilon)$. Thus, with an ϵ-*Conscientious* migration strategy, agents always try to reach out to non-visited or least visited nodes with a higher probability while reducing the drawback of a purely conscientious migration strategy.

4.2.4 Distributed Asynchronous Intelligence-Sharing and Learning

Let S^{m_i} be a set of shareable intelligence units (s_i) that a mobile agent $m_i \in M$ receives from the static agent $a_j \in A$ at a node $n_j \in W$. The set of shareable intelligence S^{m_i} is the learned information gathered as a result of the feedbacks obtained by m_i via the static agent a_j when it executes φ. Hence, the structure of individual elements $s_i \in S^{m_i}$ depends on the learning problem P of the application under consideration. For exposition, the set of shareable intelligence S^{m_i} can

be understood as the different sets of actions gathered by the mobile agents such as the sets of actions in learning to assemble a chair.

Each mobile agent m_i carries a *Bag*, B^{m_i}, (similar to the casebase of an agent as mentioned in [250]) which is a set of s_is obtained from the sets of shareable intelligence S^{m_i}, of other mobile agents as a result of sharing between m_i and the other mobile agent. Hence, B^{m_i}, which forms a part of the mobile agent's payload, can be defined as:

$$B^{m_i} = \{b_k | b_k \in S^{m_j}, i \neq j\}$$

The definitions of sharing and learning within the scope of the proposed framework are enumerated below.

Definition 4.1 *A mobile agent m_i is said to have shared its intelligence with another mobile agent m_j if*

$$m_i \otimes m_j, s_i \in S^{m_i}, s_i \notin S^{m_j}, s_i \Rightarrow m_j, \quad m_i, m_j \in M, i \neq j$$

where '$\Rightarrow$'denotes that s_i is assigned to the mobile agent m_j resulting in $s_i \in B^{m_j}$.

The sharing of information amongst the mobile agents is completely asynchronous in nature. This essentially means that there is no global clock to synchronize the sharing events among the multiple mobile agents at various nodes within the framework. Thus sharing between the several mobile agents populating the network could take place concurrently at different nodes.

Definition 4.2 *As mentioned in Section 4.2.1, φ is a set of actions provided by a mobile agent m_i to a static agent a_j at a node n_j. The execution of φ which is facilitated by a_j at n_j returns a new S^{m_i} which is passed on to m_i by a_j as the feedback. The mobile agent m_i is said to have learned new information if*

$$|S^{m_i}_{new}| > |S^{m_i}_{old}|$$

where $S^{m_i}_{old}$ is the shareable intelligence possessed by the mobile agent m_i before the execution of φ and $S^{m_i}_{new}$ is the same that the mobile agent m_i receives after this execution by the static agent a_j at node n_j.

Definition 4.3 *The problem P is said to have solved if a solution is achieved to get the desired outcome as the goal G by all the mobile agents. Hence a convergence is said to have achieved **iff***

$$\forall i, m_i \rightarrow G, m_i \in M$$

where $\rightarrow$ denotes mobile agent m_i proving solutions to get the desired response as the goal G. Thus, the main objective of the proposed framework is to ensure the convergence of all the mobile agents within the network to achieve the goal G using local sharing and consequent learning.

4.2.5 Inherent Mechanisms within the Framework

The proposed framework provides a distributed model for sharing and learning amongst a set of location-unaware nodes in a network. This framework uses asynchronous local sharing as a basis of information exchange by providing mobility to the learned information. Figure 4.2 depicts the cycle of learning that a mobile agent goes through within the proposed framework.

Each mobile agent (m_i) starts its operations initially from an *Executor* state. In this state, the mobile agent m_i residing in the queue, Q_{n_j}, of node n_j interacts with the static agent a_j also resident at the node n_j. Each mobile agent m_i is conferred with an *Execution Potential* (ξ_{m_i}) which

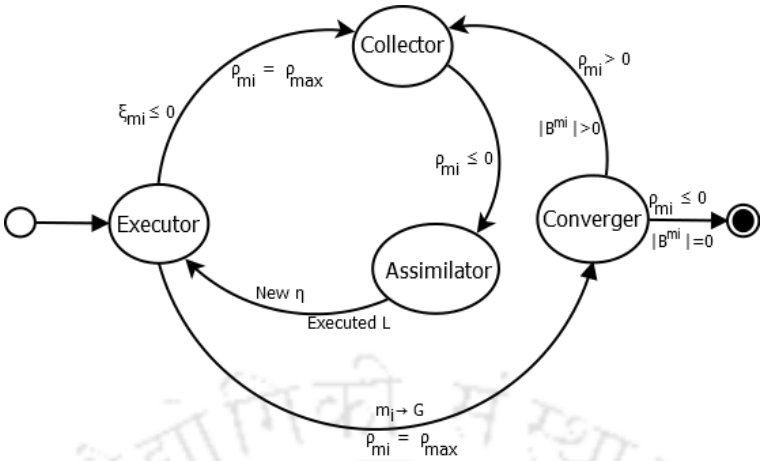


Figure 4.2: The learning cycle of a mobile agent in the proposed framework

is consumed gradually (discussed later) with every execution that the static agent a_j performs based on a request from m_i at the node n_j. The *Execution Potential* (ξ_{m_i}) restricts the mobile agent to reside at a node indefinitely. In the *Executor* state, the mobile agent m_i provides a sequence of actions derived from φ to the static agent a_j as an attempt to achieve the goal G at the node n_j. This reduces the value of ξ_{m_i} based on the length of the sequence. The static agent a_j executes these sequence of actions at the node n_j. As a result of this execution, the mobile agent m_i receives feedbacks on the derived sequence of actions from the static agent a_j. This constitutes the part of the shareable-intelligence S^{m_i} gained by the mobile agent m_i at node n_j. These feedbacks also replenish ξ_{m_i} based on criteria discussed later. A mobile agent continues to remain in the *Executor* state until its ξ_{m_i} is exhausted. Once, $\xi_{m_i} \leq 0$, the mobile agent m_i assimilates the shareable-intelligence S^{m_i} it received and transits to the *Collector* state. In this state, the mobile agent traverses the network W, to share its shareable-intelligence with other mobile agents as well as to get shareable-intelligence from them as and when they co-exist within the same queue at a node. The mobile agent communicates locally with other mobile agents in the *Collector* state to share information. As a result of these local communications within a queue, the mobile agent m_i may receive a new set of information from the shareable-intelligence of other mobile agents. This new set of information is deposited into the *Bag*, B^{m_i} which is unique to the agent.

Every mobile agent is also empowered with a quantity termed as *Migration Resource* (ρ_{m_i}) *a priori* which is also carried as its payload. The parameter ρ_{m_i} governs and regulates the duration for which the mobile agent migrates around in the network W, so as to meet and share information, in the *Collector* state. A mobile agent m_i enters the Collector state with $\rho_{m_i} = \rho_{max}$, ρ_{max} being the maximum possible value of ρ_{m_i} as defined by the user. The mobile agent m_i continues to traverse the network W, till its *Migration Resource* (ρ_{m_i}) is exhausted i.e. $\rho_{m_i} \leq 0$. This resource provides the impetus to the mobile agent to migrate within the network in search of other mobile agents having better information. The dynamics governing ρ_{m_i} have been discussed later. The structure of a mobile agent with all its components (payloads) can be seen in Figure 5.1.

The mobile agent m_i transits to the *Assimilator* state from the *Collector* state as and when ρ_{m_i} degrades to a value less than or equal to zero. This is the state where the learning takes place. In this state, the mobile agent m_i uses the learning algorithm (L) which it carries as payload. It combines the information in S^{m_i} and B^{m_i} as the input to L and churns out a new execution plan (i.e. a new sequence of actions from φ) to reach the goal G at a node n_j. The generation of the new execution plan triggers the mobile agent m_i to transit to the *Executor* state.

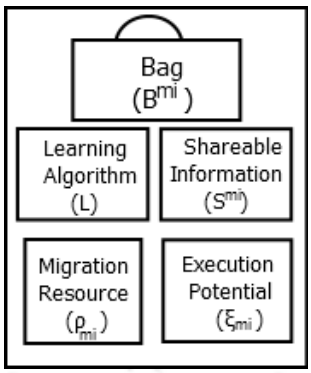


Figure 4.3: Contents within the payload of a mobile agent in the proposed framework.

The static agent a_j in turn executes this new plan within node n_j and provides the related feedback to the mobile agent m_i. This makes the mobile agent m_i transit back to the *Collector* state (unless of course the goal G is achieved) and the cycle in Figure 4.2 continues. The value of ρ_{m_i} is again changed to ρ_{max} before it enters this cycle.

If the static agent reports that the goal G has been achieved, the mobile agent then transits to the *Converger* state with $\rho_{m_i} = \rho_{max}$. Mobile agents in this state tend to verify whether the converged goal G is an optimum or not. A mobile agent m_i in the *Converger* state migrates in the network W to find other mobile agents having a better solution than the one it has found. If the mobile agent m_i finds another mobile agent m_j having a better solution (as per the criteria defined by the user) then m_i receives the shareable-intelligence from m_j into its *Bag* B^{m_i}. This makes the mobile agent m_i to transit back to the *Collector* state and once again join the cycle. However, if ρ_{m_i} becomes less than or equal to 0 for the mobile agent m_i in the *Converger* state and B^{m_i} is still empty, it triggers the mobile agent m_i to exit the learning cycle. Hence a mobile agent m_i exits the learning cycle **iff**

$$(m_i \rightarrow G) \wedge (\rho_{m_i} \leq 0) \wedge (|B^{m_i}| = 0)$$

i.e. mobile agent m_i is able to provide the desired outcome as prescribed by the goal G, it has exhausted its migration resource ρ_{m_i} and the Bag B^{m_i} has no new information.

The overall process halts when all the mobile agents exit the learning cycle through the *Converger* state. The Algorithm 1 depicts all the functions of a mobile agent in the proposed framework as described above.

4.2.6 Dynamics within the Framework

As mentioned, both ρ_{m_i} and ξ_{m_i} act as fuel for migration and on-node execution on part of the mobile agents, respectively. The dynamics that regulates the values ρ_{m_i} and ξ_{m_i} are discussed below.

Dynamics of ρ_{m_i}

It has been assumed that each $b_i \in B^{m_i}$ gained in the learning exercise by the mobile agent m_i in the *Collector* state, to achieve the goal G, has a value or weight associated to it. This weight is synonymous to the profit gained or loss incurred, as the case may be, in using this piece of intelligence to advance towards the goal G. Let $\phi_w(.)$ be the function which returns this weight for

Algorithm 1: Algorithm embedded within Mobile Agents

Input : φ, L, G, P, ρ_{max}
Output : Convergence path to G
Initialization: $B^{m_i} = \{\}$, $S^{m_i} = \{\}$

while *Not converged to* G **do**
  enter_node(n_j); `// Executor State //`
  while $\xi_{m_i} > 0$ **do**
    $Exec\_Plan$ = action_sequence(φ);
    initialize(ξ_{m_i}, $Exec\_Plan$);
    to_StaticAgent(a_j, $Exec\_Plan$) ; `// Mobile to Static Agent Interaction`
    receive_feedback(); retrieve_intelligence(S^{m_i});update_potential(ξ_{m_i});
  end
  if G *is not reached* **then**
    $\rho_{m_i} = \rho_{max}$; `// Collector State //`
    while $\rho_{m_i} > 0$ **do**
      migrate_next_node();
      en_queue(Q_{n_i});
      ρ_{m_i} = migration_penalty(ρ_{m_i}, B^{m_i});
      if *information request received* **then**
        share_intelligence(S^{m_i}); `// Mobile to Mobile Agent Interaction`
      end
      if *end_of_queue* **then**
        find_shareable_agent();
        if *agent is available* **then**
          $W_{old} = \Pi_{m_i}(B^{m_i})$;
          b_i = get_shareable_intelligence(); `// Mobile to Mobile Agent Interaction`
          $B^{m_i} = B^{m_i} \cup b_i$; $W_{new} = \Pi_{m_i}(B^{m_i})$;
          ρ_{m_i} = migration_reward(ρ_{m_i}, W_{old}, W_{new});
        end
      end
    end
  end
  else if G *is reached* **then**
    $\rho_{m_i} = \rho_{max}$; `// Converger State //`
    while $\rho_{m_i} > 0$ **do**
      migrate_next_node(); en_queue(Q_{n_i});
      ρ_{m_i} = migration_penalty(ρ_{m_i}, B^{m_i});
      validate_convergence(G);
      if G *is non-optimal* **then**
        Jump to *Collector* State ;
      end
    end
  end
  if $G \wedge (\rho_{m_i} \leq 0) \wedge (|B^{m_i}| = 0)$ **then**
    exit() ; `// Convergence`
  end
  `// Assimilator State //`
  run_learning_algorithm(L, B^{m_i}, S^{m_i});
  get_new_sequence(φ);
end

each $b_i \in B^{m_i}$. Hence, the net weight (Π_{m_i}) of the *Bag*, B^{m_i}, is calculated as:

$$\Pi_{m_i} = \sum_i \phi_w(b_i), \forall b_i \in B^{m_i} \tag{4.1}$$

The weight function $\phi_w(.)$ depends on the underlying application and can be designed based on a knowledge-sharing model as proposed in [251].

A mobile agent always enters the *Collector* state with $\rho_{m_i} = \rho_{max}$, where ρ_{max} is the maximum possible value of ρ_{m_i} conferred on it *a priori*. A migration penalty is incurred on ρ_{m_i} whenever a mobile agent m_i in the *Collector* state moves to a new node.

The value of ρ_{m_i} at the new node is computed as:

$$\rho_{m_i}(x_{n+1}) = \begin{cases} \rho_{m_i}(x_n)e^{-\Pi_{m_i}} & \text{, if } \Pi_{m_i} > 0 \\ \rho_{m_i}(x_n)(1 - \frac{1}{\rho_{max}}) & \text{, otherwise} \end{cases} \tag{4.2}$$

where x_n denotes the n^{th} instance.

As can be observed in the above equation, the value of ρ_{m_i} degrades exponentially with increase in the weight of the *Bag* B^{m_i}. The exponential reduction in ρ_{m_i} based on the weight of the bag allows for a quick examination of the information gathered so far while if there is no content within the Bag, B^{m_i}, the agents keep on looking for information within the network by expending a proportion of their current ρ_{m_i}. On the contrary, if the *Bag* B^{m_i} is empty, the mobile agent's payload is lighter and ρ_{m_i} is high, forcing it to explore for fresh information across the network W by migrating to other nodes in search of other mobile agents that can provide the same.

While the above equation tends to reduce ρ_{m_i} of an agent due to migrations, it is also *recharged* whenever a mobile agent in the *Collector* state receives information from another mobile agent as a result of local sharing. This also means that whenever there is an increase in weight Π_{m_i} of the *Bag* within a mobile agent m_i, the value of ρ_{m_i} increases empowering it to travel further into the network W in spite of its sluggishness caused by the heavy *Bag* B^{m_i}. The value of ρ_{m_i} when a mobile agent m_i receives and accumulates new information into its *Bag* B^{m_i} is calculated as:

$$\rho_{m_i}(x_{n+1}) = \rho_{m_i}(x_n) + c\frac{\lambda}{\Pi_{m_i}^{before\_sharing}}\rho_{max} \tag{4.3}$$

where

$$\lambda = \Pi_{m_i}^{after\_sharing} - \Pi_{m_i}^{before\_sharing} \tag{4.4}$$

c is a constant and $c > 0$. As can be observed, the increase in the value of ρ_{m_i} is proportional to the percentage increase in the weight of the bag and scaled by the maximum value of ρ_{m_i} i.e. ρ_{max}. The constant c acts as a tuning parameter to control the value of ρ_{m_i} which in turn controls the movement of the agents.

Dynamics of ξ_{m_i}

The *Execution Potential* (ξ_{m_i}) of a mobile agent decreases linearly with the execution of every action within the action-sequence when the mobile agent m_i is in the *Executor* state. As mentioned earlier, the action-sequence is derived from φ which is performed by the static agent.

$$\xi_{m_i}(x_{n+1}) = \xi_{m_i}(x_n) - k_d, \;\; k_d > 0 \tag{4.5}$$

However, when the mobile agent receives a positive feedback (defined by the user for the specific problem whose solutions are being learnt using the algorithm L) from the static agent as a result of executing an action, the *Execution Potential* (ξ_{m_i}) is topped up as:

$$\xi_{m_i}(x_{n+1}) = \xi_{m_i}(x_n) + k_r,\ k_r \geq 0 \tag{4.6}$$

k_d and k_r are constants whose values must be ascertained based on the problem being solved by the agents.

The increase in ξ_{m_i} allows for further exploration into the search space of the problem that has not been achieved as a result of sharing with other mobile agents. It thus aids the generation of new information and subsequent enhancement of the set S^{m_i}.

4.3 Implementation

Since the work reported herein exploits the simultaneous or concurrent executions of multiple agents running and sharing in parallel at different locations (nodes) within a network, experiments if conducted on an inherently sequential simulator would grossly undermine the strength and ability of the proposed framework. Hence, the whole framework was implemented using the *Typhon* [132] platform over a Local Area Network (LAN). *Typhon* instantiations were used to realize various overlay networks of different sizes varying from 10 nodes to 50 nodes over the LAN. Further, to evaluate the robustness of the proposed framework, experiments were conducted using a dynamic network of 50 nodes emulating a mobile computing environment. As the experimental test-bed was completely implemented over a LAN, the results gathered also involve the real-time states (such as processing speed, network conditions, etc.) of different machines used in the experiments. One separate dedicated computer served as a log server to record events such as sharing, executions, etc., at the various nodes. The events were time-stamped using the local time at the log server as and when the relevant pieces of information were received from the individual nodes in the network.

4.3.1 Terms used in the Implementation

In this section, a glossary of various terms used in the implementation has been provided for better clarity. The meaning of terms used are as follows:

- Virtual robot: This is a simulated robot which has been tasked to learn a sequence of actions to reach a specified goal.
- *Maze-World*: This is an $n \times n$ grid structure with each cell of the grid having a set of sensor vectors.
- Sensor Vector (SV): This is a set of sensor values perceived by the virtual robot in a cell within the *Maze-World*.
- Static agents: These agents have the responsibility of executing operations on the virtual robots and to interact with the mobile agents.
- Mobile agents: They act as the carrier of learned information from one virtual robot to another.
- Node: The node comprises the *Typhon* Agent Platform, Queue for mobile agents, *Maze-World* and the virtual robot managed by a static agent.

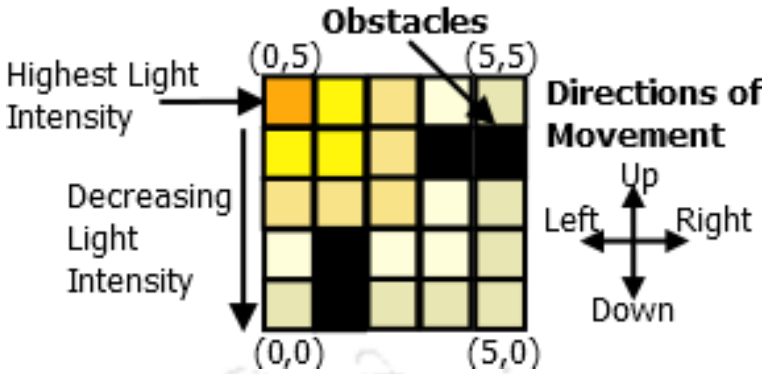


Figure 4.4: The schematic of a 5×5 *Maze-World*. The top left corner in the *Maze-World* is the location of highest light intensity and formed the Destination location. The difference in colour variation of the cells show the change in light intensity as one moves away from the destination. The black cells depict obstacles. The four directions of movement in the *Maze-World* are shown separately.

- Network: This is the inter-connection of nodes which facilitates migration of mobile agents from one node to another.

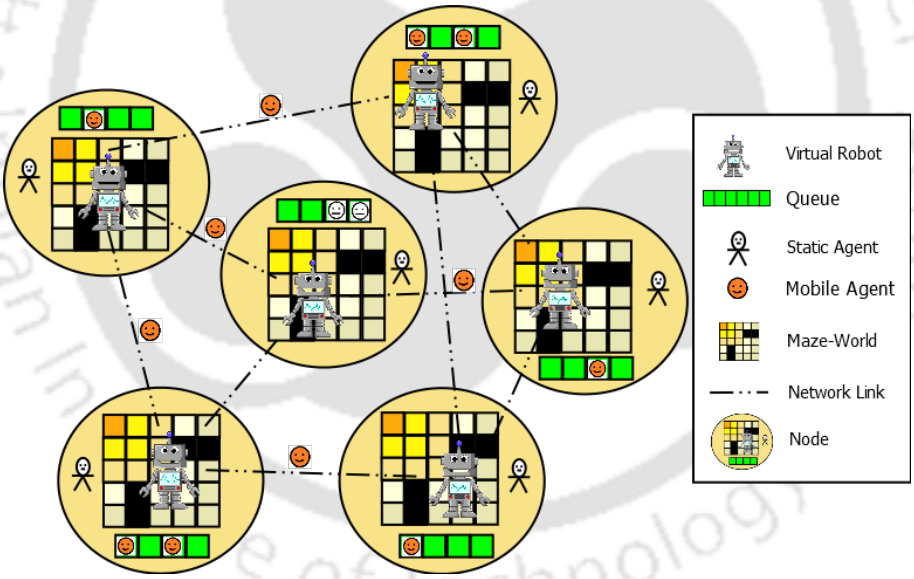


Figure 4.5: An approximate visualization of the virtual robot network along with the *Maze-World*.

4.3.2 The Distributed Learning Problem (P)

For evaluation of the proposed framework, a problem involving a set of virtual robots assigned with the task to learn a path from a fixed source (S_S - lowest light intensity) to a fixed destination (S_D - highest light intensity) in a *Maze-World* was used. The visualization of the *Maze-World* is shown in Figure 4.4 and the corresponding networked-framework is depicted in Figure 4.5. As can be seen, the *Maze-World* is available within each virtual robot locally. The set of virtual robots form the nodes of the distributed network. A static agent was also stationed at each node to manage all the functions of the virtual robot on the *Maze-World* available within the node. As can be observed,

the *Maze-World* and virtual robot replace the problem component P and the node in Figure 4.1 respectively. The network W is formed by the set of virtual robots situated within the nodes with the *Maze-World* and static agent. The mobile agents use this distributed network to disseminate the learned information received from the virtual robots via the static agents.

4.3.3 Specifications of the Virtual Robot

Each virtual robot is equipped with three virtual sensors viz. a *Light* sensor, an *Obstacle* sensor and a *Direction* sensor. The value of the *Light* sensor increases as the virtual robot moves towards the destination and is highest at the destination. Its value is inversely proportional to the distance between the destination and the current location of the virtual robot in the *Maze-World*. Obstacles may also populate the *Maze-World*. A virtual robot can sense these obstacles using its Obstacle sensor which returns only a binary value viz. *true* or *false*. The sensor returns a *true* if there is an obstacle in the direction of the virtual robot's heading or when it encounters the boundary of the *Maze-World*; else it returns a *false*. A virtual robot cannot cross any obstacle or the boundary of the *Maze-World*. Further, the value of the *Light* sensor becomes 0 if the value of *Obstacle* sensor is *true*. The *Direction* sensor returns the direction of movement of the virtual robot in the *Maze-World*. There can be four possible values of the *Direction* sensor viz. *Up* (*positive Y-axis*), *Down* (*negative Y-axis*), *Right* (*positive X-axis*) and *Left* (*negative X-axis*). Hence, a virtual robot is not allowed to move diagonally from one location to another.

The virtual robot can perform five actions within the *Maze-World*. These actions are as follows:

- *Move_Forward*: Execution of this task makes the virtual robot to change its location in the direction of its heading one step at a time. For example, if the current location of the virtual robot is (2, 3) and its current heading is *Up*, then the new location of the virtual robot would be (2, 4) after the execution of the action *Move_forward*.

- *Move_backward*: Execution of this action changes the location of the virtual robot in the diametrically opposite direction of its heading without changing the current heading of the virtual robot. For example, if the current location of the virtual robot is (3, 3) and the current heading direction is *Left* then after executing the *Move_backward* action, the new location of the agent would become (4, 3) and the heading will remain *Left*.

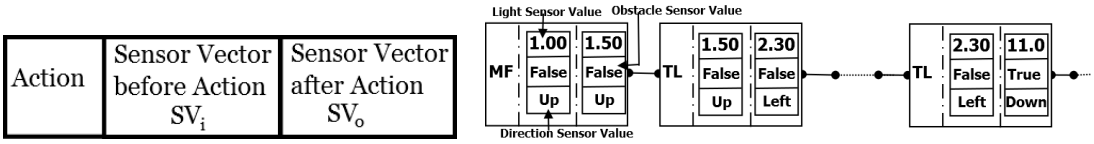
- *Turn_Left*: This action changes the heading of the virtual robot 90° towards the clockwise direction of its current heading. The location of the virtual robot within the *Maze-World* is not affected by this action.

- *Turn_Right*: This action changes the heading of the virtual robot 90° towards the anticlockwise direction of its current heading. The location of the virtual robot within the *Maze-World* is not affected by this action.

- *Turn_Back*: This action changes the heading of the virtual robot to 180° of its current heading without changing the location of the virtual robot in the *Maze-World*.

4.3.4 Embedding the Problem P in the Framework

All the virtual robots have been embedded with the sensor vectors (SVs) of the source (S_S), and the destination (S_D) within the *Maze-World* using which they come to know about the source and the destination. The goal G for the virtual robot is to find a series of SV transitions using the



(a) (b)

Figure 4.6: (a) Structure of a $s_i \in S^{m_i}$ that a mobile agent m_i shares with other mobile agents (b) An example of s_is forming a sequence

available actions to reach the destination viz. S_D from the source S_S. The network of virtual robots uses the proposed framework to realize the mentioned objective.

$\phi_w(.)$ is defined as the difference of the norm between the SV before and after an action is executed i.e.

$$\phi_w(X) = ||SV_o|| - ||SV_i|| \tag{4.7}$$

where, X is an action, SV_i is the sensor vector before the execution of the action and SV_o is the sensor vector after the execution of the action.

The S^{m_i} in this case is the set of SVs with $\phi_w(X) > 0$. The structure of $s_i \in S^{m_i}$ together with an example s_is sequence are shown in Figures 4.6(a) and (b).

4.3.5 The Algorithm (L) used for Learning the Paths

The implementation uses a greedy approach to construct the sequence of actions towards the destination. When the mobile agents are in the *Assimilator* state of the learning cycle (Figure 4.2), they perform an incremental search within their *Bag*s to find the associated transitions of SVs to reach the destination S_D. This search explores various combinations of SVs available within the *Bag*, B^{m_i} and S^{m_i}, of a mobile agent m_i and outputs the sequence of actions closest to the destination based on the SV transitions. Two elements of the Bag, $b_i, b_j \in B^{m_i}$ such that $i \neq j$, can be connected to form a link if $SV_o \in b_i$ (i.e. $SV_o^{b_i}$) is equal to $SV_i \in b_j$ (i.e. $SV_i^{b_j}$) i.e.

$$SV_o^{b_i} = SV_i^{b_j}$$

Thus, SV transitions or b_is result in a tree with either the start vector S_S (the SV of the starting point in the *Maze-World*) or an SV closest to S_S (based on the *cosine* distance among the SVs) at the root of the tree. Other b_is within the *Bag*, B^{m_i}, form the nodes of the tree. The algorithm L uses *Depth-First-Search* (*DFS*) and outputs the branch having the highest length. In case, two branches have the same length, it outputs the branch with highest weight, Π_{m_i} (calculated using Equation 4.1).

After assembling the sequence of actions towards the goal, G, the mobile agent transits to the *Executor* state. As mentioned earlier, in *Executor* state the static agent at a node executes the sequence of actions on the virtual robot thus evaluating the sequence of actions obtained using L and records the fresh SVs. If the execution of all the actions is over and the *Execution Potential*, ξ_{m_i}, is still greater than zero, then the static agent selects an action at random out of the set of actions (φ) and executes them. This allows for the self-discovery on the part of the agents which can be shared with others. If the mobile agent has an empty *Bag*, then the static agent selects actions at random from the set of available actions and executes it until $\xi_{m_i} \leq 0$.

4.3.6 Complexity Analysis within the *Maze-World*

Let us assume that the size of the *Maze-World* is $n \times n$. One may note that there could be more than one SV possible within a cell in the *Maze-World*. For example in the present *Maze-World* there are four different SVs, one for each direction within a cell. If S is the number of SVs possible within each cell of the *Maze-World*, the total possible SVs within the *Maze-World* would be Sn^2. Let $\overline{A}$ be the number of actions that can be performed and l be the length of the sequence of actions to be executed to reach the destination (S_D). If the starting point for all virtual robots is fixed (S_S) within the *Maze-World*, then the total search space would reduce to $O(\overline{A}^l)$.

If the length, l, of the sequence of actions is not known, then for a single agent, the total time complexity to reach to the destination location (S_D) in the worst-case would be:

$$\beta := O\left(\sum_{l=1}^{Sn^2-1} \overline{A}^l\right)$$

This gives us the upper bound on the time complexity of the search space using a single virtual robot positioned at a fixed location within the *Maze-World*.

Further, let us assume that α number of virtual robots at a fixed location within the *Maze-World* are trying to move towards the given destination. In the best case when no redundant executions are performed due to one-to-all sharing amongst virtual robots, the total time complexity of exploring the *Maze-World* by α virtual robots is of the order of β/α plus the overheads incurred in sharing.

Let the time taken for sharing intelligence between two virtual robots be t_r. Apparently, the sharing of information involves two virtual robots (one who provides the information and the other one who receives it) at a time. Hence, the total number of sharing events required to disseminate the intelligence within each virtual robot among the remaining virtual robots would be ${}^{\alpha}C_2$. It may be noted that the virtual robots can share intelligence concurrently. If α is even, then the number of concurrent sharing events possible is $\alpha/2$. Thus the total time required to share the intelligence among the α virtual robots would be $\frac{(2t_r \ {}^{\alpha}C_2)}{\alpha}$.

In case if α is odd, this time complexity would become $\left(\frac{2 \ {}^{\alpha-1}C_2}{\alpha-1} + \alpha - 1\right) t_r$.

Hence, the total time complexity to reach the destination in the best case for α virtual robot (α being even) would be:

$$\gamma := O\left(\frac{\beta + (2t_r \ {}^{\alpha}C_2)}{\alpha}\right)$$

This forms the lower bound on the time complexity of the overall search space. Let θ be the time complexity of solving the problem P using the proposed framework. Since, in the proposed framework $\alpha > 1$ along with localized sharing using the concept of mobility of learned information, intuitively θ will be bounded as: $\gamma < \theta << \beta$

It may be noted that these bounds are not so tight on θ, yet they give us an approximation of the reduction in the search space and time complexity when the proposed sharing framework is used.

4.4 Results

Experiments were carried out on various networks of virtual robots with different populations of mobile agents to learn a sequence of actions (i.e. the goal G in this case) to reach the destination. Each experiment was performed at least 10 times to counter any stochastic influence. The time

required to complete each of the experiments varied between 300 to 3000 seconds. The average of 10 runs has been portrayed in the results. The values of various parameters used for the experiments are:
$\epsilon = 0.2$, $\rho_{max} = 10$, $n \times n$ (Size of maze) $= 50 \times 50$, Start Location $(S_S) = (50, 0)$, Destination Location $(S_D) = (0, 50)$

The mobile agents were placed arbitrarily at different nodes within the network of virtual robots during the start of each of the experiments. All the nodes in the network of virtual robots were connected in the form of a mesh.

For experimentation, *Typhon* [132] based networks with sizes varying from 10 nodes to 50 nodes were created. The densities (D) of the mobile agents within a network, which can be defined as the ratio of number of agents to the number of nodes in the network, were varied from $D = 0.1$ to 0.5 on each of the networks and all the executions were logged. The experiments involved the problem of finding the sequence of actions required to traverse from the starting location S_S to the destination location S_D within the *Maze-World* with no obstacles. To vary the problem setting, The experiments were also conducted on a 50-node network with the *Maze-World* having contiguous obstacles occupying co-ordinate locations (40, 0) to (40, 40).

Two important factors that are crucial to verify the effectiveness of the proposed framework are - the size of the network (i.e. number of nodes in the network) and the number of mobile agents involved. While the former tests the scalability of the framework, the latter can ensure a faster convergence. Hence, as a performance yardstick, the density (D) of mobile agents was varied in networks of different sizes and the average number of executions that the virtual robots took to converge to the goal G was recorded. The number of executions of virtual robots mean the number of times each mobile agent entered the *Executor* state (Section 4.2.5). Thus, the average number of executions is the ratio of the total number of executions (until the convergence of all the mobile agents in the network) to the total number of mobile agents.

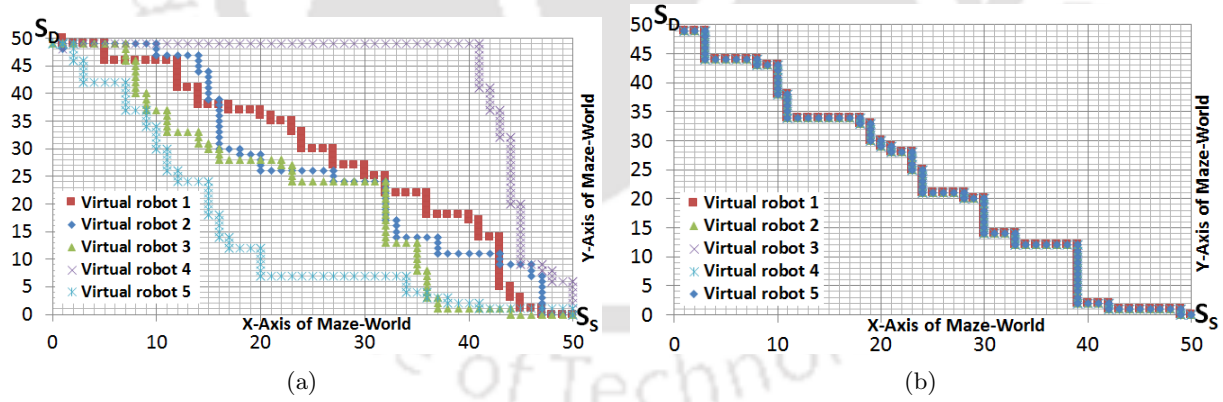


Figure 4.7: Final converged sequences of five agents in the *Maze-World* from the Source at (50, 0) to the Destination at (0, 50) in a 50-node network - (a) Without Sharing (Paths are distinct) (b) With Sharing (Paths are highly overlapped).

4.4.1 Converged Sequences of Actions

The graphs in Figures 4.7(a) and 4.7(b) depict the final converged paths taken by the virtual robots with 5 mobile agents in the 50-node network. Figure 4.7(a) shows the converged paths of five virtual robots without sharing i.e. when the proposed sharing framework was not used. While,

a total of 5 mobile agents ($D = 0.1$) in the 50-node network were used with the proposed framework for the graph in Figure 4.7(b). As can be observed in Figure 4.7(a), when the proposed sharing framework is not used by the virtual robots, all the virtual robots discover different sequences of actions (paths) to the destination S_D for the goal G. However, all the virtual robots converge to the same sequence of actions when they share information with one another. These graphs not only show the impact of sharing but also reveal the expected performance of the proposed framework. The goal G, assigned to the virtual robots, was to find a sequence of actions (as discussed in Section 4.2.5) that facilitate their movement from a source at (50,0) to a destination at (0,50). It can be seen clearly that sharing of information definitely helps these virtual robots achieve their goals in lesser time with fewer number of executions. From the logs, it has been found that the average number of executions was 195 when the virtual robots did not share information whereas it was a mere 92 when sharing was embedded using the proposed framework. It should be noted that all the virtual robots are required to find their path individually. The results indicate that the proposed framework allows mobile agents to search in different directions (possibilities) and the best amongst them is taken up by all the virtual robots to beeline towards the goal. Similar trends were observed in all other cases considered for the experimentation.

Further the converged sequence shown in Figure 4.7(b) is achieved by five different mobile agents when they are completely oblivious of the information about other mobile agents within the network along with any knowledge pertaining to the network itself (i.e. the number of virtual robots). Hence, the proposed framework augments the algorithm L and facilitates the sharing of information amongst the virtual robots in a truly distributed sense and also yields better performance.

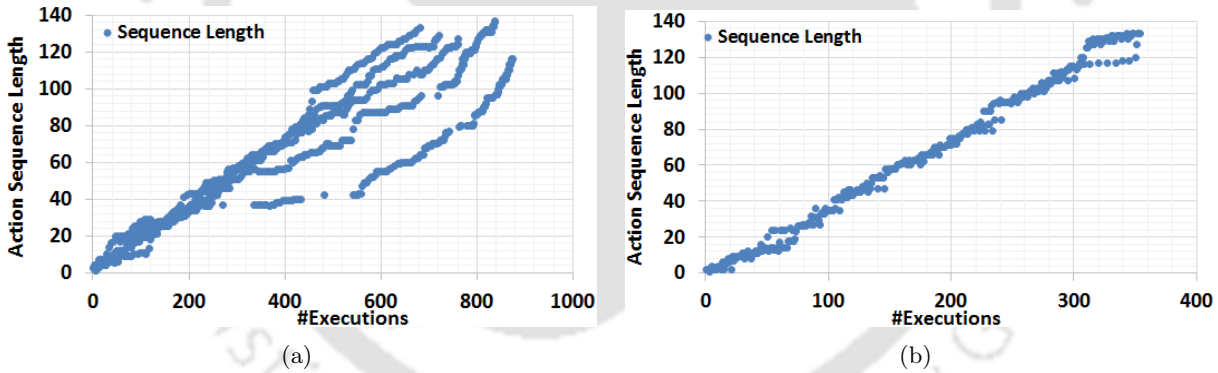


Figure 4.8: Variations in the length of the sequences of actions of the virtual robots with executions in case of five mobile agents in a 50-node network (a) Without Sharing (Mostly dissimilar sequences) (b) With Sharing (Converge on the similar sequences).

4.4.2 Length of the Sequences of Actions

The graphs in Figures 4.8(a) and 4.8(b) depict the variations in the length of the sequence of actions with respect to the executions on part of the virtual robots with five mobile agents whose converged sequences are shown in the graphs in Figure 4.7(a) and 4.7(b) respectively. The differences in the lengths of the sequences discovered by the five virtual robots against their executions can be observed in Figure 4.8(a) when they are not sharing any intelligence amongst each-other. As can be observed in the case when the mobile agents shared intelligence (Figure 4.8(b)), the lengths of the sequences of all the virtual robots are spread evenly and confined along a common line even before convergence when they are in the process of discovering the paths which is the period when

execution increases. This illustrates that with no sharing of intelligence, the virtual robots search egocentrically and explore to find their individual solutions. Whereas the virtual robots are able to effectively constrict their search space towards the goal by sharing their piecewise knowledge using the dynamics of the proposed framework. Each virtual robot thus tries to solve the sub-problems of a common agenda. It can also be observed that the total number of executions performed by all the five mobile agents taken together which is around 350 is less than half of the total number of executions taken together of all the non-sharing virtual robots (around 900). Hence, the results portrayed herein show that the proposed framework makes the virtual robots agree on a common solution and share the best available information amongst each other using the mobility of the learned information. The graphs in Figure 4.8 show one instance of the results obtained. Similar trends were observed when the number of nodes and mobile agents were varied in all other cases considered for experimentation.

4.4.3 Average Number of Executions

Figure 4.9 depicts the average number of executions required to learn a path from source to the destination within the *Maze-World* of size 50×50 with varying number of nodes and mobile agents.

As can be observed from the graph in Figure 4.9, the average number of executions (until all the mobile agents converge to a path to S_D) reduces monotonically with increase in the density of the mobile agents. Further, the trend remains the same as the size of the network (number of nodes in the network) grows. One can also observe that as the density becomes high (50% of the total number of nodes), no significant difference in the average number of executions across different network sizes is observed. It may be noted that in the real-world, the execution of a series of actions is a costly affair both in terms of energy and time. Also, $D = 0.1$ in a 10-node network depicts a scenario when there is no sharing since there is only 1 mobile agent in the network. The average number of executions in this case is 195. While the average number of execution reduces to almost half (110 executions) with $D = 0.2$ in case of 10 nodes. This depicts the huge gain in terms of average number of executions when the mobile agents are sharing information against the case when they are not doing so. Further, the average numbers of executions are higher in case of 50 nodes when there was obstacles within the *Maze-World* against the no-obstacle case. This shows that the complexity of the problem does affect the performance though the trend of executions remains the same. It may also be noted that the reduction in the average number of executions is high when the density of mobile agents varies from $D = 0.1$ to 0.2 in all the cases considered for experimentation. This reduction slows down from 0.2 onwards and almost becomes asymptotic. Moreover, increasing the density above 0.5 is not advisable as the population of mobile agents would clutter the network and entail more communication overheads [128]. Hence, the results clearly show that the proposed framework can effectively bring down the number of executions required to achieve the goal G in a fully distributed and asynchronous manner.

4.4.4 Average Number of Sharing Events per Execution

The graph in Figure 4.10 depicts the average number of times the sharing was performed per execution with varying densities and sizes of the network. The average number of sharing events per execution is calculated as the ratio of the average of the total number of sharing events performed in 10 runs of each experiment to the average of the total number of executions in 10 runs. The cases considered are same as described in the previous section. As can be observed there is an increase in the average number of sharing events per execution with increase in the density of mobile agents in each of the networks of different sizes. Further, as observed in the previous case, the trend remains

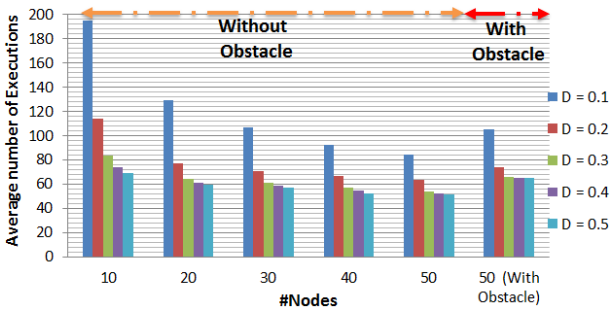


Figure 4.9: Variations in the average number of executions to converge to a path from the source to the destination within a *Maze-World* of size 50×50 by all the mobile agents for different densities. The graph is plotted by taking the average of 10 runs in each case on *Typhon* based networks with no obstacles in the *Maze-World* for the first five sets. The last set is one with obstacles stretching from (40,0) to (40,40) in the *Maze-World*.

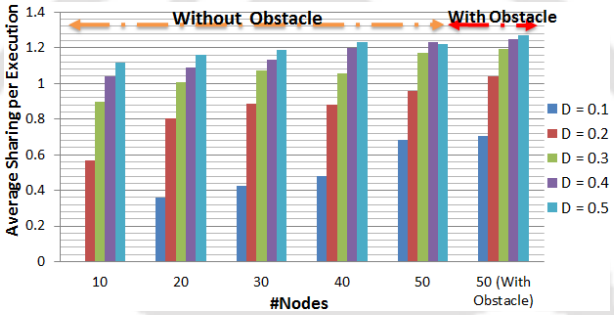
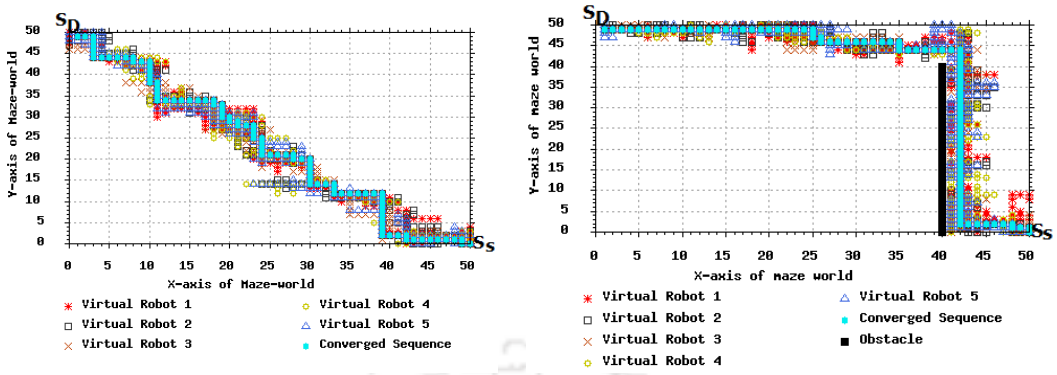


Figure 4.10: Average number of sharing events per execution with different densities. The graph is plotted by taking the average of 10 runs in each case on *Typhon* based networks with no obstacles in the *Maze-World* for the first five sets. The last set is one with obstacles stretching from (40,0) to (40,40) in the *Maze-World*.

the same across different network sizes and problem setting. Apparently, the average number of sharing events per execution is 0 in case of $D = 0.1$ in the network with 10 nodes. The graph also reveals the fact that an initial increase in the density of mobile agents aids to expand the search in multiple directions which increases the amount of sharing. However, a high density of mobile agents within the network causes redundancy in the search space and hence creates less amount of shareable intelligence amongst each other. Further, the graph also shows that a change in the problem setting (*Maze-World* with obstacles) does not alter the trend or the behaviour of the sharing among the mobile agents. It can also be observed that the amount of sharing increases when the problem becomes more complex (*Maze-World* with obstacles) because of the increase in size of the search space.

It may be noted that there is a marginal difference (< 0.05) in the average sharing per execution between $D = 0.4$ and $D = 0.5$ across all cases. These are possibly the best operating densities of the number of mobile agents in the proposed framework to effectively use distributed asynchronous sharing in the current problem setting of the *Maze-World*. Though a lower number of mobile agents could eventually find the solution (taking more number of executions and less sharing), an optimum number of mobile agents in the network could hasten the process while also effectively utilizing the network resources. A mechanism to dynamically vary the density of mobile agents based on the current size of the network as reported in [128] could aid the performance of



(a) (b)

Figure 4.11: Movements of all five virtual robots with sharing from the start of the experiment until convergence - (a) with $D = 0.1$ in a 50-node network with no obstacles in the *Maze-World* and (b) with $D = 0.1$ in a 50-node network with an obstacle stretching from location (40,0) to (40, 40) in the *Maze-World*.

the proposed framework.

4.4.5 Distinct Movements of Virtual Robots within the *Maze-World*

The graphs in Figures 4.11(a) and 4.11(b) depict the movement of all the virtual robots within the *Maze-World* of size 50×50 for $D = 0.1$ in a 50-node network without and with obstacles. The final converged sequences of actions (G) of all the virtual robots are also depicted in the graphs. These two cases of *Maze-World* exhibit different levels of complexities for the virtual robots to reach the destination location (50, 0) from the starting location (0, 50). When there is no obstacle within the *Maze-World*, the obvious way to traverse the *Maze-World* to reach the destination is to move diagonally while in the presence of an obstacle the best choice is to follow the path alongside the obstacle towards the destination. In case of an obstacle, the virtual robots need to exert more energy in terms of the number of executions required to explore the path alongside the obstacle as compared to a diagonal path in the absence of the same. Insertion of multiple obstacles along the path also gave similar results. As can be observed from the graphs, the final converged sequence of actions (G) in both without and with the obstacles, for all the virtual robots is the best possible intersection of the sequences available to reach the destination location at (50,0). Since the learning algorithm discussed in Section 4.3.5 does not ensure the optimality in the movement of virtual robots towards their destination, such an effect evolves as a result of the distributed asynchronous sharing of intelligence available within each mobile agent which in turn circulates the knowledge of the global best among all the virtual robots. Sharing of information on part of the mobile agents seems to bring back the virtual robots that drift away from the optimal path resulting in a drastic reduction in the number of otherwise futile executions. It thus motivates the entire population of virtual robots to pursue a common and possibly more optimal path in an *orderly* and *unified* manner.

4.4.6 Frequency of Sharing

Figure 4.12 shows the change in number of sharing events versus time along with the associated linear trendline for $D = 0.5$ in a 50-node network. It may be noted that the X-axis indicates

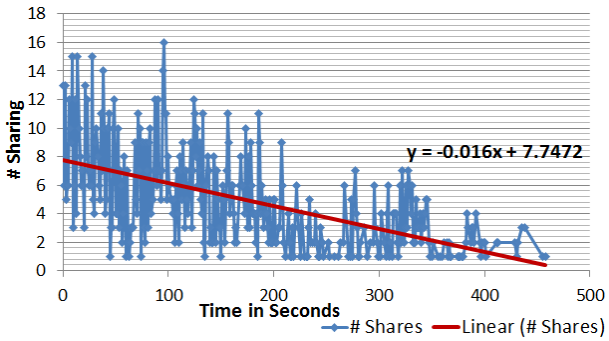
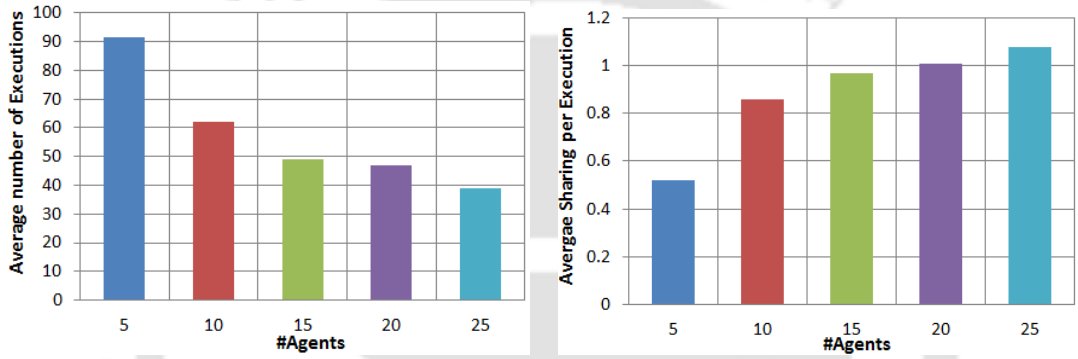


Figure 4.12: Number of sharing events among 25 agents in a 50-node network ($D = 0.5$) against time in seconds.



(a) (b)

Figure 4.13: Performance in the *Typhon* based 50-node dynamic network of virtual robots with varying densities from $D = 0.1$ (5 agents) to 0.5 (25 agents) (a) Average number of executions to converge to a path from the source to the destination within a *Maze-World* of size 50×50 by all the mobile agents (b) Average number of sharing events per execution.

the time-stamps recorded by the log server (as mentioned in Section 4.3). As can be seen, the linear trendline drawn based on the frequency of sharing events has a negative slope. Since the mobile agents try different combinations of actions on to the virtual robots initially they tend to discover diverse piecewise solutions causing the sharing events to be high. The number of sharing events decreases gradually as the virtual robots move towards the goal. This is because the initial high level of sharing causes the mobile agents to converge closer to a common path causing lesser diversity of information within their respective bags resulting in a gradual reduction in sharing. A similar trend was observed in all other cases of varying nodes and mobile agents used in the experimentation.

Sharing thus seems to constrain the mobile agents to search within a common and narrower search space. This could lead the virtual robots to follow a non-optimal path (local optima). One alternative to circumvent this could be to embed a different learning algorithm within a few mobile agents so as to force them explore the search space in a different manner ensuring diversity of the contents in the various *Bag*s for a longer time. This increase in diversity of the piecewise solutions carried within the individual *Bag*s of the mobile agents will consequently increase sharing and aid in an exit path from possible local optima.

4.4.7 Performance in a Dynamic Network

Figure 4.13 depicts the performance of the proposed distributed intelligence-sharing and learning framework in a 50-node dynamic network. The dynamic network was created using a variant of the Erdős-Rényi $G(n, p)$ model [252]. Initially a mesh network of 50 nodes was established over a LAN. Each node within the network was provisioned to break their connections with any of their current neighbours with a probability of 0.3. A node was allowed to make a new connection with any other node in the network with a probability of 0.5. Each node exercises the event of altering their connections after a time interval randomly chosen between 10 and 20 seconds in real time. Thus, the virtual robots within the dynamic network mimicked the movement or mobility of actual mobile robots just as in a mobile ad-hoc network by breaking and making new connections with other robots as they move physically. From the logs, the minimum degree of a node (d) was found to be 0 while the maximum degree was found to be 27. When a virtual robot becomes isolated ($d = 0$), then all the mobile agents within that node become *dormant* and wait till the virtual robot reconnects. Hence, the total number of nodes in the connected network keeps on varying with time.

As can be observed from Figure 4.13(a), the average number of executions (until all the mobile agents converge to G) reduces with increase in the density of the mobile agents within the 50-node dynamic network. This follows the same trend as observed in case of the static networks discussed earlier. Further as shown in Figure 4.13(b), the average number of sharing events per execution increases with increase in the number of mobile agents within the network. The trend of the graph in this case also remains similar to the previously cited cases of static networks. As can be seen from the graph, there is an initial boost in the average number of sharing events per execution in case of 5 mobile agents to 10 mobile agents while the same slows down as one moves from 10 mobile agents to 25 mobile agents. As mentioned earlier, this is due to the fact that an initial increase in the density of agents aids to expand the search in multiple directions resulting in an increase in the overall sharing. As the density of mobile agents increases, redundancy in information discovered by the mobile agents increases thereby lowering the quantum of shareable intelligence.

It may thus be observed that the proposed framework is suited to networks wherein the *nodes are mobile*. This emphasizes the robustness and flexibility of the proposed framework for intelligence-sharing and learning in distributed environments. The results substantiate the viability of using the proposed framework in distributed mobile computing environments.

4.5 Discussions

The proposed framework opens numerous avenues for testing and verifying a gamut of learning algorithms over distributed, decentralized and asynchronous environments. Convergence is hastened due to faster access to newly weaned information gained from sharing on part of the mobile entities in the network. The multi-agent paradigm coupled with this mobility of learned information, facilitates collaborative learning among a set of location-unaware entities or nodes in a large distributed, decentralized and asynchronous system. The intrinsic flexibility of agent based technology used by the framework could also allow learning of multiple goals by the various entities in a network. Thus, in an N-node network, I nodes could be trying to find a solution to a problem P_1, J nodes could be doing the same for another problem P_2, while the rest $(N - I - J)$ try to solve P_3. These problems may need to be solved concurrently or could even be made sequential using techniques cited in [135, 136]. In such a case, the N-node network would be hosting heterogeneous sets of mobile agents which use the proposed framework to solve their respective problems. Heterogeneity in this case refers to the difference in the parameters and the learning algorithms used by the three sets of mobile agents attempting to solve the problems P_1, P_2 and P_3 respectively as mentioned above. Given a heterogeneous set of algorithms say L_1, L_2 and L_3 to be used to solve a particular

problem P, such a set up of the framework could also be used to evolve the best algorithm. In this case, three sets of mobile agents could be primed to solve the problem P using an algorithm respectively within the proposed framework. Consequent to this, each set of mobile agents would evolve three solutions S_1, S_2 and S_3 based on the algorithms L_1, L_2 and L_3 respectively. In order to find the efficacy of the solutions S_i and find the best among them, an Idiotypic network [131] based mechanism such as the one proposed in the previous chapter (Chapter 3) could be used. The mechanism proposed in the previous chapter could allow the N-node network to automatically evolve the best solution out of S_1, S_2 and S_3.

In addition, if it so happens that an algorithm L_1 performs optimally during the initial phase of learning but unfortunately deteriorates in its performance at later stages while another algorithm L_2 behaves just the reverse way then running them together within the framework could be meaningful. While the set of mobile agents using L_1 will clearly dominate in performance initially, the other set of mobile agents using L_2 will use this learned information and enhance the convergence at a rapid pace. The heterogeneous set of mobile agents using different learning algorithms can thus co-operate and lead to a better solution using this framework. Since the performance of different algorithms could vary depending on the problem at hand such a hybridizing of algorithms could pave ways to newer and more efficient mechanisms.

The framework could also be enhanced by supplementing it with cloning of the best performing sets of mobile agents as described in [128]. Accordingly, the cloning of mobile agents would not only enhance concurrent processing by increasing the population of the best performing agents, but also restrict the lesser performing ones from consuming precious system resources such as bandwidth and computation times. Modifying and emulating variants of population based algorithms such as Evolutionary Computing and Genetic Algorithms (GA) including Island models [253] can also be performed using this framework.

As discussed earlier, the design methodology of the proposed framework is inspired by the interactions in social insect colonies and draws concepts from the domain of Evolutionary Computing and Genetic Algorithms (GA). Modifying and emulating variants of such population based algorithms can also be performed using this framework. If each mobile agent carrying the learned information is considered as a candidate solution, then this framework can be viewed as a real-time formulation of a distributed GA. The support for mobility of candidate solutions in this framework also allows the real implementation of Island models of parallel GA [253]. Such models have been proposed as a distributed implementation but have hardly been used, possibly due to the non-existence of a convincing platform to realize them. Looking for the analogies between the proposed framework and GA, it may be observed that the Migration Resource, ρ_{mi}, mildly mimics the crossover operator used in a GA. It also makes candidate solutions to be nomadic and forces their movement from one island population to another. Further, the Execution Potential, ξ_{m_i}, partially mimics the mutation operator used in a GA by forcing the system to try out new solutions. Given a set of actions, it makes the system try out new permutations and combinations of these actions. The feedback provided by the static agent at the nodes participating in the learning exercise can be attributed to the fitness of the solutions in a GA. Similar analogies can also be derived for other population based bio-inspired algorithms such as Particle Swarm optimization (PSO) [254]. The proposed framework can thus be effectively utilized as a potential tool for intelligence-sharing and learning in large distributed systems in a variety of scenarios.

4.6 Chapter Summary

Distributed intelligence-sharing and learning in multi-agent systems, both open up a wide variety of applications, that range from sensor beds to smart cities [24]. Mobile agent technology can play

a crucial role in optimizing the use of local computing resources and distributed decision making in highly complex and scalable systems. This chapter attempts to highlight the advantages of distributed intelligence-sharing and learning using a set of mobile agents. The *emulation* experiments presented in this chapter provide valid proof-of-concept for the same. The model of intelligence-sharing and learning proposed herein could be used as a framework to realize distributed and continuously evolving systems which in turn could exhibit emergent behaviours. A variety of user-defined learning algorithms could also be used concurrently by different sets of agents within this framework.

It may be noted that the proposed framework makes use of information available locally (within a node) to achieve a global objective. This is akin to the functioning of densely populated insect colonies such as ants, honeybees, etc. [244]. The asynchronous sharing and consequent learning observed in the proposed framework could provide insights to the manner in which these swarms interact and converge towards a goal. It further opens up avenues to explore applications in the domain of asynchronous robotic swarms wherein every execution consumes precious energy stored within their batteries. Such a system could thus enhance the capabilities of the robotic swarm several manifolds. It is possible to conjecture each robot in a swarm as a mobile agent and then embed the whole mechanism of local sharing and learning to constitute an intelligent mobile ad hoc network of robots. The framework described herein can also be used in many Internet of Things (IoT) and Cyber-Physical Systems (CPS) based applications. Some of the examples of such applications can be - energy management in smart buildings [255] wherein a set of mobile agents could be used to learn a schedule of energy consumption by various networked devices, intelligent water distribution systems [256] wherein such agents could share the information of a distributed network of various water reservoirs and optimize water distribution. Other applications could be the learning of the occupancy patterns of large buildings [257] for regulating HVAC systems and distributed intelligent traffic monitoring and management systems [206, 207] wherein the agents can share the information of heavy traffic routes and learn traffic regulation rules to cater to different times of the day thus evolving optimized route schedules. In industry automation such agents can share the information of workloads on different remotely located machines and learn an optimized schedule for job allocations on-the-fly while in intelligent warehouse management systems these agents can facilitate sharing of information of the items placed in various smart racks and shelves to come up with the better and optimal strategies to improve logistics and placement of goods, etc. Use of a set of mobile agents, empowered with the sharing and learning mechanisms, in all these scenarios is bound to instil and enhance intelligence in such networked environments.

Although this chapter alleviates the problem of learning a sequence of actions using a set of mobile agents in order to achieve an objective, synchronized execution of such a sequence of actions (tasks) in parallel by a set of asynchronous mobile agents still remains non-trivial. In addition, it may also be the case that the sequence of tasks needs to follow a particular order of execution over a large distributed network of robots. In such cases, multiple mobile agents, each of which is carrying the sequence of tasks, need to coordinate and synchronize their actions so as to avoid clashes and redundant executions over the network. A mechanism to ensure the recovery of the system in the event of a failure of a task also becomes mandatory. The next chapter presents a technique for such a synchronized execution of a sequence of tasks by a set of nodes in a distributed network using multiple mobile agents. The technique uses stigmergy based communication amongst the mobile agents for conveying task-related information to the various mobile agents within the network.

"Everything that depends on the action of nature is by nature as good as it can be."

Aristotle (384 BC – 322 BC)
Greek philosopher and scientist

5

Synchronizing the Execution of a Sequence of Tasks

Execution of tasks among networked robots is a crucial aspect of their utility and productivity in any environment. Many a time these tasks are dependent on one-another in order to achieve a desired configuration. Synchronizing the activities of each of the networked robots to make them execute a set of interdependent tasks becomes challenging since one has to take into account the complexities involved in communication, distributed sensing, control, etc. Circumventing these challenges using traditional approaches of master-slave based mechanisms [258] is not robust and does not scale well. Non-conventional approaches based on social insects, have considerable attention from the researchers to address the challenges involved in coordinating the activities of various autonomous and networked entities woven within a network fabric. Examples of highly distributed synchronized activities can be witnessed in the natural swarms such as ants hills, bee-hives and wasp-nests. The co-ordination of various activities within a colony of such insects is performed stigmergically through indirect communication.

The work presented in this chapter is inspired by the self-organized synchronous behaviours exhibited by such insect colonies. A novel technique for near synchronous execution of a sequence of tasks by a set of networked robots is described. The technique makes use of mobile agents [109] that emulate the insects and exploit the concept of indirect or stigmergy based communication so that a sequence of tasks is executed within each of networked robots in a synchronous manner. The mobile agents operate autonomously without any direct mutual interaction which tends to save communication overheads. In the proposed technique, the mobile agents remain oblivious of their total number within the network as also the total number of robotic nodes. The technique also inherently exhibits self-healing at times when a task fails for some reason.

5.1 Motivation

Simple behaviours and interactions between the individuals in a swarm of living organisms have been known to lead to the build-up of many a complex structure in nature [259]. Termite mounds, wasp nests, ant hills, bee hives, etc. form some of the most pertinent examples of such structures. These insects are limited in their capabilities and perceptions but are yet able to create such architectural monuments with finely devised techniques that can withstand high temperatures, moisture and rain [260, 261, 262], even in the absence of any global control system within. Researchers have long been trying to unravel the secrets behind how these insects manage to construct such structures in a

distributed yet decentralized manner. Various theories and models have been proposed to mimic and comprehend the techniques used by these social insects [263, 264, 265, 266, 267, 268, 269, 270, 271]. Apparently, there must be some underlying asynchronous and co-ordinated sense embedded within each insect that aids it to locally decide as to what action needs to be performed. The co-ordination of activities among these insects occurs due to their local interactions generally with those within their vicinity and also changes in the environment [272]. Interaction with the environment plays a vital role in the transfer of information throughout the insect colonies. Changes made in the environment allow for an indirect dissemination of information. Responses to such stigmergic stimuli are well organized both spatially and temporally in several species of insects. Stigmergy, thus seems to drive the various processes in such insect colonies [259].

Deneubourg et al. [263] have proposed that termites possibly build their mounds by sensing the pheromone field gradient which is mixed along with the debris before deposition. Tofts and Franks [273] have proposed a simple "Foraging for work" model wherein they generalized the understanding of task allocation among insect colonies such as ants, honey bees and naked mole-rats. According to them the tasks are spatially organized in insect colonies and bare a correlation with the age of insects. Their model was later criticized for unrealistic assumptions in the simulations that they performed [274]. Inspired by the manner in which wasps build colonies, Theraulaz and Bonabeau [266] have proposed a model that can exhibit their distributed nature of building structures. They have decomposed the entire task of building a structure into a finite number of steps. Further, it is assumed that the local configurations that are generated in a given state differ from those of an earlier or later state. This prevents any possible disorganization in the chain of events or tasks. Each individual in the colony needs to know all the rules to be followed to build the complete structure. The stigmergic stimuli cause the turning OFF or turning ON of these embedded rules within the individuals. Mason [275] discusses how these rules can be driven by pheromone gradients for continuous cellular automata. Werfel et al. [276] uses a simulation to show how various complex structures can be created using robotic swarms. They assumed the environment to be caches of blocks and beacons which send out signals while the robots perceive them. This perception triggers specific rules as to where they should place the blocks. Beshers and Fewell [269] provide a vivid description of the various models proposed to elucidate the task selection and their performance in insects colonies. They concluded that though the exact interpretation of colony dynamics is yet to be unraveled, the proposed models do point out that the behaviours of the colony are affected by both internal and external factors. The most crucial aspect to be looked into while devising a model that can mimic the stigmergic behaviour of the insects is the manner in which an entity comprising the swarm senses its environment. The strategies proposed so far employ signals or pheromones that need to be sensed from the environment so that they can trigger a set of rules within the entity. Implementing such mechanisms in *real* systems still remains a challenge.

Many researchers in the domain of robotics have tried to mimic the phenomena prevalent in insect colonies particularly those of division of labour and stigmergic communication [277, 278, 279, 280, 281, 282, 283]. The distributed nature of task performance in social insects having simple behaviours along with other interesting features, such as flexibility, robustness, decentralization and self-organization are essential in realizing a dynamic distributed application with built-in fault tolerance [81]. Krieger et al. [277] have implemented dynamic task allocation based on the fixed response threshold model [284]. They assigned a team of robots to maintain a stock of energy at a nest. Each robot is assigned a fixed threshold for foraging. Whenever the nest energy level falls below the individual activation-threshold of a robot that robot chooses the task of foraging. Thus this model dynamically allocates the task of foraging to individual robots. Jones and Mataric [281] have used simple rule based approach to vary the number of foragers in a set of dynamically evolving concurrent tasks. Groß et al. [280] have used a finite-state machine based approach to solve

complex foraging tasks using a colony of autonomous robots. They described a self-organizing and dynamically varying hierarchy among the groups and teams of robots. Inspired by the Deneubourg's learning model [285] for foraging in ants, Labella et al. [286] proposed an algorithm for division of labour in an object retrieval task where the ratio of the number of foragers to resters is altered adaptively. Liu et al. [287, 288] use a similar mechanism for the foraging activity using three different types of cues viz. internal, environmental and social.

Though much of the work has been concentrated on the division of labour in insect colonies, a few researchers have tried to model the synchronization of activities within [289]. This phenomena of synchronized colony behaviour has been found in various insects such as ants [290], fireflies [291], social spiders [292] and fiddler crabs [293]. Bonabeau et al. [289] have used simulation to portray a simple model for recruitment based foraging in ants wherein they show that the synchronization of activities increases the foraging efficiency of the colony. However they have not described how the synchronization can be achieved. Delgado et al. [294] have used a fluid neural-network in simulation to show that self-synchronization among the individuals enhances the efficiency of the colony as a whole and that there is a general relation between task performance and self-synchronization. Trianni and Nolfi [295] have studied how a group of robots having simple behaviours and minimal communication capabilities can show self-organized synchronization. They have used the dynamical system analysis to explain the self-organizing behaviour of individual robots for the synchronization activity using differently sized groups. Using neural networks and evolutionary algorithms to evolve controllers they have displayed the self-organized synchronization both in simulation and on real robots. Chevallier et al. [296] have studied the behavioural synchronization and division of labour in foraging swarms using simulations and have concluded that the synchronous flocking of individuals in the swarm reduces the chances of collision thus increasing foraging efficiency. Zecca et al. [297] have used RFID tags distributed in a simulated environment which act as information sources. Using ubiquitous computing by a swarm of simulated robots, they have portrayed synchronization among the robots with their teammates within a zone to complete a collaborative task.

Most of the work reported so far are in the form of simulation and involves some amount of direct inter-agent communication to realize the synchronization behaviour. These direct inter-agent communications become difficult to realize in real-world due to interference, loss of connectivity or bandwidth problems. Further, these works do not address the issue of detecting failures and consequent mechanism or self-healing during a synchronization activity. Though stigmergy is central to most synchronization approaches, it seems that the researchers have not sufficiently exploited the idea of indirect communication. Attempts to achieve a real-world synchronized swarm behaviour with an ability to recover from possible failures still seems grossly missing.

In the succeeding sections, the proposed technique and the results obtained from its *emulation* on real networks have been described.

5.2 System Description

The technique proposed in this chapter uses asynchronous nodes (robots) and mobile agents. This means that the nodes and the agents do not participate in any kind of exchange of information amongst each other. Both the entities - the nodes and the agents thus act independently. The goal is to execute an ordered sequence of distinct tasks using a set of mobile agents populating a network such that all the agents switch on the next task in the sequence simultaneously, thus making the executions at the various nodes synchronous. The order or sequence of the tasks is available to all the agents *a priori* as in [266]. For example assume a scenario in which tasks T_1, T_2,... T_m, need to be executed sequentially to build a structure. Let us imagine the construction of an enclosure by a

set of agents (say ants) wherein tiles are to be laid initially on the floor (T_1), then the bricks laid to form the walls around it (T_2) and finally concrete slabs that make the ceiling (T_3). These tasks need to be performed in the prescribed sequence. It is obvious that the ceiling cannot be built without the walls in place as also the walls cannot be, till the floor is in place. However, executing each task requires several tiles, bricks or slabs to be laid and cemented. Building such an enclosure could be divided into a sequence of parallel sub-tasks that could be performed by a set of mobile agents. When all the required tiles are laid on the floor by a set of agents to form the floor, a task switch needs to be performed within each agent to instigate it to perform the next task of laying bricks for the walls. Naturally an early switch to the second or third task by any of the agents can cause the enclosure to collapse. The asynchronous agents thus need to switch to the next task almost synchronously using some form of innate releaser [298] assisted by stigmergy. Further, if a few bricks in a wall come off loose when the ceiling is being built, the agents in their proximity should switch back to this previous task of cementing bricks and on completing go back to building the ceiling, thus exhibiting *self-healing*.

The main objective of the work reported herein is to synchronize the switching from one task to its successor in a given sequence of tasks, all of which need to be executed within every node within a network. The agents that switch on these tasks should also exhibit *self-healing*.

5.3 Synchronization Scenario

The strategy described herein is inspired from the mechanism on how a colony of ants or some such swarm of insects possibly build an enclosure. To simplify the process of exposition, an approximate analogy of the construction of a square enclosure, as mentioned earlier, is provided which involves the following tasks to be executed by ants in the following sequence:

1. Laying of tiles along the floor. (Construction of the floor is assumed to be complete only after this task is performed on each unit of space on the floor.)

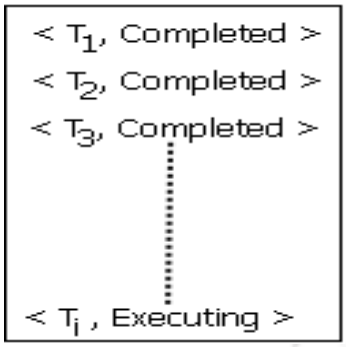
2. Laying bricks on any of the four sides of the floor to eventually, build the walls. All four walls could be built in parallel by the insects. (All four walls around the floor are built when this task is executed for every unit of space that comprises the four walls.)

3. Constructing the ceiling slab by slab. (It has been assumed that each such slab is propped on some support initially and eventually glued to the neighbouring slabs to form the ceiling.)

It may be observed that this process of performing a set of tasks in the prescribed sequence will yield an enclosed structure provided each individual comprising the colony or the swarm ensures that the switch from one task to the other is synchronized. Otherwise, it may so happen that some of them may be building the ceiling when the floor and/or the walls are not completely built. Synchronization can be achieved by having a centralized control which however would result in numerous requests and signalling between the controlling and the controlled entities. Synchronization needs to be achieved by indirect communication and control, as is possibly observed in nature. Having provided the approximate analogy, a real network-oriented example as the emulation and test scenario to demonstrate the efficacy of the proposed approach is being discussed further.

5.3.1 Agents and Tasks

The main entities in the synchronization scenario are: (a) A set of μ mobile agents, (b) A network of n nodes comprising the environment and (c) A set of m different tasks, $T = \{T_1 \rightarrow T_2 \rightarrow ... \rightarrow T_m\}$



(a)

Agent Configuration

Task {T_i}	Task Potential
T_6	100
T_5	100
T_4	100
T_3	55
T_2	0
T_1	0

(b)

Figure 5.1: Snapshot of (a) Tuples in the logged information at a node (b) The internal information within an agent (The greyed row indicates the current task being searched by the agent and its corresponding potential.)

, to be executed at all the n nodes in that order known *a priori* to the mobile agents. These tasks T are temporally distributed in the system. Both μ and n are not known to the agents and hence are not used in the entire synchronization mechanism.

The mobile agents can carry within themselves m codes or programs one for each of the m different tasks as their payload that need to be executed in the specified order at the n nodes. The code referred to here is the program associated to the concerned task which can be executed at a node. The set of μ mobile agents forms the metaphor of the swarm or colony while its elements viz. the agents form the individuals comprising the swarm. The nodes, forming the network populated by agents, metaphorize the environment. The individuals comprising the swarm inhabit the environment (nodes) where the tasks T_1, T_2,...,T_m need to be executed in the given sequence at each node. Thus, the nodes contribute to a passive environment while the mobile agents form the active entities that inhabit the nodes. Tasks are executed within these nodes but the nodes are unaware of both the task and the sequence in which they need to be executed. The nodes do not possess the code for any task (i.e. they are oblivious of any information pertaining to the execution of a task). Each task must be executed at each node in the network concurrently according to the prescribed sequence. When an agent provides code for a particular task T_i to be executed at a node, it means that the agent has *serviced* that node for the task T_i. Initially, task T_1 has to be executed in all the nodes. After a node is serviced for task T_1 by an agent and the same is executed at the node then this node is required to be serviced for the next task in sequence viz. T_2, and so on. The set of agents, each of which carries code for all the m tasks, migrate within the network of nodes in a conscientious [299] manner. This means that these agents always try to avoid the nodes which they have recently visited in their attempt at patrolling the network.

A node could either be executing a code for a particular task or have completed the execution of the same. This information is stored as a log within the node in the form of tuples (as shown in Figure 5.1(a)). On reaching a node an agent finds the current task required to be executed at this node. The agent services the node based on a set of conditions discussed later. When an agent services a node for a task T_i, it adds the tuple ($< T_i,\ Executing >$) to the log within that node. Execution of the code for the task T_i at the node completes the associated task and also updates the tuple $< T_i,\ Executing >$ to $< T_i,\ Completed >$ to reflect the completion. In the meantime the agent migrates to a neighbouring node and continues its sojourn in the network.

The synchronization process can thus be viewed as a set of mobile agents moving around in a network of nodes stigmergically sensing the tuples at a node and then deciding the tasks to be serviced therein. For example, in the analogy described earlier, after a tile is laid in a unit space on the floor (i.e. when a node in the network is serviced for a task say T_1), then that particular space (i.e. the node) no more requires the attention of any agent to lay another tile on it. The programs supplied by the mobile agents during a service cause the nodes in the network to change their state to point to the next task the moment the current task is executed. Since several agents populate the network, they encounter various nodes with different task requirements as they migrate from one node to another. This means that if one takes a peek into the network at any point in time during the task satiation period, it can be noticed that there are τ distinct tasks to be serviced within the network. If $\mu < n$ and $\tau > 2$, one can infer that the agents are servicing nodes in an asynchronous manner. Higher values of τ would indicate the haphazard manner of task execution within the network. The ideal value of τ ($\tau = 1$) will be achieved only when there are μ agents ($\mu \geq n$) with at least one agent at every node throughout. However, when this is not so ($\mu < n$) the practical minimum value of τ is 2.

The main objective of the task synchronization process described herein is to ensure that the value of τ, the number of distinct task requirements by the nodes of the network at any moment of time, is kept to its practical minimum. The next section describes the technique to achieve the same.

5.4 Synchronizing Asynchronous Mobile Agents

As already mentioned the sequence of tasks that need to be executed by the individual agents is available *a priori* within the agents. The main focus of the proposed technique is to stigmergically sense the moment to trigger a task-switch within every agent, in a uniform and synchronous manner within the network. This needs to be done with no inter-agent communication and in a completely distributed and decentralized manner. Ideally only when all nodes have executed the task T_i should the agents switch to servicing T_{i+1} thus achieving perfect synchronization across the network.

Each agent, k, carries within itself a value called *potential*, $P^k_{T_i}$, a concept similar to resource as described by Nair et al. [300] with reference to emotion generation. A potential is more like an innate drive based on which an individual agent switches its behaviour or takes an action. Each task has a dedicated potential. An agent continues to search for a j^{th} task as long as it has potential for that task. Figure 5.1(b) shows the snapshot of the internal information (Tasks and their respective potentials) along with the current task being searched for, by an agent (shaded in grey). As an agent services more and more nodes for j^{th} task, it internally generates a reward that motivates it to sustain its search for such nodes requiring service for the same task. The reward it generates is proportionate to the currently available innate potential for the j^{th} task. These rewards tend to reinforce the potential corresponding to this task. However, the presence of other peer agents that are also searching for the same task can, during its conscientious sojourn within the network, land this agent onto nodes that have already been serviced for the j^{th} task and now required to be serviced for the next i.e. the $(j+1)^{th}$ task. This situation penalizes or provides a reverse motivation to the agent thereby proportionately degrading its potential for the j^{th} task. Eventually the potential or motivation to search for the nodes required to be serviced for the j^{th} task for each agent dies down to zero. This is the point when an agent inherently switches to searching for nodes requiring service for the $(j+1)^{th}$ task. Potentials are specific to each task and their build-up and retardation are governed by the reinforcements and penalties received by the agent.

Algorithm 2: Agent behaviour in algorithmic form

Input : Sequence of tasks T_i and corresponding potential P_{T_i}, $i \in \{1, 2, 3....m\}$
Initialization: $\xi(< Task, Status >)$, Tuple at the current node N

```
1  while The agent is at node N do
2      for i = 1 : m do
3          if P_{T_i} ≠ 0 then
4              Search for task T_i;
5              Break;
6          end
7          else if i = m then
8              Exit() ;                                  // All tasks are over
9          end
10     end
11     if ξ(< T_i, Completed>) is absent at N then
12         Service and Switch ON the execution of task T_i at N;
13         Compute-reward();
14         Increment-potential(P_{T_i});
15         Migrate to a neighbouring non-visited/non-recently-visited node;
16     end
17     else if ξ(< T_i, Executing>) is present then
18         Migrate to a neighbouring non-visited/non-recently-visited node;
19     end
20     else
21         Compute-penalty();
22         Decrement-potential(P_{T_i});
23         Migrate to a neighbouring non-visited/non-recently-visited node;
24     end
25 end
26 ---------------------------------------------------------------------------
27 For Self-healing the line number 11 (above) should be replaced with the following line:
28 ---------------------------------------------------------------------------
29  if ξ(< T_j, Completed>) is absent at N for any j <= i  then set i = j
```

5.4.1 System dynamics

Let $P_{T_i}^k$ be the potential for the task T_i for the k^{th} agent. An agent k will execute a task T_i provided

$$P_{T_i}^k \neq 0 \wedge P_{T_x}^k = 0 \ \forall x < i$$

where $x, i \in \{1, 2, 3....m\}$.

The behaviour of an agent for servicing and switching ON the execution of a particular task at a node in the network by an agent is depicted in Algorithm 2. It is apparent that the agent does not switch to the next task in the sequence within it until the potential of the current task becomes zero. It may be noted that the algorithm does not allow an agent to interact with any other agent nor does it use the number of agents (μ) or nodes (n) in the network. Further none of the decisions made by an agent are probabilistic in nature.

The updates to the potential based on rewards and penalties received by the k^{th} agent are

governed by the following equations:

$$P^k_{T_i}(t+1) = P^k_{T_i}(t) \pm \theta^k_{T_i}(t) \tag{5.1}$$

where,

$$\theta^k_{T_i}(t) = \left[\frac{\{\sum_{z=(t-\omega)}^{t} \theta^k_{T_i}(z)\}}{\omega} \cdot \frac{P^k_{T_i}(t)}{P_{max}} \right] + \delta \tag{5.2}$$

δ equals zero if agent receives a reward; else a positive non-zero constant.

As discussed earlier, $P^k_{T_i}$ is the potential for the task T_i within the k^{th} agent. The value of $P^k_{T_i}$ is constrained within a range $[0, P_{max}]$. P_{max} is the maximum potential an agent can accumulate for a task. $\theta^k_{T_i}$ either acts as a reward which the agent earns on servicing a node thus incrementing $P^k_{T_i}$ or as a the penalty when it migrates to a node which needs to be serviced for another task T_j $(j > i)$ resulting in degradation of $P^k_{T_i}$. ω is the length of a sliding *history window* (H) [281] which collects the values of $\theta^k_{T_i}$.

The change in the value of $P^k_{T_i}$ (calculated as $\theta^k_{T_i}$) is based on a heuristic that essentially captures the manner in which the agents is servicing the nodes. This heuristic is derived from a collection (H) of past rewards and penalties along with the available proportional potential $P^k_{T_i}$ at the time of calculation.

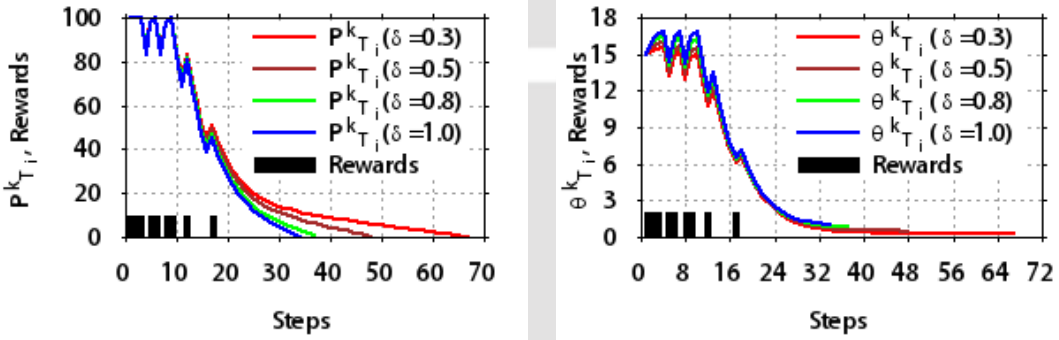
$\theta^k_{T_i}$ thus decides the amount to be extracted from the currently available potential that can be provided as the reward or penalty to the agent. The values in H provide the gradient of variations of $P^k_{T_i}$. For simplicity, the values available in the H (the first term in equation 5.2) has averaged and this averaged value has been scaled using the current normalized potential (the second term in equation 5.2). The concept is analogous to a sponge soaked in water [300] which when squeezed for the first time would release a large volume of water (high $\theta^k_{T_i}$ because of high $P^k_{T_i}$). Successive squeezing even with the same force will however yield a reduced amount of water. The potential metaphorizes the water and the $\theta^k_{T_i}$ metaphorizes the force. Rewards replenish the potential just as the sponge is replenished with water.

δ is a non-zero positive constant which acts as a penalty bias. Though $\theta^k_{T_i}$ can eventually reduce the potential to zero, δ accelerates the deterioration of the associated potential resulting in faster switching and reduced idle periods (discussed later).

At time $t = 0$ all tasks within an agent are conferred the maximum potential, P_{max}. This enables an agent to generate the required innate drive to search for nodes requiring service for these tasks. The concept of a potential for a task thus empowers an agent to stigmergically sense the temporal state of the service requirements at the nodes in the network. Since at $t = 0$, H remains empty, an initial value (a non-zero positive constant) is assigned to $\theta^k_{T_i}(0)$ to commence the process.

The proposed technique can be compared with that of the response threshold reinforcement model proposed by Theraulaz et al. [301] used to evolve specialized agents for different tasks scattered in an environment. In both the models, all the agents have *a priori* information of all the tasks needed to be executed in the environment. Each agent also carries with itself a task-specific parameter (potential or response threshold) for every task. These task-specific parameters are altered based on the reinforcements that the agents receive as a result of their interactions with the tasks and the environment. In the proposed technique, the tasks are required to be executed in a given sequence in synchronism while in the response threshold reinforcement model, the tasks are spatially distributed in the environment. In this sense the basic objective is thus grossly different. The more significant difference between these models lies in their reinforcement rules. In the proposed synchronization technique, the rewards or penalties are calculated based on the mean of

the values within the sliding window H (an average over time) and is also proportionate to the current value of potential. In the response threshold reinforcement model, the reinforcement to the task specific response thresholds is only a constant multiple of the amount of time that the agent spends on that particular task. Further, any reinforcement update is local to a particular task in case of the proposed technique whereas in case of the response threshold reinforcement model, the reinforcement updates alter the values of all the response thresholds of all the tasks within the agent. The authors in [301] have discussed the dynamics of the stimuli from the environment associated with a particular task in terms of the total number of agents available in the environment which is assumed to be known. The proposed technique does not incorporate any such assumptions regarding the number of agents or nodes in the environment which is a very distinguishing factor. It can also be noted that once an agent specializes for a particular task in response threshold reinforcement model, the probability of selecting any other task by this agent becomes very low while the case is not same with the proposed technique. In the proposed synchronization technique all the agents service all the nodes for a particular task until their respective potentials die down to zero. The proposed technique could thus be used from within the model proposed by Theraulaz et al. [301], if the tasks in response threshold reinforcement model consist of a sequence of sub-tasks which need to be executed in synchronism by the concerned agents.



(a) (b)

Figure 5.2: Nature of variations in (a) Potential $P^k_{T_i}$ for the k^{th} agent and i^{th} task (b) $\theta^k_{T_i}$ for the k^{th} agent and i^{th} task

The graph in Figure 5.2(a) shows the variation in potential ($P^k_{T_i}$) for different values of δ obtained from a simulation. The behaviour of the curve for potential follows the analogy of a sponge soaked in water as discussed earlier. The values of ω and $\theta^k_{T_i}(0)$ are set to 15. The instances when the agent received a reward are shown by solid vertical bars. For all other steps the agents receives only penalties. Since the agents may find more number of nodes that can be serviced in the initial phase of their service for a task, rewards are mostly concentrated in the initial phase. It can be observed from these graphs that initially the rate of decay of the potential ($\partial^t P^k_{T_i}$) is high but as $P^k_{T_i} \to 0$, $\partial^t P^k_{T_i}$ proportionately degrades showing an asymptotic behaviour. The graph depicts the change in convexity of the decay curve of potential with an increase in δ value. When $\delta = 0.3$, $\partial^t P^k_{T_i}$ is slow as a result $P^k_{T_i}$ takes more number of steps to converge to zero. As δ increases, $\partial^t P^k_{T_i}$ also increases resulting in faster deterioration and hence earlier switching. As can be observed the trend of $\partial^t P^k_{T_i}$ remains same for different values of δ. As can be seen, the rewards tend to increase the potential temporarily. It can thus be inferred that by changing the value of δ, the time for switching from one task to another can be customized to suit an application.

The graph in Figure 5.2(b) shows the variations in $\theta^k_{T_i}$ for the corresponding $P^k_{T_i}$ shown in Figure 5.2(a). As can be observed from the graph, the initial variations in $\theta^k_{T_i}$ for different values of δ are significant while towards the end, all the curves of $\theta^k_{T_i}$ tend to overlap with each other. It can be inferred from the graph that, as the sliding window H moves forward in time, the values within H become more coherent. Further, it can be seen from the graphs in Figures 5.2(a) and (b) that for higher values of δ, the potential $P^k_{T_i}$ dies down faster while the corresponding $\theta^k_{T_i}$ maintains a comparatively higher value than those where δ is low.

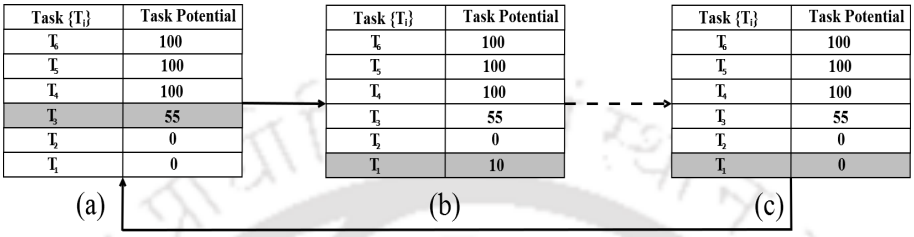


Figure 5.3: Snapshot of the internal information within an agent during self-healing

5.4.2 Self-Healing

In the analogy of the enclosure explained in Section 5.3, it may so happen that while the ants work on the ceiling, the previously made walls may collapse for some reason. Under such conditions, the ants need to switch back to the previous task of rebuilding the collapsed portions of the walls, after which they need to resume building the ceiling once again. Using the concept of potential we highlight how such a self-healing is inherently initiated by the agents without any communication amongst them. As mentioned earlier the practical value of τ is 2. Imagine a case when the nodes in the network require to be serviced for either of the tasks - T_3 or T_4 (since $\tau = 2$) and all agents within the network are currently searching for nodes required to be serviced for task T_3 as shown in Figure 5.3(a). This means that the potentials for task T_3 of none of the agents has died down to zero. Suppose at this point of time, a set of nodes fail making them to suddenly switch to a state wherein they are required to be serviced for the previous task T_1. This essentially means that the agents now need to likewise switch back to searching for nodes requiring a service for task T_1 and satiate them first. In other words if the agent is currently serving task T_i and it finds a node which is required to be serviced for a task T_j $(j < i)$, such a case is treated as the failure of a task at that node and the agent needs to switch back to service T_j.

Due to the presence of the potential for each task such a reverse switching action occurs innately within an agent. When an agent, currently searching for nodes requiring service for T_3, finds itself in a node having service requirement for T_1, its potential for T_1, $P^k_{T_1}$, is calculated using equation 5.1 by substituting $\theta^k_{T_1} = \theta^k_{T_3}$ thereby making it non-zero and positive. The resulting state of the agent is shown in Figure 5.3(b). This forces the agent to temporarily abandon its current search for nodes requiring service for T_3 (since the condition mentioned in subsection 6.3.2 fails) and switch to searching for nodes requiring service for the earlier task (in this case T_1) as indicated in the greyed portion of Figure 5.3(b). Subsequently, as can be seen from Figure 5.3(c), the potential for the task T_1 becomes zero over several transitions by the agent, making it switch back to searching for nodes requiring the service for task T_3 or more specifically, to the next task in the sequence which has a non-zero positive potential. The corresponding modification in the algorithm of the agent behaviour to enable self-healing is depicted at the bottom of Algorithm 2.

This self-healing behaviour on part of the mobile agents provides for robustness and fault tolerance to the system. The mobile agents are thus inherently programmed to restart the building process in the same sequence starting from where the failure occurred. For instance if the current task being searched for was T_5 and some of the nodes suddenly need service for T_2 due to a failure then the agents would first heal T_2, after which they would check for failures in tasks T_3 and T_4 in that order and then resume searching for nodes required to be serviced for T_5. Task switches by an agent occur only when the respective task potentials of the lower and current tasks become equal to zero. This is akin to the analogy of the enclosure wherein, if some of the tiles in the floor have broken and caused part of the wall to collapse during the building of the ceiling, then the agent would have to repair the floor first, then search for lapses within the walls and finally resume building the ceiling.

5.5 Emulating Task Synchronization

There are three ways to test an algorithm that is designed for real-world applications viz. simulation, emulation and real-world-experimentation [302]. A simulation imitates a real-world process over discrete time in a highly controlled tightly coupled environment [303]. It provides access to global variables shared among all the entities in the system. All the processes are inherently sequential in nature and the problems faced in a true parallel environment are thus not addressed appropriately. An emulation provides a certain degree of realism by combining the software and the hardware components designated for real-world implementation [302]. Hence an emulation environment uses the advantages of simulation techniques to cater to the real-world experimentation. The real-world experimentation is the ultimate test-bed to guarantee the working of an algorithm [304] since such experimentation deals with real dynamic environments comprising various unknown parameters. The results presented through simulations and their mapping to real-world processes have been critically examined in [274]. They have concluded that it is difficult to ensure that the simulation studies genuinely capture the real-world systems.

Thus a discrete simulation which is inherently synchronous may underestimate the efficacy of the proposed technique. In order to test this technique it was thus imperative that the environment comprising the nodes of a network and the mobile agents, either be emulated or alternatively real-world experimentations be performed. This chapter provides the results obtained from both emulation and real-world experimentation of the proposed synchronization technique. For experimentation, *Typhon* [132], a mobile agent framework has been used.

Six computers, connected within a LAN, were used to emulate various network topologies. An additional non-participating node within the LAN was also used as a server to log the status of the tasks and agents at various nodes, asynchronously. Status information included current tasks, rewards and penalties, current potential values, etc. which were used for plotting the relevant graphs. It may be noted that the non-participating node did not influence the agents or the participating nodes in any of their processes or actions. The services for the required tasks are provided by the agents concurrently and thus the participating nodes send their status information to the non-participating node almost simultaneously. The non-participating node thus queues them before logging and time-stamping. This could introduce a small delay in logging the information, making the data look slightly interspaced in the graphs shown later.

Conscientious mobile agents and a sequence of six tasks were used in the experiments to test the efficacy of the technique. The following values were used in the experiments unless otherwise specified:

$$m = 6,\ \theta^k_{T_i}(0) = 15,\ P^k_{T_i}(0) = 100\ (i = 1, 2..m),\ P_{max} = 100,\ \omega = 15$$

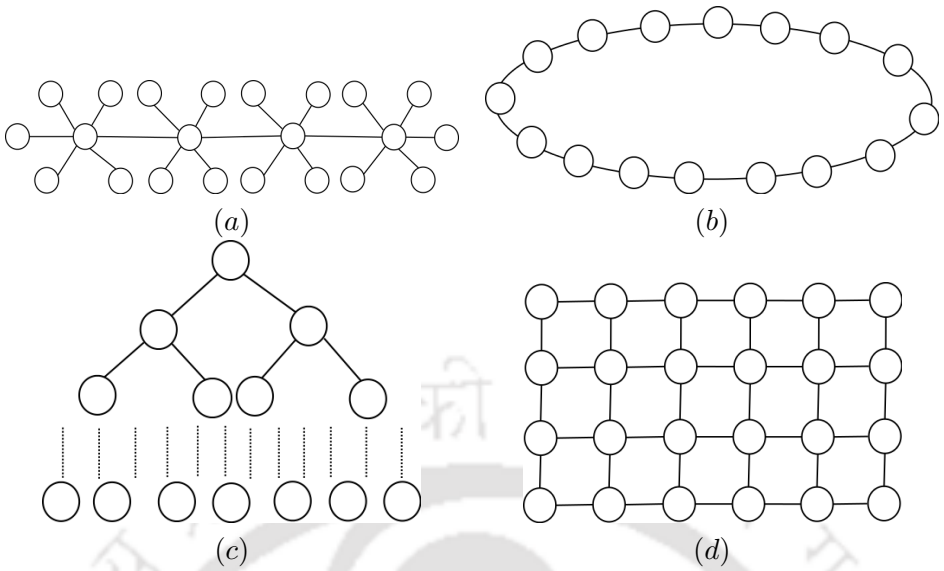


Figure 5.4: The static network topologies used in the experiments (a) Star-bus (b) Ring (c) Tree (d) Grid

The task execution time (ρ^{T_i}) for all the tasks T_i ($i = 1$ to m) is set to a constant C. The experiments were carried out on network topologies which include four static ones, samples of which have been depicted in Figure 5.4. In addition, two dynamic network topologies, also using 60 nodes each, were used. The dynamic topologies were created using the Erdős-Rényi models [252] over the Grid topology shown in Figure 5.4(d). In one of the dynamic network topologies (Dynamic Topology I), the positions of the nodes in the network do not change but their connections to neighbouring nodes may make or break. This can cause the formation of different subsets of the topologies generated by *G(n, E)* model [252] where E is the subset of all the edges (connections). In the other dynamic topology (Dynamic Topology II), nodes were made to freely connect to any other node and drop connections with any of their current neighbours based on *G(n,p)* model [252] with probabilities, $p = 0.7$ and $p = 0.5$ respectively. Unlike the previous case, here the nodes could connect to more than four others, thus forming topologies more complex than the initial Grid.

Self-healing was emulated by forcing some nodes to switch back to a previous task. A real-world experimentation using mobile robots is also presented later.

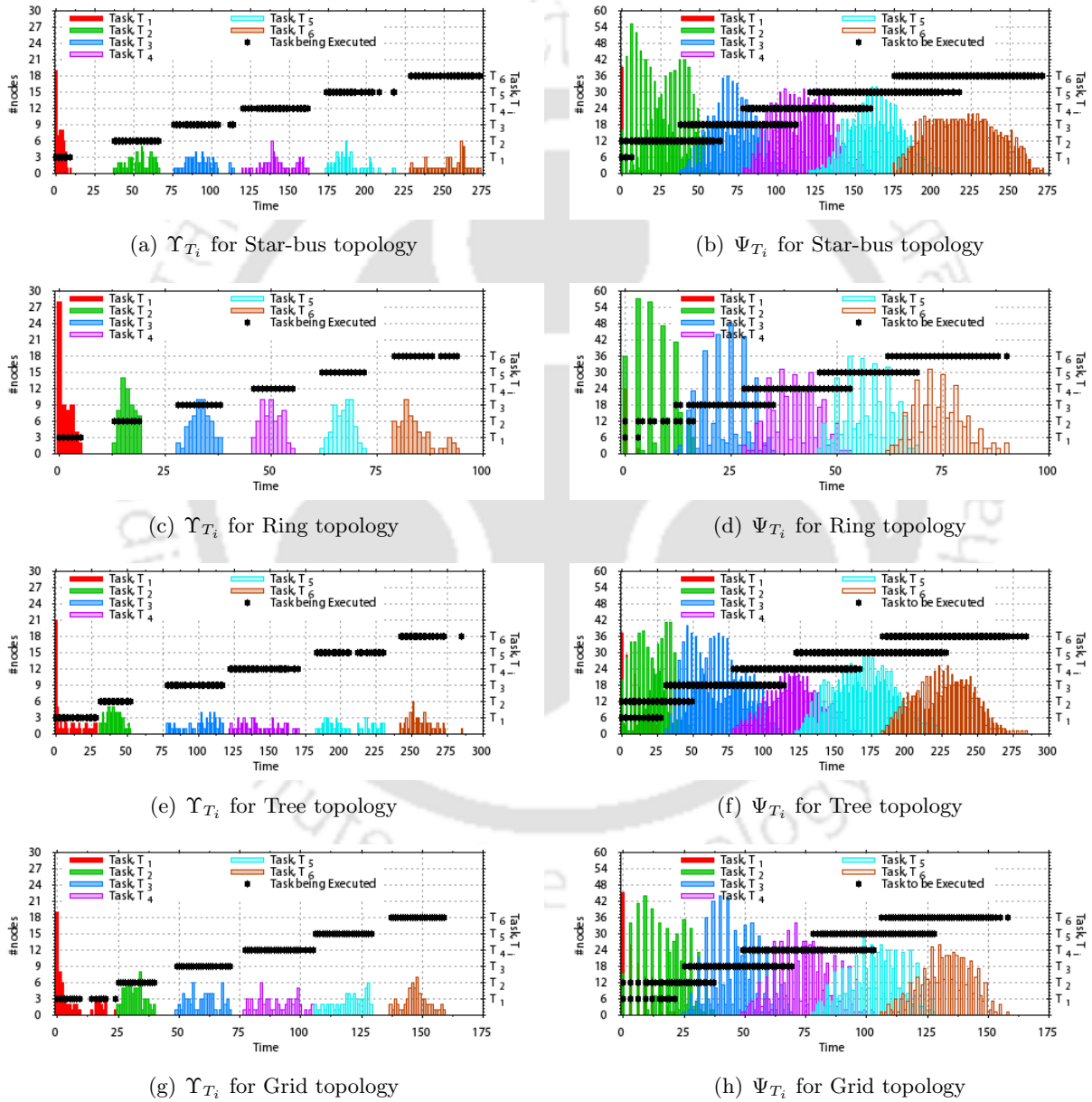
5.6 Experimental Results

The mobile agents were initially placed in different nodes and were triggered to commence their patrolling within the network. The initial placements of the agents in all the experiments carried out, were same.

To efficiently test the proposed synchronization technique, it is crucial to investigate the performance of the proposed technique with all the related parameters and environmental conditions. Since the system involves a network of nodes, the architecture or topology of the network could influence the synchronization technique. Hence a study of the proposed technique on different topologies was conducted. As the proposed technique is oblivious of the number of agents (μ) and nodes (n), performance studies by varying these numbers could provide more insights on the working of the same. As described in section 6.3.2, since δ plays a significant role in the decay of

potential, its effect on the performance of the technique has also been analyzed. Further investigations include the performance of the proposed technique in dynamic networks and also with the self-healing feature which accounts for the tolerance and robustness of the proposed technique.

The subsequent sections portray the results obtained from varying the topology of the network, the number of agents (μ), nodes (n), the value of δ and task execution times (ρ^{T_i}) to reveal the effects of each on the performance. Results obtained when the technique was used in conjunction with dynamic networks have also been presented. Self-healing was for different types of failures in the various topologies. The results obtained have also been discussed.



(a) Υ_{T_i} for Star-bus topology

(b) Ψ_{T_i} for Star-bus topology

(c) Υ_{T_i} for Ring topology

(d) Ψ_{T_i} for Ring topology

(e) Υ_{T_i} for Tree topology

(f) Ψ_{T_i} for Tree topology

(g) Υ_{T_i} for Grid topology

(h) Ψ_{T_i} for Grid topology

Figure 5.5: Tasks being executed (Υ_{T_i}) and Tasks required to be serviced (Ψ_{T_i}) at various nodes in the different network topologies for 10 agents and 60 nodes

5.6.1 Performance across the Network Topologies

The graphs in Figure 5.5(a) through (h) depict the tasks being executed (Υ_{T_i}) and tasks required to be serviced (Ψ_{T_i}) at various nodes in different network topologies shown in Figure 5.4. The experiments were conducted using 10 agents populating each of the 60-node networks. The δ for Star-bus, Ring, Tree and Grid Topologies was set to 0.2, 0.8, 0.2 and 0.5 respectively. The graph shown in Figure 5.5(a) shows the execution of each task T_i versus time (in seconds)[1] for the Star-bus topology from the agent's perspective (Υ_{T_i}). The vertical coloured bars indicate the number of nodes (#nodes) in which the execution of a particular task has been switched ON. Thus the height of these vertical coloured bars indicate the extent of parallelism of the execution of a concerned task within the network. The dots represent the specific task (T_i) that is being executed in the network at a particular instant.

In the Figure 5.5(a), between 0 to 1 second, 19 nodes start the execution of task T_1. Though there are only 10 agents in the network, some of the agents are able to migrate to other nodes and start the execution of task T_1 within this short interval making the count rise to 19. This initial burst of service can also be observed in the remaining Υ_{T_i} graphs in Figure 5.5(c), (e) and (g) for the other topologies. All the remaining nodes are serviced for task T_1 by the 9^{th} second.

It may be noted that the total time required for executing task T_1 by all the 60 nodes is 9 seconds and the execution of the next task T_2 starts at the 37^{th} second in the graph shown in Figure 5.5(a). This gap in time means that all the agents had excess potential for task T_1 even after all the nodes were serviced for the same. The period between the 9^{th} second and the 37^{th} second thus indicates an *Idle* or *non-productive* period on part of the agents during which they move conscientiously and are continuously penalized. This period helps the agents to find any remote or undiscovered node in the network which still required to be serviced for the task T_1. The *Idle* periods also serve as guard-bands which demarcate execution of two consecutive tasks within the network. The graph indicates that at the 37^{th} second, at least one agent (say k) switches to service a node that required to be serviced for the task T_2. This is the time when $P^k_{T_1} = 0$ for an agent k. Such *Idle* times can be reduced by increasing the value of δ. Further, at any point of time during the experiment, if two consecutive tasks (e.g. T_2 and T_3) are being executed within the network then that would essentially indicate a certain amount of overlapped executions and hence a loss in synchronization. This could happen because at least one agent would have exhausted its potential for task T_2 ($P^k_{T_2}$) earlier than the others. Such overlapped execution may be reduced by decreasing the value of δ, an aspect which has been discussed in the next section. None of the Υ_{T_i} graphs in Figures 5.5(a), (c), (e) and (g) depict any such overlaps indicating proper synchronization.

The graph shown in Figure 5.5(b) shows the specific tasks required to be serviced at the various nodes in the network at various points of time for the Star-bus topology from the perspective of the nodes Ψ_{T_i}.

From 0 to 6 seconds, while some nodes required to be serviced for T_1, others who have already been serviced for T_1, required to be serviced for T_2. This conforms to the best practical case wherein at any point of time the total number of distinct tasks for which services are required (τ) by the nodes in the network is 2. In the period between 6 and 37 seconds, only the service for T_2 is required but the same is not serviced in spite of the fact that agents are knitting through such nodes. This period contributes to the *Idle* period as discussed earlier. In the graph in Figure 5.5(b), the value of τ remains at most at 2 consistently throughout the run of the experiment. If at any point of time τ becomes greater than or equal to 3 then that would indicate an overlap of executions of tasks.

Figures 5.5(c) and (d), 5.5(e) and (f) and 5.5(g) and (h) portray similar graphs for the Ring, Tree and Grid topologies respectively. Execution of tasks in all these topologies are fairly syn-

[1]Note: The time on X-axis of all the graphs reported in this chapter is in Seconds.

Table 5.1: Performance based on τ for different topologies

Topology	Total *Idle* time in seconds
Star-bus	66
Ring	32
Tree	142
Grid	25

chronous with τ equal to 2 throughout. However, more *Idle* periods are observed in the Tree topology possibly because of the complexity of the network contributing to high waiting times on part of the nodes since agents need to backtrack several times to the parent nodes to reach the nodes in the other branch. Table 5.1 shows the total idle times for the various topologies. In the Star-bus, Ring and Grid topologies, the agents need to perform lesser number of back-trackings, which is possibly why the *Idle* periods are lesser than that of the Tree topology.

In Υ_{T_i} graph of Star-bus topology (5.5(a)) some gaps can be observed during the execution of task T_3 (from 104 to 112 seconds) and T_5 (from 208 to 217 seconds). During these periods all the agents in the network are not in the vicinity of any of the nodes which are required to be serviced for the concerned task. Similar gaps can be observed in Ring, Tree and Grid topologies for the tasks T_5, T_4 and T_5 and T_1 respectively. Further, the *Idle* periods in between the execution of two consecutive tasks are not evenly distributed as one moves from task T_1 to the final task T_6. This is due to the dynamic placement of agents at different nodes at the beginning of the execution of each task as a result of their conscientious migrations. As discussed earlier, since at the start of the experiment all the agents commence to switch ON the execution of task T_1 almost at the same time at different nodes, a high burst of execution is observed at the beginning of the experiments in all the topologies. However, it is rarely possible that all the agents in the network switch to start the servicing for the next task exactly at the same time. This is the reason why such bursts of executions are not seen again in the graphs.

The synchronization technique seems to perform fairly well in all the four topologies. This shows that the proposed synchronization technique is capable of managing synchronized execution of different tasks in a sequence in various networked environments. It may be noted that different values of δ are used for different topologies in order to achieve the synchronized execution of tasks. The following section discusses the effect δ and illustrates the reason to choose a particular δ for each of the topologies.

5.6.2 Effect of Varying the Penalty Bias (δ)

Figure 5.6 depicts the variations in total *Idle* periods and the overlapped executions for different topologies with $\mu = 10$ and $n = 60$. The value of δ was varied from 0.2 to 1.0. It may be noted that the non-productive *Idle* periods reduce as δ increases. However, with increase in δ there is an increase in overlapped executions of the tasks making $\tau = 3$ thus indicating loss of synchronization of tasks executions. The reason for this is that as δ increases the potential for a particular task of an agent diminishes faster thereby causing it to switch to the next task earlier than desired resulting in overlaps.

In the graph shown in Figure 5.6, there are no overlapped executions for a specific value of δ, say δ', which is 0.2 for the Tree topology. Beyond δ' overlapped i.e. out-of-order executions throw the system into a chaotic state. One should bear in mind that δ acts as an optimizing parameter which could be used to adjust the rate of decay of potential ($\partial^t P^k_{T_i}$) as discussed in Section 6.3.2. δ can thus govern the trade-off between the *Idle* periods and the overlapped executions. An increase

in δ till δ' reduces the former while beyond δ' it increases the latter. As can be seen from the graph, δ' is different for different topologies. In the Star-bus topology, $\delta' = 0.2$ while for the Grid topology $\delta' = 0.5$. Though the Ring topology does not show any overlapped executions even for $\delta = 1.0$ due to its inherent simplicity and symmetric nature, yet $\delta' = 0.8$ (Section 5.6.1) has been chosen to cater to the changing dynamics of agent movements and placements.

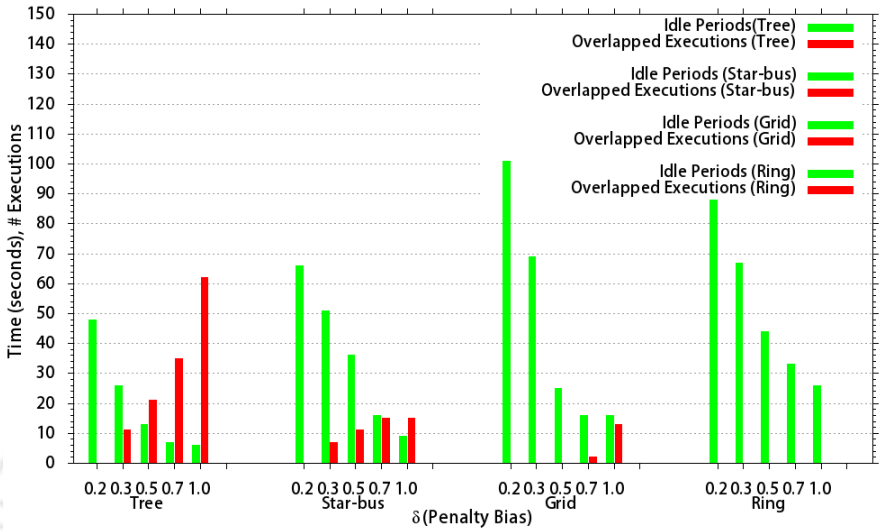
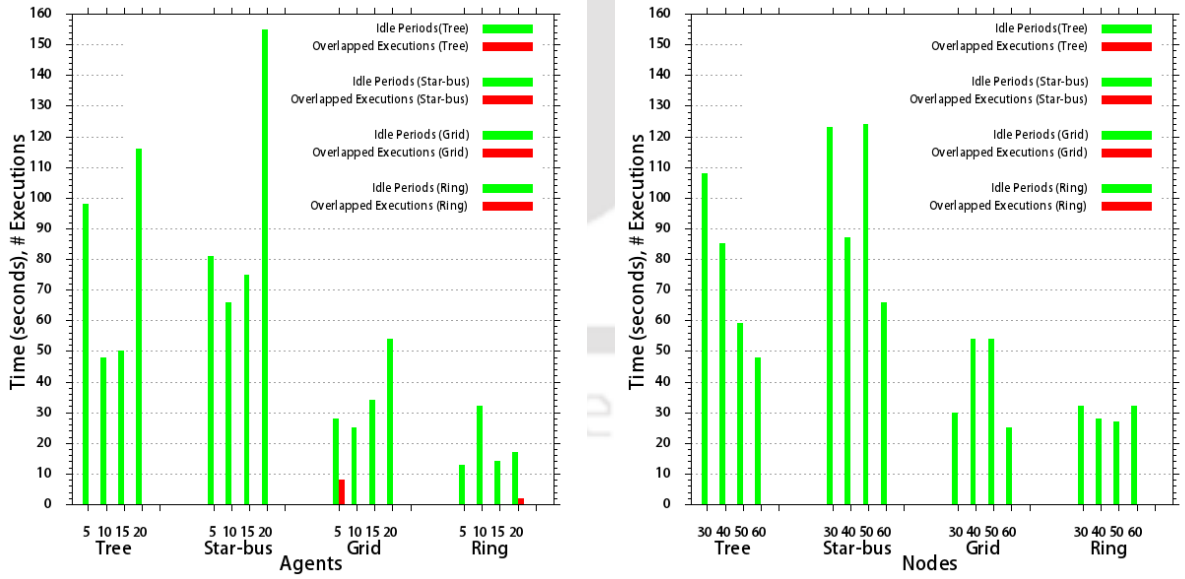


Figure 5.6: Variations in the total *Idle* periods and overlapped executions for different values of δ in the four different topologies



(a) Varying number of Agents (μ)

(b) Varying number of Nodes (n)

Figure 5.7: Variations in the *Idle* periods and the overlapped executions for (a) Varying number of Agents and (b) Varying number of Nodes, with $\delta = \delta'$ for each topology

Given a network of nodes and number of agents, if δ is varied from a very small value to

a large value, one would find that the nature of the curves of *Idle* periods and the overlapped executions are parabolic with δ' at the vertex (point of intersection). This provides an empirical method to achieve the value of δ' for a given setting. Thus δ' optimizes the performance of the synchronization technique in terms of time without compromising on the synchronization of task executions. Lower *Idle* periods apparently mean lower task completion times. A very low value of δ ($\delta \rightarrow 0$) would naturally provide for synchronization at the cost of high *Idle* periods and hence the overall completion of all the tasks. It may be noted that δ' also depends on the complexity of the underlying network. Hence, δ plays a crucial role in both synchronization and in achieving low in-order execution times of all the tasks.

5.6.3 Effect of varying the number of agents (μ) and number of nodes (n)

The population density (μ/n) of agents in the network has a high impact on the service times of tasks at each node. It may seem that a higher number of agents (μ) could bring down the total completion time of all the tasks. However too many agents are known to clutter a network [128] resulting in service delays. Figure 5.7(a) portrays the variation in the *Idle* periods and the overlapped executions for varying number of agents (μ) in different topologies with $n = 60$. The experiments were carried out using the δ' of each of the topologies as found in Section 5.6.2. It can be observed from the graph depicted in 5.7(a) that the variations in the *Idle* periods for different topologies do not follow a constant trend as μ varies however the overlapped executions are zero or negligible. In case of the Tree topology, the *Idle* periods are high for $\mu = 5$ and 20 (98 and 116 seconds respectively) while the *Idle* periods are least when $\mu = 10$ (48 seconds). A similar trend can be observed in case of the Star-bus topology. A small amount of overlapped executions can be observed in case of the Grid topology (8 instances of overlapped executions out of a total of 360 executions) for $\mu = 5$ while in the Ring topology there are 2 instances of overlapped executions for $\mu = 20$. In all these cases the number of overlaps could be reduced further by slightly varying the value of δ'.

It may be observed from the graph shown in Figure 5.7(a) that there lies a value of μ for each of the topologies for which the system performs with least *Idle* periods while successfully maintaining the desired synchronization ($\mu = 10$ for Tree, Star-bus and Grid topology while $\mu = 5$ for the Ring topology). The graph also portrays that even with different the values of μ, the synchronization technique portrays an acceptable performance with almost zero overlapped executions. It may be observed that for lower values of μ, the agents may not be able to span the search for a particular task (T_i) across the network with the given non-zero potential ($P^k_{T_i}$) thus resulting in a loss of synchronization as observed in case of Grid topology with 5 agents. While high values of μ can cause delay in agent migrations at bottleneck links of the network (such as a bridge connecting two segments of the network) thus causing delayed service to some of the nodes (Ring topology with 20 agents) and consequently increasing the service and overall task completion times. Hence, the number of agents, μ, should be chosen judiciously as per the requirements of the application at hand so as to achieve faster service and low convergence times. An adaptive method to automatically increase or decrease the number of agents [128] forming a swarm, based on the current requirement within the network could enhance the performance of this proposed synchronization technique.

The graph in Figure 5.7(b) portrays the variations in the *Idle* periods and the overlapped executions for varying number of nodes ($n = 30$ to 60) for the different topologies with $\mu = 10$. The corresponding values of δ' for the topologies were used in the experiments. The graph shows that for all values of n, there are no overlapped executions in any of the topologies indicating no loss in synchronization.

An increase in the value of n forces the agents to broaden their search for servicing each task

across the network. A high value of n with a low value of μ could mean that the agents are not able to service all nodes given a finite value of potential for each task, eventually resulting in loss of synchronization. A low value of n with a high value of μ could however increase service delays. As can be seen, even though changes in μ and n can affect the service, the proposed technique still is able to ensure synchronized execution of tasks across all the nodes in the network.

It may be observed intuitively from the graph that once the best value of δ i.e. δ' is found for a certain maximum value of nodes, n_{max}, populated by a minimum number of agents, μ_{min}, any decrease in the number of nodes or increase in the number of agents (for the same δ') does not lead to any significant loss in synchronization. However it may noted that a larger number of agents populating a small network should be avoided lest the overall execution times increase due to increase in node-to-node migration times within the network [128] .

Since both μ and n are not known, a system designer could approximate or assume an upper bound on for n viz. n' and a lower bound for μ viz. μ' and determine δ' for the given network to achieve synchronization of tasks.

5.6.4 Effect of Task-Execution Times (ρ^{T_i})

So far it has been assumed that all tasks (T_i) once initiated take the same amount of time for execution (ρ^{T_i}). Practically, this may not be the case always. The time required to execute a task T_i may differ from one node to another. This may be due to the difference in processor speeds, specific task requirements or the computing environment of the machines (nodes). In order to investigate the effect of varying task-execution times (ρ^{T_i}) across the heterogenous set of nodes or processors, experiments were conducted with task T_i having different ρ^{T_i} at various nodes in the network. The graphs in Figure 5.8 present the results obtained from a 60-node Grid topology with $\mu = 10$. The best value of δ i.e. δ' was used in the experimentation.

Table 5.2: Different Task-Execution Time ranges for individual tasks in the sequence

Task T_i	Time Range (ρ^{T_i}) in milliseconds
T_1	[500, 1400]
T_2	[1500, 2300]
T_3	[2000, 3000]
T_4	[3150, 4100]
T_5	[4500, 5200]
T_6	[6000, 8000]

In order to facilitate each task T_i having different ρ^{T_i} at different nodes, we assumed a range of possible execution times for each of the tasks in the sequence. Table 5.2 lists the ranges chosen for the all the six tasks. These ranges are selected based on an increasing complexity of tasks with T_1 being the simplest and T_6 being the most complex one and thus requiring the most time. Before initializing the execution of a task T_i at a node, each agent picks a ρ^{T_i} value from the associated time range of that task at random. The execution of that task at the concerned node thus continues until the chosen ρ^{T_i} expires. The graph in Figure 5.8 portrays the synchronized executions of the tasks for this case. The graph clearly depicts that the executions of all the tasks are performed with quite an acceptable level of synchronization.

Since the proposed synchronization technique makes no assumptions for the ρ^{T_i} values of different tasks, it works oblivious and independent of ρ^{T_i}. If ρ^{T_i} for a task T_i is set to a very high value and the agents have switched ON the execution of the task T_i at all nodes in the network,

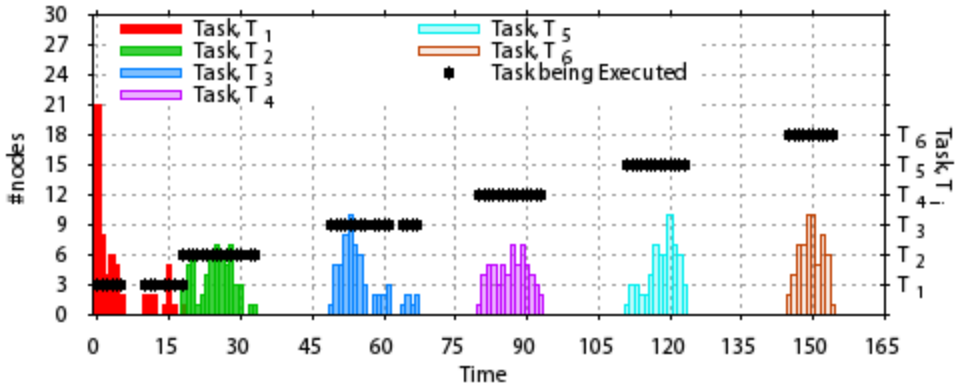


Figure 5.8: Υ_{T_i} for the tasks T_i with different ρ^{T_i} for all the n nodes

then it may happen that the whole network is executing the task T_i. In such a scenario, there would not be any nodes requiring to be serviced for any task thus making $\tau = 0$. This is the unique case when τ (number of distinct tasks required to be serviced in the network) becomes zero representing that the whole network is busy executing a particular task. The real-world experiment using robots (discussed in a later section) provides a live demonstration of how the tasks having different ρ^{T_i} synchronize using the proposed technique.

5.6.5 Dynamic Topologies

While dealing with networked distributed systems, the issues such as node failures, link failures and topological changes need to be addressed. In order to verify the robustness of the proposed synchronization technique experiments were conducted on two kinds of dynamic topologies as mentioned in Section 5.5. As explained in Section 5.6.2, an increase in δ can cause overlapped executions and loss in synchronization while a decrease in δ does not affect the same. Hence, instead of δ, the effect of change in number of agents (μ) in the Dynamic topologies I and II were studied. Figure 5.9 shows the graph of varying *Idle* times and overlapped executions in 60-node dynamic networks for different values of μ. The δ' value for these topologies were found to be 0.2.

As can be seen from the graph in Figure 5.9, both the topologies (Dynamic topology I and Dynamic topology II) do not show any instance of overlapped executions thus maintaining $\tau = 2$. This apparently means that the executions of the tasks in both of these topologies is synchronous, as desired. One can also observe the increase in the *Idle* periods with increasing number of agents. This shows that the agents are always moving within the network via. continuously changing routes thus distributing themselves in the dynamic network to the maximum possible extent. The characteristic of the Dynamic topology II, to make a connection to any other node in the network and to break a connection with any of the current neighbours, makes it a more complex network than the Dynamic topology I wherein the highest degree of a node in the network remains fixed. It may be noted that due to the dynamic nature of these topologies some nodes may get momentarily isolated from the rest of the network. In such cases, the agent may get trapped in that node until the node establishes a new connection with another node in the network. The results portrayed in the graph in Figure 5.9 shows that the proposed technique is able to cope up with all such perturbations and dynamism of the network while still maintaining the synchronous execution of tasks.

Figure 5.10 shows the time taken to complete all the six tasks at all the 60-nodes in both the

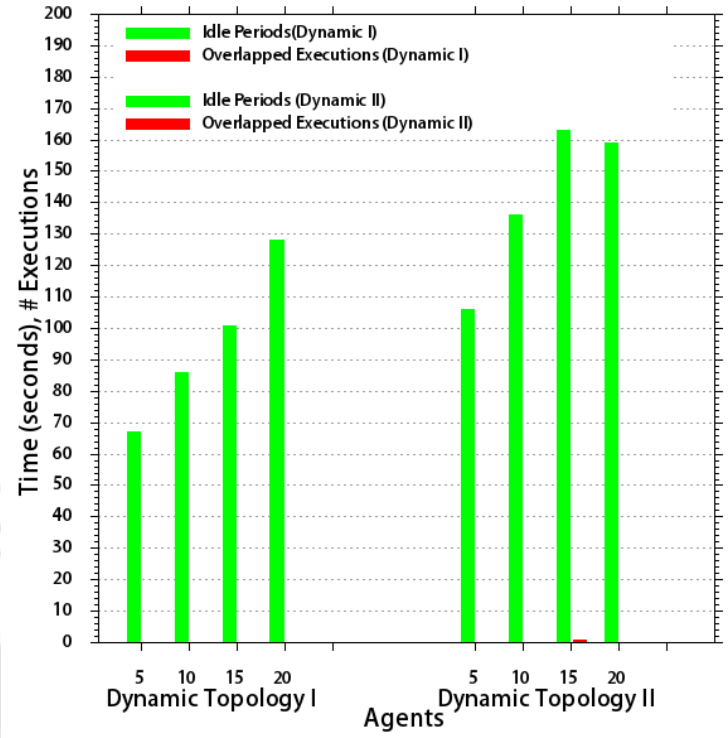


Figure 5.9: Variation in *Idle* periods and overlapped executions in Dynamic topologies I and II

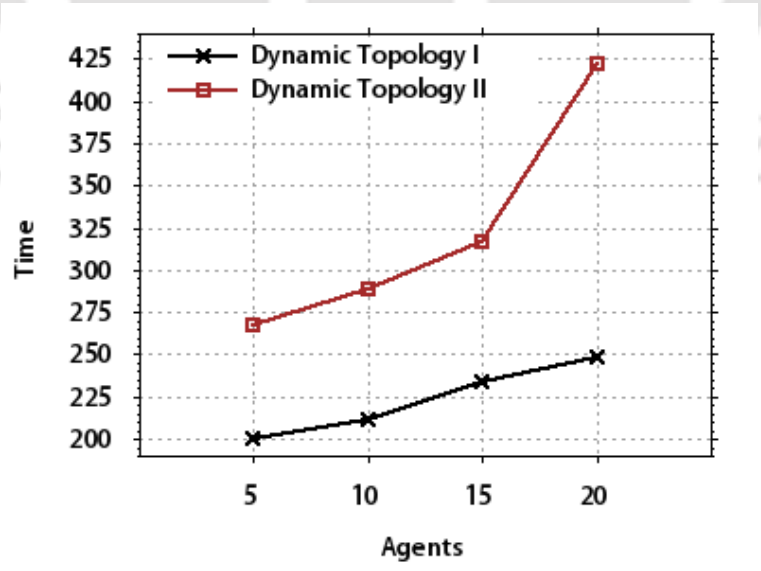


Figure 5.10: Variation in the total tasks execution times for varying number of agents (μ) in 60-node Dynamic topologies I and II

Dynamic topologies. It can be easily observed that the time taken to complete all the tasks in case of Dynamic topology II is nearly double that of the Dynamic topology I. This is due to the higher degree of dynamism within the Dynamic topology II. As this topology mimics the mobility of nodes, the agents could get trapped or choked up at some nodes making it difficult for them to migrate to other nodes. Also some non-serviced nodes may have traveled further away from those nodes which have the agents thus resulting in migration delays. These migration delays increase the overall service times at the nodes and hence the total time required to complete all the tasks in the network.

Since the proposed technique seems to maintain $\tau = 2$ i.e. synchronized execution of tasks even under high network dynamism it may be considered to be robust.

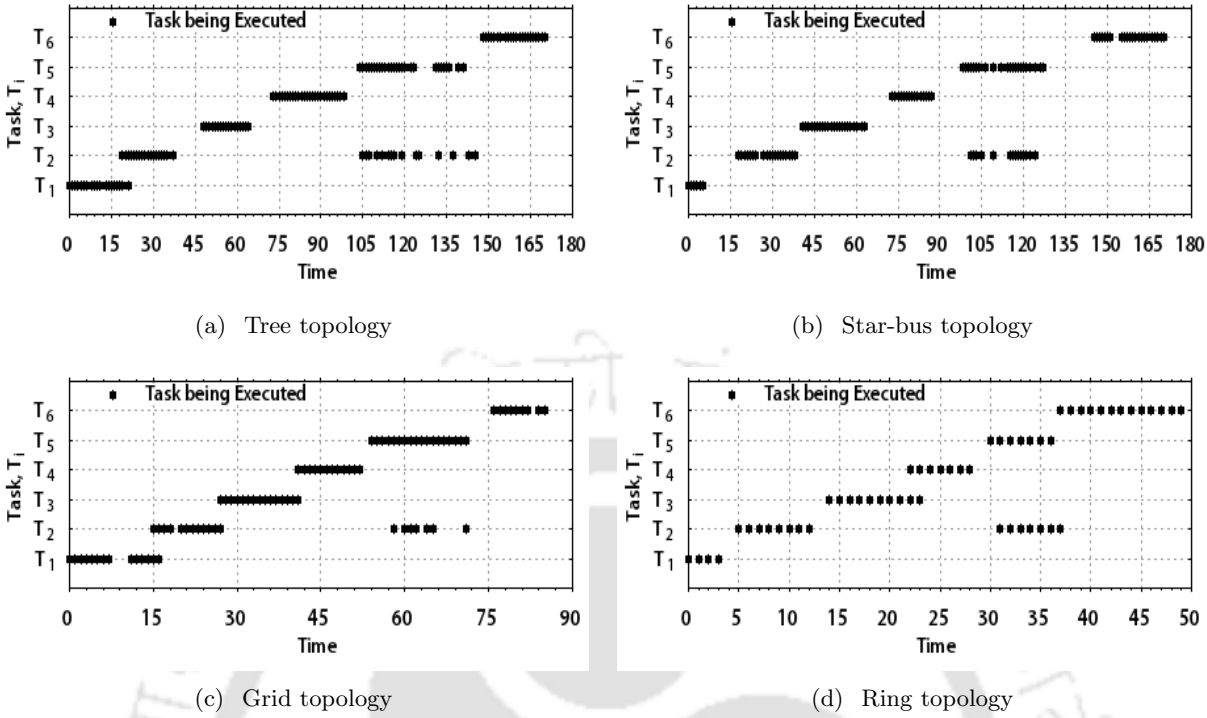
5.6.6 Self-Healing during Synchronization

To cater to the problem of failures of a completed task, the proposed synchronization technique also portrays self-healing as discussed in Section 5.4.2. This feature of the proposed technique has been evaluated by show-casing two different kinds of failures of tasks in the networks viz. *Atomic* failures and *Cascaded* failures. If the failure of a completed task does not affect other tasks in the sequence then it is called an *Atomic* failure. If the failure of a task T_i requires the agents to redo the service of all tasks from T_i to the one it was currently servicing in the given sequence then it has been referred as a *Cascaded* failure.

The graphs in Figures 5.11(a) through (d) depict self-healing in the case of Atomic failures within the topologies shown in Figure 5.4 with 60 nodes and 10 agents. In order to force self-healing for Atomic failures, 20 nodes were forced to generate a requirement for task T_2 when the agents were searching for nodes that required to be serviced for the task T_5. As described in Section 5.4.2, the agent switches to an earlier task in the sequence automatically whenever it encounters such a failure. As can be seen in Figure 5.11, when the agents currently searching for the nodes required to be serviced for the task T_5, encounter nodes now required to be serviced for task T_2, switch back to servicing task T_2. Those agents which have not encountered such nodes (when the failure has occurred) continue their search for the nodes required to be serviced for task T_5. This makes τ equal to 3 during the period of healing. Once all such failed tasks at the concerned nodes are redone, the agents involved in healing automatically switch back to servicing the interrupted task T_5. This happens when the potential for task T_2, becomes zero as discussed in Section 5.4.2.

Since the Grid topology and the Dynamic topology II are fairly more complex than the other topologies, these are used to demonstrate the effectiveness of self-healing when Cascaded failures occur. The relevant graphs are shown in Figure 5.12(a) and (b). Both the 60-node topologies were populated with 10 agents. The Cascaded failure of tasks were induced, when the agents switch to service the nodes for the task T_5 at all the nodes in the network. As can be seen from the graphs, the agents switch back to service the task T_2 as soon as the failures occur. In the case of Cascaded failures, the potential ($P^k_{T_i}$) of each of the agents is recharged to its initial value as soon as an agent discovers the failure to boost up the healing process. It can be observed from the graphs (Figures 5.12(a) and (b)) that the agents resume the usual synchronized execution of tasks as they discover and eventually rectify the failures. Since in a Cascaded failure, other tasks up in the sequence also need to be redone, the agents continues to carry out the execution of tasks after the failures in a synchronous manner.

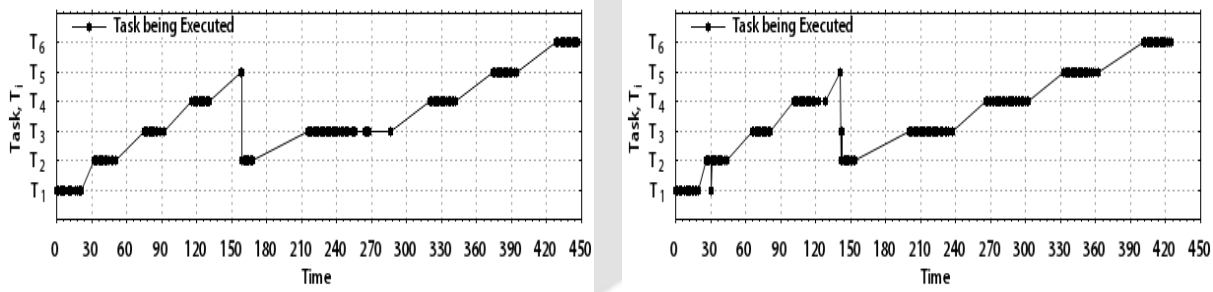
The graphs thus are in conformance to the concept of self-healing as explained earlier using Figure 5.3(a) through (c). Hence the proposed technique successfully copes up with task failures while maintaining the synchronization process as the system progresses in time. This feature of the proposed technique to handle such failures within the network makes it both fault-tolerant and



(a) Tree topology (b) Star-bus topology

(c) Grid topology (d) Ring topology

Figure 5.11: Self-healing with *Atomic* failures within 60-node networks having 10 agents



(a) Grid topology (b) Dynamic topology II

Figure 5.12: Self-healing with *Cascaded* failures within 60-node networks having 10 agents

robust.

5.7 Implementation using Real Robots

The results presented so far were obtained by emulating the proposed technique using a network of computers. Though such emulations provide fruitful insights into the technique's performance and highlights its significant properties, yet they cannot guarantee that the proposed technique will work to satisfaction in the real-world. In order to test the efficiency and viability of the synchronization technique in the real world, a simple problem has been implemented using four LEGO® MINDSTORMS® NXT [1] robots acting as nodes of a network. These robots were equipped with a touch sensor installed in their clasps to sense an object. Servo motors were used to

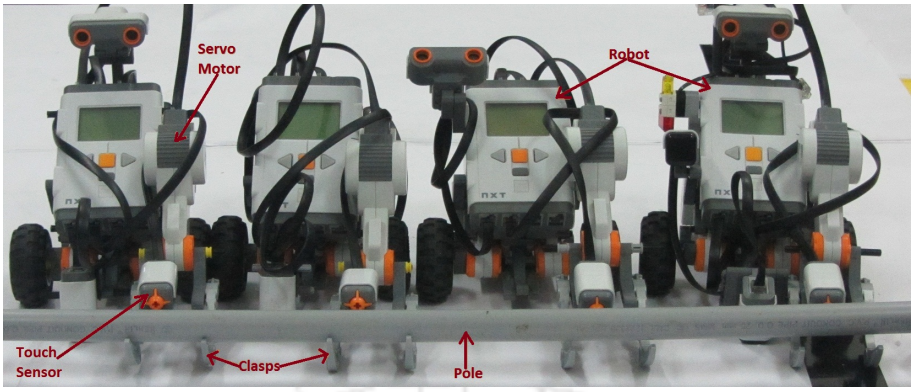


Figure 5.13: LEGO® MINDSTORMS® NXT [1] robots used in the experiment

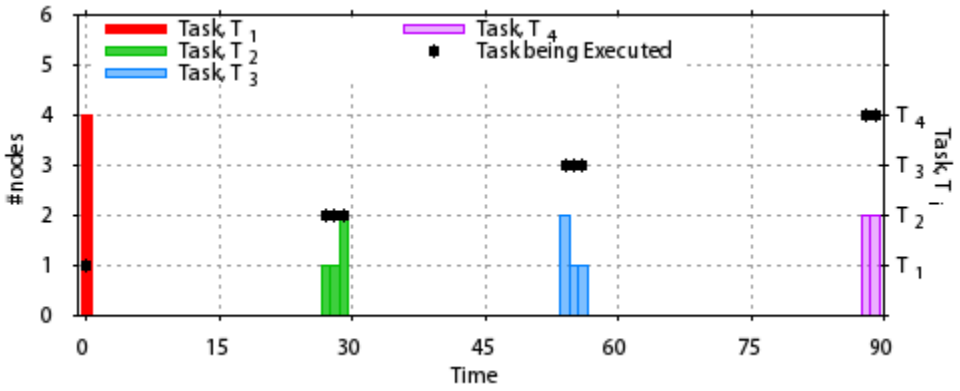


Figure 5.14: Task performance using Robots

actuate the clasps for lifting and holding objects. The LPA-PRO-NXT [241] interface was used to communicate between the robots in conjunction with Typhon [132], a mobile agent framework. The robots shown in Figure 5.13 were controlled independently by four separate computers connected in a Ring topology. Two agents ($\mu = 2$, $n = 4$ and $m = 4$) populated the network. The main goal of carrying a long pole jointly by all the four robots from one place to another was split into a sequence of four tasks viz.

- Task T_1: Move towards the pole and clasp it
- Task T_2: Lift the pole
- Task T_3: Move forward 1.2 meters with the pole in the clasp
- Task T_4: Release the clasp so as to place the pole on the ground

All the tasks (T_1, T_2, T_3 and T_4) need to be performed in synchronization by each of the four robots. In the first task (T_1), the robots were required to move forward form their current position in order to reach the pole. The touch sensor installed within the clasps of each of the robots provides a signal to the robot, as soon as the robot touches the pole, this marks the completion of the task T_1. The graph in Figure 5.14 shows Υ_{T_i} in the robotic network. It can be observed from the graph that all the four robots start the task T_1 of moving forward at the same time (at the

0^{th} second). The robots were initially placed at different distances from the pole (which essentially shows that each robot would take different time to reach to the pole). The second task of lifting the pole within the clasp starts at the 27^{th} seconds while the agents start moving forward along with the pole at around the 54^{th} second. It may be noted that the time required to execute each of the four mentioned tasks (ρ^{T_i}) is largely spaced here. The robots performed each of the tasks having different execution times, in a fairly synchronized manner proving the fact that the synchronization technique seems well applicable in real physicals systems too.[2]

5.8 Chapter Summary

While synchronized task switching is possible using a centralized control, the only way to achieve the same in a decentralized and distributed environment, is to make the individual entities communicate with one another. This calls for increased bandwidth requirements. In the real world, if all agents (or nodes) within the network were to communicate with one another to achieve synchronism, it could consume all the available bandwidth and force the network to come to a standstill. Using the concept of task-specific potentials within each agent, it has been shown that it is possible to stigmergically assess the situation of the environment and perform synchronized execution of a set of tasks.

This chapter presents a stigmergic means of synchronization on part of a set of asynchronous mobile agents populating a network. Such a method entails zero agent to agent and node to node (robot to robot) communication thereby saving on bandwidth. Results portrayed herein show that by tuning δ it is possible to achieve near perfect synchronization in task executions across the network even when the number of agents (μ) and nodes (n) are not known. Lower values of δ may be a better option to ensure no overlapped executions. It is also important to note here that the technique is oblivious of μ and n in the sense that the agents by themselves are not aware of the vastness of the network as also the presence of other agents thereby making the proposed technique a scalable one. New robots or nodes can thus be added/removed to/from the current network *on-the-fly*. The technique presented also proved to be equally efficient in both static and dynamic networks thereby enhancing its possible application areas.

Self-healing, a feature inherent to the technique, forms its forte and allows a set of mobile agents to recover from failures that could occur in a real system. Self-healing is also triggered stigmergically and in no way over-uses the available bandwidth. The dynamics of the technique seem to reveal the possible manner by which a swarm of insects in nature synchronize their actions without any global knowledge. The work presented also opens a plethora of parallel applications where failures are inevitable.

It may be noted that the performance of such a system could be enhanced if the number of agents within such a network changes based on demand. As more than required agents would clutter the network with futile migrations while a lesser number would incur delays in servicing the nodes and may involve overlapping of tasks executions. It is thus imperative to formulate an adaptive mechanism to control the number of agents (μ) based on demand. Godfrey and Nair [128] have proposed the controlled cloning mechanism for varying a population of heterogeneous set of mobile agents within a network based on the requirements of the services they provide. Such mechanisms can be coupled with the synchronization technique described herein to regulate the number of mobile agents as per the size of the network. In addition, the synchronization technique presented herein requires all the mobile agents to carry the complete information of the set of tasks along with their codes/programs thus making them homogeneous in nature. At times, such a high

[2]The video of the experiment conducted is available at http://www.iitg.ernet.in/cse/robotics/?page_id=2056

amount of payload (i.e. task related information) may make the mobile agents bulky consequently affecting their migration times within the network. The succeeding chapter presents enhancements to the synchronization technique wherein a heterogeneous set of agents, each carrying information about a specific task from the set of tasks to be executed in synchronism, populate the network. Coupled with the cloning based population control mechanism described in [128], the results of this enhanced technique have been presented therein.

"Everything is theoretically impossible, until it is done."

Robert Anson Heinlein (1907 – 1988)
American writer

6

Augmenting Population Control for Task Synchronization

A technique to synchronize the execution of a sequence of tasks using a fixed population of homogeneous agents was presented in the previous chapter. Homogeneity in the present context, means that every agent is capable of executing all the tasks in the given sequence. Thus the sequence in which the set of tasks need to be executed is also known *a priori* by all the agents. The work presented in this chapter goes a step further to address the same problem of sequential execution of a set of distinct tasks being performed at the distributed robotic nodes in a network, the synchronization of which is facilitated by a set of *heterogeneous* mobile agents. Such agents carry information that empower them to execute a distinct task from a set of interdependent tasks. Heterogeneous agents thus have a lower payload than their homogeneous counterparts, making them more agile. Instead of using homogeneous set of agents, the technique described herein focuses on the near synchronous execution of tasks by a *heterogeneous* set of mobile agents. Each heterogeneous mobile agent perceives specific stimuli from the environment and perform its assigned task accordingly. Unlike in the previously reported work, these heterogeneous and asynchronous agents are oblivious of both the number of tasks in the sequence as also the sequence in which they need to be executed thereby adding to the complexity of sequencing and synchronizing these executions. The enhanced technique presented in this chapter describes simple yet novel tweak to the technique reported in the previous chapter to handle this extra complexity.

This technique has also been augmented to manage and handle changes in the population of these heterogeneous set of mobile agents so as to suit their current demand i.e. the service requirements of a particular task from the set of interdependent tasks. The population of agents required to complete an activity is generated dynamically without any supervised control using the cloning based stigermergic population control mechanism described in [128]. The population control mechanism is embedded within each mobile agent along with the enhanced technique of synchronizing the execution of a sequence of tasks.

The technique presented in this chapter focuses on the manner in which heterogeneous mobile agents can be made to execute their assigned tasks T_1, T_2,.....,T_m respectively in a desired sequence. This essentially means that an asynchronous mobile agent needs to switch ON the execution of its assigned task (from the set of interdependent tasks) only when the previous task in the sequence is executed at every relevant point in the environment. The technique also ensures that the next task in the sequence is never triggered anywhere in the environment till the previous task execution is completed, while also ensuring distributed behaviour. This synchronizes the execution of a specific task across all points in the environment uniformly.

6.1 The Synchronization Model and Problem Description

The system under study comprises the following elements:-

- A set of m unique tasks $T = \{T_1, T_2, ..., T_m\}$.
- A network W of n nodes.
- A set of m different types of heterogeneous mobile agents $A^T = \{A_1^{T_1}, A_2^{T_2}, ...A_m^{T_m}\}$ with each $A_i^{T_i}$ capable of executing a unique task $T_i \in T$ and also cloning to form a population.

All the n nodes within the network W are passive entities i.e. they do not possess any information regarding the sequence of the tasks to be performed nor do they influence the order of their executions. Both the mobile agents and the nodes in the network are oblivious of the size of the network W and the population of each type of mobile agent $A_i^{T_i}$. Each type of mobile agent $A_i^{T_i}$ is assigned a distinct task T_i such that no two types of mobile agents have similar tasks i.e.

$$\forall A_i^{T_i}, A_j^{T_j} \in A, i \neq j, T_i \neq T_j$$

The mobile agents are capable of carrying programs (code) required to execute their assigned tasks. These agents can perceive various stimuli as and when they visit the nodes in the network W and deposit pieces of information onto the nodes as a consequence of completing the execution of their assigned tasks. A mobile agent $A_i^{T_i}$ can migrate from one node to another using the *conscientious* migration strategy [221] if there exists a link connecting the nodes. Using this strategy, they try to avoid the nodes which they have recently visited in their endeavour to patrol the network. The population of this heterogeneous set of mobile agents varies as per the stigmergic population control mechanism proposed by Godfrey *et al.* [128, 130](discussed in the next section). A mobile agent $A_i^{T_i}$ commences the execution of its assigned task T_i when it has received stimulations greater than a threshold from the environment which in this case is the network of nodes. Thus, an agent $A_i^{T_i}$ services a node by executing a task T_i and hence changes the state of that node. In the present context, servicing a node means when the agent $A_i^{T_i}$ reaches the node, it supplies the code pertaining to its assigned task T_i and execution of the code at that node completes the associated task. This execution changes the state of the node so as to generate stimulations for the next task in the sequence to be executed within that node. The nodes form metaphors for the environment of the entities viz. the mobile agents comprising the swarm. Agents encounter nodes in various states as they migrate within the network. As pointed out in the previous chapter (Chapter 5), the ideal case to commence the execution of a new task in the network is when all the nodes are in a single state. If τ is the number of distinct states that can be observed in the network at any moment of time then its value would be ideally expected to be unity. In practice, the actual value of τ is always 2 since the state of a node changes as soon as an agent services it while there may be other nodes which are yet to be serviced for the same task. Therefore, if the value of $\tau = 2$ is maintained within the network, then it would essentially mean that only one type of agents are executing their respective tasks within the nodes in the network. However, a value of $\tau > 2$ essentially infers an out of sequence execution.

Hence, the main objective of this work is to synchronously execute all the tasks in the set T on all the nodes in the network W by ensuring that the value of $\tau \leq 2$ i.e.

$$\forall T_i \in T, \forall n \in W, \text{ Execute } T_i \text{ at } n$$

$$\text{such that } \tau \leq 2$$

Synchronous execution implies that until all the nodes in the network W have finished their execution of a task T_i, the execution of the next task in the sequence of tasks i.e. T_{i+1}, should not commence. It may be noted that there ought to be no direct communication among the agents or the nodes lest this cause undesirable overheads as discussed earlier. A speed-up in the execution of a task T_i can be achieved if multiple agents capable of executing T_i exist in the network. This can be realized by cloning the agent. However, uncontrolled cloning has the obvious disadvantage of cluttering the entire network. The next section describes a stigmergy based cloning controller [128, 130] to regulate the population of heterogeneous mobile agents. Following which the synchronization technique is discussed.

6.2 Population Control of Heterogeneous Mobile Agents

Godfrey *et al.* [128, 130] have presented a model to control the population of a heterogeneous set of mobile agents in a network using stigmergy. They have used a novel concept termed *Cloning Resource* (C_R) to govern the dynamics of populations of different types of mobile agents within a network. According to their model, each node maintains a *Queue* which hosts all the mobile agents within a node. Each mobile agent carries within itself a record of its current *Lifetime* (L_t) and S along with the set of services or tasks which it intends to deliver to the requesting nodes. Whenever a mobile agent provides its services to a node, it gains a *Reward* (R_d) in return. These *Rewards* alter L_t and C_R of that agent. The increase in the population of mobile agents occurs due to cloning (creating mobile agents of their own kind) while the decrease in population happens due to the termination of mobile agents which have exhausted their *lifetimes*. The *Queue* within each node is composed of four elements:

1. The *De-Queue Controller*: This takes care of handshaking with a neighbouring node to enable the transfer of a mobile agent out of the *Queue*.

2. The *En-Queue Controller*: Its task is complementary to the De-Queue Controller and caters to the request of migrating a mobile agent into the Queue.

3. *Lifetime Monitor & Queue Compactor* : This unit ensures that the lifetimes of each of the mobile agents within the *Queue* are decremented with each instant of time.

4. *Cloning Pressure Register*: This register keeps a record of the current *Cloning Pressure* which has been explained later.

This population control model works based on a reactive mechanism to maintain the population of mobile agents within the networked systems. This mechanism embedded within each mobile agent, senses the number of existing mobile agents in the *Queue* waiting for their turn to migrate to the next node. The number of agents within the *Queue* provides an estimation of the amount of free space for agents to populate within the *Queues* of the nodes in the network. Based on this, an agent decides whether or not to clone. The extent to which an agent can clone depends on the following parameters:

- The *Cloning Resource* available within the mobile agent,

- The *Rewards* gains by servicing the nodes and

- The *Cloning Pressure* which is proportional to the number of vacant slots in the *Queue*.

Cloning Resource, C_R, is charged partly by the rewards and partly by an inherent charging mechanism embedded within the mobile agent. Apart from C_R, the *Lifetime*, L_t, of the mobile agents also increases with the rewards they acquire. The decision as to whether or not a mobile agent, resident within the *Queue* of a node at time t, should clone is made based on the *Cloning Pressure*, $\theta_c(t)$ given by Equation 6.1 [128, 130].

$$\theta_c(t) = \begin{cases} q_{th} - |q(t)| & for\ \theta_c > 0 \\ 0 & otherwise \end{cases} \tag{6.1}$$

where q_{th} is the *Queue Threshold* and determines the maximum number of agents allowed within the *Queue*. $|q(t)|$ is the number of mobile agents present within the *Queue* at time t at a node.

The number of clones, $\eta_c(t)$, that a mobile agent $A_i^{T_i}$ can generate at time t within the *Queue* of a node is determined by Equation 6.2 [128, 130].

$$\eta_c(t) = \left\lfloor \theta_c(t) \frac{C_R^{av}(t)}{C_R^{max}} \right\rfloor \tag{6.2}$$

where $C_R^{av}(t)$ is the available C_R within the agent $A_i^{T_i}$ at time t and C_R^{max} is the maximum value of C_R that a mobile agent can possess.

A mobile agent $A_i^{T_i}$ attempts to make clones if it has enough C_R to supply to the new clones as given by Equation 6.3 [128, 130].

$$C_R^{av}(t+1) = C_R^{av}(t) - \eta_c(t) C_R^{min} \tag{6.3}$$

where C_R^{min} is the minimum value of C_R that needs to be conferred to a clone.

The recharging of the C_R is governed by Equation 6.4 [128, 130].

$$C_R^{av}(t+1) = \begin{cases} C_R^{av}(t) + \alpha_c e^{-1/C_R^{av}(t)} + \alpha_r R(t) & \text{for } C_R^{av}(t) \geq y, \theta_c(t) \leq 1 \\ C_R^{av}(t) + \alpha_c + \alpha_r R(t) & \text{for } C_R^{av}(t) < y, \theta_c(t) < 1 \\ C_R^{av}(t) + \alpha_c e^{(1-1/x)} + \alpha_r R(t) & \text{for } \theta_c(t) > y \text{ where } x = \theta_c(t) \\ C_R^{max} & \text{if } C_R^{av}(t+1) > C_R^{max} \end{cases} \tag{6.4}$$

where $R_d(t)$ is taken to be 1 if the agent services a node; else as 0 at time t. α_c and α_r are constants and $y = C_R^{min}$.

The increment in the *Lifetime*, L_t, of a mobile agent in the presence of a reward is given by Equation 6.5 [128, 130].

$$L_t(t+1) = L_t(t) + C_L R_d(t) \tag{6.5}$$

where, C_L is a non-zero positive constant.

The authors in [128, 130] have successfully portrayed the performance of their population control model for a heterogeneous set of mobile agents on static as well as dynamic networks. They have show-cased the selective rise and fall in the populations of different types of mobile agents based on the demand of their services in the network.

Since the problem being tackled herein makes use of heterogeneous mobile agents to perform different tasks, such a population control framework is best suited. Further, such a framework alleviates the issue of finding the suitable number of agents of a kind to complete the service.

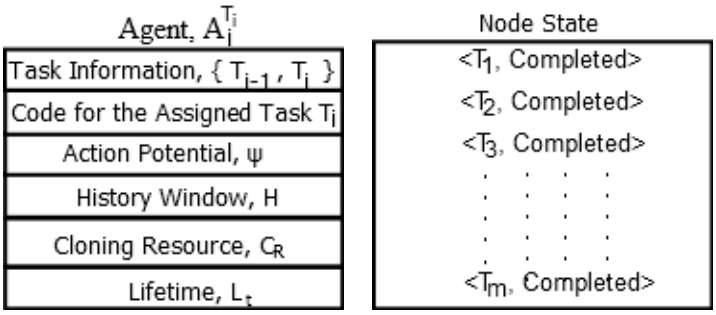


Figure 6.1: (a) Structure of the mobile agent $A_i^{T_i}$ (b) Tuples deposited at a node in the network.

6.3 Synchronizing the Heterogeneous Set of Mobile Agents

As mentioned earlier, the main focus of this work is to stigmergically sense and trigger a task-switch uniformly and synchronously across all nodes constituting the network with no inter-agent communication or centralized control. Thus, if all the nodes in the network are in a state which requires the execution of task T_2 then the concerned mobile agents (including their clones) viz. $A_2^{T_2}$ should be servicing them and migrating within the network so as to eventually complete the execution of this task at all nodes. Nodes that have been serviced for T_2 will transit to a state wherein they require the service of the next task viz. T_3. However, only when all the nodes transit to this new state should the agents assigned to service T_3 i.e. $A_3^{T_3}$, commence their executions at these nodes thereby achieving the required synchronization i.e. $\tau \leq 2$.

6.3.1 Task Distribution and Stimulations

In the proposed approach, the information about the sequence of tasks is partitioned among the different types of mobile agents. Each type of mobile agent possesses the knowledge of the task that it has been assigned to and the information about the task that occurs previous to its assigned task in the task sequence. Let $\{T_1 \rightarrow T_2 \rightarrow T_3\ \}$ in that order be the sequence of tasks assigned to agents $A_1^{T_1}$, $A_2^{T_2}$ and $A_3^{T_3}$ respectively. In this case, the mobile agents $A_1^{T_1}$, $A_2^{T_2}$ and $A_3^{T_3}$ would maintain the information $\{\phi, T_1\}$, $\{T_1, T_2\}$ and $\{T_2, T_3\}$ respectively. ϕ indicates that there is no task prior to the assigned one in the prescribed sequence. An agent carrying such an information would naturally be the first to be triggered. Whenever a mobile agent $A_i^{T_i}$ services a node for a task T_i, it deposits a tuple of the form $< T_i, Completed >$ on to that node. This stigmergic information flags the incoming agents that the task T_i has already been executed at this node. The deposition of such tuples refers to the change in the state of that node and hence provides the stimulations to the mobile agents to trigger the execution of the concerned task. For example, after the execution of task T_1 by the mobile agent $A_1^{T_1}$ on a node, the agent will deposit the tuple $< T_1, Completed >$ on that node. At a later stage, when the task T_2 is executed by the mobile agent $A_2^{T_2}$ on the same node, the latter adds the tuple $< T_2, Completed >$ so that the information available on that node becomes $[< T_1, Completed >, < T_2, Completed >]$. After the successful execution of all the tasks in the sequence of tasks T, the information on each of the nodes will be $< T_i, Completed > \forall i \in T$ as shown in Figure 6.1(b).

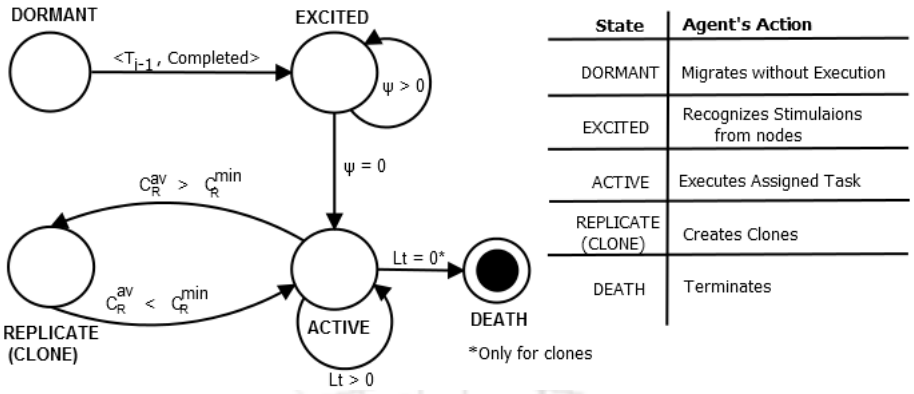


Figure 6.2: State transition diagram of the mobile agent $A_i^{T_i}$ along with its behaviour in each state in the proposed technique.

6.3.2 The Underlying Dynamics

In addition to the *Cloning Resource*, C_R, and *Lifetime*, L_t, as discussed in the previous section, each mobile agent also consists of another parameter called the *Action Potential* (Ψ) which is analogous to the response threshold [284] required to switch ON the execution of the agent's assigned task. Figure 6.1(a) depicts all the parameters carried by a mobile agent. The *Action Potential*, Ψ, of a mobile agent $A_i^{T_i}$ is responsible for the co-ordination of the synchronized executions of different tasks. Ψ for every agent is initialized to a non-zero positive value. Figure 6.2 depicts the various states of a mobile agent along with the associated transitions during the synchronization process. Initially, all types of mobile agents $A_i^{T_i}$ enter the system in the DORMANT state wherein they migrate without executing. An agent $A_i^{T_i}$ makes a transition to the EXCITED state as soon as it finds a node having the tuple $< T_{i-1}, Completed >$ which acts as a positive stimulation. These positive stimulations decrease the value of Ψ of agent $A_i^{T_i}$. On the contrary if this agent (which has already transitioned to the EXCITED state) does not find the tuple $< T_{i-1}, Completed >$ (a negative stimulation) at a node then its value of Ψ increases. The equation governing the value of Ψ is given below.

$$\Psi(t+1) = \begin{cases} \Psi(t) - (1 - e^{-\sigma(t)})\Psi(t), & \text{for Positive Stimulation} \\ \Psi(t) + (1 - e^{-\sigma(t)})\Psi(t), & \text{for Negative Stimulation} \end{cases} \tag{6.6}$$

where the parameter $\sigma(t)$ is discussed later.

The mobile agent transits to the ACTIVE state as and when Ψ equals 0. In the ACTIVE state, the mobile agent $A_i^{T_i}$ starts the execution of its assigned task T_i whenever it migrates to a node that does not have the tuple $< T_i, Completed >$. Ψ is computed and used only when the agent is in the EXCITED state. It ceases to have any effect on the behaviour of the mobile agent beyond the EXCITED state.

Let $B_{T_i}^n \in \{True, False\}$ be a binary variable which takes the value *True* (*False*) if a tuple $< T_i, Completed >$ is present (absent) at a node n. The following logical statements describe the behaviour of a mobile agent $A_i^{T_i}$ at a node $n \in W$:

S1: $\neg B_{T_{i-1}}^n \wedge (\Psi > 0) \rightarrow$ Migration without Execution

S2: $B_{T_{i-1}}^n \wedge \neg B_{T_i}^n \wedge (\Psi > 0) \rightarrow$ Positive Stimulation (Equation 6.6)

S3: $\neg B_{T_{i-1}}^n \wedge (\Psi > 0) \wedge (STATE = EXCITED) \rightarrow$ Negative Stimulation (Equation 6.6)

S4: $\neg B^n_{T_i} \wedge (\Psi = 0) \rightarrow$ Execute Task T_i

Table 6.1: Change in the state of agents $A_1^{T_1}$ to $A_5^{T_5}$ within the network for the sequence $T = \{T_1; T_2; T_3; T_4; T_5\}$

Agent's State vs System Status	$A_1^{T_1}$	$A_2^{T_2}$	$A_3^{T_3}$	$A_4^{T_4}$	$A_5^{T_5}$
Initialization	DORMANT	DORMANT	DORMANT	DORMANT	DORMANT
Stimulation for Task T_1	**EXCITED**	DORMANT	DORMANT	DORMANT	DORMANT
Execution of Task T_1, Stimulation for task T_2	ACTIVE	**EXCITED**	DORMANT	DORMANT	DORMANT
Execution of Task T_2, Stimulation for task T_3	ACTIVE	ACTIVE	**EXCITED**	DORMANT	DORMANT
Execution of Task T_3, Stimulation for task T_4	ACTIVE	ACTIVE	ACTIVE	**EXCITED**	DORMANT
Execution of Task T_4, Stimulation for task T_5	ACTIVE	ACTIVE	ACTIVE	ACTIVE	**EXCITED**
Execution of Task T_5	ACTIVE	ACTIVE	ACTIVE	ACTIVE	ACTIVE

The statement S1 describes the DORMANT state wherein the mobile agents migrate from one node to another in search of the positive stimulations required to switch to the next state. In statements S2 and S3, the agent perceives positive and negative stimulations respectively from the environment which changes the value of its Ψ (Equation 6.6). The agents following S2 and S3 are thus in the EXCITED state. When $\Psi = 0$ the agent transits to the ACTIVE state and behaves as per the statement S4 and starts to execute its assigned task. For illustration, consider the task sequence $T = \{T_1; T_2; T_3; T_4; T_5\}$ and set of agents $A = \{A_1^{T_1}, A_2^{T_2}, A_3^{T_3}, A_4^{T_4}, A_5^{T_5}\}$. Initially all the agents in A will be in DORMANT state within the network. Since the agent $A_1^{T_1}$ is assigned the first task in the task-sequence T, it will immediately switch to the EXCITED state due to the presence ϕ within its task information. While the agent $A_1^{T_1}$ is in the EXCITED state (statement S2 and S3) all other agents in A will continue to remain in the DORMANT state. Eventually the agent $A_1^{T_1}$ will transit to the ACTIVE state when its Ψ becomes 0 and start the execution of the task T_1 in the concerned nodes. Such executions would result in the deposition of the tuple $< T_1, Completed >$ on the concerned nodes in the network. Only on perceiving this tuple, will $A_2^{T_2}$ transit to the EXCITED state. The agent $A_2^{T_2}$ will continue to remain in the EXCITED state until its Ψ value becomes zero. As long as $A_2^{T_2}$ is in the EXCITED state, $A_3^{T_3}$, $A_4^{T_4}$ and $A_5^{T_5}$ will still remain in the DORMANT state. This chain of actions will continue until all the agents in A transit

to the ACTIVE state as depicted in Table 6.1. As can be observed, the agents in the task sequence create stimulations for the succeeding agents. These stimulations are distributed temporally across the network making the agents execute their assigned tasks only when their turn comes up.

The parameter $\sigma(t)$ which regulates the amount of increment or decrement in Ψ is calculated in the following manner. Let $\xi(.)$ be a function that returns the time of an event at a node $n \in W$. Let us define a parameter Δ using $\xi(.)$ as,

$$\Delta := \xi(A_i^{T_i} \text{arriving at n}) - \xi(< T_{i-1}, Completed > \text{deposited at n}) \tag{6.7}$$

Hence, Δ is essentially the difference between the time when the status of a node was updated to $< T_{i-1}, Completed >$ by the agent $A_{i-1}^{T_i-1}$ and when the mobile agent $A_i^{T_i}$ arrived at that node. Every mobile agent maintains a record of its own task-specific Δ's in a *History Window* (H) of length ω $(\omega \geq 0)$ within itself. Whenever, a mobile agent $A_i^{T_i}$ finds a node having $< T_{i-1}, Completed >$ (i.e. $B^n_{T_{i-1}} = True$), it finds the value of Δ using Equation 6.7 at that node. This new value of Δ is added to the H of $A_i^{T_i}$, if it is greater than the last recorded value i.e.

$$H_{new} \leftarrow H_{old} \cup \{\Delta^n\}, \text{ if } \Delta^n > \Delta_{\omega-1}$$

where Δ^n is the value of Δ at node n and $\Delta_{\omega-1}$ is the last recorded value of Δ in H of $A_i^{T_i}$.

Since the different types of mobile agents execute their assigned tasks in different time-slots (temporally distributed), the values of Δ within H are normalized between 0 and 1 using Equation 6.8.

$$Norm(\Delta_i) = \frac{\Delta_i - \Delta_0}{\Delta_{\omega-1} - \Delta_0}, \ \forall i \in [0, \omega - 1] \tag{6.8}$$

where Δ_i denotes the i^{th} value in H.

The value of $\sigma(t)$ is calculated using Equation 6.9 as the standard deviation of $Norm(\Delta_i)$, $\forall i \in H$. Thus, $\sigma(t)$ provides the extent of dispersion of the stimulations (requirement of a service in the network) that a mobile agent observes at time t.

$$\sigma(t) = \kappa \sqrt{\frac{\sum_{i=0}^{\omega-1} \{Norm(\Delta_i) - \overline{\mu}\}^2}{\omega}} \tag{6.9}$$

where,

$$\overline{\mu} = \frac{\sum_{i=0}^{\omega-1} Norm(\Delta_i)}{\omega} \tag{6.10}$$

κ is a non-zero positive constant and ω, as already mentioned, is the length of the *History Window*, H.

In the ACTIVE state, whenever a mobile agent $A_i^{T_i}$ executes its assigned task T_i on a node it receives a reward R_d in return. This reward recharges the *Cloning Resource*, C_R (as discussed in Section 6.2, Equation 6.4) of $A_i^{T_i}$. When the C_R of an agent $A_i^{T_i}$ crosses C_R^{min} so that it can confer enough Cloning Resource to the clone it transits to the REPLICATE state to create the clones as per Equation 6.3. The clone and the parent then transit back to the ACTIVE state as shown in Figure 6.2. Due to cloning the population of the mobile agent $A_i^{T_i}$ increases. Hence, the mobile agents whose assigned tasks are currently required to be executed on all the nodes in the network rise in population whereas the populations of the clones of the other type of agents decline due to their diminishing *Lifetimes* causing transitions from ACTIVE state to DEATH state. In the proposed technique, the system starts with a set of parent mobile agents assigned distinct tasks in T with infinite lifetimes. As the system progresses, the cloning controller ensures that the population of each type of mobile agent varies based on their demand within the network.

6.3.3 Self-Healing with Heterogeneous Agents

In real world scenarios, it may happen that a task completed by an agent at a node(s) may crash or breakdown due to some environmental effects. Consider the case of a task (T_i) of connecting two wires to make a bridge at a node. When the task is executed the change in environment within the node (i.e. $< T_i, Completed >$ is asserted at the node) allows for flow of information across the bridge. A failure could result at this node if for some reason this connection breaks (i.e. $< T_i, Completed >$ is deleted). Such problems could occur in an out-of-order manner across the network making τ to be greater than 2. In such a case, the concerned agent(s) may need to re-execute the task(s) at the respective nodes while the other nodes are being serviced for a different task as per the sequence T. The proposed technique exhibits such a *Self-Healing* mechanism inherently.

As mentioned earlier in Subsection 6.3.1, the tuple deposited at a node after the completion of a task acts as the stigmergic information at that node which indicates the completion of that task within. When a task at a node fails then the concerned tuple is deleted from that node indicating the requirement of a the task to be re-executed. Such failures may be local to a node or may involve a set of nodes. This causes the agent assigned the concerned task to behave as per statement S4 (see Subsection 6.3.2) resulting in the re-execution of this task at the failed node. This agent thus receives a reward which in turn increases its cloning resource. These agents may thus transit to the REPLICATE state and clone. This quickens re-executions at the failed nodes ensuring faster recovery. Though such re-executions introduce asynchronization ($\tau > 2$), the *Self-Healing* mechanism enhances in the fault-tolerance capability and robustness of the system.

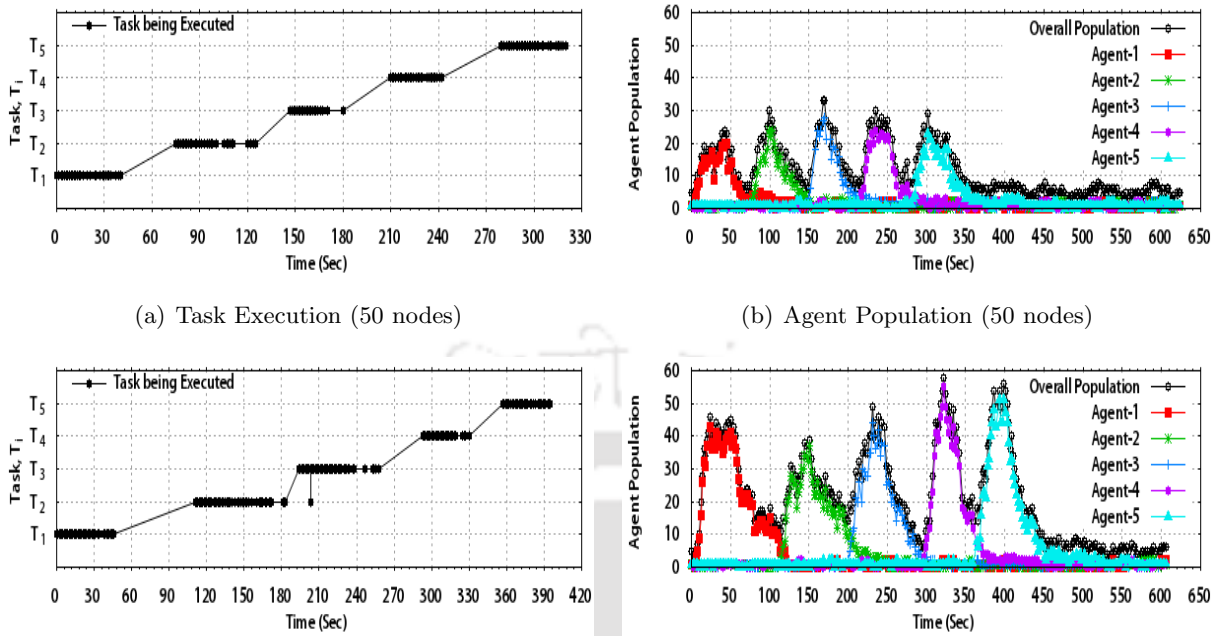
6.4 Emulating the Synchronized Task Execution

In order to test the efficacy of the proposed technique, the Typhon mobile agent framework [132] was used. Ten different physical computer systems, connected within a LAN, were used in the experimentation out of which nine were used to realize an emulated mesh overlay network. The remaining one served to asynchronously log the data generated at various nodes during run-time and as such acted as a non-participating node in the network.

6.5 Results and Discussions

The experiments were performed over two emulated mesh overlay networks having 50 and 100 nodes. Five conscientious mobile agents (parent mobile agents) each carrying a different task out of a sequence of five tasks were used in the experiments to test the efficacy of the technique. Following parameter values were used in the experiment: $\Psi(0) = 100, \kappa = 1, q_{th} = 4, C_R^{min} = 10, C_R^{max} = 100, \alpha_c = 0.1, \alpha_r = 15, C_L = 7, \omega = 5$.

All the parent mobile agents were given infinite lifetime while the initial lifetime conferred on a new clone was set to 10. Self-healing was emulated by forcing at set of 20 nodes chosen arbitrary at random in the network to crash the task T_2 while the service for the task T_4 was in progress. The mobile agents were initially placed in different nodes (chosen randomly) and were triggered to commence their patrolling within the network. Nodes and agents continuously sent information on the tasks being executed, the rewards, stimulations, Clonal Resource, etc., asynchronously to the non-participating node. It may be noted that the non-participating node was not a node in any of the networks used in the experiment nor did it influence the agents or participating nodes in any of their processes or actions.



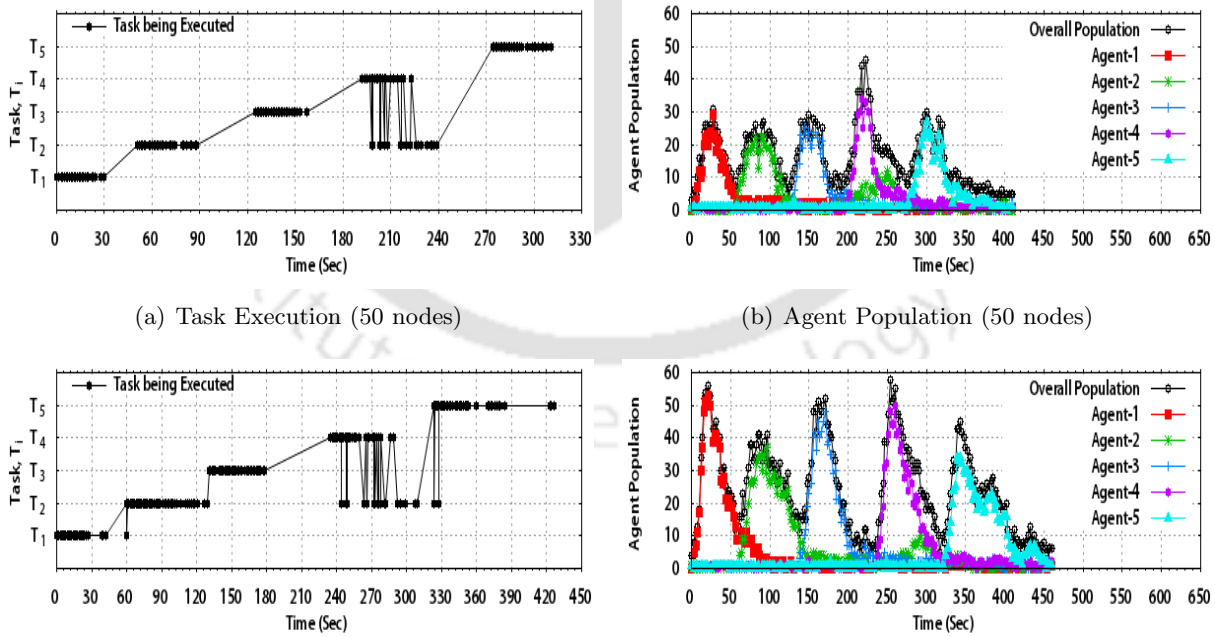
(a) Task Execution (50 nodes)

(b) Agent Population (50 nodes)

(c) Task Execution (100 nodes)

(d) Agent Population (100 nodes)

Figure 6.3: Executions of Tasks- T_1 to T_5 in (a) for 50-node and (c) 100-node mesh networks along with the corresponding change in the populations of agents *Agent-1* through *Agent-5* in (b) and (d) respectively.



(a) Task Execution (50 nodes)

(b) Agent Population (50 nodes)

(c) Task Execution (100 nodes)

(d) Agent Population (100 nodes)

Figure 6.4: Self-healing with failure of task T_2 at 20 nodes randomly selected in the network when the execution of task T_4 was in progress for the 50- and 100-node networks.

The graphs in Figures 6.3(a) and (c) depict the execution of five different tasks by five different types of mobile agents in the network of 50 and 100 nodes respectively using the proposed synchronization technique. As can be observed from the graph, the execution of each type of task (T_1 through T_5) occurs almost without any overlaps (there is only one instance of overlap of execution of task T_2 with task T_3 in the 100-node network) with other tasks in the sequence. Further, the time for execution of each type of task (T_i) on all the nodes for both the networks varies in between 30 to 60 seconds. Some gaps can be noted in between the executions of consecutive tasks. These gaps, refereed as the *idle periods* are generated as a result of the time taken by the mobile agents to acquire the level of stimulation to commence the execution of their assigned tasks.

The graphs in Figures 6.3(b) and (d) depict the change in the populations of different types of agents, *Agent-1* through *Agent-5* assigned with tasks T_1 through T_5, along with the change in the overall population of all the mobile agents in the network. One can easily observe the successive peaks in the graph which illustrates the rise in the population of agents which were currently servicing the nodes for their assigned tasks. This rise in agent population accounts for the parallel and concurrent executions of a task at different nodes in the network. It may be observed that the rise in population of agents is more in the 100-node network (around 60) than that of in the 50-node network (around 30) which shows the on-demand generation of agents.

The graphs in Figure 6.4 depict the order of execution of tasks and the change in population of agents during *self-healing*. As can be observed in Figures 6.4(a) and (c), when the mobile agent assigned to service task T_2, $A_2^{T_2}$, finds a node where the task T_2 has crashed, it automatically starts its service of the task T_2 and hence gain rewards which charges its C_R. This allows the mobile agent to clone and expand its search and service for the crashed task at other nodes in the network. A small rise in the population of *Agent-2* can be observed in Figures 6.4(b) and (d) at around 250^{th} and 300^{th} second respectively due to such cloning activities. Apparently, the crashed tasks does not affect the order of execution of tasks of other agents in the network.

Hence the results portray that the proposed technique is capable of streamlining the execution of different tasks for a heterogeneous set of mobile agents. Further the *self-healing* feature ensures that the failure of a task from the sequence T does not adversely affect the progression of executions. The equivalent range of *idle-times* in different networks shows the adaptiveness of the proposed technique in different environments.

6.6 Chapter Summary

This chapter describes a technique for near synchronous execution of a sequence of tasks using a heterogeneous set of mobile agents wherein each agent is capable of performing a distinct task from the sequence of tasks. Having been augmented by a cloning controller, it is capable of regulating the population of each type of mobile agent. These populations grow and diminish based on the current demand of the specific agent within the network. Thus, using stigmergy and discrete stimulations coupled with the concept of a cloning resource within each mobile agent, the technique presented herein assesses the situation of the environment and achieves synchronized execution with selective rise and fall in the population of heterogeneous mobile agents. Stigmergic communication emerges when the agents sense the number of serviced and unserviced nodes and orient themselves accordingly to serve for global synchronization across the network. Emulation results gathered on networks of varying sizes have also fortified the applicability of the presented technique in real systems. In addition, the proposed technique also inherently exhibits *self-healing* making the overall system fault-tolerant.

The technique described for the synchronized execution of a sequence of tasks makes an inher-

ent assumption that the nodes (i.e. robots) are already available in the environment to carry out the task executions. However, in real systems comprising large, distributed and heterogeneous sets of networked robots, the availability of the robots having a set of desired resources and capabilities needed to carry out the execution of tasks must be ensured *a priori* within the environment. This calls for a mechanism to acquire the required number of unoccupied or idling robots in the environment which would be needed to commence the execution of the set of tasks. Further, there can be more than one instance of sets of tasks to be carried out in different parts of the environment. Hence the mechanism should be robust enough to resolve contentions that may arise when several agents try to reserve a common set of robots within their environment, so as to execute the respective tasks allocated to them. Contention could crop up when many agents require the services of the same robot(s) at the same time. The next chapter describes one such mechanism using mobile agents to dynamically acquire a required set of robots within a distributed robotic network. The contentions among multiple mobile agents vying for the identical resources (robots with the desired skills) are resolved using asynchronous and voluntary back-offs.

"The true sign of intelligence is not knowledge but imagination."

Albert Einstein (1879 – 1955)
German-born theoretical physicist

7

A Strategy for Multiple Task Allocations

Efficient arbitration of resources and allocation of jobs is always at the core of any distributed system. These functions can become difficult to model and realize when the entities within the system are dynamic in nature due to the ensuing non-determinism and uncertainties. The allocation of different sets of tasks within a network of heterogeneous robots based on their capabilities and available resources is an example of one such function within a dynamic and distributed system. Investigating novel mechanisms for the allocation of tasks within MRSs form a prominent area of research. Task allocation within MRSs has been mostly tackled using either a self-organizing approach or by a market-based auctioning strategy [305]. These approaches scarcely address the issue of competing instances of concurrent task allocations occurring within the system wherein all allocations vie for the same set or subsets of robots or associated resources.

This chapter presents a novel strategy for the allocation of task(s) within a large heterogeneous MRS in the presence of competing task allocation instances. The problem of contentions among the concurrent competing instances of task allocations is tackled using voluntary and adaptive back-offs. The proposed strategy of task allocation is completely distributed and is oblivious of the number of robots and their locations in the environment, the number of tasks to be allocated, the number of competing instances of task allocations available in the system and the stability of the communication network among the robots within the MRS. The proposed strategy also provides for better scalability and robustness in the allocation of task(s) in an MRS.

7.1 Motivation

Coordination of activities in an MRS is a challenging task since it involves numerous sequential and parallel asynchronous processes running concurrently. In particular, allocation of tasks or a set of tasks to a set of robots within an MRS has been a notable area of interest for researchers. Various techniques and methodologies have been proposed in the literature [305] to facilitate task allocation in MRS while also taking in consideration the payoffs of the individual robots, the global cost and utility, group and team dynamics, etc. The task allocation approaches used in MRSs can be broadly classified into two categories viz. Self-organizing or emergent approaches and Intentional or explicit approaches [305].

In the self-organizing approach, the behaviours of individual robots are evolved in a manner so that the global objective is achieved. These approaches are usually inspired from biology, more often by mimicking techniques learnt from insect societies such as ant colonies and bee swarms. The tasks are dynamically allocated to the individual robots based on their perceived stimuli from

the environment. The complexity involved in the design and implementation of self-organized approaches of task allocation is in the embedding of task-based stimuli within the environment along with the stimulus sensing capabilities of the individual robot. Further, these approaches assume a homogenous team of robots which are coherent in their capabilities and perception. In addition, there is less control over the MRS in terms of optimal allocations of tasks as the global coordination of robots emerges from the individual actions in these fully distributed approaches.

On the other hand, the intentional or explicit approaches use market-based auction methods for allocating tasks to an MRS. In these approaches, initially an auction for a task or set of tasks is announced by an auctioneer and the individual robots bid based on their respective capabilities, current state and estimated costs for the task(s). The auctioneer awards the task(s) to the robots after assessing their bids and utilities. Hence, unlike the self-organizing approach, the interdependencies among the tasks can be easily taken care of. Further, the coordination among the robot teams can be planned using optimal task allocation strategies. Future tasks can also be allocated to the robots based on their current state and available resources. Dias et al. [306] point out that market based approaches hybridize the centralized and distributed mechanisms depending on the application under consideration. These approaches are thus best suited for multi-robot systems used in tightly monitored and controlled environments such as an assembly line automation or in an autonomous warehouse or port.

Most task allocation methods proposed in the literature [305] always assume a single instance for the allocation of task(s) under consideration or in other words have just one auction or task allocation at any moment of time. However, in a large heterogeneous MRS, there could be initiations of multiple instances of task allocation processes running and competing concurrently; some or all of them vying to acquire the same sets of robots. Solving such contentious scenarios of task(s) allocations in an MRS is very critical for their deployment in real-world applications. Such concurrent task allocation scenarios have hardly been addressed by the MRS researchers. A solution to this calls for new strategies that can facilitate the allocation of tasks in a large heterogeneous MRS wherein concurrent, asynchronous and competing instances of task allocation processes emanate. Further, the robots in an MRS are always assumed to be embodied physical agents that can evolve multi-agent strategies for their coordination and control. The use of mobile agents [90] however has never been explored to alleviate the problem of such task(s) allocations in an MRS.

The next section discusses the relevant literature of market based task allocation mechanisms in MRSs. Succeeding sections present the proposed strategy of task allocation in a large heterogeneous MRS along with its inherent dynamics.

7.2 Background

Dias et al. [306] provide a survey on the market-based multi-robot coordination approaches and analyses their performance in various application domains. According to them, market based approach hybridize the centralized and distributed methodologies for multi-robot coordination. They discuss the suitability of the market based approaches in terms of heterogeneous robot teams and their scalability. The market based coordination mechanisms are conceptually derived from the Contract Net Protocol (CNP) [307] which was designed as a task-sharing protocol in multi-agent systems. Botelho and Alami [308] have proposed a decentralized multi-robot task decomposition and coordination mechanism (M+) which has been adapted from the CNP. M+ has three layers: a task allocation layer, a fault-tolerance layer and a task executor layer. In this scheme, all the robots receive the same mission (set of tasks). Each robot decomposes the tasks in the set of actions. In case of any failure on part of the robot or the action being performed, the robots contact one another

and try to find cooperative schemes for the execution of tasks and thus overcome the problem. This scheme assumes that any robot is capable of executing a task on its own and only one robot is allowed to issue a first-offer broadcast for a given task. In [309], the authors have presented an auction-based task allocation mechanism nicknamed Murdoch based on a publish/subscribe communication model. They use a greedy algorithm to allocate tasks and issue time-limited contract to the robots. Murdoch has been shown to be applied successfully in heterogeneous robot teams in different applications. However, the presence of multiple auctioneers targeting similar resources has not been handled therein. Zlot and Stentz [310] have used task trees for auctioning of time-extended task allocations [305]. These task trees represent a possible decomposition of the complex interdependent tasks. The agents auction and bid on task trees instead of simple tasks. The bidding can be done on a branch or some specific node of the task trees based on the current state and resources of the robots. The auctioneer announces the winner(s) after receiving the bids based on the lowest cost solutions. Although their scheme tries to globally optimize task allocations in a time-extended manner, the authors assume that only one auctioneer manages the auction process for a particular task tree.

In [311], the authors use a single round auction scheme to discover and form robot teams. The coordination among the robots are modelled based on the constraints imposed over the shared resources. They enabled robots to perform multiple tasks concurrently by bringing together the requirements of tasks and subsequent constraints in their execution. However, they have assumed that only a single auctioneer or task allocator agent enters tasks into the environment within a minimum time interval. Zhang et al. [312], portray a stochastic algorithm for centralized and distributed task allocation using Markov-chain Monte Carlo methods. The focus of their algorithm is to produce globally optimal allocations of the robots. For distributed task allocation, they have applied the centralized algorithm in spatially segregated regions wherein each robot serves as a task allocator in a fixed rotation pattern. The authors do mention that in future their distributed algorithm needs be altered to cater to multiple auctioneers working in parallel with a mechanism for conflict resolution and eventually be implemented using real robots. Keshmiri and Payandeh [313] propose dynamic task allocation strategy with elements of centralized and distributed mechanisms. While the centralized unit coordinates the functioning of the robotic agents through task sub-grouping and agent cost permutation processes, the decision making and execution of the allocated tasks are carried out in a distributed manner by the individual robots. They assume synchronous communication among the robots without any delays, an assumption which is difficult to hold in the real world. Their method is suited when a single mission can be divided into several sub-goals wherein the sub-goals may change their locations in the environment overtime. The coordination strategy uses the votes of the robots for a centralized mediator to allocate the available subgroups.

In [314], the authors assume that each task can be completed by a single robot, however, they use task sharing to reduce task completion times. Each robot is made to auction its own tasks and clear its own auctions. The auction mechanism is based on a combination of sequential single-item auctions and repeated parallel single item auctions. The auctions are repeated after a fixed interval of time. The robots bid for tasks based on their current commitment and location. If a robot fails, it auctions all its remaining tasks. Further, the robots do the reordering of tasks in the sequence of their execution. Here, authors assume a homogeneous set of robots with time-extended task allocations. In [315], the authors provide an ad hoc network based task allocation (ANTA) while considering resource aware cost generation. The ANTA mechanism constructs a minimal spanning tree network among the robots and determines the robot with the lowest cost bid to be allocated for the task through multi-hop communications. The task allocation mechanism is not controlled by any task allocator since all the robots receive bid messages from other robots and forward the same to others in a decentralized manner similar to the way robots exchange eligibility information

in [316]. The authors empirically evaluate their strategy using simulation experiments based on foraging swarms. Although, the authors have discussed the creation of minimum spanning trees using broadcast tree communication, they have not clarified how the tree is managed when the robots move in the environment. Further, this mechanism focuses on the assignment of a single task to a single robot and does not consider the allocation of a coalition of robots for tasks. The case of multiple instances of task allocations competing for similar resources is also not considered.

Auction based multi-robot task allocation mechanisms always use the auctioneer as a central entity to announce auctions, collect bids and allocate tasks to the winning robots. The whole network of robots is assumed to be static while the task negotiations are in progress, a situation which is hard to enforce in real world. The concept of exploiting the features of a mobile agent to carry out negotiations with the robots in the network and eventually allocating them for a set of tasks has never been attempted. Such an agent can act as a non-static negotiator that can knit through the dynamic network of mobile robots. Hence, the whole task allocation process can be realized in a truly distributed and decentralized manner. Further, there seem to be hardly any effort to address the issue of multiple concurrent and competing task allocation instances that vie for the same or subsets of robots. This chapter provides an approach that uses mobile agents to facilitate concurrent and competing task allocation instances in the MRS domain. The succeeding sections elaborate the mechanism for the same.

7.3 System Model

This section provides the basic definitions and concepts along with the building blocks of the system under consideration for the proposed task allocation strategy. The following characterization exists in the system under consideration:

1. The system consists of a network of heterogeneous mobile robots $R = \{r_i | i \in I\}$, where I is a set of positive Integer. Heterogeneous robots essentially mean that the robots are of different types and thus may possess different capabilities. Further, a robot only executes the tasks that it is capable of.

2. There may be same or similar robots in R which can be segregated based on their capabilities. Hence there exists overlapping subgroups of spatially separated robots.

3. A robot r_i can only be assigned a single task at a time.

4. The robots in R are oblivious of any information about the location or capabilities of other robots or the subgroups in the network. They are also not aware of their total number.

5. The robots in R possess communication capabilities and hence can make or break connections with other robots as per the requirement.

6. The robots in R are equipped with an agent platform which supports all the functionalities of mobile agents and provides these agents with an execution environment.

7. The mobile agents can be spawned and released into the network of robots R from other auxiliary networked devices (such as a PC or a laptop connected to R) by a system user. There can be one or more such devices available and distributed across the network. These devices do not control any other process except for releasing the agents.

8. The robots in R can also spawn the mobile agent into their network, if required.

9. The task(s) to be executed using the robots in R may require one or more robots. However, the execution mechanism of the task(s) or the decomposition of task(s) into subtasks is assumed to be either provided by the user or embedded within the robots.

10. Each task T_i to be allocated to the robots contains the associated task descriptors which enlist the dependencies and requirements for executing that task (such as the state or location of the robot having capability, for instance, to lift a bottle) along with other essential parameters. Such descriptors can be derived from ontologies referred to streamline the interaction among the mobile agent and robots.

11. New tasks can appear at any moment when the system is running.

12. The mobile agents use conscientious migration strategy [221] to navigate into the network of robots R. This migration strategy ensures that the agents evenly distribute their frequency of visits amongst all robots in R.

In the next section, we present the proposed strategy of task allocation for the robots in R using mobile agents.

7.4 The Proposed Task Allocation Mechanism

The proposed approach regulates the allocation of tasks when multiple instances of this allocation process are present in the system. For illustration, let us assume a network of 100 robots out of which 10 robots possess the resources required to execute say a *pushing* task, T_1. Similarly there can be another 20 robots equipped with resources to execute both tasks T_2 and T_3. It may be noted that the proposed approach makes no assumption regarding the diversity in the capabilities of robots in terms of their resources and skills. Since the network of robots is fully distributed, several task allocation requests (requiring more than one robot having similar or dissimilar capabilities) can arrive asynchronously at any moment of time. Further, there could be requests whose resource requirements could overlap causing contention. Suppose at some time there are three concurrent allocation instances or requests for a *pushing* task viz. T_1, to be executed, each requiring the services of 7, 5 and 9 robots. Evidently, all the three instances of task allocation cannot be satisfied simultaneously since the total number of robots that have the necessary resources to execute *pushing* task T_1 is only 10. Hence, each instance of task allocation needs to reserve the robots in a particular order one after the other so that these allocations are satisfied. In the proposed approach, such an order of task allocations emerges autonomously as a result of the resolution of contentions for the resources required.

Within the proposed approach, each robot is equipped with an agent framework to support all the functionalities of mobile agents. For each instance of the task allocation, a dedicated mobile agent is spawned which has the details of the type and number of robots along with the resources that the robots need to possess encapsulated within it. Thus each such agent released into the system represents an instance of an autonomous task allocation request. The scenario now becomes synonymous to a multiple task auctioneering one as predicted in [312]. These agents then migrate into the network of robots to find those free or unoccupied robots that fulfil all the requirements. These agents interact locally with these robots and negotiate whether or not they can be reserved for the required task. The negotiation involves the estimated cost of performing the task in question depending upon the current state of the robot (such as the charge in its batteries or its location) and the resources required for executing the task (such as a sensor to measure the depth of objects from a distance, etc.). If the negotiation succeeds, the agent reserves the

robot for the task allocation and continues its migration within the network in search of other such robots. This process continues till all the required robots are reserved. Once all the required robots are reserved, the mobile agent initiates the execution of the allocated task(s) using these reserved robots. As the robots make progress in executing the specified task, the mobile agent monitors their status and records any failures or disruptions. The agent de-allocates these robots after the execution of the specified task(s).

Since multiple mobile agents may vie for similar robots concurrently, it is highly possible that no agent gets the required number of robots initially. Under such conditions an agent voluntarily backs off from the search so as to create asynchrony, by remaining dormant for an interval of time. During this period it remains within the last robot visited and idles. However, just before a back-off, the mobile agent spawns child agents each containing the information of the already reserved robots (if any). The spawned child agents migrate into the network of robots and ensure the de-allocation of the specified robots from the task allocation process so that they be available to the other active mobile agents. Spawned child agents are programmed to die (*apoptosis*) after the completion of their de-allocation target(s).

7.4.1 Inherent Dynamics

There are two crucial aspects that require attention in the proposed scheme of task allocation using mobile agents. The first aspect is the extent to which the agents need to migrate within the network while the second involves the manner in which the computing of the back-off interval needs to be performed to ensure the required asynchrony. In this section, we describe the dynamics that regulate these two aspects within the proposed approach.

Agent Migration

The mobile agents carry within themselves a parameter termed the *Mobility Resource* (χ). This resource acts as a fuel for the agent to migrate into the network and determines the extent to which it penetrates into the network. Whenever an agent reserves a robot, the value of χ of that agent is reinforced by a reward. On the contrary if an agent finds a desirable robot which is already reserved, the agent receives a penalty which decrements the value of χ. In addition, the value of χ also decays inherently as the agent migrates into the network of robots. An agent goes for a back-off when its χ becomes almost equal to 0. It may be noted that since the allocation and de-allocation of robots is dynamic and asynchronous, it may happen that some unallocated robots become allocated while some become available for allocation. The increase in number of rewards constituting an increase in χ motivates the agent to explore the network for the remaining resources. In the absence of such a rewarding mechanism it may happen that an agent explores and reserves the majority of the robots required only to find that it cannot penetrate the network further due to decay of χ. Rewards thus avoid such a situation by boosting the value of χ for every robot allocated by an agent.

The opposite happens when the number of penalties increases thereby forcing the agent to go for an early back-off and allowing other such agents to allocate their remaining robots. Penalties also restrict futile migrations on part of the agents and avoid any possible cluttering within the network.

The following equations contribute to the change in χ.

Reward:

$$\chi(n+1) = \chi(n)\left\{1 + \left(\mu^r \frac{N^R - N^Q}{N^R}\right)\right\} \tag{7.1}$$

Penalty:

$$\chi(n+1) = \chi(n)\left\{1 + \left(\mu^p \frac{\Lambda}{\Lambda + (N^R - N^Q)}\right)\right\} \tag{7.2}$$

Decay:

$$\chi(n+1) = \begin{cases} \chi(n)\left\{1 - \left(\frac{\tan(\frac{\pi}{4+U_R-V_R})}{C_d}\right)\right\}, & U_R - V_R \geq 0 \\ \chi(n)\left\{1 - \frac{1}{C_d}\right\}, & otherwise \end{cases} \tag{7.3}$$

where N^R is the total number of robots required by the agent and N^Q is the number of robots reserved by the agent. Λ denotes the number of times the agent received penalties while C_d is a non-zero positive constant. μ^r and μ^p are the frequencies at which the rewards and penalties are received respectively. μ^r (or μ^p) is calculated as the ratio of the number of rewards (or penalties) to the number of hops travelled by the agent. As can be seen, the values of reward and penalty are directly proportional to their respective frequencies (Equations 7.1 and 7.2) and thus form a heuristic that estimates the number of unallocated robots within the network. Further, the values of reward and penalty are scaled based on the current requirement of robots $(N^R - N^Q)$ by the agent. U_R is the number of distinct robots discovered while V_R is the total number of robots visited by the agent.

In addition, whenever an agent commences its migration into the network after completion of a back-off, the value of $\chi(0)$ is calculated by solving the recurrence relation from the decay equation as:

$$\chi(0) = \frac{\alpha_k C_d^{U_R}}{\prod_{r=0}^{U_R}[C_d - \tan(\frac{\pi}{4+U_R-r})]}, \alpha_k \sim 0 \tag{7.4}$$

As can be observed, the value of $\chi(0)$ is always calculated based on the number of distinct robots that the agent has encountered so far within the network. The value of $\chi(0)$ is indicative of the current size of the network with higher values signifying a larger network. It may be noted that the decay of χ (Equation 7.3) also adapts based on the number of unique nodes discovered by the agent. This decay component increases as $(U_R - V_R)$ tends to 0 and becomes constant beyond. Thus if U_R is high, the decay of resource remains slow per hop. This allows the agent to migrate and explore deeper into the network.

Contention Resolution

As mentioned earlier, voluntary back-offs are used to resolve contentions among the agents. The back-off mechanism used herein is inspired from the binary back-off mechanism used in CSMA/CD [317]. The mechanism CSMA/CD is used by the computers within a LAN to send data in distributed manner over the common bus. This back-off mechanism is very crucial to the success of the task allocation process.

Imagine a case when there are n agents and $(n-1)$ agents back-off. This means that the $n-1$ agents become dormant for a particular interval of time (back-off interval) leaving the remaining active agent to explore the network for resources. However, if the back-off interval of $n-1$ agents is lower than the time it takes for the single active agent to explore the network, then there will be contentions. In this case, such contentions would result in repeated back-offs by all the agents while none of them are able to reserve their required resources.

On the other hand, a large back-off interval would lead to unnecessary delays as the $n-1$ agents would still be dormant even the single active agent has finished acquiring its resources. This

calls for embedding an adaptive and egocentric mechanism for computing the back-off interval in each agent based on the way it perceives the network. This essentially means that the back-off interval is based on the part of network covered by the agents since the complete knowledge about the whole network is not available.

Taking this into consideration the back-off interval (BI^i) is calculated by each agent, i, individually using the following equation:

$$BI^i = \begin{cases} \Theta + \Phi(2^{\beta_i}\Theta), & 2^{\beta_i}\Theta \leq BI_{max} \\ \Theta + \Phi(BI_{max}), & otherwise \end{cases} \tag{7.5}$$

$$\Theta = U_R^i \Delta^R \tag{7.6}$$

where U_R^i is the number of distinct robots discovered by the agent i during its migration in the network. Δ^R is the current hop-time at the robotic node R wherein the agent i decides to back-off (i.e. $\chi \sim 0$). The hop-time denotes the average time required by an agent to migrate from one robot to another within the network. These hop-times are maintained by the agent framework hosted within each robot. β_i refers to the number of back-offs that the agent i has experienced so far. $\Phi(X)$ is a function which returns a random value in the range from 0 to X. BI_{max} is a non-zero positive constant that signifies the maximum back-off interval.

The first term of Equation 7.5 ensures that the back-off lasts for at least the time required for the spawned child agents to de-allocate the robots reserved by the agent i. Since the values of U_R and V_R almost become coherent after some certain number of back-offs, the random function $\Phi(.)$ makes sure that the BI values remain diverse across the agents. The value of BI_{max} needs to be provided by the user and must be selected based on parameters such as the maximum allowable execution time for a task, the maximum waiting time to commence the execution of a task, the communication bandwidth, etc.

7.5 Emulation Setup

The proposed approach was emulated using *Typhon* mobile agent framework [132] over a LAN. In the emulation experiments, an instance of *Typhon* on a computer is modelled as a robot in the network. Several such instances were used to emulate the complete MRS. To emulate a dynamic and mobile ad-hoc network of robots, the links among the *Typhon* based robotic nodes within the emulated MRS were altered by the individual emulated robots using a variant of the Erdős-Rényi $G(n, p)$ model [252]. Hence, while the mobile agents were made to migrate in a real dynamic network environment, the robots were emulated as computer processes in the emulation experiment. Two other auxiliary *Typhon* instances were connected to the emulated MRS - one for releasing mobile agents into the network and the other to act as a Data Logging Server which can log the various events (such as the allocation and de-allocations of robots, back-offs, task executions, etc.) occurring within the MRS. These logs were then used to plot the graphs depicted in the results. It may be noted that these auxiliary instances did not influence or affect any process of the system or the proposed decentralized and distributed multiple task allocation approach in any manner. Each experiment was performed five times to account for any possible stochastic effects.

To evaluate the performance of the proposed multiple task allocation approach, two other prominent auction based task allocation techniques via. Murdoch [309] and ANTA [315] were also implemented within the *Typhon* framework. Since these approaches cater to a single instance of task allocation at any instant of time, for comparison they were augmented with a back-off mechanism to suit multiple competing instances of task allocations.

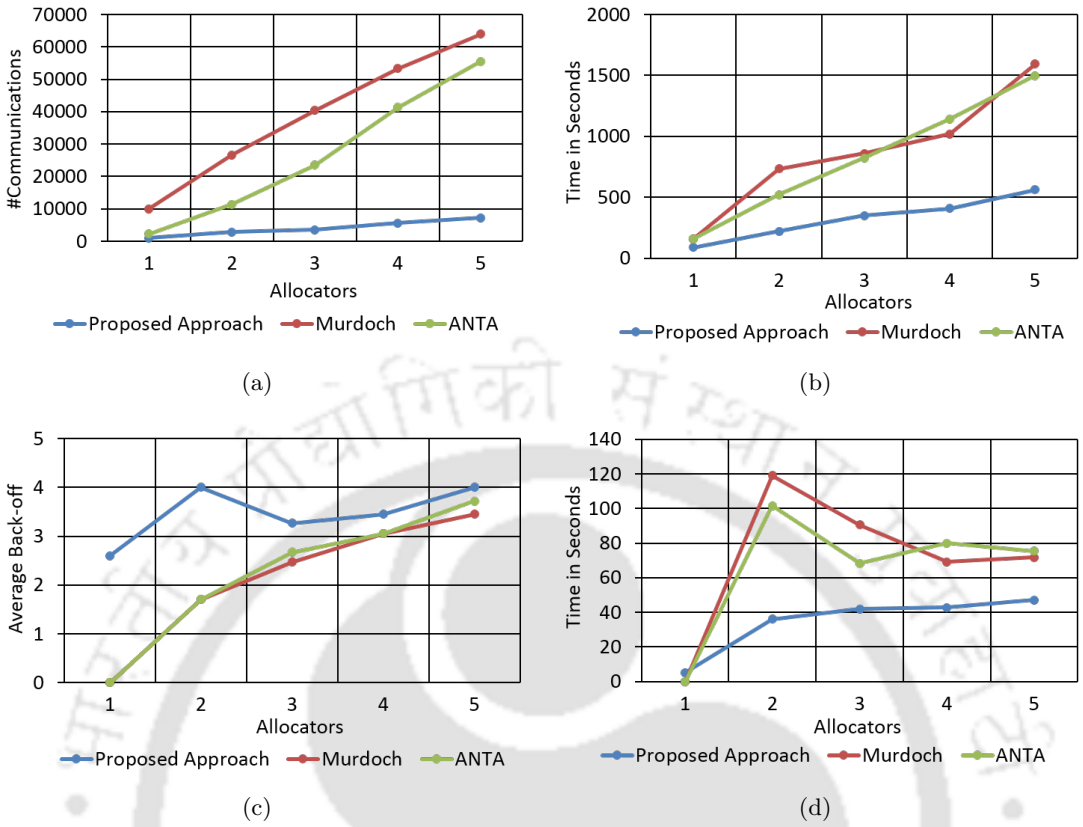


Figure 7.1: Results gathered from the experiments on 100-robot dynamic network out of which 15 were eligible for the required task and a total of 10 robots were required by each allocator. (a) Average inter-node communications (b) Average completion time of all allocators (c) Average number of back-offs (d) Average back-off interval.

In Murdoch, the authors have proposed a broadcast based auction method for task allocation. Although, they do not define the range of the message, we have implemented a multi-hop message broadcast for all kinds of communications using the Murdoch approach. The size and the heterogeneity of the MRS was kept the same. The multiple auctioneers (task allocators) were selected at random within the MRS. For contention resolution in this *augmented* Murdoch implementation, we have used the same back-off mechanism as mentioned in Equation 7.5. However, since this mechanism does not keep track of the number of distinct robots visited, U_R, Θ was calculated as:

$$\Theta = \nu BI_{min} \tag{7.7}$$

where ν denotes the number of bids received by the auctioneer and BI_{min} is a non-zero positive constant signifying the minimum back-off interval. BI_{min} was ascertained based on the size of the network.

The ANTA approach uses a broadcast tree based method to disseminate the trade messages and elicit bids from the interested robots within the network. This method uses a fixed tree level to limit the extent of message broadcasts within the network. The authors have shown that a higher tree level results in better performance of their task allocation method. Hence, in this augmented ANTA implementation, the tree level is made equal to the diameter of the network of robots. Further, the authors do not mention the consequences of the messages if the tree breaks

due to the movement of the robots in the mobile ad-hoc network. To tackle such scenarios in the implementation, the robots were made to broadcast their messages in case they did not find the intended parent robot as their neighbour. The task allocators were selected at random and the back-off mechanism was implemented in a manner similar to that in the case of Murdoch.

7.6 Results and Discussions

Experiments were conducted on networks spread across more than 10 PCs within a LAN. The parameters viz. the average number of inter-node communications, average task completion times, average number of back-offs and average back-off intervals are considered as the performance yardstick to evaluate the efficacy of the proposed approach. Following parameter values were used in the experiments: $C_d = 5$, $\alpha_k = 0.01$, $\chi_{initial}(0) = 10$, $BI_{min} = 10$ seconds, $BI_{max} = 500$ seconds.

7.6.1 Varying Number of Task Allocators

The graphs shown in Figures 7.1(a) through (d) depict the performance of the proposed approach as compared to that of Murdoch and ANTA with increasing number of allocators (auctioneer robots in case of the Murdoch and ANTA approaches and mobile agents in case of proposed approach) in a dynamic network of 100 robots. In the experiments, each allocator required 10 robots of a specific type and there were 15 such pertinent robots present in the network. The graph in Figure 7.1(a) shows the average number of inter-node communications incurred during the experiments. An instance of interaction between two robots forms a unit of inter-node communication. Such interactions between two robots can occur during exchange of messages (in case of Murdoch and ANTA) or when a mobile agent migrates (in case of the proposed approach). As can be observed from the graph, the average number of inter-node communications increases with increase in the number of allocators. This result may seem intuitive since the increase in the number of allocators increases the competition to acquire the robots. However, as can be noted, the proposed approach incurs least inter-node communications (< 10000 for 5 allocators) as compared to the other approaches (> 50000 for 5 allocators). It may also be observed that as the number of allocators increases, there is a sharp increase in the number of inter-node communications in case of Murdoch and ANTA as compared to that of proposed approach. This decreased number of inter-node communications in case of proposed approach is due to the limited migrations by the mobile agents as a consequence of their adaptive mobility resources (χ); which control the extent to which they explore the network after every back-off. Further in the proposed approach, if there are k number of mobile agents (k allocators) present in the network at any moment of time, then there can be a maximum of k possible inter-node communications. In case of Murdoch and ANTA with k-allocators, multiple robots ($>> k$) across the network exchange messages concurrently leading to an escalation of inter-node communications which could cause cluttering and subsequent reduction in the available bandwidth. Since the number of inter-node communications is proportional to the energy consumed within the robotic network, it can be inferred that the proposed approach is also energy efficient.

The graph in Figure 7.1(b) depicts the average time required to complete the task allocation process for the three approaches considered. This completion time is calculated as the difference between the time-stamps at the start and end of an experiment. These completion times include - the time taken by the allocators to find the required robots, the back-off intervals, communication delays due to the size of the messages or agents and the available bandwidth. As can be observed from this graph, the proposed approach consistently outperforms the other two approaches as the number of allocators increase. This saving in time is not only because of the adaptive mobility resource concept and fewer inter-node communications in proposed approach but also due to its

adaptive back-off mechanism. The graphs in Figures 7.1(c) and 7.1(d) depict the average number of back-offs and the average back-off intervals respectively with increasing number of allocators. As can be observed from these graphs, the proposed approach incurs more number of back-offs (Figure 7.1(c)) than the other two approaches but the average back-off intervals are always less than that of the other approaches. The higher back-offs in case of the proposed approach is due to the initial exploration of the network by the agents which is unknown to them. The average back-off interval for the proposed task allocation approach can be seen to be increasing with increase in the number of allocators. When there is only one allocator, the average number of back-offs (2.6 in Figure 7.1(c)) is high but the corresponding average back-off interval is low (5.16 sec in Figure 7.1(d)). This happens because the proposed approach selects the back-off interval based on how deep an agent has penetrated into the network. Such an adaptive concept has not been used in the other two approaches.

The graphs shown in Figures 7.2(a) through (d) depict the results gathered using a static network of 100 robots with increasing number of allocators. The static case was experimented just to investigate whether the other two approaches have an edge under such conditions. As can be observed from the graph in Figure 7.2(a), the average number of inter-node communications for the proposed and Murdoch approaches remains similar to that of the corresponding dynamic cases shown in Figure 7.1(a). However, there seems to be a large reduction in the average number of inter-node communications in case of the ANTA approach as compared to that when it was used in the dynamic scenario. This occurs as a result of the tree-based broadcast strategy used in the ANTA approach. Nonetheless, the proposed approach still incurs the least inter-node communications. The graph depicting average completion times in the Figure 7.2(b) and those of average number of back-offs and average beck-off interval in Figures 7.2(c) and (d) respectively, show similar trends as described earlier in case of the dynamic scenario. These results reveal that the proposed approach is oblivious of the nature of network and performs equally well in both static as well as dynamic environments.

7.6.2 Adaptive versus Constant Back-off

In order to remove any possible bias to the proposed approach caused by augmenting the Murdoch and ANTA approaches with a naïve constant back-off mechanism (Equation 7.7), we modified the proposed approach by using Equation 7.7 in lieu of Equation 7.6. The graphs in Figure 7.3 compare the performances of the proposed approach with adaptive and constant back-off, respectively on a dynamic network of 100 robots with increasing number of allocators under the same conditions as described in Section 7.6.1. It can be observed from the graph in Figure 7.3(a) that the average inter-node communications for the proposed approach with adaptive back-off is always higher than that of the proposed approach with constant back-off. This is so because the value of BI_{min} needs to be fixed *a priori*. This value, which needs to be found based on several empirical runs, requires, unlike in proposed approach with adaptive back-off, the knowledge of the number of robots in the network. The value may need to be recomputed for every change in the size of the network. Contrary to this, the proposed approach with adaptive back-off is scalable and adapts to differently sized networks.

It can also be noted from the graph in Figure 7.3(b) that the total completion times for both adaptive and constant back-off based approaches follow almost the same trend. These experiments clearly indicate that, the proposed approach has not been biased in any way to perform better than the other approaches augmented by the constant back-off mechanism.

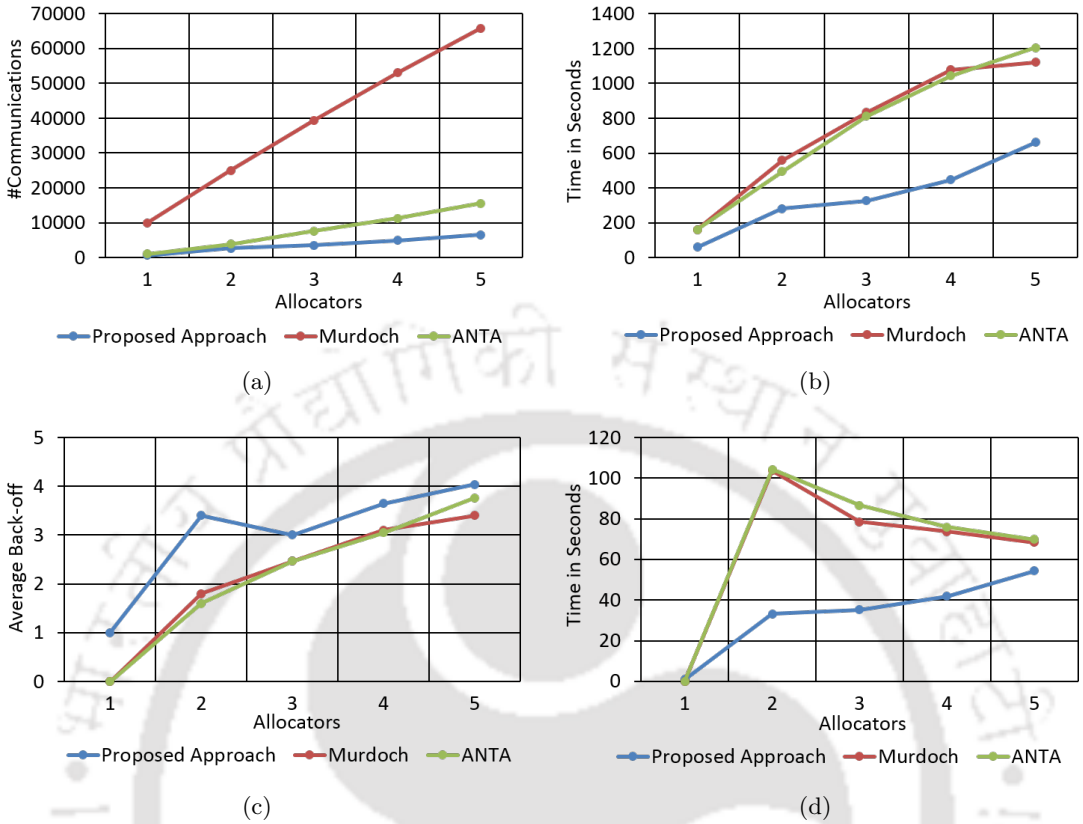


Figure 7.2: Results gathered from the experiments on 100-robot static network out of which 15 were eligible for the required task and a total of 10 robots were required by each allocator. (a) Average inter-node communications (b) Average completion time of all the allocators(c) Average number of back-offs (d) Average back-off interval.

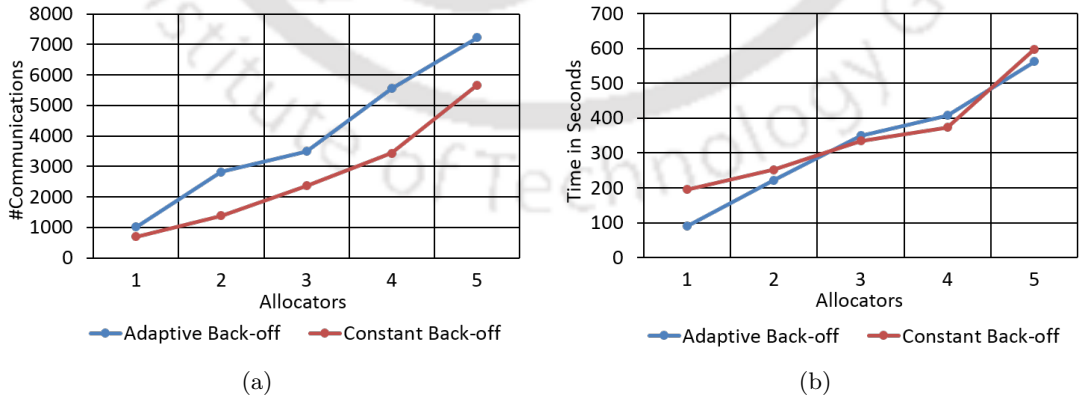


Figure 7.3: Comparing the proposed task allocation approach having adaptive back-off versus a constant back-off for the proposed approach (a) Average number on inter-node communications (b) Average completion time.

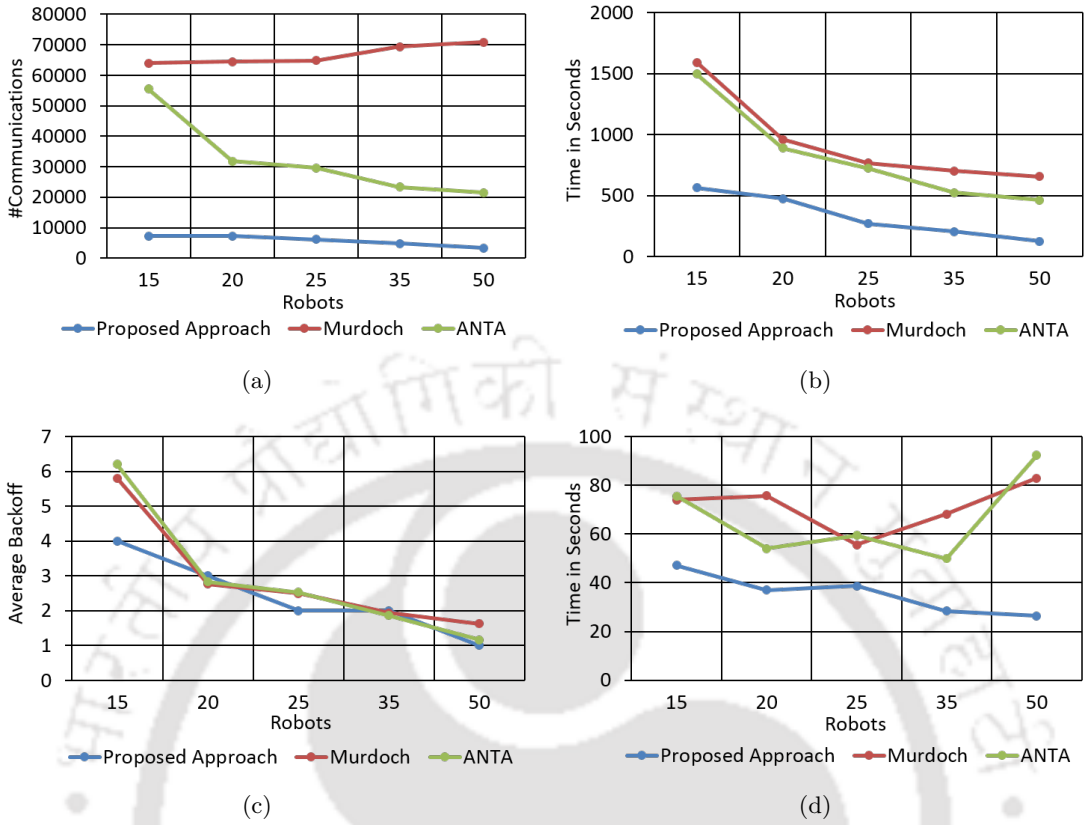
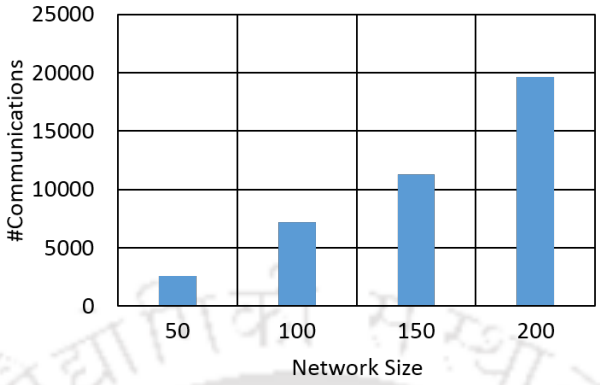


Figure 7.4: Results gathered from the experiments on varying the density of pertinent robots in the 100-robot dynamic network. Five allocators with each requiring a total of 10 robots were present in all the cases. (a) Average inter-node communications (b) Average completion time of all the allocators (c) Average number of back-offs (d) Average back-off interval.

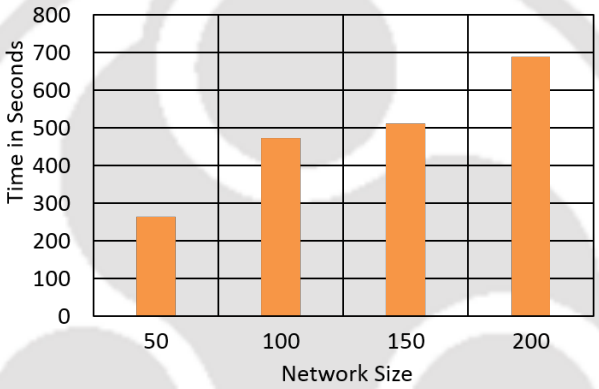
7.6.3 Varying Density of Robots

The density of the pertinent robots required by the competing allocators plays a crucial role in the performance of the proposed approach. If this density is high, the chances of fulfilling the requirements of most of the allocators will also be high thereby diminishing competition. The graphs in Figures 7.4(a) through (d) depict the performance of the proposed approach along with that of Murdoch and ANTA with increasing number of pertinent robots (density) in a 100-robot dynamic network. Five allocators, each requiring 10 robots were present in the network in each of the experiments.

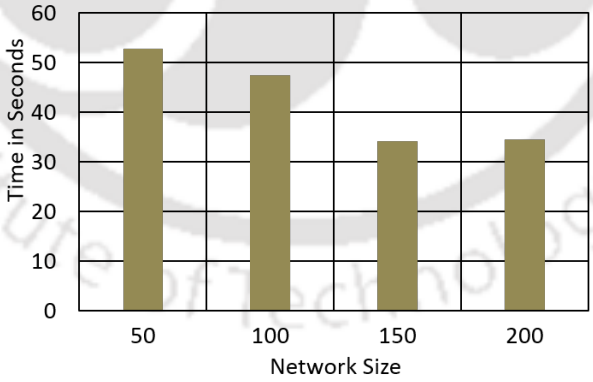
As can be observed from the graph in Figure 7.4(a), the average number of inter-node communications decreases with increase in the density of the pertinent robots for the proposed and ANTA approaches. In case of Murdoch, the number of inter-node communications increases with increasing density of the pertinent robots due to the increase in unrestricted broadcast based communication by them. As can be noted, the proposed approach incurs the least inter-node communications out of the three approaches demonstrating distinctly the effectiveness of this approach. The completion times of the three approaches can be seen in the graph in Figure 7.4(b) with increasing number of pertinent robots. It can be observed that the increase in density of pertinent robots decreases the completion times for all the three approaches. However, the proposed approach still outperforms



(a)



(b)



(c)

Figure 7.5: Results gathered from the experiments on varying size of the dynamic robotic network. A ratio of 1:2:10 is maintained for robots required: pertinent robots: network size. (a) Average inter-node communications (b) Average completion time (c) Average completion time per unit allocator requirement.

the rest by taking the least time. The graphs in Figures 7.4(c) and 7.4(d) depict the average number of back-offs and the average back-off intervals with increasing number of the pertinent robots in the network. As can be seen, in case of the proposed approach, the average number of back-offs and the corresponding average back-off intervals decrease with increase in the density of pertinent robots. For the other approaches, although a downward trend can be noted in the average number of back-offs, the corresponding average back-off intervals do not seem to follow any meaningful pattern. This shows that the effectiveness of the adaptive back-off mechanism used in the proposed approach which, unlike its constant back-off counterpart, changes the back-off interval based on the available number of pertinent robots. The proposed approach is thus more robust as it functions oblivious of the number of pertinent robots in the network and discovers them *on-the-fly*.

7.6.4 Varying Network Size

This section presents the performance of the proposed approach with varying number of robots forming the network (Network Size). The ratio of the requirements of the allocator, the density of the pertinent robots and the network size was maintained as $1 : 2 : 10$. For instance, 10 pertinent robots present in a 50-robot network would have each allocator explore for 5 pertinent robots. All the experiments were performed in dynamic networks with five allocators. The graphs in Figures 7.5(a) and 7.5(b) depict the average inter-node communications and average completion times for the proposed approach. Intuitively, it can be seen from the graphs that with increase in the size of the network, there is a proportionate increment in both the average number of inter-node communications (Figure 7.5(a)) and the average completion times (Figure 7.5(b)). This is bound to happen since the corresponding requirement of the pertinent robots was also made to increase proportionately as per the ratio. It may be noted that the maximum number of inter-node communications (< 20000) recorded in case of the 200-robot network in Figure 7.5(a) is far less than that of the other approaches as depicted in Figure 7.4(a) wherein the network was half the size (100-robot network).

The graph in Figure 7.5(c) depicts the average completion time per robot required by an allocator. This is basically the ratio of the average completion time to the requirement of pertinent robots by an allocator and signifies the rate at which the pertinent robots are allocated. As can be noted from this graph, there is a slight downward trend in this rate with increase in the size of the network. This is so because the constant ratio maintained in these experiments increases the density of the pertinent robots along with the network size and requirements of the allocators. The increase in density hastens the process of allocation as discussed in the previous section. It can thus be inferred that the proposed approach is not only adaptable but also scalable since it can cater to varying network sizes.

7.7 Implementation on Real-Robots

The proposed approach of task allocation was implemented in real-world[1] using a total of seven Lego® Mindstorms® NXT robots [1]. The multi-robot box pushing problem at multiple locations within an environment was chosen for the real-world implementation of the proposed approach. Each Lego robot was equipped with four sensors viz. a compass sensor, a touch sensor and two colour sensors. While the compass and colour sensors helped the robots to navigate within the environment, the touch sensor was used to detect the boxes. The environment consisted of black

[1]The video depicting the complete experiment can be accessed in the following link - http://www.iitg.ernet.in/cse/robotics/?page_id=2056

lined paths with colour markers to indicate the positions of the job locations. Figures 7.6 and 7.7 depict the details of the robots used in the experiment along with the environment for the box pushing task respectively. As can be observed in Figure 7.7, there are two job locations (# 1 & 2) where boxes are placed to be moved to the respective destinations. The red markers over the black-lined path near to the job locations indicate the entry points for the robots to the job location. The number of robots required to push a box depends on the size of that box. In the experiment, we had used boxes requiring 2, 3, 4 or 5 robots to push them to the destination locations. In Figure 7.7, Box1 in job location #1 requires three robots while Box2 in job location #2 requires four. Unallocated robots were stationed at the Robot Docking Station. All the robots were configured within a dynamic *Typhon* based network using LPA-PRO-NXT interface [241] over the Bluetooth®. Once a mobile agent successfully procures the requisite number of robots, each robot uses the black-lined path to navigate to the respective job location. After finishing the box pushing task, the robots again follow the black-lined path back to the Docking Station.

Experiments were performed by placing boxes of different sizes at the two job locations. Whenever a box is placed at a job location, a mobile agent having information about the requirements of the current pushing task along with the job location and entry point, is released into the *Typhon* based network of robots. Initially the experiment commenced with only five robots available in the environment. Two mobile agents for each of the two pushing jobs at job locations #1 and #2 were released into the network. One agent required 3 robots (at job location #1) while the other required 4 (at job location #2). Contention was bound to occur since there were only five robots in the environment. The proposed task allocation approach was used to resolve the conflicts and complete both the jobs. Further, two new robots were added to the system while the other robots were performing their tasks. These new robots were seamlessly integrated into the system in an autonomous manner. In the experiment, we initially placed two boxes requiring three robots each followed by two boxes one requiring 3 robots while the other requiring 4 robots and a new robot. Afterwards, another new robot and two more boxes were placed at the job location #1 and #2 requiring 2 and 5 robots respectively.

This experiment demonstrated the viability of the proposed technique of task allocation in real-environments. The use of mobile agents simplifies the process of delegating the task requirements and acquiring robots from the network. The integration of additional robots *on-the-fly* in the existing network was seamless and did not in any way affect the stability or working of the system. The manner in which the proposed approach functions in the emulated and the real-worlds are similar indicating the feasibility of its implementation in real systems.

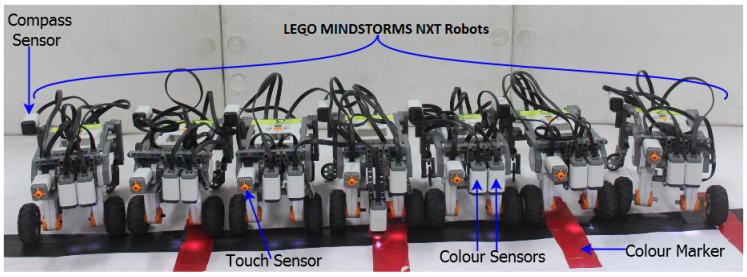


Figure 7.6: Lego® Mindstorms® NXT robots used in the multi-robot box pushing experiment.

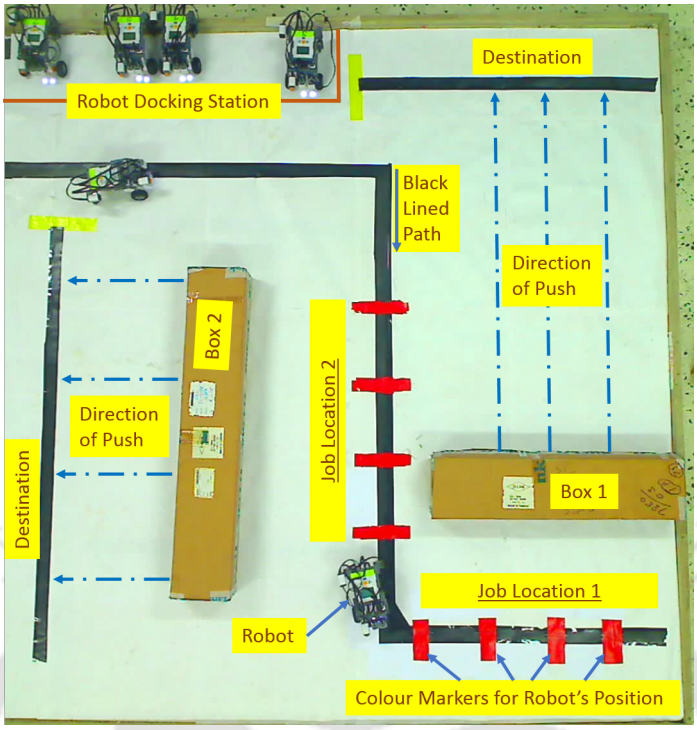


Figure 7.7: The boxing pushing experiment environment with two Job locations.

7.8 Chapter Summary

An efficient task allocation mechanism is a fundamental requirement to regulate the processes and make productive use of an MRS. In large heterogeneous MRSs, the requirements of various tasks may overlap resulting in competing instances of task allocations. The task allocation approach presented in this chapter describes a mechanism that uses mobile agents for task allocation within a network of robots constituting an MRS. The proposed approach supports multiple and concurrent competing instances of task allocations using voluntary and adaptive back-offs. The approach can adapt to changes in the environment such as varying number of robots, varying density of pertinent robots, varying number of task allocators, etc. The results gathered from the emulation experiments conclusively indicate that the proposed approach uses the least number of inter-node communications for its execution thus saving on both time and power consumed. The practical viability of the use of the proposed approach for real MRSs, is also endorsed by an experiment using real-robots capable of pushing boxes.

The proposed task allocation approach can still be improved by forcing *on-the-fly* sharing of information amongst the mobile agents so as to accelerate the completion of the tasks while also saving futile migrations on part of the agents. The sharing of information can also be used to carry out optimal task allocations by improving on the agent-to-robot and agent-to-agent negotiation strategies.

"The woods are lovely, dark, and deep,
But I have promises to keep,
And miles to go before I sleep, And miles to go before I sleep."

Robert Frost (1874 – 1963)
American poet

8

Conclusions and Avenues for Future Research

With rapid advancements in computing and communication technologies, researchers are once again looking into how the features of mobile agent technology can be exploited. In the context of network robots, these soft agents with mobility, offer various latent characteristics essential to realize truly distributed and decentralized mechanisms. The contributions made in this thesis concentrate on utilizing some of the characteristics of the mobile agents in order to mitigate the challenges involved in distributed coordination, sharing and learning within the networked robots. This chapter provides a summary of the contributions made in this thesis and discusses their impact and applications. The chapter ends by opening up the directions for future research with possible extensions to the work carried out in this thesis.

8.1 Summary of the Contributions

The prime aim behind the work done in this thesis was to craft mechanisms which could aid in developing integrated solutions for networked robots including various devices and sensors embedded within the environment. Previous to this, Godfrey [318] developed mechanisms that improve migration strategies of mobile agents so as to deliver services faster within the network of robots. He went on to formulate a stigmergy based mechanism to control the population of different sets mobile agents wherein each set carried a distinct solution. These mechanisms improved the performances by satiating the requirements of the robots within a network by using mobile agents for delivering the services (solutions). However, it was inherently assumed that the solutions carried by the mobile agents are the best suited ones and hence would always result in providing optimal services. Such an assumption may hold true for a static environment wherein one solution has been verified *a priori*, to be the best one. This would mean that the system would always exhibit the same level of performance all through. In dynamic environments involving a network of mobile robots, one needs to provide solutions based on the current parameters of the system. Obviously it would be an arduous task to arrive at a single or generic solution which could work in all situations. The first contribution of this thesis addresses this challenge of selecting a solution among a set of solutions which can provide the best performance as per the demand of the current situation and parameters of the environment. Taking inspirations from the manner in which the biological immune system manages a repertoire of antibodies to form an Idiotypic Network and combat foreign particles invading the human body, a mechanism to conceive and emulate the same in the computational world has been attempted. The requirements of the robots are modelled as the antigens (foreign particles) while the solutions or services carried by the mobile agents are

modelled as antibodies. As prescribed in the Idiotypic Network model, the interactions among the antibodies, resulting in the exchange of stimulations and suppressions signals depending upon their affinities and performance in countering the antigens, are used to select the dominant solution among a given set of solutions available for the same objective. The results gathered by emulating the mechanism in the real networked environment revealed that the mechanism not only selects the best performing solutions but also the most generic ones which cater to a range of problems or requirements of the entities comprising the network. In addition, the concentration of antibodies i.e. the population of mobile agents carrying the solutions are realized in a completely distributed fashion without involving the use of any global parameter.

This mechanism can be applied not only for networked robots but also for a distributed congregation of various devices forming an Internet-of-Things or a Cyber-Physical System. In a sensor network which has been installed to monitor the parameters of environment such as temperature, humidity, etc., this mechanism can be used to trigger a quick response for a problem caused by any one of the environmental parameters. For illustration, if the sensors report a jump in the temperature value which could be due to fire, the relevant mobile agents would become dominant entities within the network to raise alarm and perform curative actions. Similar applications can be envisioned to improve the throughput in a grid environment for the arbitration of resources based on the needs of the current demand or for managing processes within an automated warehouse.

Although the first contribution of the thesis looked only in the quality of solutions being carried by the mobile agents so as to select the best ones, it did not do anything to improve the solutions carried by the mobile agents. The mobility of these agents can facilitate their meeting at various nodes within the network and consequent sharing and learning based on their respective experiences and hence enhance the solutions they carry. This led to the second contribution of this thesis wherein mobile agents were made to share information when they meet at a node based on some conditions. The shared information was then used by the individual agents to improve the performance of the solutions they carry. A framework was thus developed which could facilitate such mutual interactions among the mobile agents within a node in the network along with the associated dynamics involved in making the agents migrating into the network to collect information. The conditions under which they are required to trigger the execution of a learning algorithm were also propounded. The framework was realized over a LAN for the problem of finding a path from a source to a destination within a *Maze-World*. The emulation experiments conducted validate the applicability of the proposed framework for localized sharing of information and the consequent learning among a set of mobile agents having the same objective. The framework allows the user to define the objective along with the associated learning algorithm and the feedback function based on the application under consideration. This framework is specifically suited for those applications wherein a solution must be learnt by the collective feedback from a set of location-unaware and spatially segregated entities within a network of nodes. In addition, the number of such entities within the network can vary with time and a solution must be learnt by discovering those entities currently present within the reach of the network. Hence, the networked robotic systems form a well-suited application domain for the use of this sharing and learning framework. However, in this age of growing hand-held and mobile devices, similar scenarios can be visualized in the domain of mobile computing systems which involves various dynamic mobile units joining and leaving the system with time. The framework can be exploited to learn and deliver specific user oriented services whenever a mobile device enters an autonomous system such as one in a campus or a building.

The next contribution of this thesis deals with the synchronization of the executions of a set of tasks by a network of robots. The specific objective of this contribution was to complete the execution of a task throughout the environment before the execution of the next task could com-

mence following a sequential order of the set of tasks. The developed technique is based on the method of indirect stigmergic communication (prevalent in the social insect colonies) among the mobile agents to minimize the communication overheads and realize a truly distributed solution. The mobile agents which carry the set of tasks act as ants within the network of robots and the synchronization among the execution of tasks is achieved by making the use of task specific potentials. These potentials are regulated by the respective rewards and penalties that a mobile agent receives whenever it visits a robot (node) in the network. The technique also exhibits self-healing by taking care of any task failures in the sequence of tasks executed without completely stalling the synchronization process. The technique was evaluated by emulating the same on various networks with varying sizes, network topologies and number of mobile agents. The results gathered validate the objectives of the techniques to achieve near synchronous execution of a sequence of tasks. Further, to verify the applicability of the developed technique for real world applications, a proof-of-concept implementation was performed using real robots. In the robotic implementation four robots were used to carry a long pole from one location to another in a synchronized manner. The technique was found to be performing equally well as it performed in the emulations thus endorsing its use in real world applications. This synchronization technique has a range of applications from industrial robots working on an assembly line to a swarm of robots scavenging a floor. The technique also applies to the domain of Cyber-Physical Systems wherein a set of mobile agents could be assigned the responsibility to execute a set of tasks using the actuating devices based on the values reported by the sensors embedded within the environment. One such scenario could be to orient a set of solar panels to face the sun based on the current intensity of sun-light or to regulate the irrigation valves on a field based on temperature and humidity.

A limitation of the technique developed in the previous contribution is that each mobile agent needs to carry the information about all the tasks available within the set of tasks. In addition, the number of mobile agents required to carry out the task synchronization process is also required to be ascertained *a priori*. Hence, the technique developed in the previous contribution was extended to address these limitations. The enhanced technique used a heterogeneous set of mobile agents in which each set was equipped with the information to execute a distinct task from the set of tasks. The objective although remained same as that in the previous contribution to carry out the executions of a set of tasks in a given sequential order by a network of robots. The problem of finding the number of robots is ameliorated by integrating the cloning resource based population control model proposed by Godfrey *et al.* [128] into the enhanced technique. The concept of task specific *potentials* and their regulation based on the rewards and penalties received by visiting the nodes was preserved. However, several improvements were added. This included the manner in which the rewards and penalties were calculated to cater to the heterogeneous sets of mobile agents in the network. The enhanced technique was tested on dynamic networks of varying sizes using emulation experiments. The results clearly depicted that the enhanced technique could successfully carry out the synchronized execution of a set of tasks in the specified sequential order. In addition, the population of mobile agents were found to be varying based on the current demand of the network of nodes. The mobile agents which carry the tasks currently required by the network of nodes rise in their respective populations while those of the others recede, thus dynamically managing the number of mobile agents in the network. The enhanced technique inherently exhibited self-healing for mending any task failures in the sequence of tasks executed by the nodes. This enhanced technique thus provides additional features for the process of synchronized task execution. Due to the presence of heterogeneous mobile agents with segregated tasks, the sequence of tasks can be easily extended or reduced as per the need while the synchronization process is in action. Further, new nodes can join the network any time and become a participant in the task execution process without adversely affecting the performance since the number of mobile agents required to cater to

the number of nodes in the network is automatically managed by the integrated population control mechanism.

An inherent assumption made in the last two contributions was the availability of the robots (nodes) within the network capable of executing the set of tasks. However, in systems comprising large number of heterogeneous robots having varying capabilities, there may be a continuous arrival of sets of tasks to be executed. In such situations, it is important to first find the requisite set of robots capable of performing the desired set of tasks before commencing the task execution. Hence, the last contribution of this thesis deals with finding the required set of unoccupied robots available within the network having the necessary capabilities and resources that are needed in order to allocate them for executing a desired set of tasks. The mechanism presented in this contribution also addresses the contentious issue of the competing instances of task allocations present within the system. While the mobile agents are used to search and reserve the robots, a method of voluntary and asynchronous back-off has been used to resolve contentions and deadlocks ensuring progress. The mechanism has been emulated on real networked environment along with two other prominent multi-robot task allocation approaches available in the literature which have been used for comparison. In the results, the mobile agents based mechanism developed to allocate the required number of robots to execute a set of tasks was found to outperform the other two approaches. Further, the developed mechanism was found to be more adaptive with respect to the size of the network and density of the required robots. The implementation was also supplemented by a real world experiment using real robots. The multi-robot box pushing problem was considered for implementing the proof-of-concept experiment. A total of seven robots were used with two simultaneous job locations in the environment. The developed mechanism seamlessly carried out the allocation of tasks to the desired set of robots thus fortifying the practical utility of the mechanism developed. The distinguishing characteristic of this mechanism is its least amount of inter-node communication and a completely distributed approach. Along with applications into the multi-robot task allocation problems, this mechanism can also be employed in allocating jobs to different machines within a factory automation system.

8.2 Future Research Scopes

The research work reported in the contributing chapters of this thesis provides ample scope to advance the work in various research directions. One of the primary domains of research is to extend the mechanisms presented in this thesis for their applications in the areas of Internet-of-Things (IoT) and Cyber-Physical Systems (CPS). The mobile agent technology has the latent characteristics to become a key technology for addressing the application level issues in the domain of IoT and CPS. The Idiotypic Network based mechanism can be used to evolve better solutions in order to choose the best algorithm for managing devices within an IoT and CPS. The Idiotypic Sieve presented in this thesis can also be extended to use the clonal selection theory for designating the best performing solutions as memory cells. The mobile agents designated as memory cells can be used to trigger a quicker secondary response whenever similar antigens (requirements in the network) are encountered again in the environment. Similarly, the framework on sharing information and mutual learning can be extended to clone those mobile agents which show better progress than the others towards finding a solution. Though such cloning of mobile agents with better solutions may hasten the process of learning, excessive cloning could clutter the network. The population control model proposed by Godfrey *et al.* [128] can also be integrated within the learning framework to regulate the population of learning agents within bounds. This framework can also be used as a test-bed to find the best among a set of given learning algorithms. Further,

a possible extension of this framework could be to learn a solution by hybridizing the intermediate solutions of various algorithms. It will be interesting to see how the different algorithms use the intermediate information of other algorithms in order to construct their own solutions.

The techniques for synchronized task executions can be extended in a manner that can cater to a set of spatially segregated robots. The technique can be used to synchronize the execution of tasks at selected points in the environment using mobile agents. In addition, multiple sets of such synchronized tasks executions can be carried out concurrently by different sets of mobile agents thus creating a complex meta-level network of task executions. A further research study can be carried out to find the value of the task specific potential required to achieve the optimal performance of the system. The model driven analysis and theoretical guarantees on the time and space complexities of the technique can throw more light into its intricacies.

The work on multiple task allocations can be exploited to bring together the robots with complementary capabilities required to finish a task. The task allocation mechanism presented in this thesis does not consider any priority among the tasks nor does it put any time limit on the search for the required set of robots. As such without priority and a deadline, the mechanism described could be rendered useless, if the set of tasks need to be executed on an emergency basis. With a selected set of mobile agents empowered with higher priorities in the task allocation process and a prescribed deadline for the execution of their tasks the mechanism may be moulded to suit time critical applications as well. In addition, the task allocation mechanism can be coupled with the population control based synchronization technique to create an integrated approach from task announcement to finding the required set of robots and the ensuing task executions.

Another direction of future research for using the mobile agent technology could be a hierarchical architecture of system development wherein different levels of hierarchies of mobile agents are used to aid in the functioning of the system as a whole. In such systems, while the top level mobile agents could be tasked to perform decision making and learning activities, the mobile agents at the lower levels of the hierarchy could carry out mundane activities such as dissemination of information or updating nodes to cater to the new changing dynamics of the environment. A stronger control over such systems can be enforced by putting an entity at the top of the hierarchy which observes the system performance and perform critical decision making functions.

In this growing age of digitization and proliferation of a gamut of devices, the mobile agent technology can be used as a versatile tool by system developers in order to provide personalized services within the boundaries of an autonomous system. There is a need to overhaul mobile agent technology based on the demands and scope of the current technological landscape. Although the researchers are revisiting this technology with a new perspective, a greater thrust is required to roll out mobile agents as an acceptable framework for system development.

Bibliography

[1] LEGO Robot. Lego Mindstorms Education NXT Base Set (9797) - Robotic Platform. http://www.lego.com/en-us/mindstorms/. [Online; accessed 21-December-2015].

[2] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer Science & Business Media, 2008.

[3] Yudhijit Bhattacharjee. Mars rover plans roll while asteroid acrimony continues. *Science*, 341(6143):229–229, 2013.

[4] Doru Panescu and Gabriela Varvara. On applying csp for coordination of a multi-robot holonic manufacturing execution system. In Theodor Borangiu, editor, *Advances in Robot Design and Intelligent Control*, volume 371 of *Advances in Intelligent Systems and Computing*, pages 3–12. Springer International Publishing, 2016.

[5] G. Demesure, M. Defoort, A. Bekrar, D. Trentesaux, and M. Djemai. Cooperation mechanisms in multi-agent robotic systems and their use in distributed manufacturing control: Issues and literature review. In *40th Annual Conference of the IEEE Industrial Electronics Society, IECON 2014* -, pages 2538–2543, 2014.

[6] A. English, P. Ross, D. Ball, and P. Corke. Vision based guidance for robot navigation in agriculture. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1693–1698, May 2014.

[7] Jrgen Altmann, Peter Asaro, Noel Sharkey, and Robert Sparrow. Armed military robots: editorial. *Ethics and Information Technology*, 15(2):73–76, 2013.

[8] A. Bekku, J. Kim, Y. Nakajima, and K. Yonenobu. A body-mounted surgical assistance robot for minimally invasive spinal puncture surgery. In *2014 5th IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 19–23, Aug 2014.

[9] A. Broggi, P. Cerri, S. Debattisti, M.C. Laghi, P. Medici, D. Molinari, M. Panciroli, and A. Prioletti. Proud - public road urban driverless-car test. *Intelligent Transportation Systems, IEEE Transactions on*, 16(6):3508–3519, Dec 2015.

[10] Si Jung Kim, Jeong Hyeon Cheon, S. Forsyth, and Eunsook Jee. Research trends in art and entertainment robots (ane robots). In *2013 IEEE RO-MAN*, pages 360–361, Aug 2013.

[11] R. Borja, J.R. de la Pinta, A. lvarez, and J.M. Maestre. Integration of service robots in the smart home by means of upnp: A surveillance robot case study. *Robotics and Autonomous Systems*, 61(2):153 – 160, 2013.

[12] Roger Bemelmans, Gert Jan Gelderblom, Pieter Jonker, and Luc de Witte. Socially assistive robots in elderly care: A systematic review into effects and effectiveness. *Journal of the American Medical Directors Association*, 13(2):114 – 120.e1, 2012.

[13] Markus Bernard, Konstantin Kondak, Ivan Maza, and Anibal Ollero. Autonomous transportation and deployment with aerial robots for search and rescue missions. *Journal of Field Robotics*, 28(6):914–931, 2011.

[14] Raffaello D'Andrea. Guest editorial: A revolution in the warehouse: A retrospective on kiva systems and the grand challenges ahead. *IEEE Transactions on Automation Science and Engineering*, 4(9):638–639, 2012.

[15] Henry W Stone. *Kinematic modeling, identification, and control of robotic manipulators*, volume 29. Springer Science & Business Media, 2012.

[16] Minoru Asada, Karl F. MacDorman, Hiroshi Ishiguro, and Yasuo Kuniyoshi. Cognitive developmental robotics as a new paradigm for the design of humanoid robots. *Robotics and Autonomous Systems*, 37(23):185 – 193, 2001. Humanoid Robots.

[17] TylerD. Wortman, Avishai Meyer, Oleg Dolghi, AmyC. Lehman, RyanL. McCormick, ShaneM. Farritor, and Dmitry Oleynikov. Miniature surgical robot for laparoendoscopic single-incision colectomy. *Surgical Endoscopy*, 26(3):727–731, 2012.

[18] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013.

[19] Sigal Berman. Design considerations of dexterous telerobotics. In Aly El-Osery and Jeff Prevost, editors, *Control and Systems Engineering*, volume 27 of *Studies in Systems, Decision and Control*, pages 193–203. Springer International Publishing, 2015.

[20] Linamara Rizzo Battistella, Guang-Zhong Yang, Paolo Dario, Eiichi Saitoh, Andrew Schwartz, and Pyung Hun Chang. Multifactorial approach to rehabilitation robotsfrom clinic to research. In *2014 5th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 1–10. IEEE, 2014.

[21] J. Fasola and M.J. Mataric. Using socially assistive human-robot interaction to motivate physical exercise for older adults. *Proceedings of the IEEE*, 100(8):2512–2526, 2012.

[22] S.C. Lenaghan, Yongzhong Wang, Ning Xi, T. Fukuda, T. Tarn, W.R. Hamel, and Mingjun Zhang. Grand challenges in bioengineered nanorobotics for cancer therapy. *Biomedical Engineering, IEEE Transactions on*, 60(3):667–673, 2013.

[23] J.R. De La Pinta, J. M. Maestre, E. F. Camacho, and I. Gonzalez Alonso. Robots in the smart home: a project towards interoperability. *Int. J. Ad Hoc Ubiquitous Comput.*, 7(3):192–201, May 2011.

[24] M. Brenna, M. C. Falvo, F. Foiadelli, L. Martirano, F. Massaro, D. Poli, and A Vaccaro. Challenges in energy systems for the smart-cities of the future. In *2012 IEEE International on Energy Conference and Exhibition (ENERGYCON)*, pages 755–762, Sept 2012.

[25] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805, 2010.

[26] Ragunathan (Raj) Rajkumar, Insup Lee, Lui Sha, and John Stankovic. Cyber-physical Systems: The Next Computing Revolution. In *Proceedings of the 47th Design Automation Conference*, DAC '10, pages 731–736, New York, NY, USA, 2010. ACM.

[27] ITU Strategy and Policy Unit. Itu internet reports 2005: The internet of things. *Geneva: International Telecommunication Union (ITU)*, 2005.

[28] Chen Wen-lin, Cao Rui-min, Hao Li-na, and Wang Qing. Researches on robot system architecture in cps. In *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, pages 603–607, 2015.

[29] RoboEarth. What is Cloud Robotics? http://roboearth.org/cloud_robotics/. [Online; accessed 22-December-2015].

[30] Robohow. European research project: Robohow. https://robohow.eu/. [Online; accessed 22-December-2015].

[31] Rapyuta Robotics. Cloud-connected low-cost multi-robot systems for security and inspection. http://www.rapyuta-robotics.com/. [Online; accessed 22-December-2015].

[32] Martin Buss and Michael Beetz. Cotesyscognition for technical systems. *KI-Künstliche Intelligenz*, 24(4):323–327, 2010.

[33] Guoqiang Hu, Wee Peng Tay, and Yonggang Wen. Cloud robotics: architecture, challenges and applications. *IEEE Network*, 26(3):21–28, 2012.

[34] MyRobots. Social network of robots. http://www.myrobots.com/wiki/About. [Online; accessed 22-December-2015].

[35] Erico Guizzo. How googles self-driving car works. *IEEE Spectrum Online*, 18, October 2011.

[36] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura. The intelligent asimo: system overview and integration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2478–2483, 2002.

[37] R. Playter, M. Buehler, and M. Raibert. Bigdog. In *Proc. SPIE*, volume 6230, pages 62302O–62302O–6, 2006.

[38] S. Shamsuddin, L.I. Ismail, H. Yussof, N. Ismarrubie Zahari, S. Bahari, H. Hashim, and A. Jaffar. Humanoid robot nao: Review of control and motion exploration. In *2011 IEEE International Conference on Control System, Computing and Engineering (ICCSCE),*, pages 511–516, 2011.

[39] J. L. Jones. Robots at the tipping point: the road to irobot roomba. *IEEE Robotics Automation Magazine*, 13(1):76–78, 2006.

[40] Peidong Liang, Chenguang Yang, Ning Wang, Zhijun Li, Ruifeng Li, and Etienne Burdet. Implementation and test of human-operated and human-like adaptive impedance controls on baxter robot. In *Advances in Autonomous Robotics Systems*, pages 109–119. Springer, 2014.

[41] A Glamnik and R Šafarič. Control of kuka kr 5 robot with a haptic device. In *2012 9th International Conference on Remote Engineering and Virtual Instrumentation (REV),*, pages 1–7. IEEE, 2012.

[42] Personal Robotics. Pr2 robots. https://www.willowgarage.com/pages/pr2/overview. [Online; accessed 22-December-2015].

[43] Robin Murphy. *Introduction to AI robotics*. MIT press, 2000.

[44] Rodney A Brooks. A robot that walks; emergent behaviors from a carefully evolved network. *Neural computation*, 1(2):253–262, 1989.

[45] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

[46] Rodney Allen Brooks. *Cambrian intelligence: the early history of the new AI*, volume 97. Mit Press Cambridge, MA, 1999.

[47] Ronald C Arkin and Douglas C Mackenzie. Planning to behave: A hybrid deliberative/reactive robot control architecture for mobile manipulation. In *International Symposium on Robotics and Manufacturing, Maui, HI*, 1994.

[48] Toshiyuki Yasuda and Kazuhiro Ohkura. Autonomous role assignment in a homogeneous multi-robot system. *Journal of Robotics and Mechatronics*, 17(5):596, 2005.

[49] Paul Levi and Serge Kernbach. Heterogeneous multi-robot systems. In *Symbiotic Multi-Robot Organisms*, volume 7 of *Cognitive Systems Monographs*, pages 79–163. Springer Berlin Heidelberg, 2010.

[50] Luca Iocchi, Daniele Nardi, and Massimiliano Salerno. Reactivity and deliberation: a survey on multi-robot systems. In *Balancing reactivity and social deliberation in multi-agent systems*, pages 9–32. Springer, 2001.

[51] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, 10:399, 2013.

[52] Nader Mohamed, Jameela Al-Jaroodi, and Imad Jawhar. Middleware for robotics: A survey. In *Robotics, Automation and Mechatronics, 2008 IEEE Conference on*, pages 736–742. IEEE, 2008.

[53] Noriaki Ando, Takashi Suehiro, Kosei Kitagaki, Tetsuo Kotoku, and Woo-Keun Yoon. Rt-middleware: distributed component middleware for rt (robot technology). In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005 (IROS 2005)*, pages 3933–3938. IEEE, 2005.

[54] Toby H.J. Collett, Bruce A. MacDonald, and Brian P. Gerkey. Player 2.0: Toward a practical robot programming framework. In *Proc. of the Australasian Conf. on Robotics and Automation (ACRA)*, Sydney, Australia, 2005.

[55] Jet Propulsion Laboratory NASA. Claraty: Coupled-layer architecture for robotic autonomy. https://claraty.jpl.nasa.gov/man/overview/. [Online; accessed 22-December-2015].

[56] H. Utz, S. Sablatnog, S. Enderle, and G. Kraetzschmar. Miro - middleware for mobile robot applications. *IEEE Transactions on Robotics and Automation*, 18(4):493–497, 2002.

[57] D. Calisi, Andrea Censi, L. Iocchi, and D. Nardi. Openrdk: A modular framework for robotic software development. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1872–1877, 2008.

[58] Carle Cote, Yannick Brosseau, Dominic Letourneau, Clément Raïevsky, and Francois Michaud. Robotic software integration using marie. *International Journal of Advanced Robotic Systems*, 3(1):55–60, 2006.

[59] Alex Brooks, Tobias Kaupp, Alexei Makarenko, Stefan Williams, and Anders Orebck. Orca: A component model and repository. In Davide Brugali, editor, *Software Engineering for Experimental Robotics*, volume 30 of *Springer Tracts in Advanced Robotics*, pages 231–251. Springer Berlin Heidelberg, 2007.

[60] Olivier Michel. Webots: Symbiosis between virtual and real mobile robots. In Jean-Claude Heudin, editor, *Virtual Worlds*, volume 1434 of *Lecture Notes in Computer Science*, pages 254–263. Springer Berlin Heidelberg, 1998.

[61] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3(2), page 5, 2009.

[62] H. Bruyninckx. Open robot control software: the orocos project. In *Proceedings 2001 (ICRA) IEEE International Conference on Robotics and Automation*, volume 3, pages 2523–2528, 2001.

[63] iRobot. irobot aware 2 robot intelligence software. http://www.irobot.com/For-Defense-and-Security/Developers.aspx. [Online; accessed 22-December-2015].

[64] Jaesoo Lee, Jiyong Park, Seunghyun Han, and Seongsoo Hong. Rsca: Middleware supporting dynamic reconfiguration of embedded software on the distributed urc robot platform. In *The First International Conference on Ubiquitous Robots and Ambient Intelligence (ICURAI)*, pages 426–437, 2004.

[65] Nader Mohamed, Jameela Al-Jaroodi, and Imad Jawhar. A review of middleware for networked robots. *International Journal of Computer Science and Network Security*, 9(5):139–148, 2009.

[66] Ayssam Elkady and Tarek Sobh. Robotics middleware: A comprehensive literature survey and attribute-based bibliography. *Journal of Robotics*, 2012.

[67] A. Prorok, A. Bahr, and A. Martinoli. Low-cost collaborative localization for large-scale multi-robot systems. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4236–4241, 2012.

[68] M.S. Couceiro, R.P. Rocha, C.M. Figueiredo, J.M.A. Luz, and N.M.F. Ferreira. Multi-robot foraging based on darwin's survival of the fittest. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 801–806, 2012.

[69] Cyril Robin and Simon Lacroix. Multi-robot target detection and tracking: taxonomy and survey. *Autonomous Robots*, pages 1–32, 2015.

[70] Ken Sugawara, Nikolaus Correll, and Dustin Reishus. Object transportation by granular convection using swarm robots. In M. Ani Hsieh and Gregory Chirikjian, editors, *Distributed Autonomous Robotic Systems*, volume 104 of *Springer Tracts in Advanced Robotics*, pages 135–147. Springer Berlin Heidelberg, 2014.

[71] Praneel Chand and Dale A. Carnegie. Mapping and exploration in a hierarchical heterogeneous multi-robot system using limited capability robots. *Robotics and Autonomous Systems*, 61(6):565 – 579, 2013.

[72] Tiago P. Nascimento, Antnio Paulo Moreira, and Andr G. Scolari Conceio. Multi-robot nonlinear model predictive formation control: Moving target and target absence. *Robotics and Autonomous Systems*, 61(12):1502 – 1515, 2013.

[73] Alessandro Farinelli, Luca Iocchi, and Daniele Nardi. Multirobot systems: a classification focused on coordination. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34(5):2015–2028, 2004.

[74] S.A. Reveliotis and E. Roszkowska. Conflict resolution in free-ranging multivehicle systems: A resource allocation paradigm. *IEEE Transactions on Robotics*, 27(2):283–296, 2011.

[75] M. Bowling and M. Veloso. Motion control in dynamic multi-robot environments. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation, CIRA '99*, pages 168–173, 1999.

[76] Ignacio Mas and Christopher Kitts. Object manipulation using cooperative mobile multi-robot systems. In *Proceedings of the World Congress on Engineering and Computer Science*, volume 1, 2012.

[77] Minoru Asada, Manuela Veloso, Gerhard K Kraetzschmar, and Hiroaki Kitano. Robocup: Today and tomorrow. *Experimental Robotics Six*, 6:369, 1999.

[78] Y Uny Cao, Alex S Fukunaga, and Andrew Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous robots*, 4(1):7–27, 1997.

[79] Pierre-P Grasse. Reconstruction of the nest and coordination between individuals in terms. natalensis and cubitermes sp. the theory of stigmergy: Test interpretation of termite constructions. *Social Insects*, 6:41–80, 1959.

[80] Kristina Lerman, Chris Jones, Aram Galstyan, and Maja J Matarić. Analysis of dynamic task allocation in multi-robot systems. *The International Journal of Robotics Research*, 25(3):225–241, 2006.

[81] Marco Dorigo, Eric Bonabeau, and Guy Theraulaz. Ant algorithms and stigmergy. *Future Generation Computer Systems*, 16(8):851–871, 2000.

[82] Guenther Witzany. Communicative coordination in bees. In Guenther Witzany, editor, *Biocommunication of Animals*, pages 135–147. Springer Netherlands, 2014.

[83] Benjamin J. Taylor. An investigation of the role of signals and cues in the coordination of foraging in the social wasps, 2012. Copyright ProQuest, UMI Dissertations Publishing 2012.

[84] Erol Şahin. Swarm robotics: From sources of inspiration to domains of application. In *Swarm robotics*, pages 10–20. Springer, 2005.

[85] Flaminio Borgonovo, Antonio Capone, Matteo Cesana, and Luigi Fratta. Adhoc mac: new mac architecture for ad hoc networks providing efficient and reliable point-to-point and broadcast services. *Wireless Networks*, 10(4):359–366, 2004.

[86] B Gerkey and Maja J Mataric. Are (explicit) multi-robot coordination and multi-agent coordination really so different? In *Proceedings of the AAAI spring symposium on bridging the multi-agent and multi-robotic research gap*, pages 1–3, 2004.

[87] Brian P Gerkey and Maja J Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004.

[88] Vijay Kumar, Daniela Rus, and Gaurav S Sukhatme. Networked robots. In *Springer Handbook of Robotics*, pages 943–958. Springer, 2008.

[89] Peter Stone and Manuela Veloso. Multiagent Systems: A Survey from a Machine Learning Perspective. *Autonomous Robots*, 8(3):345–383, 2000.

[90] Vu Anh Pham and A. Karmouch. Mobile software agents: an overview. *IEEE Communications Magazine*, 36(7):26–37, 1998.

[91] Alberto Sanfeliu, Norihiro Hagita, and Alessandro Saffiotti. Network robot systems. *Robotics and Autonomous Systems*, 56(10):793–797, 2008.

[92] Carol Cheung and Benjamin Grocholsky. Uav-ugv collaboration with a packbot ugv and raven suav for pursuit and tracking of a dynamic target. In *Proc. SPIE*, volume 6962, pages 696216–696216–10, 2008.

[93] Mitsushige Oda. Experiences and lessons learned from the ets-vii robot satellite. In *IEEE International Conference onRobotics and Automation*, volume 1, pages 914–919. IEEE, 2000.

[94] Michael Ackerman, Richard Craft, Frank Ferrante, Mary Kratz, Salah Mandil, and Hasan Sapci. Chapter 6: telemedicine technology. *Telemedicine Journal and e-health*, 8(1):71–78, 2002.

[95] IEEE RAS TC. Networked robots. http://www.ieee-ras.org/networked-robots. [Online; accessed 22-December-2015].

[96] Katherine M Tsui, Munjal Desai, Holly Yanco, Chris Uhlik, et al. Exploring use cases for telepresence robots. In *2011 6th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 11–18. IEEE, 2011.

[97] Pierre-Jean Bristeau, François Callou, David Vissiere, Nicolas Petit, et al. The navigation and control technology inside the ar.drone micro uav. In *18th IFAC world congress*, volume 18(1), pages 1477–1484, 2011.

[98] Dimitrios Koutsonikolas, Saumitra M Das, Y Charlie Hu, Yung-Hsiang Lu, and CS George Lee. Cocoa: Coordinated cooperative localization for mobile multi-robot ad hoc networks. In *26th IEEE International Conference on Distributed Computing Systems Workshops, ICDCS Workshops 2006,*, pages 9–9. IEEE, 2006.

[99] CRISTINA Turcu, CORNEL Turcu, and VASILE Gaitan. Merging the internet of things and robotics. *Recent Researches in Circuits and Systems*, pages 499–504, 2012.

[100] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. A survey on sensor networks. *IEEE Communications magazine*, 40(8):102–114, 2002.

[101] Takayuki Kanda, Masahiro Shiomi, Zenta Miyashita, Hiroshi Ishiguro, and Norihiro Hagita. An affective guide robot in a shopping mall. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 173–180. ACM, 2009.

[102] Akiko Yamazaki, Keiichi Yamazaki, Matthew Burdelski, Yoshinori Kuno, and Mihoko Fukushima. Coordination of verbal and non-verbal actions in human–robot interaction at museums and exhibitions. *Journal of Pragmatics*, 42(9):2398–2414, 2010.

[103] Savioke. Savione: robot butler starts room service deliveries. http://www.savioke.com/about/. [Online; accessed 22-December-2015].

[104] Sara Ljungblad, Jirina Kotrbova, Mattias Jacobsson, Henriette Cramer, and Karol Niechwiadowicz. Hospital robot at work: something alien or an intelligent colleague? In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 177–186. ACM, 2012.

[105] B Mazzolai, V Mattoli, C Laschi, P Salvini, G Ferri, G Ciaravella, and P Dario. Networked and cooperating robots for urban hygiene: the eu funded dustbot project. In *Proceedings of the 5th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 2008.

[106] K. Kamei, S. Nishio, N. Hagita, and M. Sato. Cloud networked robotics. *IEEE Network*, 26(3):28–34, 2012.

[107] Miki Sato, Koji Kamei, Shuichi Nishio, and Norihiro Hagita. The ubiquitous network robot platform: common platform for continuous daily robotic services. In *2011 IEEE/SICE International Symposium on System Integration (SII)*, pages 318–323. IEEE, 2011.

[108] Colin G Harrison, David M Chess, and Aaron Kershenbaum. *Mobile Agents: Are they a good idea?* IBM TJ Watson Research Center Yorktown Heights, New York, 1995.

[109] Abdelkader Outtagarts. Mobile agent-based applications: A survey. *International Journal of Computer Science and Network Security*, 9(11):331–339, 2009.

[110] Butler Lampson. Remote procedure calls. *Lecture Notes in Computer Science*, 105:365–370, 1981.

[111] Steve Vinoski. Corba: integrating diverse applications within distributed heterogeneous environments. *Communications Magazine, IEEE*, 35(2):46–55, 1997.

[112] Danny B. Lange. Mobile objects and mobile agents: The future of distributed computing? In Eric Jul, editor, *ECOOP98 Object-Oriented Programming*, volume 1445 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 1998.

[113] Kurt Rothermel, Fritz Hohl, and Nikolaos Radouniklis. Mobile agent systems: What is missing? *Distributed Applications and Interoperable Systems (DAIS'97), Chapman & Hall*, pages 111–124, 1997.

[114] Liam Cragg and Huosheng Hu. Application of mobile agents to robust teleoperation of internet robots in nuclear decommissioning. In *2003 IEEE International Conference on Industrial Technology*, volume 2, pages 1214–1219. IEEE, 2003.

[115] Liam Cragg and Huosheng Hu. A multi-agent system for distributed control of networked mobile robots. *Measurement and Control*, 38(10):314–319, 2005.

[116] Liam Cragg and Huosheng Hu. Mobile agent approach to networked robots. *The International Journal of Advanced Manufacturing Technology*, 30(9-10):979–987, 2006.

[117] Liam Cragg, Huosheng Hu, and Norbert Voelker. Modularity and mobility of distributed control software for networked mobile robots. In *Software Engineering for Experimental Robotics*, pages 459–484. Springer, 2007.

[118] Yasushi Kambayashi, Yoshikuni Harada, Osamu Sato, and Munehiro Takimoto. Design of an intelligent cart system for common airports. In *Consumer Electronics, 2009. ISCE'09. IEEE 13th International Symposium on*, pages 523–526. IEEE, 2009.

[119] Yasushi Kambayashi, Yuki Tsujimura, Hidemi Yamachi, Munehiro Takimoto, and Hiroshi Yamamoto. Design of a multi-robot system using mobile agents with ant colony clustering. In *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on*, pages 1–10. IEEE, 2009.

[120] Masashi Mizutani, Munehiro Takimoto, and Yasushi Kambayashi. Ant colony clustering using mobile agents as ants and pheromone. In *Intelligent Information and Database Systems*, pages 435–444. Springer, 2010.

[121] Tsukasa Abe, Munehiro Takimoto, and Yasushi Kambayashi. Searching targets using mobile agents in a large scale multi-robot environment. In *Agent and Multi-Agent Systems: Technologies and Applications*, pages 211–220. Springer, 2011.

[122] Jaafar Gaber and Mohamed Bakhouya. Mobile agent-based approach for resource discovery in peer-to-peer networks. In *Agents and Peer-to-Peer Computing*, pages 63–73. Springer, 2008.

[123] Yongsheng Ding and Lei Gao. Macrodynamics analysis of migration behaviors in large-scale mobile agent systems for the future internet. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 41(5):1032–1036, 2011.

[124] W. Wilfred Godfrey and Shivashankar B. Nair. A pheromone based mobile agent migration strategy for servicing networked robots. In Junichi Suzuki and Tadashi Nakano, editors, *Bio-Inspired Models of Network, Information, and Computing Systems*, volume 87 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 533–541. Springer Berlin Heidelberg, 2012.

[125] W. Wilfred Godfrey and Shivashankar B. Nair. Mobile agent cloning for servicing networked robots. In Nirmit Desai, Alan Liu, and Michael Winikoff, editors, *Principles and Practice of Multi-Agent Systems*, volume 7057 of *Lecture Notes in Computer Science*, pages 336–339. Springer Berlin Heidelberg, 2012.

[126] W.Wilfred Godfrey and ShivashankarB. Nair. An immune system based multi-robot mobile agent network. In Peter J. Bentley, Doheon Lee, and Sungwon Jung, editors, *Artificial Immune Systems*, volume 5132 of *Lecture Notes in Computer Science*, pages 424–433. Springer Berlin Heidelberg, 2008.

[127] W Wilfred Godfrey and Shivashankar B Nair. A bio-inspired technique for servicing networked robots. *International Journal of Rapid Manufacturing*, 2(4):258–279, 2011.

[128] W. Wilfred Godfrey, Shashi Shekhar Jha, and Shivashankar B. Nair. On stigmergically controlling a population of heterogeneous mobile agents using cloning resource. In Ngoc Thanh Nguyen, editor, *Transactions on Computational Collective Intelligence XIV*, volume 8615 of *Lecture Notes in Computer Science*, pages 49–70. Springer Berlin Heidelberg, 2014.

[129] W. W. Godfrey, S. S. Jha, and S. B. Nair. On a mobile agent framework for an internet of things. In *2013 International Conference on Communication Systems and Network Technologies (CSNT)*, pages 345–350, 2013.

[130] W Wilfred Godfrey and Shivashankar B Nair. A Mobile Agent Cloning Controller for Servicing Networked Robots. In *Proceedings of 2011 International Conference on Future Information Technology, IPCSIT 2011.*, pages 81–85. IACSIT Press, 2011.

[131] Niels K Jerne. Towards the network theory of the immune system. *Ann. Immunol.(Inst. Pasteur)*, 125:373–389, 1974.

[132] Jatin Matani and Shivashankar B. Nair. Typhon - a mobile agents framework for real world emulation in prolog. In Chattrakul Sombattheera, Arun Agarwal, SibaK Udgata, and Kittichai Lavangnananda, editors, *Multi-disciplinary Trends in Artificial Intelligence*, volume 7080 of *Lecture Notes in Computer Science*, pages 261–273. Springer Berlin Heidelberg, 2011.

[133] Shashi Shekhar Jha and Shivashankar B. Nair. An idiotypic solution sieve for selecting the best performing solutions in real-world distributed intelligence. In *Proceedings of the Third International Conference on Artificial Intelligence, Modelling and Simulation*, AIMS 2015. IEEE Computer Society, 2015.

[134] Shashi Shekhar Jha and Shivashankar B. Nair. On a multi-agent distributed asynchronous intelligence-sharing and learning framework. In Ngoc Thanh Nguyen, editor, *Transactions on Computational Collective Intelligence XVIII*, volume 9240 of *Lecture Notes in Computer Science*, pages 166–200. Springer Berlin Heidelberg, 2015.

[135] Shashi Shekhar Jha, W. Wilfred Godfrey, and Shivashankar B. Nair. Stigmergy-based synchronization of a sequence of tasks in a netowrk of asynchronous nodes. *Cybernetics and Systems:An International Journal*, 45(5):373–406, 2014.

[136] Shashi Shekhar Jha and Shivashankar B. Nair. Orchestrating the sequential execution of tasks by a heterogeneous set of asynchronous mobile agents. In JrgP. Mller, Michael Weyrich, and AnaL.C. Bazzan, editors, *Multiagent System Technologies*, volume 8732 of *Lecture Notes in Computer Science*, pages 103–120. Springer International Publishing, 2014.

[137] David Kotz and Robert S Gray. Mobile agents and the future of the internet. *Operating systems review*, 33(3):7–13, 1999.

[138] Dejan S Milojičić, Fred Douglis, Yves Paindaveine, Richard Wheeler, and Songnian Zhou. Process migration. *ACM Computing Surveys (CSUR)*, 32(3):241–299, 2000.

[139] David Chess, Colin Harrison, and Aaron Kershenbaum. Mobile agents: Are they a good idea? In *Mobile Object Systems Towards the Programmable Internet*, pages 25–45. Springer, 1997.

[140] Agostino Poggi and Michele Tomaiuolo. Mobile agents: Concepts and technologies. *Handbook of Research on Mobility and Computing: Evolving Technologies and Ubiquitous Impacts: Evolving Technologies and Ubiquitous Impacts*, 1:343, 2011.

[141] Danny B Lange and Mitsuru Oshima. Seven good reasons for mobile agents. *Communications of the ACM*, 42(3):88–89, 1999.

[142] Andrzej Bieszczad, Bernard Pagurek, and Tony White. Mobile agents for network management. *IEEE Communications Surveys*, 1(1):2–9, 1998.

[143] Nelson Minar, Matthew Gray, Oliver Roup, Raffi Krikorian, and Pattie Maes. Hive: Distributed agents for networking things. In *First International Symposium on Agent Systems and Applications and Third International Symposium on Mobile Agents*, pages 118–129. IEEE, 1999.

[144] Adam Pridgen and Christine Julien. A secure modular mobile agent system. In *Proceedings of the 2006 international workshop on Software engineering for large-scale multi-agent systems*, pages 67–74. ACM, 2006.

[145] Niklas Borselius. Mobile agent security. *Electronics & Communication Engineering Journal*, 14(5):211–218, 2002.

[146] Joris Claessens, Bart Preneel, and Joos Vandewalle. (how) can mobile agents do secure electronic transactions on untrusted hosts? a survey of the security issues and the current solutions. *ACM Transactions on Internet Technology (TOIT)*, 3(1):28–48, 2003.

[147] Wayne A Jansen. Countermeasures for mobile agent security. *Computer Communications*, 23(17):1667–1676, 2000.

[148] William M Farmer, Joshua D Guttman, and Vipin Swarup. Security for mobile agents: Authentication and state appraisal. In *Computer SecurityESORICS 96*, pages 118–130. Springer, 1996.

[149] Neeran M Karnik and Anand R Tripathi. Design issues in mobile agent programming systems. *Concurrency, IEEE*, 6(3):52–61, 1998.

[150] J Sebastian Terence. Secure route discovery against wormhole attacks in sensor networks using mobile agents. In *2011 3rd International Conference on Trendz in Information Sciences and Computing (TISC)*, pages 110–115. IEEE, 2011.

[151] Timon C Du, Eldon Y Li, and An-Pin Chang. Mobile agents in distributed network management. *Communications of the ACM*, 46(7):127–132, 2003.

[152] Thomas Herlea, Joris Claessens, Bart Preneel, Gregory Neven, Frank Piessens, and Bart De Decker. On securely scheduling a meeting. In *Trusted Information*, pages 183–198. Springer, 2001.

[153] Martyn Fletcher, Robert W Brennan, and Douglas H Norrie. Modeling and reconfiguring intelligent holonic manufacturing systems with internet-based mobile agents. *Journal of Intelligent Manufacturing*, 14(1):7–23, 2003.

[154] Munehiro Fukuda, Koichi Kashiwagi, and Shinya Kobayashi. Agentteamwork: Coordinating grid-computing jobs with mobile agents. *Applied Intelligence*, 25(2):181–198, 2006.

[155] Zehua Zhang and Xuejie Zhang. Realization of open cloud computing federation based on mobile agent. In *IEEE International Conference on Intelligent Computing and Intelligent Systems, ICIS 2009*, volume 3, pages 642–646. IEEE, 2009.

[156] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, pages 1–10. IEEE, 2010.

[157] James White. Telescript technology: An introduction to the language. *General Magic White Paper, General Magic*, 1995.

[158] Robert S Gray. Agent tcl: A flexible and secure mobile-agent system. Technical report, Dartmouth College Hanover, NH, SAD, 1997.

[159] Robert S Gray, George Cybenko, David Kotz, Ronald A Peterson, and Daniela Rus. D'agents: applications and performance of a mobile-agent system. *Pract. Exper. Softw.*, 32(6):543–573, 2002.

[160] Danny B Lange and Oshima Mitsuru. *Programming and Deploying Java Mobile Agents Aglets*. Addison-Wesley Longman Publishing Co., Inc., 1998.

[161] IBM. Aglets: Java based mobile agents. http://aglets.sourceforge.net/. [Online; accessed 23-December-2015].

[162] Holger Peine and Torsten Stolpmann. The architecture of the ara platform for mobile agents. In *Mobile Agents*, pages 50–61. Springer, 1997.

[163] Graham Glass. Objectspace voyagerthe agent orb for java. In *Worldwide Computing and Its ApplicationsWWCA'98*, pages 38–55. Springer, 1998.

[164] Alberto Silva, Miguel Mira Da Silva, and José Delgado. An overview of agentspace: a next-generation mobile agent system. In *Mobile Agents*, pages 148–159. Springer, 1998.

[165] Ichiro Satoh. A mobile agent system - agentspace. http://research.nii.ac.jp/~ichiro/. [Online; accessed 23-December-2015].

[166] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Jade: a fipa2000 compliant agent development environment. In *Proceedings of the fifth international conference on Autonomous agents*, pages 216–217. ACM, 2001.

[167] IEEE Computer Society. The foundation of intelligent physical agents (fipa). http://www.fipa.org/. [Online; accessed 23-December-2015].

[168] Peter Braun, Christian Erfurth, and Wilhem Rossak. *An introduction to the Tracy mobile agent system*. Friedrich-Schiller-Universität, 2000.

[169] Sergio Ilarri, Raquel Trillo, and Eduardo Mena. Springs: A scalable platform for highly mobile agents in distributed computing environments. In *Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, pages 633–637. IEEE Computer Society, 2006.

[170] Bo Chen, Harry H Cheng, and Joe Palen. Mobile-c: a mobile agent platform for mobile c/c++ agents. *Software: Practice and Experience*, 36(15):1711–1733, 2006.

[171] U C Davis Integration Engineering Laboratory. Mobile-c. http://www.mobilec.org/. [Online; accessed 23-December-2015].

[172] Logic Programming Associates. LPA WIN-Prolog. http://www.lpa.co.uk/. [Online; accessed 21-December-2015].

[173] Tushar Semwal, Manoj Bode, Vivek Singh, Shashi Shekhar Jha, and Shivashankar B. Nair. Tartarus: A multi-agent platform for integrating cyber-physical systems and robots. In *Proceedings of the 2nd International Conference of Robotics Society of India - ADVANCES IN ROBOTICS*, AIR 2015. ACM, 2015.

[174] Cornell and Tromso. The tacoma project (troms and cornell moving agents). http://www.tacoma.cs.uit.no/. [Online; accessed 23-December-2015].

[175] Christoph Baumer, Markus Breugst, Sang Choy, and Thomas Magedanz. Grasshoppera universal agent platform based on omg masif and fipa standards. In *First International Workshop on Mobile Agents for Telecommunication Applications (MATA99)*, pages 1–18. Citeseer, 1999.

[176] Anand R Tripathi, Neeran M Karnik, Manish K Vora, Tanvir Ahmed, and Ram D Singh. Mobile agent programming in ajanta. In *19th IEEE International Conference on Distributed Computing Systems*, pages 190–197. IEEE, 1999.

[177] Hairong Qi. Mobile agent framework (maf). http://maf.sourceforge.net/. [Online; accessed 23-December-2015].

[178] Fredrik Kilander, Patrik Werle, and Karin Hansson. Jima-a jini-based infrastructure for active documents and mobile agents. In *In Proceedings of the Personal Computing and Communication (PCC) Workshop*. Citeseer, 1999.

[179] Chien-Liang Fok, Gruia-Catalin Roman, and Chenyang Lu. Agilla: A mobile agent middleware for self-adaptive wireless sensor networks. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 4(3):16, 2009.

[180] Xining Li. Imago: A prolog-based system for intelligent mobile agents. In *Mobile Agents for Telecommunication Applications*, pages 21–30. Springer, 2001.

[181] H.VanDyke Parunak and Sven Brueckner. Concurrent modeling of alternative worlds with polyagents. In *Multi-Agent-Based Simulation VII*, volume 4442 of *Lecture Notes in Computer Science*, pages 128–141. Springer Berlin Heidelberg, 2007.

[182] H Van Dyke Parunak. Interpreting digital pheromones as probability fields. In *Winter Simulation Conference*, pages 1059–1068. Winter Simulation Conference, 2009.

[183] Tom Holvoet, Danny Weyns, and Paul Valckenaers. Patterns of delegate mas. In *Self-Adaptive and Self-Organizing Systems, 2009. SASO'09. Third IEEE International Conference on*, pages 1–9. IEEE, 2009.

[184] R. Claes, T. Holvoet, and D. Weyns. A decentralized approach for anticipatory vehicle routing using delegate multiagent systems. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):364–373, 2011.

[185] Shaza Hanif and Tom Holvoet. Dynamic scheduling of ready mixed concrete delivery problem using delegate mas. In *Advances in Practical Applications of Heterogeneous Multi-Agent Systems, The PAAMS Collection*, pages 146–158. Springer, 2014.

[186] Pattie Maes, Robert H. Guttman, and Alexandros G. Moukas. Agents that buy and sell. *Commun. ACM*, 42(3):81–ff., March 1999.

[187] Mike P. Papazoglou. Agent-oriented technology in support of e-business. *Commun. ACM*, 44(4):71–77, April 2001.

[188] Ryszard Kowalczyk, Mihaela Ulieru, and Rainer Unland. Integrating mobile and intelligent agents in advanced e-commerce: A survey. In *Agent Technologies, Infrastructures, Tools, and Applications for E-Services*, pages 295–313. Springer, 2003.

[189] UP Kulkarni, Padmashree D Desai, J Ahmed, J V Vadavi, and A Yardi. Mobile agent based distributed data mining. In *International Conference on Conference on Computational Intelligence and Multimedia Applications*, volume 2, pages 18–24. IEEE, 2007.

[190] Athanasios Kinalis, Sotiris Nikoletseas, Dimitra Patroumpa, and Jose Rolim. Biased sink mobility with adaptive stop times for low latency data collection in sensor networks. *Information fusion*, 15:56–63, 2014.

[191] Kai Lin, Min Chen, Sherali Zeadally, and Joel JPC Rodrigues. Balancing energy consumption with mobile agents in wireless sensor networks. *Future Generation Computer Systems*, 28(2):446–456, 2012.

[192] Muhammad Arif, Manzoor Illahi, Ahmad Karim, Shahaboddin Shamshirband, Khubaib Amjad Alam, Shahid Farid, Salman Iqbal, Zolkepli Buang, and Valentina Emilia Balas. An architecture of agent-based multi-layer interactive e-learning and e-testing platform. *Quality & Quantity*, pages 1–24, 2014.

[193] Mueen Uddin, Azizah Abdul Rahman, Naeem Uddin, Jamshed Memon, Raed A Alsaqour, and Suhail Kazi. Signature-based multi-layer distributed intrusion detection system using mobile agents. *IJ Network Security*, 15(2):97–105, 2013.

[194] Azzedine Boukerche, Renato B. Machado, Kathia R.L. Juc, Joo Bosco M. Sobral, and Mirela S.M.A. Notare. An agent based and biological inspired real-time intrusion detection and security model for computer network operations. *Computer Communications*, 30(13):2649 – 2660, 2007. Sensor-Actuated NetworksSANETs.

[195] Ismail Butun, Salvatore D Morgera, and Ravi Sankar. A survey of intrusion detection systems in wireless sensor networks. *IEEE Communications Surveys & Tutorials*, 16(1):266–282, 2014.

[196] Umar Manzoor and Samia Nefti. Quiet: A methodology for autonomous software deployment using mobile agents. *Journal of Network and Computer Applications*, 33(6):696 – 706, 2010. Advances on Agent-based Network Management.

[197] Xi Vincent Wang and Xun William Xu. Dimp: an interoperable solution for software integration and product data exchange. *Enterprise Information Systems*, 6(3):291–314, 2012.

[198] Yefim H Kats. Enabling intelligent agents and the semantic web for e-commerce. *International Journal of Computing*, 2(3):153–159, 2014.

[199] H Souraya, Kazar Okba, and Amira Youssef. Integration of mobile agent and semantic web service in a mobile environment. In *Innovative Computing Technology (INTECH), 2012 Second International Conference on*, pages 349–354. IEEE, 2012.

[200] Michal Laclavik, Zoltan Balogh, Marian Babik, and Ladislav Hluchỳ. Agentowl: Semantic knowledge model and agent architecture. *Computing and Informatics*, 25(5):421–439, 2012.

[201] Ichiro Satoh. Building and selecting mobile agents for network management. *Journal of Network and Systems Management*, 14(1):147–169, 2006.

[202] David L Tennenhouse and David J Wetherall. Towards an active network architecture. *ACM SIGCOMM Computer Communication Review*, 37(5):81–94, 2007.

[203] R. Aversa, B. Di Martino, M. Rak, and S. Venticinque. Cloud agency: A mobile agent based cloud system. In *International Conference on Complex, Intelligent and Software Intensive Systems (CISIS)*, pages 132–137, 2010.

[204] Laura Jarvenpaa, Markku Lintinen, Anna-Liisa Mattila, Tommi Mikkonen, Kari Systa, and Jari-Pekka Voutilainen. Mobile agents for the internet of things. In *2013 17th International Conference on System Theory, Control and Computing (ICSTCC)*, pages 763–767. IEEE, 2013.

[205] Yingying Wang, Hehua Yan, Jiafu Wan, and Keliang Zhou. Mobile agents for cps in intelligent transportation systems. In *Advanced Technologies, Embedded and Multimedia for Human-centric Computing*, pages 721–729. Springer, 2014.

[206] Manoj Bode, Shashi Shekhar Jha, and Shivashankar B. Nair. On movement of emergency services amidst urban traffic. *EAI Endorsed Transactions on Ambient Systems*, 15(8), 12 2015.

[207] Manoj Bode, Shashi Shekhar Jha, and Shivashankar B. Nair. A mobile agent based autonomous partial green corridor discovery and maintenance mechanism for emergency services amidst urban traffic. In *Proceedings of the First International Conference on IoT in Urban Space*, URB-IOT '14, pages 13–18. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2014.

[208] Chao-Lin Wu, Chun-Feng Liao, and Li-Chen Fu. Service-oriented smart-home architecture based on osgi and mobile-agent technology. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(2):193–205, 2007.

[209] Umar Manzoor and Samia Nefti. idetect: content based monitoring of complex networks using mobile agents. *Applied Soft Computing*, 12(5):1607–1619, 2012.

[210] Imen Brahmi, Sadok Ben Yahia, and Pascal Poncelet. Mad-ids: novel intrusion detection system using mobile agents and data mining approaches. In *Intelligence and Security Informatics*, pages 73–76. Springer, 2010.

[211] Oscar Urra, Sergio Ilarri, Thierry Delot, and Eduardo Mena. Mobile agents in vehicular networks: Taking a first ride. In *Advances in Practical Applications of Agents and Multiagent Systems*, pages 119–124. Springer, 2010.

[212] ZhenJiang Li, Cheng Chen, Ka Wang, et al. Cloud computing for agent-based urban transportation systems. *IEEE Intelligent Systems*, 26(1):73–79, 2011.

[213] Liang Zhou and Han-Chieh Chao. Multimedia traffic security architecture for the internet of things. *IEEE Network*, 25(3):35–40, 2011.

[214] Sergio González-Valenzuela, Min Chen, and Victor CM Leung. Programmable middleware for wireless sensor networks applications using mobile agents. *Mobile Networks and Applications*, 15(6):853–865, 2010.

[215] Min Chen, Wei Cai, Sergio Gonzalez, and Victor CM Leung. Balanced itinerary planning for multiple mobile agents in wireless sensor networks. In *Ad Hoc Networks*, pages 416–428. Springer, 2010.

[216] Chuan-Jun Su and Chang-Yu Chiang. Pervasive community care platform: Ambient intelligence leveraging sensor networks and mobile agents. *International Journal of Systems Science*, 45(4):778–797, 2014.

[217] Teemu Leppnen, Jos lvarez Lacasia, Yoshito Tobe, Kaoru Sezaki, and Jukka Riekki. Mobile crowdsensing with mobile agents. *Autonomous Agents and Multi-Agent Systems*, pages 1–35, 2015.

[218] Xiang-Jun Shen, Lu Liu, Zheng-Jun Zha, Pei-Ying Gu, Zhong-Qiu Jiang, Ji-Ming Chen, and John Panneerselvam. Achieving dynamic load balancing through mobile agents in small world p2p networks. *Computer Networks*, 75:134–148, 2014.

[219] Tetsuo Otani and Hiromu Kobayashi. A scada system using mobile agents for a next-generation distribution system. *IEEE Transactions on Power Delivery*, 28(1):47–57, 2013.

[220] Stephen S Nestinger, Bo Chen, and Harry H Cheng. A mobile agent-based framework for flexible automation systems. *IEEE/ASME Transactions on Mechatronics*, 15(6):942–951, 2010.

[221] Nelson Minar, KwindlaHultman Kramer, and Pattie Maes. Cooperating mobile agents for dynamic network routing. In AlexL.G. Hayzelden and John Bigham, editors, *Software Agents for Future Communication Systems*, pages 287–304. Springer Berlin Heidelberg, 1999.

[222] Dipankar Dasgupta, Senhua Yu, and Fernando Nino. Recent advances in artificial immune systems: Models and applications. *Applied Soft Computing*, 11(2):1574 – 1587, 2011.

[223] Leandro Nunes De Castro and Jonathan Timmis. *Artificial immune systems: a new computational intelligence approach*. Springer Verlag, 2002.

[224] Ali Raza and Benito R. Fernandez. Immuno-inspired robotic applications: A review. *Applied Soft Computing*, 37:490 – 505, 2015.

[225] Samir Kr Borgohain and Shivashankar B. Nair. An immuno-inspired approach towards sentence generation. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 97–104, New York, NY, USA, 2015. ACM.

[226] Akshat Kumar and Shivashankar B. Nair. An artificial immune system based approach for english grammar checking. In Leandro Nunes de Castro, Fernando Jos Von Zuben, and Helder Knidel, editors, *Artificial Immune Systems*, volume 4628 of *Lecture Notes in Computer Science*, pages 348–357. Springer Berlin Heidelberg, 2007.

[227] J Doyne Farmer, Norman H Packard, and Alan S Perelson. The immune system, adaptation, and machine learning. *Physica D: Nonlinear Phenomena*, 22(1):187–204, 1986.

[228] Dipankar Dasgupta. *Artificial Immune Systems and Their Applications*. Springer Publishing Company, Incorporated, 2014.

[229] Julie Greensmith, Amanda Whitbrook, and Uwe Aickelin. Artificial immune systems. In *Handbook of Metaheuristics*, pages 421–448. Springer, 2010.

[230] M. Satyanarayanan. Pervasive computing: vision and challenges. *IEEE Personal Communications*, 8(4):10–17, 2001.

[231] Dipankar Dasgupta. Immunity-based intrusion detection system: a general framework. In *Proc. of the 22nd NISSC*, volume 1, pages 147–160, 1999.

[232] Leandro N De Castro and Fernando J Von Zuben. Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation*, 6(3):239–251, 2002.

[233] Paul K. Harmer, Paul D. Williams, Gregg H. Gunsch, and Gary B. Lamont. An artificial immune system architecture for computer security applications. *IEEE transactions on Evolutionary computation,*, 6(3):252–280, 2002.

[234] Scott A DeLoach. agentMom Users Manual. *Air Force Institute of Technology*, 2000.

[235] Leandro N De Castro and Jon Timmis. Artificial immune systems: a novel approach to pattern recognition. Technical report, University of Paisley, 2002.

[236] Amanda M Whitbrook, Uwe Aickelin, and Jonathan M Garibaldi. Two-timescale learning using idiotypic behaviour mediation for a navigating mobile robot. *Applied Soft Computing*, 10(3):876–887, 2010.

[237] Mingxin Yuan, Sunan Wang, Canyang Wu, and Naijian Chen. A novel immune network strategy for robot path planning in complicated environments. *Journal of Intelligent & Robotic Systems*, 60(1):111–131, 2010.

[238] M. Bakhouya, M. Nemiche, and J. Gaber. An adaptive regulation approach of mobile agent population size in distributed systems. *International Journal of Intelligent Systems*, 00:1–16, 2015.

[239] Shashi Shekhar Jha, Kunal Shrivastava, and Shivashankar B. Nair. On emulating real-world distributed intelligence using mobile agent based localized idiotypic networks. In Rajendra Prasath and T. Kathirvalavakumar, editors, *Mining Intelligence and Knowledge Exploration*, volume 8284 of *Lecture Notes in Computer Science*, pages 487–498. Springer International Publishing, 2013.

[240] Robson A Santana, Murilo Rebelo Pontes, and Carmelo JA Bastos-Filho. A multiple objective particle swarm optimization approach using crowding distance and roulette wheel. In *Ninth International Conference on Intelligent Systems Design and Applications, ISDA'09*, pages 237–242. IEEE, 2009.

[241] Shashi Shekhar Jha and S. B. Nair. A logic programming interface for multiple robots. In *2012 3rd National Conference on Emerging Trends and Applications in Computer Science (NCETACS)*, pages 152–156, 2012.

[242] Richard W Hamming. Error detecting and error correcting codes. *Bell System technical journal*, 29(2):147–160, 1950.

[243] Ellouise Leadbeater and Lars Chittka. Social learning in insectsfrom miniature brains to consensus building. *Current Biology*, 17(16):R703–R713, 2007.

[244] R. Dukas. Insect social learning. In Michael D. BreedJanice Moore, editor, *Encyclopedia of Animal Behavior*, pages 176 – 179. Academic Press, Oxford, 2010.

[245] P.J.A.M. Korst and H.H.W. Velthuis. The nature of trophallaxis in honeybees. *Insectes Sociaux*, 29(2):209–221, 1982.

[246] Nigel R Franks and Tom Richardson. Teaching in tandem-running ants. *Nature*, 439(7073):153–153, 2006.

[247] H. Van Dyke Parunak. "Go to the ant": Engineering principles from natural multi-agent systems. *Annals of Operations Research*, 75(0):69–101, 1997.

[248] Owen E Holland. Multiagent systems: Lessons from social insects and collective robotics. In *the 1996 AAAI Spring Symposium on Adaptation, Coevolution and Learning in Multiagent Systems*, pages 57–62, 1996.

[249] VincentA. Cicirello and StephenF. Smith. Wasp-like agents for distributed factory coordination. *Autonomous Agents and Multi-Agent Systems*, 8(3):237–266, 2004.

[250] Andrew Garland and Richard Alterman. Autonomous agents that learn to better coordinate. *Autonomous Agents and Multi-Agent Systems*, 8(3):267–301, 2004.

[251] Dominik Fisch, Martin Jnicke, Edgar Kalkowski, and Bernhard Sick. Learning from others: Exchange of classification rules in intelligent distributed systems. *Artificial Intelligence*, 187188(0):90 – 114, 2012.

[252] Paul Erdős and A Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci*, 5:17–61, 1960.

[253] Erick Cantú-Paz. A survey of parallel genetic algorithms. *Calculateurs paralleles, reseaux et systems repartis*, 10(2):141–171, 1998.

[254] James Kennedy. Particle swarm optimization. In *Encyclopedia of Machine Learning*, pages 760–766. Springer, 2010.

[255] Peng Zhao, S. Suryanarayanan, and M.G. Simoes. An energy management system for building structures using a multi-agent decision-making control methodology. *IEEE Transactions on Industry Applications*, 49(1):322–330, Jan 2013.

[256] Lucilla Giannetti, FranciscoP. Maturana, and FrederickM. Discenzo. Agent-based control of a municipal water system. In Michael Pchouek, Paolo Petta, and LszlZsolt Varga, editors, *Multi-Agent Systems and Applications IV*, volume 3690 of *Lecture Notes in Computer Science*, pages 500–510. Springer Berlin Heidelberg, 2005.

[257] Frauke Oldewurtel, David Sturzenegger, and Manfred Morari. Importance of occupancy information for building climate control. *Applied Energy*, 101(0):521 – 532, 2013. Sustainable Development of Energy, Water and Environment Systems.

[258] O. Beaumont, A. Legrand, and Y. Robert. The master-slave paradigm with heterogeneous processors. *IEEE Transactions on Parallel and Distributed Systems*, 14(9):897–908, Sept 2003.

[259] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm intelligence: from natural to artificial systems*. OUP USA, 1999.

[260] S. Camazine, J.L. Deneubourg, N.R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau. *Self-organisation in biological systems*. Princeton University Press, 2001.

[261] Nigel R Franks, Jean-Louis Deneubourg, et al. Self-organizing nest construction in ants: individual worker behaviour and the nest's dynamics. *Animal Behaviour*, 54(4):779–796, 1997.

[262] Mitchel Resnick. *Turtles, termites, and traffic jams: Explorations in massively parallel microworlds*. The MIT Press, 1997.

[263] J. Deneubourg. Application de l'ordre par fluctuations a la description de certaines tapes de la construction du nid chez les termites. *Insectes Sociaux*, 24:117–130, 1977.

[264] P.N. Kugler and M.T. Turvey. *Information, natural law, and the self-assembly of rhythmic movement*. Resources for ecological psychology. L. Erlbaum Associates, 1987.

[265] Peter N. Kugler, Robert E. Shaw, Kim J. Vincente, and Jeffrey Kinsella-Shaw. Inquiry into intentional systems i: Issues in ecological physics. *Psychological Research*, 52:98–121, 1990.

[266] G. Theraulaz and E. Bonabeau. Modelling the collective building of complex architectures in social insects with lattice swarms. *Journal of Theoretical Biology*, 177:381–400, 1995.

[267] Edward O Wilson. The relation between caste ratios and division of labor in the ant genus pheidole (hymenoptera: Formicidae). *Behavioral Ecology and Sociobiology*, 16(1):89–98, 1984.

[268] Gene E Robinson. Regulation of division of labor in insect societies. *Annual review of entomology*, 37(1):637–665, 1992.

[269] Samuel N Beshers and Jennifer H Fewell. Models of division of labor in social insects. *Annual review of entomology*, 46(1):413–440, 2001.

[270] NR Franks. *Ants*, pages 29 – 32. Academic Press, 2003.

[271] Elva JH Robinson, Ofer Feinerman, and Nigel R Franks. Flexible task allocation and the organization of work in ants. *Proceedings of the Royal Society B: Biological Sciences*, 276(1677):4373–4380, 2009.

[272] B. Hölldobler and E.O. Wilson. *The superorganism*. W W Norton & Company Incorporated, 2009.

[273] Chris Tofts and Nigel R Franks. Doing the right thing: ants, honeybees and naked mole-rats. *Trends in Ecology & Evolution*, 7(10):346–349, 1992.

[274] Simon K Robson and Samuel N Beshers. Division of labour and foraging for work: simulating reality versus the reality of simulations. *Animal behaviour*, 53(1):214–218, 1997.

[275] Zachary Mason. Programming with stigmergy: using swarms for construction. In *Proceedings of the eighth international conference on Artificial life*, ICAL 2003, pages 371–374, Cambridge, MA, USA, 2003. MIT Press.

[276] Justin Werfel, Yaneer Bar-Yam, and Radhika Nagpal. Building patterned structures with robot swarms. In *Proceedings of the 19th international joint conference on Artificial intelligence*, IJCAI'05, pages 1495–1502, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[277] Michael JB Krieger, Jean-Bernard Billeter, and Laurent Keller. Ant-like task allocation and recruitment in cooperative robots. *Nature*, 406(6799):992–995, 2000.

[278] William Agassounon and Alcherio Martinoli. Efficiency and robustness of threshold-based distributed allocation algorithms in multi-agent systems. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3*, pages 1090–1097. ACM, 2002.

[279] Frederick Ducatelle, Alexander Förster, G Di Caro, and Luca Maria Gambardella. New task allocation methods for robotic swarms. In *9th IEEE/RAS Conference on Autonomous Robot Systems and Competitions*, 2009.

[280] Roderich Groß, Shervin Nouyan, Michael Bonani, Francesco Mondada, and Marco Dorigo. Division of labour in self-organised groups. In *From Animals to Animats 10*, pages 426–436. Springer, 2008.

[281] Chris Jones and Maja J Mataric. Adaptive division of labor in large-scale minimalist multi-robot systems. In *Proceedings of 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS 2003).*, volume 2, pages 1969–1974. IEEE, 2003.

[282] Yongming Yang, Changjiu Zhou, and Yantao Tian. Swarm robots task allocation based on response threshold model. In *4th International Conference on Autonomous Robots and Agents, 2009. ICARA 2009.*, pages 171–176. IEEE, 2009.

[283] Dandan Zhang, Guangming Xie, Junzhi Yu, and Long Wang. Adaptive task assignment for multiple mobile robots via swarm intelligence approach. *Robotics and Autonomous Systems*, 55(7):572–588, 2007.

[284] Eric Bonabeau, Guy Theraulaz, and Jean-Louis Deneubourg. Fixed response thresholds and the regulation of division of labor in insect societies. *Bulletin of Mathematical Biology*, 60(4):753–807, 1998.

[285] Jean-Louis Deneubourg, Simon Goss, Jean-Michel Pasteels, Dominique Fresneau, and Jean-Paul Lachaud. Self-organization mechanisms in ant societies (ii): learning in foraging and division of labor. *From individual to collective behavior in social insects*, page 177, 1987.

[286] Thomas H Labella, Marco Dorigo, and Jean-Louis Deneubourg. Division of labor in a group of robots inspired by ants' foraging behavior. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 1(1):4–25, 2006.

[287] Wenguo Liu, Alan Winfield, Jin Sa, Jie Chen, and Lihua Dou. Strategies for energy optimisation in a swarm of foraging robots. In *Swarm Robotics*, pages 14–26. Springer, 2007.

[288] Wenguo Liu, Alan FT Winfield, Jin Sa, Jie Chen, and Lihua Dou. Towards energy optimization: Emergent task allocation in a swarm of foraging robots. *Adaptive Behavior*, 15(3):289–305, 2007.

[289] E Bonabeau, G Theraulaz, and J L Deneubourg. The synchronization of recruitment-based activities in ants. *Bio Systems*, 45(3):195–211, March 1998.

[290] Nigel R Franks, Steve Bryant, Richard Griffiths, and Lia Hemerik. Synchronization of the behaviour within nests of the ant *Leptothorax acervorum* (fabricius)–I. Discovering the phenomenon and its relation to the level of starvation. *Bulletin of mathematical biology*, 52(5):597–612, 1990.

[291] John Buck. Synchronous rhythmic flashing of fireflies. ii. *Quarterly Review of Biology*, pages 265–289, 1988.

[292] B Krafft and A Pasquet. Synchronized and rhythmical activity during the prey capture in the social spideranelosimus eximius (araneae, theridiidae). *Insectes Sociaux*, 38(1):83–90, 1991.

[293] Patricia Backwell, Michael Jennions, Neville Passmore, and John Christy. Synchronized courtship in fiddler crabs. *Nature*, 391(6662):31–32, 1998.

[294] J Delgado and R V Solé. Self-synchronization and task fulfilment in ant colonies. *Journal of theoretical biology*, 205(3):433–41, August 2000.

[295] V. Trianni and S. Nolfi. Self-Organizing Sync in a Robotic Swarm: A Dynamical System View. *IEEE Transactions on Evolutionary Computation*, 13(4):722–741, August 2009.

[296] Sylvain Chevallier, Nicolas Bredeche, Hélène Paugam-Moisy, and Michèle Sebag. Emergence of temporal and spatial synchronous behaviors in a foraging swarm. In *European Conference on Artificial Life*, 2011.

[297] Giulio Zecca, Paul Couderc, Michel Banatre, and Roberto Beraldi. Swarm robot synchronization using rfid tags. In *IEEE International Conference on Pervasive Computing and Communications, PerCom 2009.*, pages 1–4. IEEE, 2009.

[298] EJ. Gibson. The development of perception as an adaptive process. *American Scientist*, 58:98–107, 1970.

[299] Hoang-Nam Chu, A. Glad, O. Simonin, F. Sempe, A. Drogoul, and F. Charpillet. 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. ICTAI 2007.*, volume 1, pages 442 –449, oct. 2007.

[300] Shivashankar B Nair, W W Godfrey, and D H Kim. On realizing a multi-agent emotion engine. *International Journal of Synthetic Emotions (IJSE)*, 2(2):1–27, 2011.

[301] Guy Theraulaz, Eric Bonabeau, and JN Denuebourg. Response threshold reinforcements and division of labour in insect societies. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 265(1393):327–332, 1998.

[302] Wolfgang Kiess and Martin Mauve. A survey on real-world implementations of mobile ad-hoc networks. *Ad Hoc Networks*, 5(3):324–339, 2007.

[303] Jerry Banks. Introduction to simulation. In *Simulation Conference Proceedings, 1999 Winter*, volume 1, pages 7–13. IEEE, 1999.

[304] Md Omar Faruque Sarker. *Self-regulated Multi-Robot Task Allocation*. PhD thesis, PhD thesis, University of Wales, Newport, UK, 2010.

[305] G. Ayorkor Korsah, Anthony Stentz, and M. Bernardine Dias. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, 32(12):1495–1512, 2013.

[306] M.B. Dias, Robert Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006.

[307] R.G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, 29(12):1104–1113, 1980.

[308] S.C. Botelho and R. Alami. M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1234–1239, 1999.

[309] B.P. Gerkey and M.J. Mataric. Sold!: auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768, 2002.

[310] Robert Zlot and Anthony Stentz. Market-based multirobot coordination for complex tasks. *The International Journal of Robotics Research*, 25(1):73–101, 2006.

[311] P.M. Shiroma and M.F.M. Campos. Comutar: A framework for multi-robot coordination and task allocation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, pages 4817–4824, 2009.

[312] Kai Zhang, Emmanuel G. Collins, Jr., and Dongqing Shi. Centralized and distributed task allocation in multi-robot teams via a stochastic clustering auction. *ACM Trans. Auton. Adapt. Syst.*, 7(2):21:1–21:22, 2012.

[313] Soheil Keshmiri and Shahram Payandeh. Multi-robot, dynamic task allocation: a case study. *Intelligent Service Robotics*, 6(3):137–154, 2013.

[314] Maitreyi Nanjanath and Maria Gini. Repeated auctions for robust task execution by a robot team. *Robotics and Autonomous Systems*, 58(7):900 – 909, 2010.

[315] Dong hyun Lee, S.A. Zaheer, and Jong-Hwan Kim. Ad hoc network-based task allocation with resource-aware cost generation for multirobot systems. *IEEE Transactions on Industrial Electronics*, 61(12):6871–6881, 2014.

[316] Barry Brian Werger and Maja J. Mataric. Broadcast of local eligibility: Behavior-based control for strongly cooperative robot teams. In *Proceedings of the Fourth International Conference on Autonomous Agents*, AGENTS '00, pages 21–22, 2000.

[317] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall, 4th edition, 2002.

[318] W. Wilfred Godfrey. *Mobile Agent based Bio-inspired Mechanisms for Servicing Networked Robots*. PhD thesis, Department of Computer Science & Engineering, Indian Institute of Technology Guwahati, Guwahati, Assam, India, 2012.

Publications

Journals

- **Shashi Shekhar Jha**, Shivashankar B. Nair, *On a Multi-Agent Distributed Asynchronous Intelligence-Sharing and Learning Framework*, Transactions on Computational Collective Intelligence XVIII, Lecture Notes in Computer Science, Vol. 9240, Chapter 9, pp. 166-200, 2015, Springer Berlin Heidelberg.

- **Shashi Shekhar Jha**, W. W. Godfrey and Shivashankar B. Nair, *Stigmergy based Synchronization of a Sequence of Tasks in a Network of Asynchronous Nodes*, Cybernetics & Systems: An International Journal, Vol. 45, Issue 5, pp. 373-406, Taylor & Francis Publn.

- W. W. Godfrey, **Shashi Shekhar Jha** and Shivashankar B. Nair, *On Stigmergically Controlling a Population of Heterogeneous Mobile Agents Using Cloning Resource*, Transactions on Computational Collective Intelligence XIV, Lecture Notes in Computer Science, 2014, pp. 49-70, Springer-Verlag Berlin Heidelberg.

- Manoj Bode, **Shashi Shekhar Jha**, Shivashankar B. Nair, *On Movement of Emergency Services amidst Urban Traffic*, EAI Endorsed Transactions on Ambient Systems 15(8): e4, Dec. 2015, EAI.

- **Shashi Shekhar Jha**, Shivashankar B. Nair, *TANSA: Task Allocation using Nomadic Soft Agents for Multi-Robot Systems*, IEEE/CAA Journal of Automatica Sinica, Special Issue on Human-Centered Intelligent Robots: Issues And Challenges, 2016 (**Major Revision Submitted**).

Conferences

- **Shashi Shekhar Jha** and Shivashankar Nair, *An Idiotypic Solution Sieve for Selecting the Best Performing Solutions in Real-World Distributed Intelligence*, IEEE 3rd International Conference on Artificial Intelligence, Modelling and Simulation, AIMS 2015, 2-4 December 2015, Malaysia. [To appear in IEEE Xplore]

- **Shashi Shekhar Jha** and Shivashankar Nair, *Orchestrating the Sequential Execution of Tasks by a Heterogeneous set of Asynchronous Mobile Agents*, [12th German Conference on Multiagent System Technologies, Stuttgart, Germany, 23-25th September 2014], Multiagent System Technologies, Lecture Notes in Computer Science, Volume 8732, 2014, pp 103-120, Springer.

- **Shashi Shekhar Jha**, *On Realizing an intelligent Internet of Things using Intelligent Agents*, MATES Doctoral Consortium 2014, Stuttgart, Germany, pp. 23-30, IfI Technical Report Series (IfI-14-04).

- **Shashi Shekhar Jha**, Kunal Shrivastava and Shivashankar Nair, *On Emulating Real-world Distributed Intelligence using Mobile Agent based Localized Idiotypic Networks*, The First International Conference on Mining Intelligence and Knowledge Exploration (MIKE 2013), Virudhunagar, India, LNAI, Vol. 8284, pp. 487-498, Springer International Publishing.
- **Shashi Shekhar Jha** and Shivashankar Nair, *A logic programming interface for multiple robots*, 3rd National Conference on Emerging Trends and Applications in Computer Science (NCETACS), 2012, pp.152,156, 30-31 March 2012, IEEE.
- W.W. Godfrey, **Shashi Shekhar Jha**, Shivashankar B. Nair, *On A Mobile Agent Framework for an Internet of Things*, Proceedings of the International Conference on Communication System and Technologies, CSNT 2013, Gwalior, India, pp. 345-350, IEEE.
- Manoj Bode, **Shashi Shekhar Jha**, Shivashankar B. Nair, *A Mobile Agent based Autonomous Partial Green Corridor Discovery and Maintenance for Emergency Services amidst Urban Traffic*, In Proceedings of the First International Conference on IoT in Urban Space 2014 (URB-IOT 14), Rome, Italy. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, pages 13-18, ACM.

Patents Filed

- Shivashankar B. Nair, **Shashi Shekhar Jha**, W.W. Godfrey, *A Technique for Synchronizing Task execution by a set of Asynchronous Nodes*, No. 625/KOL/2012, dated 31/05/2012, Patent Pending.
- Shivashankar B. Nair, W.W. Godfrey, **Shashi Shekhar Jha**, *A Mobile Agent Framework for an Internet of Things*, No. 674/KOL/2012, dated 18/06/2012, Patent Pending.

Other publications (Not related to thesis work)

- **Book Chapter**: Saurabh K. Singh, **Shashi Shekhar Jha** and Shivashankar B. Nair, *On Realizing Emotional Memories*, Handbook of Research on Synthesizing Human Emotion in Intelligent Systems and Robotics, Editors Jordi Vallverd et al., Chapter 5, pages 116-151, IGI Publications
- **Shashi Shekhar Jha**, Shrinivasa Naika C.L. and Shivashankar Nair, *Driving Robots Using Emotions*, Transactions on Computational Science XXI, Special Issue on Innovations in Nature-Inspired Computing and Applications, Lecture Notes in Computer Science Vol. 8160, 2013, pp 64-89, Springer.
- **Shashi Shekhar Jha**, Shrinivasa Naika C. L., Vibhor Nikhra and Shivashankar B. Nair, *On Using Emotions in Decision-Making by Situated Robots*, Proceedings of the 12th International Conference on Hybrid Intelligent Systems (HIS 2012), 4-7 December, 2012, Pune, India, pp. 372-377, IEEE Xplore.
- Tushar Semwal, Nikhil S., **Shashi Shekhar Jha**, Shivashankar B. Nair, TARTARUS: A Multi Agent Platform for Bridging the gap between Cyber and Physical Systems, Proceedings of the 2016 Autonomous Agents and Multi Agent Systems Conference (AAMAS-2016), International Foundation for Autonomous Agents and Multiagent Systems, pp. 1493-1495, Singapore.

- Tushar Semwal, Manoj Bode, Vivek Singh, **Shashi Shekhar Jha**, Shivashankar B. Nair, *Tartarus: A Multi-Agent Platform for Integrating Cyber-Physical Systems and Robots*, ADVANCES IN ROBOTICS 2nd International Conference of Robotics Society of India, (ACM In co-operation & Korea Robotics Society) 2nd to 4th July, 2015, Goa, India.

- Kunal Shrivastava, **Shashi Shekhar Jha** and Shivashankar Nair, *Autonomous Mobile Robot Navigation using an Artificial Immune System*, Proceedings of Conference on Advances In Robotics (AIR 13), International Conference of the Robotics Society of India, Article 82, 7 pages, ACM.

- Saurabh Prajapati, Shrinivasa Naika C.L., **Shashi Shekhar Jha** and Shivashankar Nair, *On Rendering Emotions on a Robotic Face*, Proceedings of Conference on Advances In Robotics (AIR 13) , International Conference of the Robotics Society of India, Article 19, 7 pages, ACM,.

Vitae

Shashi Shekhar Jha joined the PhD programme at the Department of Computer Science and Engineering (CSE) of Indian Institute of Technology (IIT) Guwahati, India in July 2010 where he was affiliated to the Robotics Lab. of the Department of CSE. Prior to joining PhD, he did his Bachelors of Engineering degree in Information Technology from Shri Vaishnav Institute of Technology and Science, Indore, India. He was awarded the Tata Consultancy Services (TCS) *Research Fellowship* under TCS-RSP scheme during his PhD. He was also awarded an international travel grant in October 2014 by the Department of Science and Technology, India for one of his papers. He has served as a member of Departmental Postgraduate Programme Committee and Departmental Disciplinary Committee in 2013-2014 and 2012-2013 respectively at the Department of CSE, IIT Guwahati. In addition, he was the founding Doctoral representative at the Students' Academic Board of IIT Guwahati during 2014-2015. A *certificate of merit* from Central Board of Secondary Education, India has been awarded to him in 2006. He has keen interests in pursuing the field of robotics and practical aspects of artificial intelligence. His contemporary research interests includes robotics, multi-agent systems, bio-inspired algorithms and reinforcement learning. He enjoys reading books, playing badminton and travelling to places where nature has bestowed its bounty.

Contact Information

Email	:	j.shashi@iitg.ernet.in, ss.jha@hotmail.com
Web	:	http://iitg.ernet.in/stud/j.shashi/
Address	:	Q.No. E/101, Miners Colony, Jamuna Colliery, Distt.-Anuppur, Madhya Pradesh (M.P.) - 484444, INDIA