Communication Strategies for Self-regulated Multi-robot Task Allocation

Md Omar Faruque Sarker

University of Wales, Newport

Email: Mdomarfaruque.Sarker@students.newport.ac.uk

June 16, 2010

Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

1.1 Multi-robot task allocation (MRTA)

Robotic researchers generally agree that multiple robots can perform complex and distributed tasks more conveniently. Multi-robot systems (MRS) can provide improved performance, fault-tolerance and robustness through parallelism and redundancy (Arkin 1998, Parker & Tang 2006, Mataric 2007). However, in order to get potential benefits of MRS in any application domain, we need to solve a common research challenge i.e., *multi-robot task allocation* (MRTA) (Gerkey & Mataric 2004). MRTA can also be called as *division of labour* (DoL) analogous to DoL in biological social insect and human societies (hereafter the term MRTA is used to denote an instance of social DoL). It is generally identified as the question of assigning tasks in an appropriate time to the appropriate robots considering the changes of the environment and/or the performance of other team members. This is a NP-hard optimal assignment problem where optimum solutions can not be found quickly for large complex problems (Gerkey & Mataric 2003, Parker 2008). The complexities of MRTA arise from the fact that there is no central planner or coordinator for task assignments and the robots are limited to sense, to communicate and to interact locally. None of them has the complete knowledge of the past, present or future actions of other robots. Moreover, they don't have the complete view of the world state. The computational and communication bandwidth requirements also restrict the solution quality of the problem (Lerman et al. 2006).

Researchers from multi-robot or multi-agent systems, operations research and other disciplines have approached the MRTA or task-allocation in multi-agents issue in many different ways. Traditionally task allocation in a multi-agent systems has been divided into two major categories: 1) Predefined (off-line) and 2) Emergent (real-time) task-allocation (Shen et al. 2001). However predefined task- allocation approach fails to scale well as the number of tasks and robots becomes large, e,g,, more than 10 (Lerman et al. 2006). On the other hand emergent task-allocation approach relies on the emergent group behaviours e.g., (Kube & Zhang 1993), such as emergent cooperation (Lerman et al. 2006), adaptation rules (Liu et al. 2007) etc. They are more robust and scalable to large team size. However most of the robotic researchers found that emergent task-allocation approach is difficult to design, to analyse formally and to implement in real robots. The solutions from these systems are also sub-optimal. It is also difficult to predict exact behaviours of robots and overall system performance.

Within the context of the Engineering and Physical Sciences Research Council (EPSRC) project, "Defying the Rules: How Self-regulatory Systems Work", we have proposed to solve the above mentioned MRTA problem in a new way (Arcaute et al. 2008). Our approach is inspired from the studies of emergence of task-allocation in both biological insect societies and human social systems. Biological studies show that a large number of animal as well as human social systems grow, evolve and generally continue functioning well by the virtue of their individual self-regulatory task-allocation systems. The amazing abilities of biological organisms to change, to respond to unpredictable environments, and to adapt over time lead them to sustain life through biological functions such as self-recognition, self-recovery, self-growth etc. It is interesting to note that in animal societies task-allocation has been accomplished years after years without a central authority or an explicit planning and coordinating element. Direct peer-to-peer (P2P) and indirect communication such as stigmergy is used to exchange information among individuals (Camazine et al. 2001). The decentralized self-growth of Internet and its bottom-up interactions of millions of users around the globe present us similar evidences of task-allocation in human social systems (Andriani & Passiante 2004). These interactions of individuals happen in the absence of or in parallel with strict hierarchy. Moreover from the study of sociology e.g., (Sayer & Walker 1992), cybernetics e.g., (Beer 1981), strategic management e.g., (Kogut 2000) and related other disciplines we have found that decentralized self-regulated systems exist in nature and in man-made systems which can grow and achieve self-regulated division of labour over time.

From the above mentioned multi-disciplinary studies of various complex systems, we believe that a set of generic rules can govern the self-regulated task-allocation in MRS. Primarily these rules should deal with the issue of deriving local control rules for facilitating the task-allocation of an entire robot team.

The outcome of our research can be applied to solve generic task-allocation problem in numerous multi-agent systems. As an example, our technique can be useful in automated manufacturing (AM) which faces all the existing challenges of traditional centralized and sequential manufacturing processes such as, insufficiently flexible to respond and adapt changes in production styles of high-mix low-volume production environments (Shen et al. 2006). We believe that our approach can help AM industries to overcome many of these challenging issues, such as flexibility to change the manufacturing plant layouts on-the-fly, adaptability for high variation in product styles, quantities, and active manufacturing resources e.g., robots, AGVs etc.

1.2 Communications for self-regulated task-allocation

In MRS research, robotic researchers have been using various forms of communications e.g., (Bonabeau et al. 1999, Labella 2007). Two widely used forms of communications are: 1) direct or explicit communication and 2) indirect or implicit communication. *Direct communication* is an intentional communicative act of message passing that aims at one or more particular receiver(s) (Mataric 1998). It typically exchanges information through physical signals. In contrast, indirect communication, sometimes termed as *stigmergic* in biological literature, happens as a form of modifying the environment (e.g., pheromone dropping by ants) (Bonabeau et al. 1999). In ordinary sense, this is an observed behaviour and many robotic researchers call it as *no communication* (Labella 2007). In order to avoid ambiguity, by the term *self-regulated MRTA* (or *MRTA* for short) we refer to those

MRS where robots can exhibit most common self-regulatory properties (Bonabeau et al. 1999) in their task-allocation process. Also in this thesis, by the term *communication*, we always refer to direct communication and we confine our discussion on MRTA within the context of direct communication only.

In the process of pursuing self-regulated MRTA, robots can receive information from a centralised source (Krieger & Billeter 2000) or from their local peers (Agassounon et al. 2004). In (Sarker & Dahl n.d.), we reported a steady-state convergence of MRTA in a practical MRS using a centralized information source. This centralized communication system is easy to implement. It simplifies the overall design of a robot controller. However this system has disadvantage of a single point of failure and it is not scalable. The increased number of robots and tasks cause inevitable increase in communication load and transmission delay. Consequently, the overall system performance degrades. On the other hand, uncontrolled reception of information from decentralized or local sources is also not free from drawbacks. If a robot exchanges signals with all other robots (hereafter called as *peers*), it might get the global view of the system quickly and can select an optimal or near optimal task. This can produce a great improvement in overall performance of some types of tasks e.g., in area coverage (Rutishauser et al. 2009). But this is also not practical and scalable for a typically large MRS due to the limited communication and computational capabilities of robots and limited available communication bandwidth of this type of system.

A potential alternate solution to this problem can be obtained by decreasing the number of message recipient peers on the basis of a local communication radius (r_{comm}). This means that robots are allowed to communicate only with those peers who are physically located within a pre-set distance. When this strategy is used for sharing task information among peers, MRTA can be more robust and efficient (Agassounon et al. 2004). However it is not well-defined how the selection of communication range can be made despite the significant differences in various implementation of MRS. In case of biological social insects, the concept of *active space* explains how each individual set their dynamic communication radius (Holldobler & Wilson 1990, McGregor & Peake 2000) (see Section **??**). In this thesis, we present a locality based dynamic P2P communication model that design a desired communication range by considering both biological inspirations and geometric relationships of the environment particularly, the shapes and communication capabilities of robots. Along with a practical insight for selecting r_{comm} value, various other design issues have been tackled. The recursion-free design of local communication channels is also achieved by a dynamic publish/subscribe model of communication. We also compare this system with our baseline centralized communication based MRS in terms of convergence of MRTA, communication load, robot motions and their task specializations.

1.3 Contributions

The main contributions of this thesis are as follows:

- Introduction of attractive field model (AFM), an inter-disciplinary generic model of division of labour, as a basic mechanism of self-regulated MRTA.
- Validation of the model through experiments with reasonably large number of real robots.
- Development of a centralized and a local P2P communication model and their respective implementation algorithms that satisfy the requirement of system-wide continuous flow of information for self-regulated task-allocation.
- Comparisons of performances of both communication models in achieving similar self-regulated MRTA.
- Development of a point-to-point signal based multi-robot control architecture using D-Bus inter-process communication technology.

1.4 Thesis outline and relevant publications

This report has been organized as follows. Chapter 2 reviews the reviews the related literature on general terms, key issues of MRS and MRTA. This also includes the review of communication for self-regulated task-allocation in biological societies. This chapter concludes by discussing the related work on communication for self-regulated MRTA. Chapter 3 describes the attractive filed model in details. Chapter 4 presents the our centralized and local communication models and analyse it from the geometric and biological view-point. Chapter 5 includes experiment tools used in this research. Chapter 6 describes the design of our experiments. Chapter 7 describes the results of our experiments. Chapter 8 concludes this thesis with a summary and future research directions.

CHAPTER 2

Background and Related Work

2.1 Definition of key terms

2.1.1 Self-regulation

Animals and flying beings, that live on or above earth, form social communities similar to human societies (Ali 1995). In recent years, the biological study of social insects and other animals reveals us that simple individuals of these self-organized societies can solve various complex and large everyday-problems with a few behavioural rules, relying on their minimum sensing and communication abilities (Camazine et al. 2001). Some common tasks of these biological societies include: dynamic foraging, building amazing nest structures, maintaining division of labour among workers (Bonabeau et al. 1999). These tasks are done by colonies ranging from a few animals to thousands or millions of individuals. Despite their huge colony size, they easily achieve surprising efficiency in those tasks with many common features, e.g. robustness, flexibility, synergy (for an example in ants, see Fig. 2.1). Today, these findings have inspired scientists and engineers to use this knowledge of biological self-organization in developing solutions for various problems of artificial systems, such as routing traffics in telecommunication and vehicle networks, designing control algorithms for large groups of autonomous robots, automating industrial shop-floor tasks and so forth (Garnier et al. 2007).

Self-organization (SO) in biological and other systems are often characterized

(a) Weaver ants (b) Green-tree ants

Figure 2.1: (a) During nest construction, weaver ants combine two leaves by pulling them from two sides, reproduced from Yahya (2000). (b) Green-tree ants retrieve a relatively large prey. From http://www.spacecollective.org, last seen on 01/05/2010.

in terms of four major ingredients: 1) positive feedback, 2) negative feedback, 3) presence of multiple interactions among individuals and their environment, and 4) amplification of fluctuations e.g., random walks, errors, random task-switching (Camazine et al. 2001). An external observer, as if looking through transparent glasses, can recognize a self-organized system by observing the individual interactions of that system from four interlinked perspectives (Fig. 2.2). The first perspective is the *positive feedback* or amplification that can be resulted from the execution of simple behavioural "rules of thumb". For example, recruitment to a food source through trail laying and trail following in some ants is due to positive feedback that attract other ants to follow the trail and to lay more pheromones over time. The second perspective is the *negative feedback* that counterbalances positive feedback. This usually occurs to stabilize collective patterns, e.g., crowding at the food sources (saturation), competition between paths to food sources etc. The third perspective is the *presence of multiple interactions* that can be direct peer-to-peer (P2P) or indirect *stigmergic*, e.g. ants pheromone laying. Finally, the fourth perspective is the *amplification of fluctuations* that comes from various stochastic events. For example, errors in trail following of some ants may lead some foragers to get lost and later on to find new, unexploited food sources and recruit other ants to those sources.

In a self-organized system, an individual agent may have limited cognitive, sensing and communication capabilities. But they are collectively capable of solving

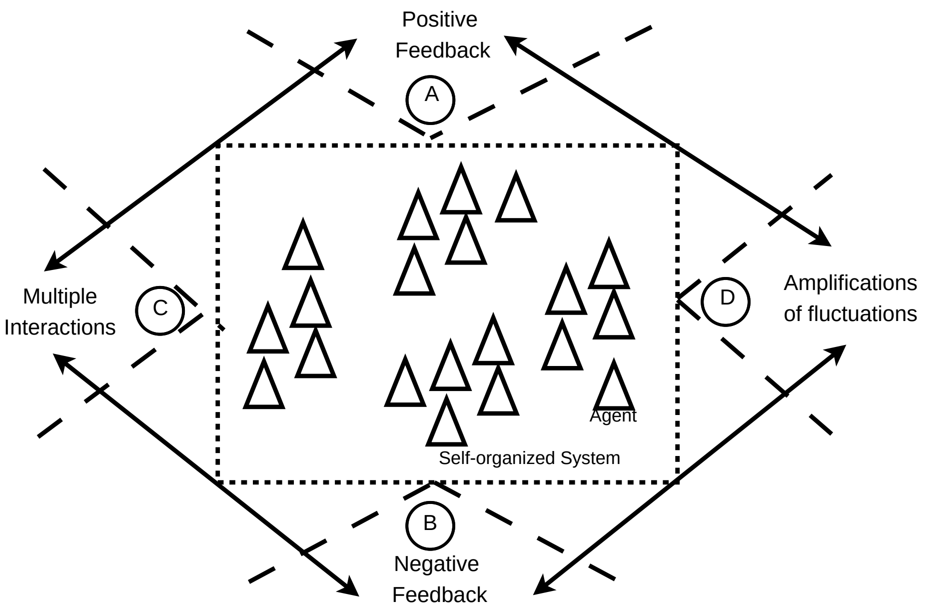


Figure 2.2: Self-organization viewed from four (A-D) inseparable perspectives. Adopted from Camazine et al. (2001).

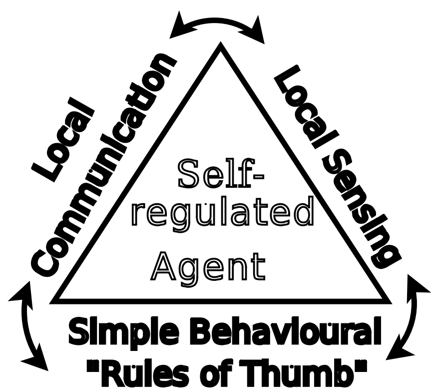


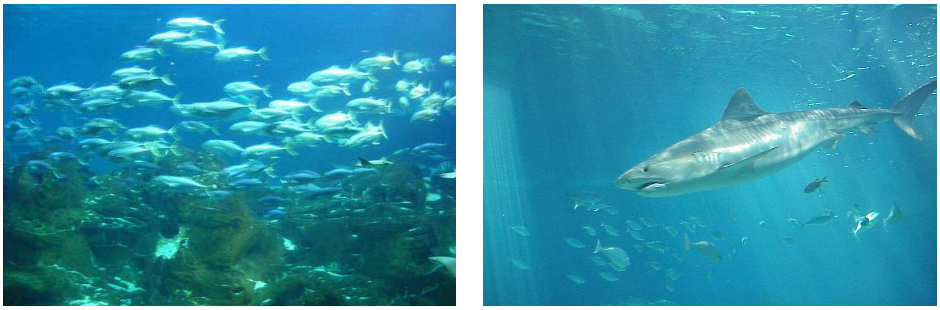
Figure 2.3: Three major interfaces of a self-regulated agent.

(a) Honey-bee nest (b) Construction of honey-combs

Figure 2.4: (a) A Honey-bee colony has built a nest on a tree-branch. From http://www.harunyahya.com, last seen on 01/05/2010. (b) Honey-bees are constructing honey combs. From http://knol.google.com, last seen on 01/05/2010.

complex and large problems, e.g. coordinated nest construction of honey-bees (Fig. 2.4), collective defence of school fishes upon predator attack (Fig. 2.5), ordered homing of bats (Fig. 2.6). Since the discovery of these collective behavioural patterns of self-organized societies, scientists observed modulation or adaptation of behaviours in the individual level (Garnier et al. 2007). For example, in order to prevent a life-threatening humidity-drop in the colony, cockroaches maintain a locally sustainable humidity level by increasing their tendency to aggregate, i.e. by regulating their individual aggregation behaviours . As shown in Fig. 2.43, this self-regulation (SR) of an individual agent is depicted through a triangle where its base-arm of simple behavioural rules of thumb (e.g. in this case, intense aggregation of cockroaches in low humidity) is supported by two side-arms: local communication and local sensing. This local sensing is sometimes also referred to as sensing or information gathering from the work in progress, e.g.stigmergy, and the local communication mentioned here is an instance of direct communication with neighbours (Camazine et al. 2001).

SR has been studied in many other branches of knowledge. In most places of literature, SR refers to the exercise of control over oneself to bring the self into line with preferred standards (Baumeister & Vohs 2007). One of the most notable self-regulatory process is the human body's homoeostatic process where the human body's inner process seeks to return to its regular temperature when it gets overheated or chilled. Baumeister & Vohs (2007) has referred self-regulation to goal-directed behaviour or feedback loops, whereas self-control may be associated

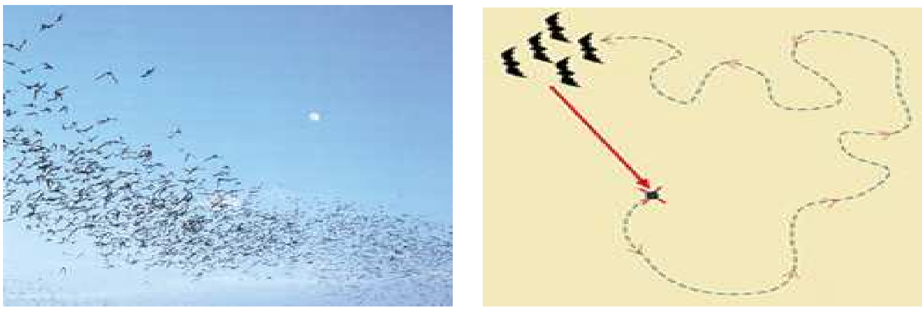


(a) A moving school of fish (b) A shark attacked a school of fish

Figure 2.5: (a) Group cohesion of a school of fish. (b) When a shark attacks a school of fish, they misguide the attacker by swift random movements. From http://www.travelblog.org last seen on 01/05/2010.

with conscious impulse control. In psychology, SR denotes the strenuous actions to resist temptation or to overcome anxiety. SR is also divided into two categories: 1) conscious and 2) unconscious SR. Conscious SR puts emphasis on conscious, deliberate efforts in self-regulation. On the other hand, unconscious self-regulation refers to the automatic self-regulatory process that is not labour intensive but operate in harmony with unpredictable, unfolding events in the environment. This uses the available informational input in ways that help to attain an activated goal.

The concepts of SR is also commonly used in cybernetic theory where SR in inanimate mechanisms show that they can regulate themselves by making adjustments according to pre-programmed goals or set standards. A common example of this kind can be found in a thermostat that controls a heating and cooling system to



(a) A bat colony (b) Navigation of bats

Figure 2.6: (a) One of the largest bat colony with about 50 million bats, (b) These bats can show amazing navigation abilities: they always fly back to their nest on a straight route from wherever they are, reproduced from **?**.

maintain a desired temperature in a room. In physics, chemistry, biology and some other branches of natural sciences, the concept of SR is centred around the study of self-organizing individuals. SR has also been studied in the context of human social systems where it originates from the division of social labour that creates self-organized process that has self-regulating effects (Kppers et al. 1990). Two types of SR have been reported in many places of literature of sociology: 1) SR from SO and 2) SR from activities of components in a heterarchical organization. It is interesting to note that SR in biological species provides the similar evidences of bottom-up approach of SR of heterarchical organization through interaction of individuals or the absence of strict hierarchy (Beer 1981).
From the above discussion, we see that the term *self-regulation* carries a wide range of meaning in different branches of knowledge. In psychology and cognitive neuroscience, SR is discussed within an individual's perspective whereas, in biology and social sciences, SR is discussed within the context of a group of individuals or society as a whole. In this thesis, the latter context is more appropriate where SR covers both aspects of monitoring one's own state and environmental changes in relation to the communal goal and thus making adjustments in self behaviours with respect to the changes found.

2.1.2 Communication

What is Communication ? Defining *communication* can be challenging due to the use of this term in several disciplines with somewhat different meanings. This has been potrayed in the writing of Sarah Trenholm (West 2003) who describes communication as piece of luggage overstuffed with all manner of odd ideas and meanings. West (2003) defines communication as:

> "A social process where individuals employ symbols to establish and interpret meaning in their environment."

The notion of being a "social process" involves, at least two or more individuals, and interactions that are both dynamic and ongoing. Moreover, symbols can be simply some sort of arbitrary labels given to a phenomena and they can represent concrete objects or an abstract ideas. Encyclopaedia Britannica also defines communication as "the exchange of meanings between individuals through a common

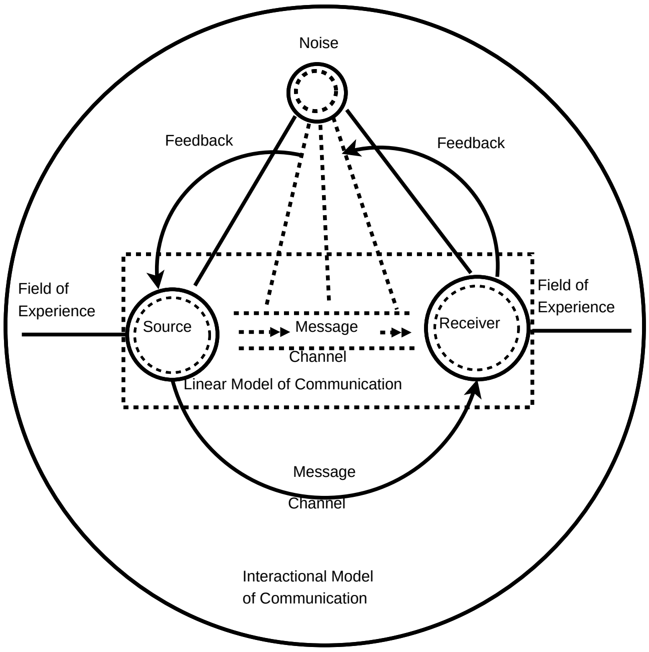


Figure 2.7: General models of communication, adopted from West (2003).

system of symbol". But since this definition lacks the notion of sociality we find it incomplete. There are many other debates related to communication, e.g. the intentionality debate (West 2003), symbol grounding (Mataric 2007) and so on. However, in order to draw some tractable boundaries , we consider communication process within the context of symbol or message exchange between two or more parties with a clear intent to influence each others' behaviours.

According to a biological model of communication (Fig. 2.8), communication is a biological process where an individual (sender) intentionally transmits encoded message though physical signal and that, on being received and decoded by another individual of same species (receiver), influences receiver's behaviour (Frings 1977). Note that, here individuals are of same species and thus they have a shared message vocabulary and mechanism of message encoding/decoding. Although this definition has not included the dynamics of a communication process, it is more precise for low-level biological and artificial systems. It accounts for the behavioural changes during communication process. These changes can be tracked though observing states of individuals.

Models of Communication. If we explore the elements of communication, we

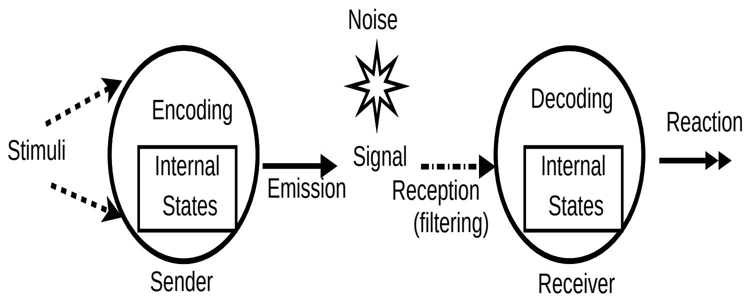


Figure 2.8: A biological model of communication, adopted from Frings (1977).

can grab the whole picture involved in the communication process. This can be explained through the study of the models of communication. There exists a plenty of models of communication. For this thesis, here we briefly discuss three prominent models: 1) linear model, 2) interaction model and 3) transaction model. Fig. 2.7 embeds first two models inside a single circle. In linear model, as introduced by Claude Shanon and Warren Weaver in 1949, communication is a one way process where a *message* is sent from a *source* to a *receiver* through a *channel*. In Fig. **??**, around this linear view, interaction model has been drawn. This model, proposed by Wilber Schramm in 1954, views communication as a two-way process with an additional *feedback* element that links both source and receiver. This feedback is a response given to the source by the receiver to confirm how the message is being understood. Here, during message passing, both source and receiver utilize their individual *field of experiences* that describe the overlap of their common experiences, cultures etc. Unlike separate filed of experiences and discrete sending and receiving of message, in transactional model, introduced by Barnlund in 1970, the sending and receiving of message is done simultaneously. Here, the field of experiences of source and receiver can overlap to some degrees. In all of the above three models, *noise* or a common message distorting element is present in the communication process. This noise can be occurred from the linguistic influences, i.e. message semantics, physical or bodily influences, cognitive influences or even from biological or physiological influences e.g., anger or shouting voice while talking.

The above models of communication describe the incremental complexities of message exchanging in the communication process. Surely, the transactional model is comparatively the most sophisticated model that prescribes adjusting the sender's message content while receiving an implicit or explicit feedback in real-time. For example, while speaking with her son for advising to read a story book, a mother may alter her verbal message as she simultaneously "reads" the non-verbal message of her child from his face. However, in case of a MRS, such sophistication may not be required or realizable by the current state of the art in multi-robot communication technology. In this study, we follow the simple linear model that meets the most of communication requirements of a MRS. The feedback has not been considered as we have assumed that all robots of our artificial MRS have a same shared vocabulary such that a message is understood as it is sent. Source never waits for an additional feedback from receiver to terminate sending a message.

Measuring communicated information. Following the linear model of communication, the amount of transferred information associated with a certain random variable X can be calculated by the concept of *Shanon entropy*. Adopting the notation of Feldman (Feldman 1997), and indicating a discrete random variable with the capital letter X, that can take values $x \in \chi$, the information entropy is defined as:

$$H[X] = -\sum_{x \in \chi} p(x) \cdot \log_2 p(x) \tag{2.1}$$

where $p(x)$ is the probability that X will take the value of x. $H[X]$ is also called the *marginal* entropy of X , since it depends on only the marginal probability of one random variable. The marginal entropy of the random variable X is zero if X always assumes the same value with $p(X = x) = 1$, and maximum if X assumes all possible states with an equal probability.

For example, in order to measure information flow in an elementary communication system, let *bit* be the unit amount of information needed to make a choice between two equiprobable alternates. If n alternates are present, a choice provides the following quantity of information: $H = log_2\, n$. Thus sending of n equiprobable messages reduces $log_2\, n$ amount of uncertainty and thus the amount of information is $log_2\, n$ bit. Similarly, according to Eq. 2.1, the value of $H[X]$ depends on the discretization of x. For instance, if the value of random variable x is discretized into 4, then $p(x)$ becomes $\frac{1}{4}$ leading to $H[X]$ = - 4 · $\frac{1}{4}$ · log_2 $\frac{1}{4}$ = 2.

Communication modes and strategies. The communication structure of a system can broadly be classified into two major categories: centralized communi-

Table 2.1: General characteristics of common communication modes

Type	**Indirect or implicit communication**	**Direct or explicit communication**
Centralized Communication Mode (CCM)	Typically a central entity modifies the environment. It facilitates passive forms of communications, i.e. communication without specific target recipient.	Both global and local broadcast communications are commonly used. P2P communication can also occur. Here, exchange of messages occurs through a central entity.
Decentralized or Local Communication Mode (LCM)	All individuals are free to modify the environment and convey information to others.	P2P and local-broadcast are most commonly used forms. Global broadcast occurs to handle emergency situations. All communications are local without requiring a central entity.

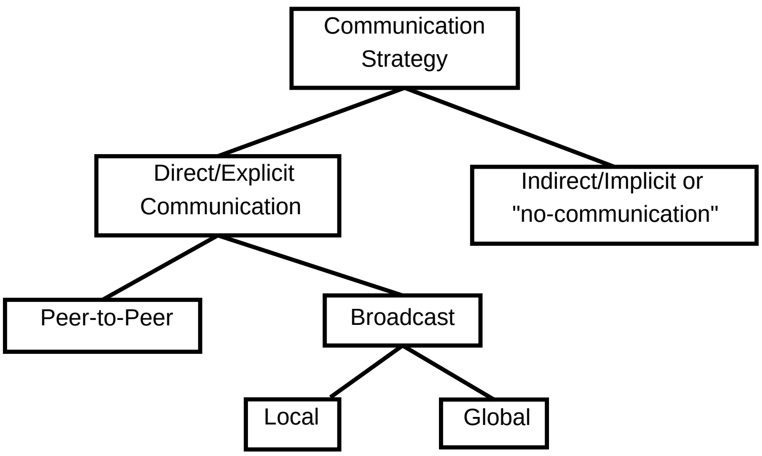


Figure 2.9: Common communication strategies observed in social systems.

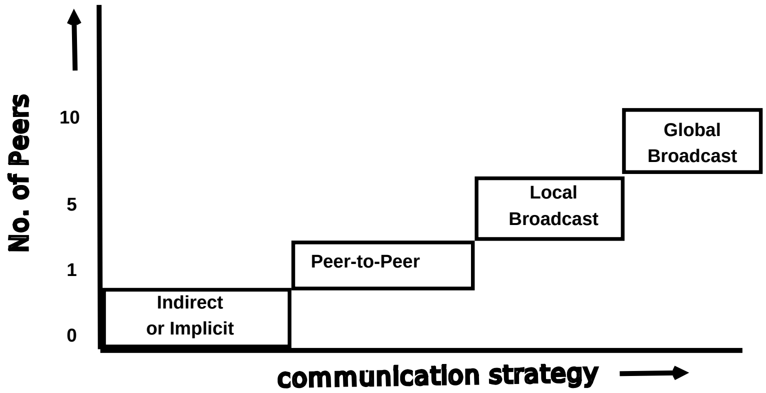


Figure 2.10: Number of recipients involved in various communication strategies.

cation mode (CCM) and decentralized or local communication mode (LCM). A centralized communication system generally has a central entity, e.g. gateway, that routes all incoming and outgoing communications of the system. Individual nodes of this system often do not communicate each other directly. But they can send and receive messages through this central gateway. Central gateway can play many roles such as, access control, resource allocation and so on. On the other hand, in LCM, there is no central entity and each node can independently route message to each other.

In both biological and robotic literature two basic types of communication are often discussed: 1) direct or explicit communication and 2) indirect or implicit communication. As defined in (Mataric 1998), *direct communication* is an intentional communicative act of message passing that aims at one or more particular receiver(s). It typically exchanges information through physical signals. In contrast, *indirect communication*, sometimes termed as *stigmergy* in biological literature, happens as a form of modifying the environment, e.g. pheromone dropping by ants (Bonabeau et al. 1999). In an ordinary sense, this is an observed behaviour and many robotic researchers call it as *no communication* (e.g. Labella 2007). In order to avoid ambiguity, in this dissertation, by the term *communication*, we always refer to *direct* communication.

Direct or explicit communication can be limited by a communication range and thus by a number of target recipients. Under both CCM and LCM, nodes can select a certain number of target recipients of their messages. This process specifies *to whom* a node intends to communicate. In this thesis, we have denoted this mechanism of target recipient(s) selection as *communication strategy*. Fig. 2.9 shows the most common communication strategies found in a social system. In the simplest case, when only two nodes can communicate we call this *peer-to-peer* (P2P) communication. When nodes can spread information to a limited number of peers of their locality, the communication takes the form of *local broadcast*, i.e. one sender and a few receivers within a certain locality. For example, when a foraging honey-bee gives the information of flower sources to a number of peers through various dances, it conveys this information to a few peers through a local broadcast. However, giving the sample of nectar through tactile or taste to its peers can be considered as an instance of P2P communication. The *global broadcast* strategy can be found in almost all social species to handle emergency situations, e.g. emitting alarm signal in danger. Table 2.1 shows the relationship between various communication modes and their ways of adopting different strategies. Fig. 2.10 shows a typical count of average number of peers in various communication strategies. The actual number of peers under local broadcast strategy is dependent upon a particular social system and it changes over time in different levels of interactions among individuals. Sec. 2.2 and 2.5 reviews communication in biological social system (BSS) and MRS respectively.

2.1.3 Division of labour or task-allocation

Encyclopaedia Britannica serves the definition of *division of labour (DOL)* as the "separation of a work process into a number of tasks, with each task performed by a separate person or group of persons". Originated from economics and sociology, the term division of labour is widely used in many branches of knowledge. As mentioned by the Scottish philosopher Adam Smith, the founder of modern economics :

> The great increase of the quantity of work which, in consequence of the division of labour, the same number of people are capable of performing, is owing to three different circumstances; first, to increase the dexterity in every particular workman; secondly, to the saving of the time which is commonly lost in passing from one species of work to another; and lastly, to the invention of a great number of machines

which facilitate and abridge labour, and enable one man to do the work of many.
(Adam Smith (1776) in Sendova-Franks & Franks (1999))

In sociology, DOL usually denotes the work specialization (Sayer & Walker 1992). Basically, it answers three major questions:

1. *What task?* i.e., the description of the tasks to be done, service to be rendered or products to be manufactured.
2. *Why dividing it to individuals?* i.e., the underlying social standards for this division, such as task appropriateness based on class, gender, age, skill etc.
3. *How to divide it?* i.e.,the method or process of separating the whole task into small pieces of tasks that can be performed easily.

(a) A termite nest

(b) Two Skyscrapers

Figure 2.11: (a) A termite colony constructs their nest through bottom-up approach, i.e. without a central planner. (b) Humans construct skyscrapers using a top-down plan. From http://www.harunyahya.com, last seen on 01/05/2010.

From the study of biological social insects and other BSSes, we can find that two major metrics of DOL have been established in literature: 1) task-specialization and 2) plasticity. *Task-specialization* is an integral part of DOL where a worker usually does not perform all tasks, but rather specializes in a set of tasks, according to its morphology, age, or chance (Bonabeau et al. 1999). This DOL among nestmates, whereby different activities are performed simultaneously by groups of specialized individuals, is believed to be more efficient than if tasks were performed sequentially by unspecialised individuals. DOL also has a great *plasticity* where the removal of one class of workers is quickly compensated for by other workers.

Thus distributions of workers among different concurrent tasks keep changing according to the external (environmental) and internal conditions of a colony (Garnier et al. 2007).

In artificial social systems, like multi-agent or MRS, the term “division of labour” is often found synonymous to “task-allocation” (Shen et al. 2001). However, some researchers (e.g. (Labella 2007)) argued to distinguish these terms due to the origin and particular contextual use of these terms. Particularly, DOL adopts the biological notion of collective task performance with little or no communication. On the other hand, task allocation follows the meaning of assigning task(s) to particular robot(s) based on individual robot capabilities, typically through explicit communication, such as *intentional cooperation* (Parker 1998). Generally, the former is considered by *swarm robotic system (SRS)* and latter is done under *traditional MRS*. Sec. 2.3.1 covers both of these approaches and Sec. 2.4 provides critical review on DOL under these approaches.

In this dissertation, for defining DOL we closely follow the SR approach that emphasizes on having task-allocation and plasticity among workers. However, we do not put any restriction on the use of communication. In fact, we view DOL as a group-level phenomenon which occur due to the individual agent’s self-regulatory task selection behaviour. But, unlike SR approach that view communication as expensive and hence try to find solutions avoiding it, we do not advocate for restricting the use of communication. Rather, along with our generic mechanism of division of labour, i.e. AFM (Chapter 4), we have proposed some self-regulatory communication strategies to vary communication load dynamically (Chapter 5).

2.2 Communication in biological social systems

Communication plays a central role in self-regulated division of labour of biological societies.In this section communication among biological social insects are briefly reviewed within the context of self-regulated division of labour.

2.2.1 Purposes, modalities and ranges

Communication in biological societies serves many closely related social purposes. Most peer-to-peer (P2P) communication include: recruitment to a new food source

Table 2.2: Common communication modalities in biological social systems

Modality	**Range**	**Information type**
Sound	Long[a]	Advertising about food source, danger etc.
Vision	Short[b]	Private, e.g. courtship display
Chemical	Short/long	Various messages, e.g. food location, alarm etc.
Tactile	Short	Qualitative info, e.g. quality of flower, peer identification etc.
Electric	Short/long	Mostly advertising types, e.g. aggression messages

[a] Depending on the type of species, long range signals can reach from a few metres to several kilometres.
[b] Short range typically covers from few mm to about a metre or so.

or nest site, exchange of food particles, recognition of individuals, simple attraction, grooming, sexual communication etc. In addition to that colony-level broadcast communication include: alarm signal, territorial and home range signals and nest markers, communication for achieving certain group effect such as, facilitating or inhibiting a group activity (Holldobler & Wilson 1990).

Biological social insects use different modalities to establish social communication, such as, sound, vision, chemical, tactile, electric and so forth. Sound waves can travel a long distance and thus they are suitable for advertising signals. They are also best for transmitting complicated information quickly (Slater 1986). Visual signals can travel more rapidly than sound but they are limited by the physical size or line of sight of an animal. They also do not travel around obstacles. Thus they are suitable for short-distance private signals such as in courtship display.

In ants and some other social insects chemical communication is dominant. Any kind of chemical substance that is used for communication between intra-species or inter-species is termed as semiochemical (Holldobler & Wilson 1990). A pheromone is a semiochemical, usually a glandular secretion, used for communication within species. One individual releases it as a signal and others responds it after tasting or smelling it. Using pheromones individuals can code quite complicated messages in smells. For example a typical an ant colony operates with somewhere between 10 and 20 kinds of signals (Holldobler & Wilson 1990). Most of these are chemical in nature. If wind and other conditions are favourable, this type of signals emitted by such a tiny species can be detected from several kilometres away. Thus chemical signals are extremely economical of their production and transmission. But they are quite slow to diffuse away. But ants and other social insects manage to create sequential and compound messages either by a graded reaction of different concentrations of same substance or by blends of signals. Tactile communication is also widely observed in ants and other species typically by using their body antennae and forelegs. It is observed that in ants touch is primarily used for receiving information rather than informing something. It is usually found as an invitation behaviour in worker recruitment process. When an ant intends to recruit a nest-mate for foraging or other tasks it runs upto a nest-mate and beats her body very lightly with antennae and forelegs. The recruiter then runs to a recently laid pheromone trail or lays a new one. In this form of communication limited amount of information is exchanged. In underwater environment some fishes and other species also communicate through electric signals where there nerves and muscles work as batteries. They use continuous or intermittent pulses with different frequencies learn about environment and to convey their identity and aggression messages.

(a) Flashing fireflies

(b) A firefly emitting light

Figure 2.12: (a) Flashing lights of fireflies displaying their synchronous behaviours (b) A firefly can produce light to signal other fireflies. From http://www.letsjapan.markmode.com, last seen on 01/06/2010.

2.2.2 Signal active space and locality

The concept of active space is widely used to describe the propagation of signals by species. In a network environment of signal emitters and receivers, active space is defined as the area encompassed by the signal during the course of transmission

(McGregor & Peake 2000). In case of long-range signals, or even in case of short-range signals, this area include several individuals where their social grouping allows them to stay in cohesion. The concept of active space is described somewhat differently in case some social insects. In case of ants, this active space is defined as a zone within which the concentration of pheromone (or any other behaviourally active chemical substances) is at or above threshold concentration (Holldobler & Wilson 1990). Mathematically this is denoted by a ratio:

$$\frac{\textit{The amount of pheromone emitted (Q)}}{\textit{The threshold concentration at which the receiving ant responds (K)}} \tag{2.2}$$

Q is measured in number of molecules released in a burst or in per unit of time whereas K is measured in molecules per unit of volume. Fig. 2.13 shows the use of active spaces of two species of ants: (a) *Atta texana* and (b) *Myrmicaria eumenoides*. The former one uses two different concentrations of *4-methyl-3-heptanone* to create attraction and alarm signals while the latter one uses two different chemicals: *Beta-pinene* and *Limonene* two create similar kinds signals, i.e. alerting and circling.

The adjustment of this ratio enables individuals to gain a shorter fade-out time and permits signals to be more sharply pinpointed in time and space by the receivers. In order to transmit the location of the animal in the signal, the rate of information transfer can be increased by either by lowering the rate of emission of Q or by increasing K, or both. For alarm and trail systems a lower value of this ratio is used. Thus, according to need, individuals regulate their active space by making it large or small, or by reaching their maximum radius quickly or slowly, or by enduring briefly or for a long period of time. For example, in case of alarm, recruitment or sexual communication signals where encoding the location of an individual is needed, the information in each signal increases as the logarithm of the square of distance over which the signal travels. From the precise study of pheromones it has been found that active space of alarm signal is consists of a concentric pair of hemispheres (see Fig. 2.13). As the ant enters the outer zone she is attracted inward toward the point source; when she next crosses into the central hemisphere she become alarmed. It is also observed that ants can release pheromones with different active spaces.

Active space has strong role in modulating the behaviours of ants. For example,

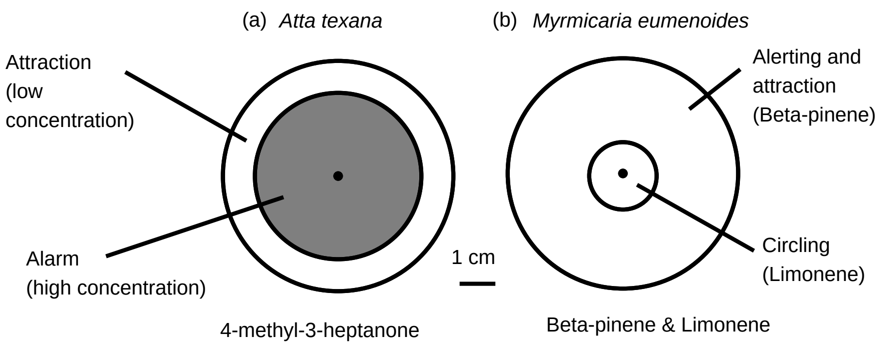


Figure 2.13: Pheromone active space observed in ants, reproduced from Holldobler & Wilson (1990).

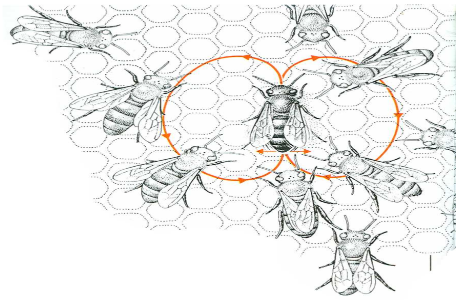
when workers of *Acanthomyops claviger* ants produce alarm signal due to an attack by a rival or insect predator, workers sitting a few millimetres away begin to react within seconds. However those ants sitting a few centimetres away take a minute or longer to react. In many cases ants and other social insects exhibit modulatory communication within their active space where many individuals involve in many different tasks. For example, while retrieving the large prey, workers of *Aphaeonogerter* ants produce chirping sounds (known as stridulate) along with releasing poison gland pheromones. These sounds attract more workers and keep them within the vicinity of the dead prey to protect it from their competitors. This communication amplification behaviour can increase the active space to a maximum distance of 2 meters.

2.2.3 Common communication strategies

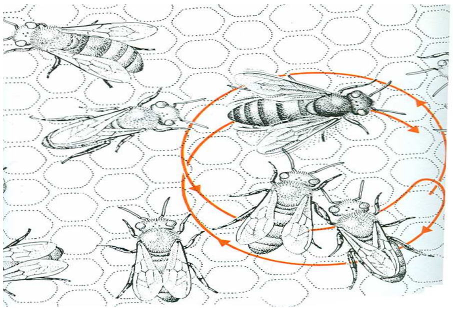
In biological social systems, we can find all different sorts of communication strategies ranging from indirect pheromone trail laying to local and global broadcast of various signals. Sec. 2.1.2 discusses the most common four communication strategies in natural and artificial world, i.e. indirect, P2P, local and global broadcast communication strategies. Table 2.3 lists the use of various communication modalities under different communication strategies. Here we give a few real examples of those strategies from biological social systems. In biological literature, the pheromone trail laying is one of the most discussed indirect communication strategy among various species of ants. Fig. 2.14 shows a pheromone trail follow-

Figure 2.14: A group of ants following pheromone-trail. From http://www.bioteams.com, last seen on 01/06/2010.

Figure 2.15: A dancing honey-bee (*centre*) and its followers. From http://knol.google.com, last seen on 01/06/2010.



(a) Honey-bee's waggle dance



(b) Honey-bee's round dance

Figure 2.16: Examples of local broadcast communication of honey-bees: (a) Honey-bees show waggle-dance (making figure of 8) when food is far and (b) they show round-dance without any waggle when food is closer (within about 75m of hive). From Slater (1986).

ing of a group of foraging ants. This indirect communication strategy effectively helps ants to find a better food source among multiple sources, find shorter distance to a food source, marking nest site and move there etc. (**?**).

Direct P2P communication strategy is also very common among most of the biological species. Fig. **??** and Fig. **??** shows P2P communication of ants and honey-bees respectively. This tactile form of communication is very effective to exchange food item, flower nectar with each-other or this can be useful even in recruiting nest-mates to a new food source or nest-site.

Table 2.3: Common communication strategies in biological social systems

Communication strategy	**Common modalities used**
Indirect	Chemical and electric
Peer-to-peer (P2P)	Vision and tactile
Local broadcast	Sound, chemical and vision
Global broadcast	Sound, chemical and electric

(a) Two honey-bees (b) Two ants

Figure 2.17: Example of P2P tactile communication: (a) Honey-bees exchange nectar samples by close contact (b) ants also exchange food or information via tactile communication From http://www.harunyahya.com/ last seen 01/05/2010 .

Table 2.4: Self-regulation of communication behaviours based on task-urgency perception

Example event	**Strategy**	**Modulation of communication based-on task-urgency**
Ant's alarm signal by pheromones	Global broadcast	High concentration of pheromones increase aggressive alarm-behaviours
Honey-bee's round dance	Local broadcast	High quality of nectar source increases dancing and foraging bees
Ant's tandem run for nest selection	P2P	High quality of nest increases traffic flow
Ant's pheromone trail-laying to food sources	Indirect	Food source located at shorter distance gets higher priority as less pheromone evaporates and more ants joins

2.2.4 Roles of communication in task-allocation

Communication is an integral part of the DOL process in biological social systems. It creates necessary preconditions for switching from one tasks to another or to attend dynamic urgent tasks. Suitable communication strategies favour individuals to select a better tasks. For example, Garnier et al. (2007) has reported two worker-recruitment experiments on black garden ants and honey-bees. The scout ants of *Lasius niger* recruit uninformed ants to food source using a well-laid pheromone trails. *Apis mellifera* honey-bees also recruit nest-mates to newly discovered distant flower sources through waggle-dances. In the experiments, poor food sources are given first to both ants and honey-bees. After some time, rich food has been introduced to both of them. It has been found that only honey-bees can switch from poor source to a rich source. The sophisticated dance communication of honey-bees favours them to get a better solution.

Table 2.4 presents the link between the task-urgency perception and self-regulation of communication behaviours in biological social systems. Here, we can see that communication is modulated based on the perception of task-urgency irrespective of the communication strategy of a particular species. This dissertation takes this biological evidence as the baseline of our hypothesis that communication behaviours of a self-regulated system must be linked to the task-requirements of the society. Under indirect communication strategy of ants, i.e. pheromone trail-

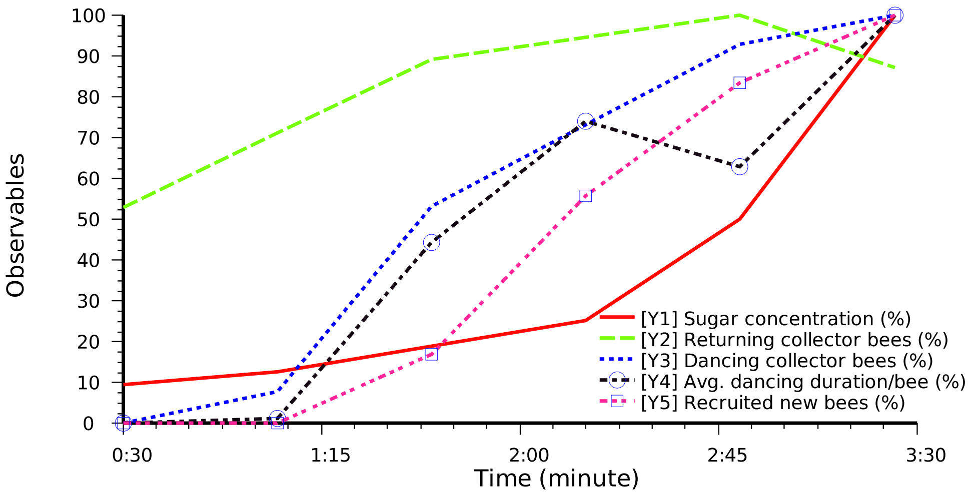


Figure 2.18: Self-regulation in honey-bee's dance communication behaviours, produced after the results of Von Frisch's (1967) honey-bee round-dance experiment performed on 24 August 1962.

laying, we can see the that principles of self-organization, e.g. positive and negative feedbacks take place due to the presence of different amount of pheromones for different time periods. Initially, food source located at shorter distance gets relatively more ants as the ants take less time to return nest. So, more pheromone deposits can be found in this path as a result of positive feedback process. Thus, the density of pheromones or the strength of indirect communication link reinforces ants to follow this particular trail.

Similarly, perception of task-urgency influences the P2P and broadcast communication strategies. *Leptothorax albipennis* ant take lees time in assessing a relatively better nest site and quickly return home to recruit its nest-mates (Pratt et al. 2002). Here, the quality of nest directly influences its intent to make more "tandem-runs" or to do tactile communication with nest-mates. We have already discussed about the influences of the quality of flower sources to honey-bee dance. Fig. 2.18 shows this phenomena more vividly. It has been plotted using the data from the honey-bee round-dance experiments of Von Frisch (1967, p. 45). In this plot, Y1 line refers to the concentration of sugar solution. This solution was kept in a bowl to attract honey-bees and the amount of this solution was varied from $\frac{3}{16}$M to 2M (taken as 100%). The variation of this control parameter influenced honey-bees communication behaviours and thus they produce varying degree of division of labour. Y2 line in Fig. 2.18 represents the number of collector bees that return home. The total number of collectors was 55 (taken as 100%). Y3 plots the percent of collectors displaying round dances. We can see this dancing collectors is directly proportional to the concentration of sugar solution or task-urgency in this case. Similarly the average duration of dance per bee is plotted in Y4 line. The maximum dancing period was 23.8s (taken as 100%). Finally, from Y5 line we can see the outcome of the round-dance communication as the number of newly recruited bees to the feeding place. The maximum number of recruited bees was 18 (taken as 100%). So, from an overall observation, we can see that bees sense the concentration of food-source as the task-urgency and they self-regulate their round-dance behaviour according to this task-urgency. Thus, this self-regulated dancing behaviour of honey-bees attracts an optimal number of inactive bees to work.

Broadcast communication is one of the classic way to handle dynamic urgent tasks in biological social systems. It can be commonly observed in birds, ants, bees and many other species. Table 2.4 mentions about the alarm communication of ants. Similar to the honey-bee's dance communication, ants has a rich language of chemical communication that can produce words through blending different glandular secretions in different concentrations. Fig. 2.13 shows how ants can use different concentrations of chemicals to make different stimulus for other ants. From the study of ants, it is clear to us that taking defensive actions, upon sensing a danger, is one of the highest-priority tasks in an ant colony. Thus, in this highly urgent task, ants almost always use their global broadcast communication strategy through their strong chemical signals and they make sure all individuals can join in this task. This gives us a coherent picture of self-regulation of animal communication based on the perception of task urgency.

2.2.5 Effect of group size on communication

The performance of cooperative tasks in large group of individuals also depends on communication and information sharing. From the study of social wasps, Jeanne (1999) has reported that, depending on the group size, different kinds of information flow occur in two groups of social wasps: 1) independent founders of *Polistes* (Fig. **??**) 2) swarm founders of *Polybia* (Fig. **??**). Independent founders (IF) are species in which reproductive females establish colonies alone or in small groups,

(a) Polybia wasps (b) Polistes wasps

Figure 2.19: Colony founding in two types of social wasps (a) *Polybia occidentalis* founds colony by swarms (b) *Polistes* founds colony by a few queens independently. From http://www.discoverlife.org, last seen 01/05/2010.

but independent of any sterile workers and in the range of 10^2 individuals at maturity. Swarm founders (SF) initiates colonies by swarm of workers and queens. They have a large number of individuals, in the order of 10^6 and 20% of them can be queen. The most notable difference in the organization of work of these two social wasps is IF does not rely on any cooperative task performance while SF interact with each-other locally to accomplish their tasks. The work mode of IF can be considered as *global sensing no communication (GSNC)* where individuals sense the task-urgencies throughout a small colony and do these tasks without communicating with each other. On the other hand, the work mode of SF can be treated as *local sensing local communication (LSLC)* where individuals can only sense locally due to large colony-size and they communicate locally to exchange information, e.g. task-urgency (although their exact mechanism is unknown).

Fig. 2.20 compares the occurrence of information flow among IF and SF. In case of SF information about nest-construction or broods food-demand can not reach to foragers directly. Fig. 2.21 shows the path of information flow among SF for nest construction. The works of *pulp foragers* and *water foragers* depend largely on their communication with *builders*. On the other hand, in case of IF there is no such communication present among individuals. This phenomena raises the question of how these individuals can select the best work modes from GSNC and LSLC.

Garnier et al. (2007) tried to answer this question in terms of task-specialization. In case of large colonies, many individuals repeatedly performs same tasks as this

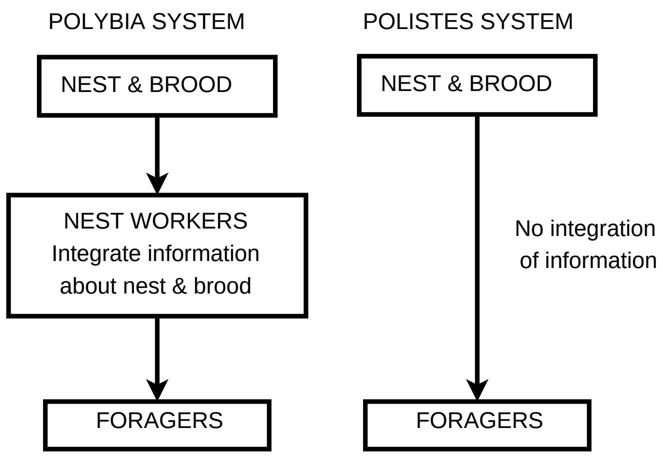


Figure 2.20: Different patterns of information flow in two types of social wasps: polybia and polistes, reproduced from Jeanne (1999).

minimizes their interferences, although they still have a little probability to select a different task randomly (Jeanne 1999). But because of the large group size, the queuing delay in inter-task switching keeps this task-switching probability very low. Thus, in SF, task-specialization becomes very high among individuals. On the other hand, in small group of IF, specialization in specific tasks is costly because these prevents individuals not to do other tasks whose task-urgency becomes high. Thus they becomes generalist and do not communicate task-urgency to each other.

The above interesting findings on GSNC and LSLC in social wasps have been linked up with the group productivity of wasps. Fig. 2.22 illustrates high group productivity in case of LSLC of SF. The per capita productivity was measured as the number of cells built in the nest in (a) and the weight of dry brood in grams in (b). In case of IF this productivity is much lesser (max. 24 cells per queen at the time the first offspring observed) comparing to the thousands of cells produced by SF (Jeanne 1999). This shows us the direct link between high productivity of social wasps and their local communication and information strategy.

2.3 Overview of multi-robot systems (MRS)

2.3.1 MRS research paradigms

Historically the concept of multi-robot system comes almost after the introduction of behaviour-based robotics paradigm (Brooks 1986, Arkin 1990). In 1967, us-

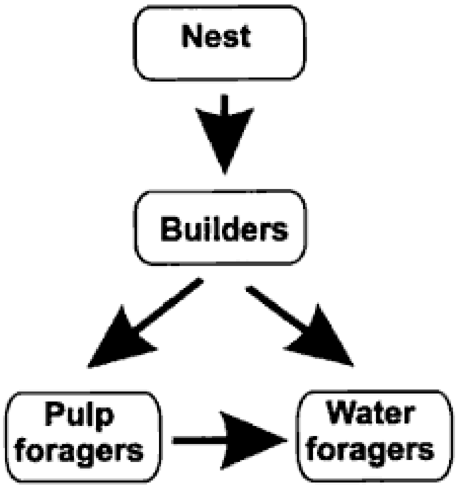


Figure 2.21: Information flow in polybia social wasps, reproduced from Jeanne (1999).

ing the traditional sense-plan-act or hierarchical approach (Murphy 2000), the first Artificially Intelligent (AI) robot, Shakey, was created at the Stanford Research Institute. In late 80s, Brooks influenced this entire field of mobile robotics by his layered, behaviour based robot-control approach that acted significantly differently than the hierarchical approach. At the same time, Braitenberg (1984) described a set of experiments where increasingly complex vehicles are built from simple mechanical and electrical components. Around the same time and with similar principles, Reynolds (1987) developed a distributed behavioural model for a bird in a flock that assumed that a flock is simply the result of the interactions among the individual birds (see Sec. **??**). Early research on multi-robot systems also include the concept of cellular robotic system (Fukuda & Nakagawa 1987, Beni 1988) multi-robot motion planning (Arai et al. 1989, Premvuti & Yuta 1990, Wang 1989) and architectures for multi-robot cooperation (Asama et al. 1989). Fig. **??** and **??** present us the two earliest MRS system in foraging and box-pushing task domains, developed by the pioneers in this filed, Mataric (1994) and Kube (1997) respectively.

From the beginning of the behaviour based paradigm, the biological inspirations influenced many cooperative robotics researchers to examine the social characteristics of insects and animals and to apply them to the design multi-robot systems (Arkin 1998). The underlying basic idea is to use the simple local control rules of

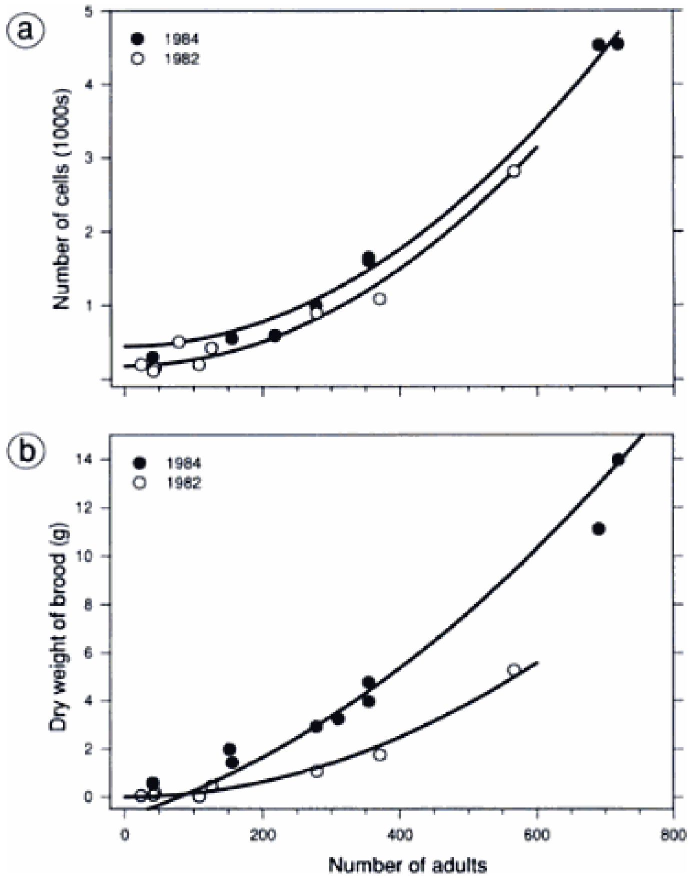


Figure 2.22: Productivity of social wasps as a function of group size, reproduced from Jeanne (1999).

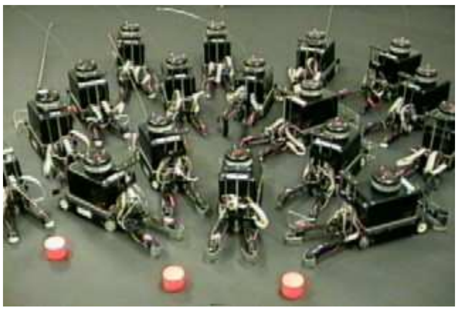


Figure 2.23: The Nerd-Herd. From Mataric (1994)

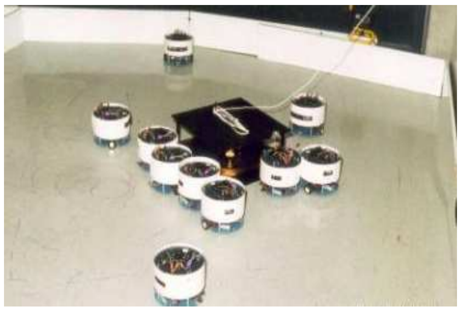


Figure 2.24: A group of 10 box pushing robots. From Kube (1997)

(a) A disaster site: buring oil rig

(b) An array of Seaglider underwater robots

Figure 2.25: Example of MRS working at a disaster site (a) British Petroleum's oil rig sinks in the Gulf of Mexico after explosion. From http:///news.bbc.co.uk, reported on 22/04/2010. (b) IRobot's Seaglider fleet that was surveying oil spill in the Gulf of Mexico at depths of up to 1,000 meters. From http:///www.irobot.com, reported on 25/05/2010.

(a) Robot teams at Robocup soccer

(b) Robot team at Robocup rescue

Figure 2.26: Example of MRS at sports and rescue (a) Robot dogs playing at Robocup soccer. From http:///news.bbc.co.uk, reported on 11/05/2005. (b)Rugbot robot team was competing at Osaka's Robocup rescue league from **?**.

Figure 2.27: Hundreds of Centibots robots worked at indoor search, navigation and mapping tasks. From Ortiz et al. (2005).

Figure 2.28: Pioneer robots operating at outdoor uncertain environment of Georgia Tech. Mobile Robot Lab. From http://news.cnet.com, reported on 05/04/2010.

various social species, such as ants, bees, birds etc., to the development of similar behaviours in multi-robot systems. In multi-robot literature, there are many examples that demonstrate the ability of multi-robot teams to aggregate, flock, forage, follow trails etc. (Bonabeau et al. 1999, Mataric 1994). The dynamics of ecosystem, such as cooperation, has also been applied in multi-robot systems that has presented the emergent cooperation among team members (McFarland 1994, Martinoli et al. 1996). On the other hand, the study of competitive behaviours among animal and human societies has also been applied in multi-robot systems, such as that found in multi-robot soccer (Asada et al. 1999). From Fig. **??** and 2.26 show us that the applications of MRS can be ranging from human disaster recovery to games and entertainment.

As discussed above, there are several research groups who follow different approaches to handle multi-robot research problems. Based on the underlying philosophies and principles, We have classified them into two broad paradigms: 1) Traditional MRS and 2) Swarm-robotic system (SRS). Below we have highlighted them briefly.

1. Traditional MRS paradigm

As discussed above, unlike SRS, traditional MRS does not directly take inspiration from BSS. Rather it follows the organizational, social, knowledge-based and multi-agent based approaches to solve problems of MRS. Explicit modelling of environment, tasks, robots can be the main features of these systems. According to Parker (2008), traditional MRS can be classified into two following categories.

Organizational and social approaches: Organizational and social paradigms are typically based on organization theory derived from human systems that reflects the knowledge from sociology, economics, psychology and other related fields. To solve complex problems this paradigm usually follows the cooperative and collaborative forms of distributed intelligence. In multi-robot systems the example of this paradigm is found in two major formats: 1) the use of roles and value system and 2) market economics. In multi-robot applications under this paradigm, an easy division of labour is achieved by assigning roles depending on the skills and capabilities in individual team member. For example, in multi-robot soccer (Stone & Veloso 1999, Asada et al. 1999) positions played by different robots are usually considered as defined roles. On the contrary, in market economics approach (Gerkey & Mataric 2002, Dias et al. 2006) task allocation among multiple robots are done via market economics theory that enables the selection of robots for specific tasks according to their individual capabilities determined by a bidding process.

Knowledge-based and multi-agent based approaches: This paradigm, commonly used for developing multi-agent systems, is knowledge-based, ontological and semantic paradigm. Here knowledge is defined as ontology and shared among robots/agents from disparate sources. It reduces the communication overhead by utilizing the shared vocabulary and semantics. Due to low bandwidth, limited power, limited computation and noise and uncertainty in sensing/actuation, the use of this approach is usually restricted in multi-robot systems.

2. SRS paradigm

In bio-inspired, emergent swarms paradigm local sensing and local interaction forms the basis of collective behaviors of swarms of robots. Many researchers addressed the issues of local interaction, local communication (i.e., stigmergy) and other issues of this paradigm (Mataric 1995), (Kube & Zhang 1993). Today, this paradigm has been emerged as a sub-field of robotics called swarm robotics (Sahin & Spears 2005). This is a powerful paradigm for those applications that require performing shared common tasks over distributed workspace, redundancy or fault-tolerance without any complex interaction of entities. Some examples include flocking, herding, searching, chaining, formations, harvesting, deployment, coverage etc.

Although our approximate classification of MRS includes most of the research directions it is very hard to specifically categorize all diverse researches on multi-robot systems. However, most of the researchers select a suitable paradigm to abstract the problem from an specific perspective with common fundamental challenges of MRS discussed in the later sections.

2.3.2 MRS taxonomies

The vast amount of research in MRS makes it necessary to use well-established classifications or taxonomies in order to specify and design the target MRS. In Section 2.3.1 we see that the main-stream research in MRS can be classified into two distinct paradigms. However, these paradigms have certain assumptions, often unspecified or implicit, regarding the design of MRS hardware, software, communication and interaction etc. Thus, MRS taxonomies can be useful for many purposes, e.g. to avoid ambiguities in system specification by reducing the size and complexity of possible design spaces, and to use certain trade-off among various features for achieving overall system performance.

While earlier MRS taxonomies, e.g. one proposed by Premvuti & Yuta (1990), Cao et al. (1997) discuss very fundamental design issues of MRS, recent taxonomies e.g. proposed by Dudek et al. (2002), Gerkey & Mataric (2004), Balch (2002), Farinelli et al. (2004) etc. gives us the detail design choices for making useful system specifications. We classify these recent taxonomies into two groups: 1) generalized taxonomies and 2) specialized taxonomies. We consider taxonomies of Dudek et al. (2002) and Parker (2008) as generalized taxonomies since they can be used to specify almost all necessary features of a MRS. On the other hand, specialized taxonomies provide MRS specification with respect to particular system features, e.g., the taxonomy of Balch is only useful in a MRS with reinforcement learning, the taxonomy of Gerkey & Mataric gives us the specification of tasks in a MRTA context. Other less common taxonomies e.g., one proposed by Farinelli et al. (2004) or another proposed by Cao et al. (1997) are centred around the coordination (weak or strong or none), communication (implicit or explicit or none), architecture (centralized or decentralized) etc. Here we briefly describe the major axes of leading taxonomies of MRS.

Dudek's generalized taxonomy of MRS

Dudek et al. (2002) provides seven main axes of MRS specification. We have re-grouped them into the following three major areas.

1. Collective or group size, composition and reconfigurability: A MRS can be formed by one, two, multiple or effectively infinite number of autonomous robots. Composition refers to the homogeneity of the group members. Robots can be identical in both form and function (hardware and software), homogeneous (consisting of same physical hardware) or physically heterogeneous. Collective reconfigurability refers to the rate at which robots can spatially re-position themselves. It can be completely static, coordinated or dynamically arranged.

2. Communication range, topology and bandwidth : The maximum distance between two robots, required for effective communication, can be zero (i.e. they can not communicate directly) or infinite (i.e. all robots can communicate to any other robot) or in-between these two. Communication topology determines the style of addressing target peers e.g., through broadcast messaging, individual addressing by name or address or, following tree-like hierarchy or redundant graphs. Communication bandwidth provides the measure of costs associated with communication. This can be no cost (i.e. infinite bandwidth) or high cost (i.e. limited or no bandwidth) or something in between these two extreme cases.

3. Processing abilities: This refers to the software architectures that can be used for controllers of the robots. General models are finite state automata, a push-down automata, neural-networks or Turing machines (most common assumption).

Specialized taxonomies of Gerkey and Balch

With this general speculation of MRS we can specify the tasks or rewards in the system using any specialized taxonomies. For example, taxonomy of Gerkey & Mataric can be helpful to understand the target domain of application of MRS. He defined three axes of possible tasks and robot capabilities of MRS:

Single-task robots (ST) *vs.* multi-task robots (MT): ST (MT) means a robot can

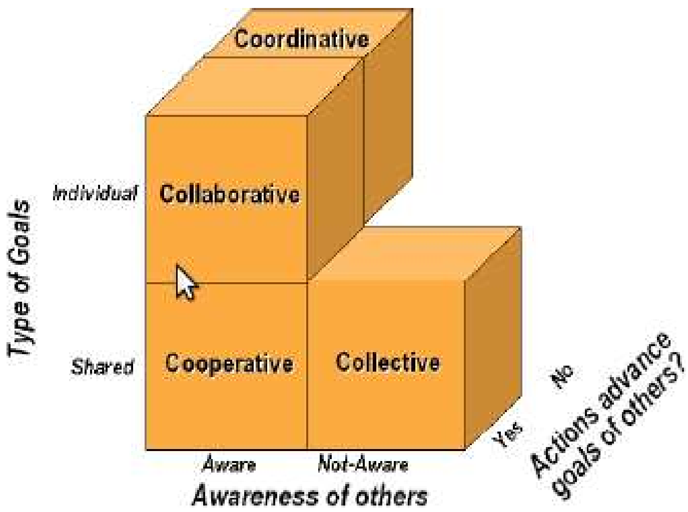


Figure 2.29: Categorization of types of interactions in MRS, reproduced from Parker (2008).

perform one (multiple) tasks at a time.

Single-robot tasks (SR) *vs.* multi-robot tasks (MR): while under SR each task requires only one robot, under MR, multi-robots may be required.

Instantaneous assignment (IA) *vs.* time-executed assignment (TA): While IA refers a situation when planning for future task-allocations is not possible, under TA planning is possible.

Balch extended this taxonomies of tasks and applied to multi-robot learning cases. His taxonomy of reward include: source of reward (internal or external or both), rewarding time (immediate or delayed), continuity of reward (discrete or continuous), locality of reward (global or local or a combination of both) relation to performance (tied to performance or based-on intuitive state-value).

Parker's taxonomy of interaction

In addition to the above two classes of taxonomies, we present here Parker's (2008) notion of *interaction* in a robot team in four levels: 1) collective, 2) cooperative 3) collaborative and 4) coordinative. Although Parker did not claim it to be a MRS taxonomy, we found this very much useful to describe the high-level relationships of individuals of the system. Moreover this removes the ambiguities among these overused terms and makes them precise for future use. While analysing the role and application of distributed intelligence on MRS, Parker presented an excellent classification of interactions of entities of MRS. As seen in Fig. **??** she viewed the interactions along three different axes:

1. the types of goals of entities (either shared goal such as, cleaning a floor, or, individual goal)

2. whether entities have awareness of others on the team (either aware such as, in cooperative transport, or, unaware such as, in a typical foraging)

3. whether the action of one entity advances the goal of others (e.g., one robot's floor cleaning helps other robots not to clean that part of the floor)

Based on this approximate observation Parker classified interactions into four categories:

1. Collective interaction: Entities are not aware of others on the team, yet do share goals and their actions are beneficial to team-mates. Mostly, swarm-robotic work of many researchers follow this kind of interaction to perform biologically-relevant tasks, such as foraging, swarming, formation keeping and so forth.

2. Cooperative interaction: Entities are aware of others on the team, they share goals and their actions are beneficial to their team-mates. This type of interaction is used to reason about team-mates capabilities multiple robots works together, usually in shared workspace, such as cleaning a work-site, pushing a box, performing search and rescue, extra-planetary exploration and so forth.

3. Collaborative interaction: Having individual goals (and even individual capabilities), entities aware of their team-mates and their actions are beneficial to their team-mates. One example of this kind of interaction is a team of collaborative robots where each must reach a unique goal position by sharing sensory capabilities to all members such as illustrated as coalition formation in (Parker & Tang 2006).

4. Coordinative interaction: Entities are aware of each other, but they do not share a common goal and their actions are not helpful to other team members. For example, in a common workspace, robots try to minimize interference by coordinating their actions as found in multi-robot path planning techniques, traffic control techniques and so on.

Beyond this four most common types of interactions Parker also described another kind of interaction in adversarial domain where entities effectively work each other such as multi-robot soccer. Here entities have individual goals, they are aware of each other, but their actions have a negative affect on other robots' goals.
In this dissertation, we use the taxonomy of Dudek et al. (2002) for specifying our MRS (Sec. **??**. The taxonomy of **?** is used to analyse the dependence of MRS on various levels of interactions in Sec. 2.4.

2.3.3 Traditional MRS

MRS not only shares the problem of controlling a single robot but also it amplifies the problem to several orders or magnitude. Below we list a few major challenges of any MRS:
Increased uncertainty about environment: When multiple robots work in a partially observable world, the environmental view becomes severely restricted due to both in terms of noisy sensor readings and frequent obstacle detections. Thus, in MRS, the uncertainty about the environment increased in may folds.
Increased dynamic changes of the environment: Since many robots work in a shared environment, the dynamic movements and physical interferences among the robots becomes more frequent and robots are required to change their course of action more frequently.
Decreased communication throughput: Interference in communication is inescapable for a team of robots. Since the typical bandwidth of a communication channel is fixed, adding more robots reduces the effective communication throughout and thus increased latency in robot-robot or robot-computer communications. If the robots are required to coordinate their action then the saturation of the communication channel affects the overall team-performance.
Decreased real-time performance: In a functional MRS, autonomous mobile robots need to do some tasks in real-time, e.g. identifying their current poses (*localization*) to determine next motions, avoiding obstacles etc. However, when the number of robots increases the real-time performance can be poor due to the above factors.
Increased sensor failures and break-downs: This is also common in a MRS that the real-time interaction of large number robots can decrease the life of their hard-

ware as they become subject to more collisions and interferences. Thus overall reliability of the MRS can be decreased gradually.

Despite the above big challenges, robotic researchers design and operate MRS successfully using a number of intelligent solutions since the last few decades. In the previous subsections we have seen how the researches are classified into distinctive paradigms and can be specified by precise taxonomies. Here we list a number of typical issues that any MRS may encounter from its inception to implementation.

- *Motion control*. How to use sensor values to produce real-time motions avoiding obstacles ?
- *Localization*. How to find out the self-position in the world so that reaching to a specified target becomes possible ?
- *Navigation/map-building*. How to integrate sensor values to build maps or representation of the environment for further exploration ?
- *Task-selection*. How to plan/predict and select a particular high-level task (e.g. find a red object or picking up a stick) provided that a number of tasks present in the environment ?
- *Interaction and communication*.How to interact or communicate with other robots for cooperating, collaborating or coordination in doing tasks ?
- *Adaptation/learning* How to remember things so that future robot actions or behaviours become improved ?

Not all of the above issues are present in all MRS. Many MRSes do not use any form of navigation, or communication or learning and yet they do some useful tasks. However it is important to understand how these issues can be solved in a structured, modular and timely manner. Integrating the solutions of these issues and resolving the conflicts among them also appear to be the major functions of multi-robot control architectures. For example, conflicts occur if a resource is required by or, a unique single task is distributed to, more than one robot at any given time. Several resources such as bandwidth, space etc. may be needed by more than one robot (Cao et al. 1997). The sharing of bandwidth among robots is a great problem in case of applications like multi-robot mapping (Konolige et al. 2003).

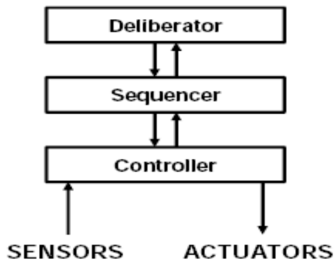


Figure 2.30: A typical hybrid robot control architecture, adopted from Mataric (2007).

As shown in Fig. **??**, in large multi-robot team such as in Centibots system (Ortiz et al. 2005), task interference and high bandwidth communication between 100s of robots appear as a significant research challenge.

Whatever the principle characteristics of a MRS, e.g., homogeneity, coupling, communication methods etc., each MRS must address some degree to those problems. For example, usually every MRS adopts a control architecture under a specific paradigm to organize its hardware, software and communication system. Similarly every MRS address the issues of communication, localization, interaction in a way specific to the application and underlying design principles (or philosophies). In the following subsections, we have attempted to summarize the key MRS research issues that would influence the selection and implementation our research. In this initiative we have deliberately omitted the non-central or very specific issues, such as collaborative transport or reconfigurable MRS, that does not directly relate to our research.

Architecture and control

In MRS, two high-level control strategies are very common: 1) centralized and 2) decentralized or distributed. Under a specific control strategy, traditionally three basic system architectures are widely adopted: deliberative, reactive and hybrid (Mataric 2007, Arkin 1998). Deliberative systems based on central planning are well suited for the centralized control approach. The single controller makes a plan from its Sense-Plan-Act (SPA) loop by gathering the sensory information and each robot performs its part. Reactive systems are widely used in distributed con-

trol where each robot executes its own controller maintaining a tight coupling between the system's sensors and actuators, usually through a set of well-designed behaviours. Here, various group behaviour emerges from the interactions of individuals that communicate and cooperate when needed. Hybrid systems are usually the mixture of the two above approaches; where each robot can run its own hybrid controller with the help of a plan with necessary information from all other robots. Behaviour-based control architecture can also be considered as a separate category of distributed control architecture (Mataric 2007), where each robot behaves according to a behaviour-based controller and can learn, adapt and contribute to improve and optimize the group-level behaviour.
Although most of the MRS control architectures share some common characteristics (such as distributed and behaviour-based control strategy) based on their difference of underlying design principles we have put them into three groups:

1. Behaviour-based classical architectures
2. Market-based architectures
3. Multi-agent based architectures

Due to the overwhelming amount of literature on MRS architectures it is not possible to include all of them. However, below some representative key architectures, strictly designed for MRS, are described.

Behaviour-based classical architectures

The ALLIANCE architecture (Parker 1998) is one of the earliest behaviour-based fully distributed architectures (Fig. **??**). This architecture has used the mathematically modelled behaviour sets and motivational system (Fig. 2.40). The primary mechanism for task selection of a robot is to activate the motivational behaviour partly based on the estimates of other robots behaviour. This architecture was designed for heterogeneous teams of robots performing loosely coupled tasks with fault-tolerance and co-operative control strategy. Broadcast of local eligibility (BLE) (Werger & Mataric 2001) is another behaviour-based architecture that uses port-attributed behaviour technique through broadcast communication method. It was demonstrated to perform coordinated tasks, such as multi-target

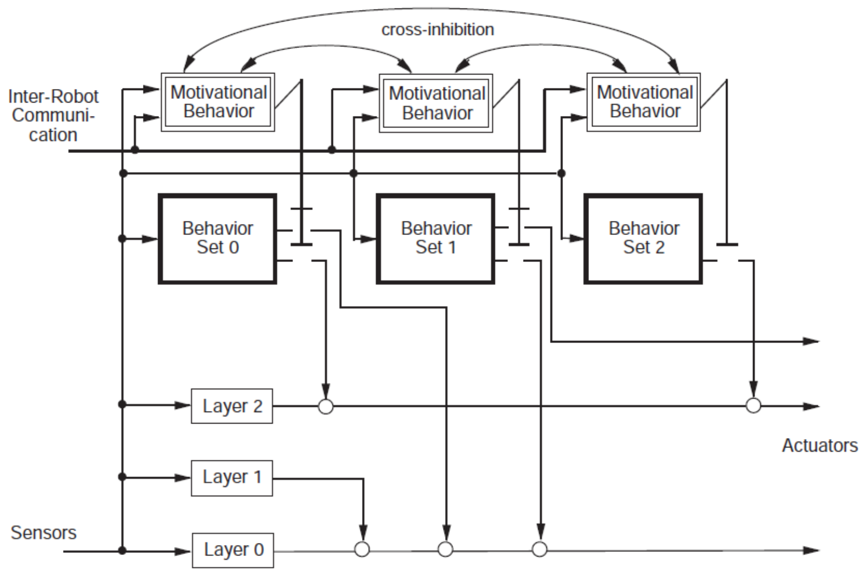


Figure 2.31: ALLIANCE architecture. From Parker (1998).

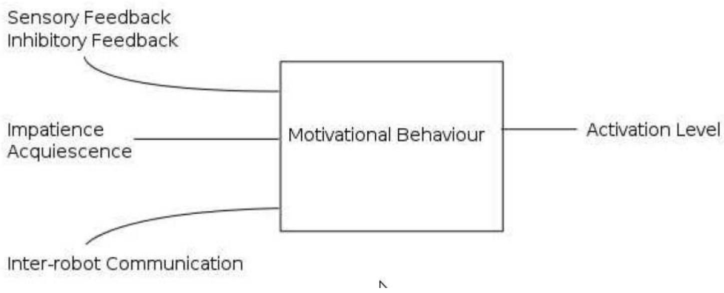


Figure 2.32: Motivational behaviour in ALLIANCE. From Parker (1998).

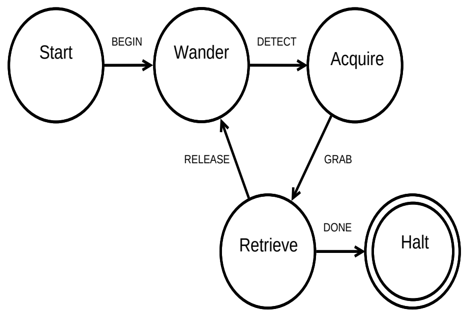


Figure 2.33: Finite state machine for foraging task, reproduced from Arkin (1998).

observation tasks. Major differences between this two behaviour-based systems include the need in ALLIANCE for motivational behaviours to store information about other individual robots, the lack of uniform inter-behaviour communication, and ALLIANCE's monitoring of time other robots have spent performing behaviours rather than BLE's local eligibility estimates. Similar to the above two architectures, many other researchers proposed and implemented many variants of behaviour-based architectures. Some of them used the classic three layer (plan-sequence-execute) approach, (e.g. Simmons et al. 2002, Gat et al. 1997) used a *layered architecture* where each layer interact directly to coordinate actions at multiple levels of abstraction.

Market-based architectures

Using the theory of marker economics and well-known Contract Net Protocol (CNP) (Davis & Smith 1988), these architectures solve the task-allocation problem by auction or bidding process. Major architectures following market-based approaches include MURDOCH (Gerkey & Mataric 2002), M+ system (Botelho et al. 1999), first-piece auction (Zlot et al. 2002), dynamic role assignments (Chaimowicz et al. 2002) among others.

Multi-agent based architectures

Some MRS architectures are influenced by multi-agent systems (MAS). For example, CHARON is a hierarchical behaviour-based architecture that rely on the notion of agents and modes. Similarly CAMPOUT is another distributed behaviour-based architecture that provide high-level functionality by making use of basic low-level behaviours in downward task decomposition of a multi-agent planner. It is comprised of five different architectural mechanisms including, behaviour representation, behaviour composition, behaviour coordination, group coordination and communication behaviours.

In this dissertation, we closely follow the behaviour-based hybrid architecture with an event-driven mechanism for activating behaviours. Our architecture and robot controllers are illustrated in Sec. 3.4.

Interaction and learning

According to the Oxford Dictionary of English the term interaction means reciprocal action or influence. In MRS research, such as in (Mataric 1994), interaction is referred to as mutual influence on behaviour. Following this definition, it is obvious that objects in the world do not interact with agents, although they may affect on their behaviour. The presence of an object affects the agent, but the agent does not affects the object since objects, by definition, do not behave, only agents do. However many other researchers acknowledge that interactions of robots with their environment (as found in stigmergic communication) have a great impact on their behaviours. Therefore, we adopt the broad meaning of interaction that is reciprocal action or influence among robots and their environment. From the above review of MRS system architecture, task allocation and communication, it is obvious that interaction among robots and their environment is the core of the dynamics of MRS. Without this interaction, it can not be a functioning MRS. We have split our discussions of MRS interaction into multi-robot learning and communication. Multi-robot learning is described here and communication in MRS is discussed in Sec. 2.5. A great deal of research on multi-robot learning has been carried out since the inception of MRS (Mataric 2001, Yang & Gu 2004, Parker 1995). Learning, identified as the ability to acquire new knowledge or skills and improve one's performance, is useful in MRS due to the necessity of robots to know about itself, its environment and other team-members (Mataric 2007). Learning can improve performance since robot controllers are not perfect by design and robots are required to work in an uncertain environment that all possible states or actions can not be predicted in advance. Besides learning a new skill or piece of knowledge it is also important to forget learned things that are no longer needed or correct as well as, to make room for new things to be learned and stored in a finite memory space of a robot.

Several learning techniques are available in robotics domain, such as reinforce or unsupervised learning, supervised learning and learning by imitation. Although reinforce learning, or learning based on environmental or peer feedback, is a good option for MRS, it has been found that in large teams the ability to lean in this way is restricted due to large continuous state and action space (Yang & Gu 2004). Several other learning techniques are also available to explore in MRS domain including Markov models, Q-learning, fuzzy logic, neural nets, game theory, probabilistic or Bayesian theory among others (Mataric 2007).

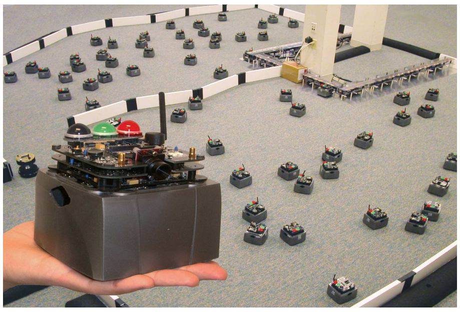
Localization and exploration

Mobile robot systems highly rely on precise localization for performing their autonomous activities in indoor or outdoor. Localization is the determination of exact pose (position and orientation) with respect to some relative or absolute coordinate system. This can be done by using proprioceptive sensors that monitor motion of a robot or exteroceptive sensors that provide information of world representation, such as global positioning system (GPS) or indoor navigation system (INS). Many other methods are also available, such as landmark recognition, cooperative positioning and other visual methods (Arkin & Diaz 2002).

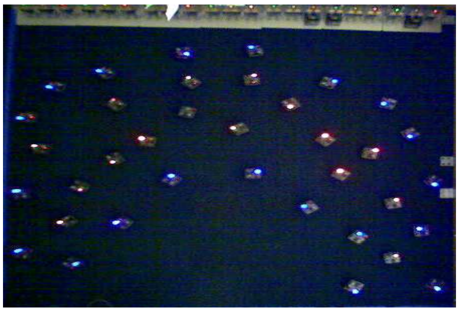
Localization issue of MRS also invites researchers to examine specific areas like exploration and map generation. In exploration problem, robots need to minimize the time needed to explore the given area. Many researchers uses various kinds of exploration algorithms for solving this NP-hard problem, such as line-of-sight constrained exploration algorithm , collaborative multi-robot exploration (Burgard et al. 2000). In mapping problem, mostly inaccurate localization information from teams of robots are accumulated and combined to generate a map by various techniques, such as probabilistic approaches (Thurn et al. 2000).

Example areas of MRS research

Researches on MRS have been targeted for numerous application domains that all can not be listed here altogether. Rather than listing all of areas explored by



(a) Swarmbot SRS



(b) Robots detecting boundary

Figure 2.34: (a) A Swarmbot and the Swarms (b) Swarmbots detecting boundary using distributed algorithms, from **?**.

researchers, here we have included a few major areas that have received highest attention in the MRS research community. Cooperative transport of large objects (that one robot is unable to handle) by multi-robots was investigated by many researchers such as, following a formal model of cooperative transport in ants (Kube & Zhang 1993), box-pushing by six-legged robots (Mataric et al. 1995). Another kind of object transport problem include clustering objects into piles e.g., (Beckers et al. 1994), collecting waste or trash e.g., (Parker 1994), sorting coloured objects e.g., (Melhuish et al. 1998), constructing a building site collectively (**?**) and so on. It has also been observed that multi-robot teams as micro or mini machines are helpful to improve the control and efficiency of mining and its processing operations (Dunbar & Klein 2002). Many researchers address MRS research issues under the requirements of a military or space application. Behaviour-based formation control (Balch & Arkin 1998), landmine detection (Franklin et al. 1995), multiple planetary rovers for various missions (Huntsberger et al. 2004) and so forth, all are the examples of this areas.

Although research on MRS has been becoming more mature since last decades, it is not easy to find many industrial applications relying on multiple autonomous mobile robots. We have found one exceptional application developed by Kiva Systems in the domain of multi-robot material handling in warehouses (Wurman et al. 2008). Along with this, Sec. bg:mrs-industry reviews some possible applications of MRS in automation industry.

2.3.4 Swarm robotic systems (SRS)

Background of SRS: When many traditional MRSes showed serious scalability failures, the necessity of adopting a new paradigm becomes obvious (Lerman et al. 2006). Researchers of traditional MRS approach realized that executing their time and processing intensive algorithms in large number of real robots (≥ 10) could be a nightmare. Adding more robots almost exponentially amplified their inherent problems e.g. physical and communication interferences with an ever-ending hunger of more CPU powers. Traditional approach became infeasible for some other reasons too. For example, one of the early inspirations for constructing a MRS is to own 10-20 cheaper and simple robots is preferable to manage 1-2 expensive and complex robots. But under traditional approach, when the robot-team size increases, robots require more sensory and on-board processing power for maintaining large internal and environmental sates. This goes against the original spirit of MRS. Moreover, many traditional MRSes, that relied upon a centralized system for communication, localization and other necessary supports, often failed under heavy-stress conditions of large MRS.

In early and mid-90s, many researchers found that applying biological principles of swarm intelligence effectively removed and reduced many bottlenecks of traditional MRS. In 1995, Maja Mataric published that complex group behaviours could be produced by the appropriate combinations of more simple "basis behaviours" (Mataric 1995). The idea of using simple biological behaviours, such as avoidance and following, to create complex flocking and foraging behaviours inspired many other researchers to search solution for controlling large MRS in this direction. The early research of Reynold (1987) guided many others to apply biological principles of self-organization aka swam intelligence (SI) in MRS. The term *swarm intelligence* was first coined by Gerado Beni (Beni 2005) in late 1980s and it represents the effort of designing algorithms or distributed problem solving devices inspired by the collective behaviours of social insect colonies (Bonabeau et al. 1999). The idea of using simple robots to create complex patterns or structures was also studied under *cellular robotics* (Fukuda & Nakagawa 1987). During recent years the term swarm robotics (SR) emerged as an application of swarm intelligence to multi-robot systems with the emphasis on physical embodiment of entities and realistic interactions among the entities and between the entities and their environment.

These systems of swarm robotics or minimalist robotics [1] can be represented by a common term swarm robotic system (SRS).

Advantages of SRS: The simplicity of SRS approach inspires robotic researches to build MRS with cheap robotic hardware, to equip them with simple controllers and control them through local informations without, creating any explicit model of the environment or using any sophisticated central control. The redundancy of robots, parallelism in their task-executions and an overall distributed control architecture, support addition or removal (or failure) of any robots in run-time. Moreover the control algorithms are now decoupled from the model of the environment or other robots. Thus this system now becomes more robust, fault-tolerant, scalable and adaptive to unknown and dynamic environment.

Distinct features of SRS: In order to distinguish swarm robotics from other branches of robotics such as collective robotics, distributed robotics, robot colonies and so forth, Sahin & Spears (2005) proposed a formal definition and a set of criteria for swarm robotics research .

> "Swarm robotics is the study of how large number of relatively simple physically embodied agents can be designed such that a desired collective behaviour emerges from the local interactions among agents and between the agents and the environment."

And the notable criteria of swarm robotics research are listed as follows.

Autonomous robots that exclude the sensor networks and may include metamorphic robotic system without having no centralized planning and control element.

Large number of robots, usually $\geq$ 10 robots, or at least having provision for scalability if the group size is below this number.

Mostly homogeneous groups of robots that typically exclude the multi-robot soccer teams having heterogeneous robots.

Relatively incapable of inefficient robots that is the task complexity enforces either cooperation among robots or increased performance or robustness with-

[1] although both SR and minimalist robotics follow similar approaches for solving similar problems, the latter does not explicitly relate its origin to SI

Figure 2.35: A group of Swarmbots are crossing rough terrain, from **?**.

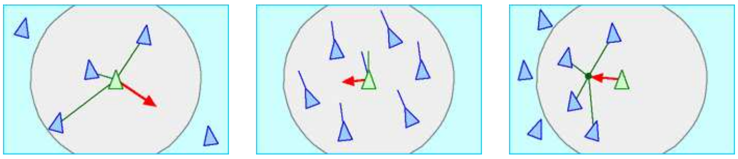
Figure 2.36: A Swarmbot of 18 s-bots pulls a child to a safe location from **?**.

out putting no restriction on individual robot's hardware/software complexity.

Robots with local sensing and communication capabilities that does not use global coordination channel to coordinate among themselves, rather enforces distributed coordination.

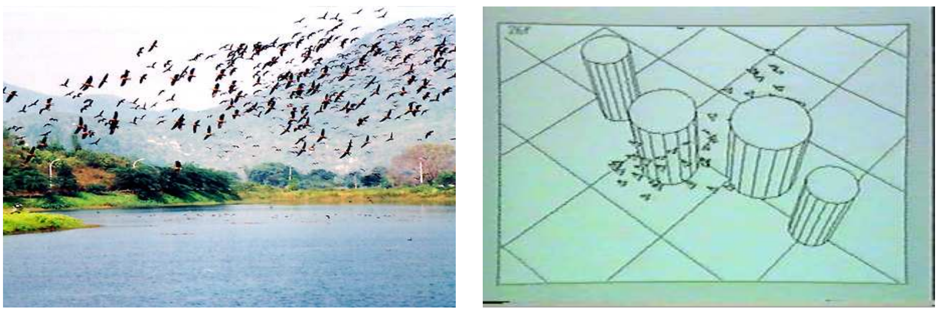
Classification and application of SRS: SRS can broadly be classified into two distinct classes. The first class consists of simple and relatively inexpensive mobile robots that are fully autonomous and can work in isolation. For example, E-puck robots (Cianci & Martinoli 2004) fall under this class. Other class of robots include self-reconfigurable (Fukuda & Nakagawa 1987) and self-assembling robots which can be built by coupling several identical units together, e.g. a robotic snake. In this dissertation, we have limited our focus to the former class of robots alone.
The potential applications of SRS include: spatially distributed tasks (e.g., environment monitoring), dangerous tasks (e.g., robotic de-miner), tasks that scale-up or scale-down over time, and tasks that require redundancy (Sahin & Spears 2005). Fig. 2.36 shows amazing abilities of Swarmbot that is crossing a rough terrain. Fig 2.37 shows another interesting demonstrations of SRS that a team of 18 Swarmbots pulls a child to a safe place.

Modelling Swarms: Modelling both the behaviour of an individual robot controllers and system-level collective behaviours have become an interesting issue in SRS. This is due to the fact that, in this kind of system, collective behaviours, e.g. flocking or aggregation, can significantly be changed by of just changing one or few parameters of individual robot controllers. Thus modelling SRS can give us



(a) Separation (b) Alignment (c) Cohesion

Figure 2.37: Reynold's simulated flocking of boids Reynolds (1987). (a) Separation: steer to avoid crowding local flockmates, (b) Alignment: steer towards the average heading of local flockmates and (c) Cohesion: steer to move toward the average position of local flock-mates, from http://www.red3d.com/cwr/boids/, last seen on 01/06/2010.



(a) A flock of birds (b) Simulated boids

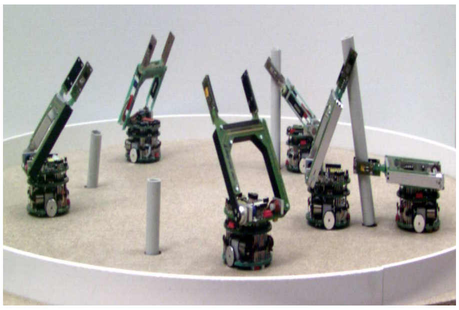
Figure 2.38: (a) Real flocks of birds, from http://www.travelblog.org, and (b) simulated flock of birds, from http://www.red3d.com/cwr/boids, last seen on 01/06/2010.

an early insight about a target system before its implementation before doing any time-consuming simulation or expensive real-experiment. This is very important since in real-experiments or in simulations the parameter space usually becomes huge and it could be infeasible to run significant number of iterations of these real or simulated trials. From mathematical models, it can be easy to find out which parameters of interest dominate the system performance or which resources are more important to do a particular type of tasks. In some cases, it can also predict the system performance to some degree.

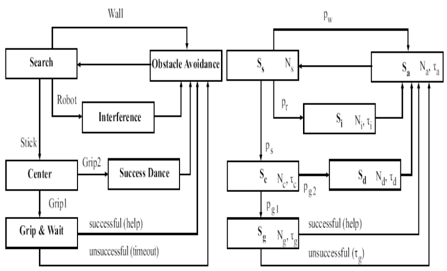
A plenty of approaches for modelling SRS exists in MRS literature (Gazi & Fidan 2006). Most common modelling approaches include: behaviour-based approach, probabilistic models, potential-function based approach, asynchronous swarm models, multi-agent based swarm models. Behaviour-based approach can be found in the study of Reynolds (1987) who has simulated the flocking behaviours of birds. Fig. 2.33 illustrates how three simple rules can produce a coordinated flocking motion (Fig. **??**(b)). From biological observation of flocking birds it is obvious that collective behaviours can be generated through a local control and interaction rules. Similar to this study, many other researchers tries to apply behaviours based local strategy for formation control (e.g. (Balch & Arkin 1998)), aggregation (e.g. Mataric 1995), sorting (e.g. (Melhuish et al. 1998)), foraging (e.g.Liu+2007), co-operative transport (e.g. (Kube 1997)) etc.

In 1999, Martinoli (1999) proposed probabilistic modelling of SRS. This probabilistic approach often has two major aspects: controller design through probabilistic finite state machine (PFSM) (e.g. see Fig. 2.35) and automated parameter adaptation through genetic algorithm. This approach has been adopted by many recent SRS research (e.g. Agassounon et al. 2004, **?**, **?**).

SRS models can be classified into many distinct classes. Firstly, they can be classified into: spatial and non-spatial models. *Spatial models* keep track of the agent's trajectories and perhaps use that spatial distribution. However, in *non-spatial models* it is assumed that agents occupy independent, random positions at consecutive time-steps. SRS models can also be classified into embodied and non-embodied models. Here *non-embodied models* consider agents as points and their physical characteristics are ignored, whereas *embodied models* take the physical characteristics or interferences of agents into account. Thus spatial models with embodied



(a) Robots pulling sticks



(b) Probabilistic FSM

Figure 2.39: (a) Stick-pulling experiments by a group of Khephera robots equipped with gripper turrets and (b) Probabilistic finite state machie (PFSM) of robot controllers. From Agassounon et al. (2004).

agents are chosen in typical simulations.

As another distinct classification, SRS models can be classified into two major groups: 1) microscopic models and 2) macroscopic models. *Macroscopic models* focus on individual robots and state transitions of each robot controller are updated based on the stochastically approximated robot-robot and robot-environment interactions. The probabilities of state transitions are calculated from simple geometric configurations and with few trial experiments. Here, no group-level sensing or actuation is taken into account. On the other hand, *macroscopic models* captures the snapshot-by-snapshot pictures of whole SRS. Each snapshot presents the total number of robots in a given state. Fig. 2.35(b) shows the probabilistic macroscopic model where S_x denotes a particular state x and N_s denotes the number of robots under state S_x. Here τ represents the corresponding probability density function derived from a set of Master-Equations (Agassounon et al. 2004). Despite a lot of attractive benefits of SRS modelling approaches, they have some notable shortcomings. Formal models of SRS, particularly probabilistic models, may not be attractive or useful for many reasons. Firstly, constructing a functional model takes time due to the need for accurate calibration of necessary parameters (which also involves running several real-experiments or simulations). Secondly, most of probabilistic models rely upon some assumptions, e.g. coverage of the area should be spatially uniform or the system should follow Markov properties i .e. the robot's future state depends only on its current state and how much time it has spent in that state. These can not be satisfied in many practical applications e.g. open space exploration or robots with memory. As task complexity increases the parameters space becomes large and searching good combinations of parameters by some means, e.g. genetic algorithms, becomes more complex.

Similar to traditional MRS, SRS faces great challenges in enabling localization, communication and interaction in group-level. For example, without the presence of any centralized localization module, such as GPS or indoor navigation system (INS), it is not easy to localize precisely and locally the position of a robot with respect to other robots or environment. Recently researchers investigated these issues and reported some novel solutions, e.g. Spears et al. (2006) presented a novel technique based on trilateration for localization of swarm robots using ultrasonic and RF transceivers. Schmickl et al. (2006) reported hop-count and bio-inspired strategies for collective perception or how a swarm robot can join multiple instances of individual perception to get a global picture. (Rothermich et al. 2005) presented a collaborative localization algorithm using landmark based localization technique. Because of the similarity of the problems in both traditional MRS and SRS, we have presented the issues of task-allocation and communication of both types of MRS in Sec. 2.4 and Sec. **??** respectively.

In this dissertation, we closely follow the SRS approach of designing robot group and solving related issues. We have followed the behaviour-based approach for designing robot controllers (Chapter **??**) that rely on GPS-like overhead camera-based solution to fulfil their localization needs. We have modelled our real robotic system considering their spatial, embodied and microscopic properties. No macroscopic simulation or analysis of the robot group has been conducted. Our autonomous robot group meets all the criteria of a SRS mentioned by (Sahin & Spears 2005) except the communication issue. We do not restrict our robots always to do local communication alone for solving their MRTA problem. Our solutions to MRTA and multi-robot communication problems have been presented in Chapter 4 and Chapter 5.

2.4 Multi-robot task allocation (MRTA)

Since 90s multi-robot task allocation (MRTA) is a common research challenge that tries to define the preferred mapping of robots to tasks in order to optimize some objective functions (Gerkey & Mataric 2004). Many control MRS architectures have been solely designed to address this task-allocation issue from different perspectives . Based-on the high-level design of those solutions, here we have classified researches on MRTA into two major categories: 1) predefined or intentional task-allocation and 2) Bio-inspired self-organized task-allocation. Fig. **??** illustrates our classification. This classification has been adopted from Shen et al. (2001), but our sub-categories are different since Shen et al. proposed the classification for multi-agent system alone that does not take the spatiality and embodiment of agents into account. Under each of our sub-categories of MRTA there are many inter-connected issues that need to be addressed by the system designer. We have put major issues into three major axes: 1) organization of task-allocation,

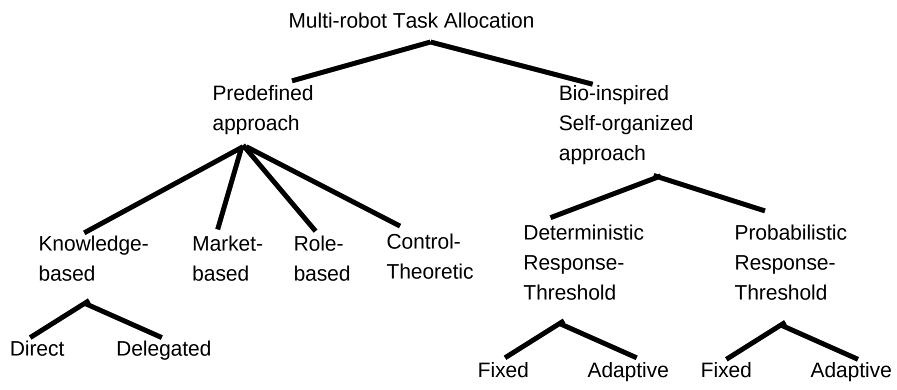


Figure 2.40: Classification of MRTA

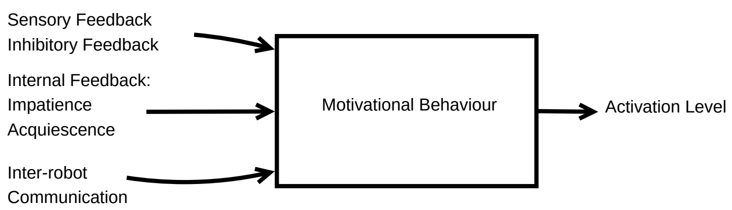


Figure 2.41: Motivational behaviour in ALLIANCE. From Parker (1998).

2) degree of communication and 3) degree of interaction. In the following subsections, we have discussed these two categories and their key issues with some example MRSes and their comparisons.

2.4.1 Predefined task-allocation

In most of the traditional MRS, task allocation is done using a well-defined models of tasks and environments. Here it is assumed that the system designer has the precise knowledge about tasks, robot-capabilities etc. Many flavours of the type of task-allocation can be found in the literature. Below we briefly discussed a few well-acknowledged works.

Knowledge-based and multi-agent based approaches:

In this approach knowledge-based techniques are used to represent tasks, robot capabilities etc. One of the early well-known MRTA architecture of this category was Parker's ALLIANCE (Parker 1998) in which each robot models the ability of team-members to perform tasks by observing their current task performances and collecting relevant task quality statistics e.g. time to complete tasks. Robots use these models to select a task that benefit the group as a whole. As shown in Fig. 2.39, ALLIANCE architecture, implemented in each robot, delineates several mathematically modelled behaviour sets, each of which corresponds to some high-level task-achieving function. The concept of motivational behaviour was introduced as a mechanism to choose among these high level behaviours. As shown in Fig. **??**, each motivational behaviour had a number of inputs and one output. The output, i.e. the activation level corresponding behavioural set, was activated once a predefined threshold was passed. In the same time, all other behavioural sets became inhibited for allowing that selected behavioural set to complete its task. The input of the behavioural sets was ranged from sensory reading to robot-robot broadcast communication of state-information. Internal behaviours e.g. *impatience* and *acquiescence* were also used to evaluate the motivation of a robot to select a high-level behaviour set. Impatience encouraged individual robots to perform a task that was not selected by any other robot of the team and a robot's acquiescence of a task was increased when a robot selected to perform it. Moreover, robots had the ability to override the inhibitory signal from another robot if a task assigned to other robot was not being completed to a desired level (e.g. when a robot stalled). In case of unsatisfactory self-progress, i.e. not doing any significant progress in a task, robots were able to switch from that task to a different one. This system was deployed on a mock hazardous waste clean up and achieved fault-tolerant distributed task performance of the robot team. Later on, L-ALLIANCE, an extension of this system was also developed to enable robots to learn from the observations of a set of task-performance metrics.

Similar to ALLIANCE, multi-agent based task allocation also use both centralized and decentralized approaches for allocating tasks among its peers. Shen et al. (2001) presented a detailed categorization where in a multi-agent system task allocation can be done by using various agents ranging from a central supervising agent or a few mediator agents to all independent agents. In case of centralized systems, the central supervisor (or a group of mediators) must have the necessary system knowledge, e.g. the capabilities and availabilities of all agents, descriptions of tasks, This system gives a well coordinated,consistent and optimized task-allocation but reduces the reliability and fault-tolerance and scalability of the system. On the other hand, in case of distributed task-allocation, each agent can directly assign a task *directly* to another agent provided that all of them have precise knowledge about others. This approach is very expensive for large number of agents since it requires all agents to have huge processing power and communication bandwidth which is not practical. Alternatively, agents can know only a few agents and *delegate* a task to these known peers so that a suitable agent can be found who has sufficient capabilities and free resources to do this task. This task-allocation by delegation also suffers from poor performance due to the use of time-consuming search algorithms. This approach also assumes the availability of high communication bandwidth which is not true in large systems.

Market-based approaches:

As a feasible alternative to the above common multi-agent based task-allocation techniques, many researchers have been following the market-based bidding approach. Dias et al. (2006) provided a survey on it. Originated from Contract-Net Protocol, market-based approach can be implemented as a centralized auctioning system or as a combination of *a few auctioneers – all bidders* or, independently *all auctioneers – all bidders*. For example, in a completely distributed system, when a robot needs to solve a problem or task for which it does not have necessary expertise or resources, it broadcasts a task-announcement message, often with a expiry time of that message. Robots that received the message and can solve that task return a bid message. The initiating robot or *manager* selects one (or more) bidder, called as *contractor*, and offers the opportunity to complete the task. The choice of contractor is done after selection by the manager and mutual agreement that maximizes the individual profits. High-level communication protocol is necessary to define several types of messages with structured content. In centralized market-based approach there is only one manager that can be an external supervising agent or one of the robots. While market-based approach consume more resources it usually produces more efficient task-allocations. Anonymous robots can be selected for tasks and these can be different in each bidding cycle.

Role or value-based task-allocation:

In this type of task-allocation each role assumes several specific tasks and each robot selects roles that best suit their individual skills and capabilities (Chaimowicz et al. 2002). In this case, robots are typically heterogeneous, each one having variety of different sensing, computation and effector capabilities. Here robot-robot or robot-environment interactions are designed as a part of the organization. In multi-robot soccer (e.g. (Stone & Veloso 1999)), positions played by different robots are often defined as roles, e.g. goal-keeper, left/right defender, left/right forwarder etc. The robot, best suited and in closest proximity to available roles/positions, selects to perform that role.

Control-theoretic approaches:
In this type of task-allocation systems, a model of the system is usually developed that converts the task specification into an objective function to be optimized. This model typically uses the rigid body dynamics of the robots assuming the masses and other parameters well-known. Control laws of individual robots are derived either by analytically or by run-time iterations. Unlike most other approaches where task-allocation problem is taken as discrete, control-theoretic approaches can produce continuous solutions. The formalisms of these systems allow system designer to check the system's controllability, stability and related other properties. These systems typically use some degree of centralization, e.g. choosing a leader robot. Example of control-theoretic approach include: multi-robot formation control (Belta & Kumar 2004), multi-robot box-pushing (**?**) etc.

2.4.2 Bio-inspired self-organized task-allocation

Task performance in self-organized approaches relies on the collective behaviours resulted from the local interactions of many simple and mostly homogeneous (or interchangeable) agents. Robots choose their tasks independently and asynchronously using the principles of self-organization, e.g. positive and negative feedback mechanisms, randomness (see Sec. 2.1.1 for details). Moreover interaction among individuals and their environment are modulated by the stigmergic, local and broadcast communications (more in Sec. 2.5). Among many variants of self-organized task-allocation, most common type is threshold-based task-allocation (Bonabeau et al. 1999). Here, a robot's decision to select a particular task depends largely on its perception of stimulus or demand for a task and its corresponding response threshold for that task. Below, we describe most common forms of threshold-based task-allocation: deterministic response-threshold and probabilistic response-threshold techniques. Both of them can use the fixed values of response-thresholds or they can adapt their response-thresholds over time based on a suitable learning or adaptation mechanisms.

$$\sigma(r,e) = \frac{1}{d(r,e)} \tag{2.3}$$

$$\theta_e = \frac{1}{\mid D_r \mid} \tag{2.4}$$

Deterministic response-threshold:

In this approach, each robot has a fixed or deterministic activation threshold for each task that needs to be performed. It continuously perceives or monitors the stimulus of all tasks that reflect the relative urgencies of tasks. When a particular task-stimuli exceeds a predefined threshold the robot starts working on that task and gradually decreases it stimuli. When the task-stimuli falls below the fixed threshold it becomes inactive for that task. This type of approach has been effectively applied in foraging (e.g. Krieger & Billeter 2000, **?**), aggregation (e.g. Agassounon & Martinoli 2002). This fixed response-threshold can initially be same for all robots (e.g. in (Jones & Mataric 2003)), or they can be different according robot capabilities or configuration of the system (e.g. Krieger & Billeter 2000).

From a simple example of this approach in event-handling domain (Kalra & Martinoli n.d.), we can see how task-stimulus can be encoded in mathematical equations. For example, Eq. 2.3 encodes the stimuli of robot r for task-urgency perception event e, ($\sigma(r,e)$) as inversely proportional to the distance between the robot and the event occurring place. Eq. 2.4 gives the threshold value θ_e (based on a predefined distance value D_r) under which robot selects this particular task or event.

Unlike maintaining a fixed response-threshold, adaptive response-threshold model changes or adapts the threshold over time. Response-threshold decreases often due to performance of a task and this enables a robot to select that particular task more frequently or in other words it learns about that task. Examples of this type of task-allocation can be found in (e.g. Bonabeau et al. 1999, Agassounon & Martinoli 2002).

$$p_e = \frac{\sigma(r,e)^n}{\sigma(r,e)^n + \theta_e^n} \tag{2.5}$$

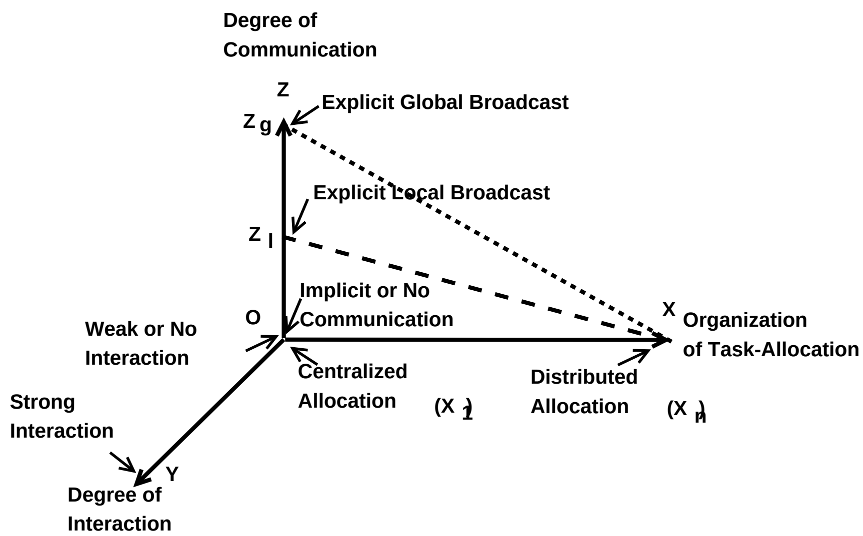


Figure 2.42: Three major axes of complexities in MRTA

Probabilistic Response-Threshold:

Unlike deterministic approach, where robots always respond to a task-stimuli that has a largest stimulus above the threshold, probabilistic approach offers a selection process based-on a probability distribution. For example, in probabilistic response, robots can respond to an event e with probability p_e as in Eq. 2.5 where θ_e is the threshold and n is the non-linearity of the response. Thus, robots always have small nonzero probabilities for all tasks.

In this dissertation we have closely followed this approach with an on-line adaptation mechanism which has been outlined in Chapter 4.

2.4.3 Key issues in MRTA

From the vast amount of literature on MRTA, we can easily infer the level of complexities exist in MRTA. In fact researchers generally agree that the MRTA is a *NP-hard* problem where optimal solutions can not be found quickly for large problems (e.g. See Gerkey & Mataric 2004, Parker 2008). But why do we find so many variants of MRTA solutions ? In order to answer this question, first we need to look into the contexts from where the solutions are made. Most predefined task-allocation solutions are proposed within the context of a known or controlled environment where the modelling of tasks, robots, environments etc. becomes feasible. Note that here tasks can be arbitrarily complex that often require relatively higher sensory and processing abilities of robots. Robot-team can be consists of homogeneous or heterogeneous individuals, having different capabilities based on the variations in their hardware, software etc. But the uncertainty of the environment is assumed to be minimum. On the other hand, bio-inspired self-organized MRTA solutions are free from extensive modelling of environment, tasks or robot capabilities. Most of the existing research considers very simple form of one global task e.g. foraging, area cleaning, box-pushing etc. This is due to the fact that major focus of this approach is limited mainly to design individual robot controllers in such a way that a few simple or *specific* tasks can be accomplished. More research is needed to verify the capabilities of self-organized approach in doing multiple complex tasks. At this moment, the bottom line remains as "select simple robots for simple tasks (self-organized approach) and complex robots for complex tasks (predefined approach)".

Both of the above task-allocation approaches expose their relative strengths and weaknesses when they are put under real-time experiments with variable number or robots and dynamic tasks. In an arbitrary event handling domain, Kalra & Martinoli (n.d.) compared between self-organized and predefined market-based task-allocation, where they found that predefined task-allocation was more efficient when the information was accurate, but threshold-based approach offered similar quality of allocation at a fraction of cost under noisy environment. Gerkey & Mataric (2003) presented a comparative study of the complexity and optimality of key architectures, e.g. ALLIANCE (Parker 1998), BLE (Werger & Mataric 2001), M+ (Botelho et al. 1999), MURDOCH (Gerkey & Mataric 2002), First piece auctions (Zlot et al. 2002) and Dynamic role assignment (Chaimowicz et al. 2002), all of them relied upon predefined task-allocation methods. The computational and communication requirements were expressed in terms of number of robots and tasks. Although this study does not explicitly measures the scalability of those key architectures, it clearly shows us that many predefined task-allocation solutions will fail to scale well in challenging environments when the number of robots (and tasks) will increase, under the given limited overall communication bandwidth and processing power of individual robots. In this regard, self-organized task-allocation methods are advantageous as they can provide fully distributed, scalable and robust

MRTA solutions through redundancy and parallelism in task-executions. Moreover, the interaction and communication requirements of robots can also be kept under a minimum limit. Thus we can say that for large MRS, self-organized task-allocation methods can potentially be selected, if a system designer can divide his complex tasks into simple pieces that can be carried out by multiple simple robots in parallel with limited communication and interaction needs.

In order to characterize both predefined and self-organized approaches in terms of their deployment, we propose three distinct axes: 1) organization of task-allocation (X), 2) degree of interaction (Y) and 2) degree of communication (Z). Fig. 2.41 depicts these axes with a reference point O. These axes can be used to measure the complexities involved in various kinds of MRTA problems and the design of their solutions. In this figure, X axis represents the number of active nodes that provides the task-allocation to the group. For example, in any predefined task-allocation approach, we can use one external centralized entity or an one of the robots (aka leader) to manage the task-allocation. This can optimize the MRTA solution globally, but is subject to single point of failure. This is also not feasible for large systems where the number of robots and the descriptions of tasks are large. In many predefined methods, e.g. in market-based systems, multiple nodes can act as mediators or task-allocators that we have discussed before. Under predefined task-allocation approach and for a small number of robots, fully distributed task-allocation can also be possible where all nodes can act as independent task-allocator, e.g. as found in ALLIANCE architecture. Most of the self-organized task-allocation methods are fully distributed, i.e. they allocate their tasks independently without the help of a centralized entity. However, they might be dependent on external entities for getting status or descriptions of tasks. Recent studies on swarm-robotic flocking by (Çelikkanat et al. n.d.) shows that a swarm can be guided to a target by a few informed individuals (or leaders) while maintaining the self-organizing principles for task-allocation. Task-allocation of a swarm of robots just by one central entity may be rare since one of the major spirits of swarm robotic system is to become fully distributed.

The Y axis of Fig. 2.41, corresponds to the level of robot-robot interaction present in the system. As we have mentioned before in Sec. 2.3.2, interaction can be classified into various levels: collective, cooperative, coordinative and collaborative. The presence of interaction can be due to the nature of the problem, e.g. a cooperation is necessary in co-operative transport tasks. Alternately, this interaction can be a design choice where interaction can improves the performance of the team, e.g. cooperation in cleaning a work-site is not necessary but it can help to improve the efficiency of this task. Y axis can also be used to refer to the degree of coupling present in the system (Mataric 2007). In case of collective interaction, robots merely co-exist, i.e. they may not be aware of each other except treating others as obstacles. Many other MRS are loosely-coupled where robots can indirectly infer some states of the environment from their team-mates' actions. But in many cases, e.g. in co-operative transport, robots not only recognize others as their team-mates, but also they coordinate their actions. Thus they form a tightly coupled system. This level of interaction and coupling also gives us the information about potential side-effects of failure of an individual robot. Tightly coupled systems where high degrees of interactions among the robots are present suffer from performance loss if some of the robots removed from the system.

The Z axis of Fig. 2.41 represents the communication overhead of the system. This can be the result of the interactions of robots under a given task-allocation method. As we have discussed before various task-allocation methods rely upon variable degrees of robot-robot communications. On the other hand, the communication capabilities of individual robots can limit (or expand) the level of interaction can be made in the given group. Thus in one way, considering the interaction requirements of a MRTA problem, the system designer can select suitable communication strategies that both minimizes the communication overhead and maximizes performance of the group. And in other way, the communication capabilities of robots can guide a system designer to design interaction rules of robot teams, e.g. the specification of robot's on-board camera can determine the degree of possible visual interactions among robots. The suitable trade-offs between these two axes: communication and interaction can give us a balanced design of our MRTA method.

Finding suitable communication strategies under the adaptive response-threshold task-allocation method is the central issue of this thesis. So we have focused to examine the benefits of traversing along the various axes of Fig. 2.41. In this dissertation, we are interested on two distinct lines: 1) distributed task-allocation, with no direct robot-robot interaction and communication, say line OX_n (n being the number of robots) and 2) distributed task-allocation, with no direct robot-robot interaction, but varying degrees of local communications, say line X_nZ_l (Z_l being a local broadcast communication strategy that involves l number of peers in communication). Our MRTA experiments along OX_n and X_nZ_l can be found in Chapter 4 and Chapter 5. The issue of multi-robot communication is presented in more detail in next section.

2.5 Communication in MRS

Communication is extremely important for any high-level interaction (e.g. co-operative, coordinative) among a multi-robot team (Arkin 1998). This is not a prerequisite for the group to be functioning, but often useful component of MRS (Mataric 2007). The characteristics of communication in MRS can be presented in terms of these issues: rationale of communication (*why to communicate*), message content (*what to communicate*), communication modalities (how to communicate), and target recipients (*with whom to communicate*) . Below we have described why communication is important, how to design this and usually how this is archived in MRS and related other issues.

2.5.1 Rationale of communication

Researchers generally agree that communication in MRS usually provides several major benefits, such as:

Information exchange improves perception: Robots can exchange potential information (as discussed below) based on their spatial position and knowledge of past events. This, in turn, leads to improve perception over a distributed region without directly sensing it.

Synchronization of actions: In order to perform (or stop performing) certain tasks simultaneously or in a particular order, robots need to communicate, or signal, to each other.

Enabling interactions and negotiations: Communication is not strictly necessary for collective interactons of a robot team. From a set of multi-robot communication experiments Arkin (1998) concluded that for certain classes of tasks,

Figure 2.43: A team of s-bots communicating by light signals. From http://lis.epfl.ch, last seen on 01/06/2010.

Figure 2.44: A fleet of robots relying on camera (vision) for search operation. From http://www.cs.utk.edu, last seen on 01/06/2010.

explicit communication is not strictly necessary. However, higher level interaction, such as cooperative task-performance or coordinative actions are almost always designed with communication support built-into the robots (see Sec. 2.3.2 for definition of various kinds of interactions). Thus communication can help a lot to influence each-other in a team that, in turn, enables robots to interact and negotiate their actions effectively.

2.5.2 Information content

Although communication provides several benefits for team-work it is costly to provide communication support in terms of hardware, firmware as well as runtime energy spent in communication. So robotic researchers carefully minimize the necessary information content in communications by using suitable communication protocols and high-level abstractions. For example in foraging, grazing and consuming experiments Balch & Arkin (1994) introduced state and goal communications. In state communication, a single bit is transmitted indicating the current state of robot (e.g. 0 transmitted when robot was in *Wander* state and 1 transmitted when robot was in *Acquire* or *Retrieve* states). In case of goal communication, the localition of task was also transmitted. From similar instances of these experiemnts below is a brief summary of information contents.
Individual state: ID number, battery level, task-perfomance statistics, e.g. number of tasks done
Goal: Location of target task or all tasks discovered

(a) E-puck robots with table-lamps

(b) Sniffing Khepera III

Figure 2.45: (a) A fleet of mobile “lighting” robots moving on a large table, such that the swarm of robots form a distributed table light and (b) Distributed odor source localization by Khephera robot equipped with volatile organic compound sensor and an anemometer (wind sensor). From http://http://disal.epfl.ch, last seen 01/06/2010.

Task-related state: The amount of task completed, number of other robots present there etc.

Environmental state: Free and blocked paths, level of interference found, any urgent event or dangerous changes found in the environment.

Intentions: Detail plan for doing a task, sequences of selected actions etc.

Since a MRS can be comprised of robots of various computation and communication capabilities, it communication content can vary greatly based on their individual communication modules and channel capacities.

2.5.3 Communication modalities

Robotic researchers typically use robot’s on-board wireless radio, infrared (IR), vision and sound hardware modules for robot-robot and robot-host communication. The reduction in price of wireless radio hardware chips e.g. wifi (ad-hoc WLAN 802.11 network) or bluetooth makes it possible to use wireless widely. Inexpensive infrared communication module is also typically built into almost all mobile robots due to its low-cost and suitability for ambient light and obstacle detection. IR can also be used for low bandwidth communication in short-range, e.g. keen-recognition. Most robots can also produce basic sound waves and detect it with suitable configuration of on-board microphones. Although spech-recognition is not still commonly found in mobile robots, detection of pre-recorded sound

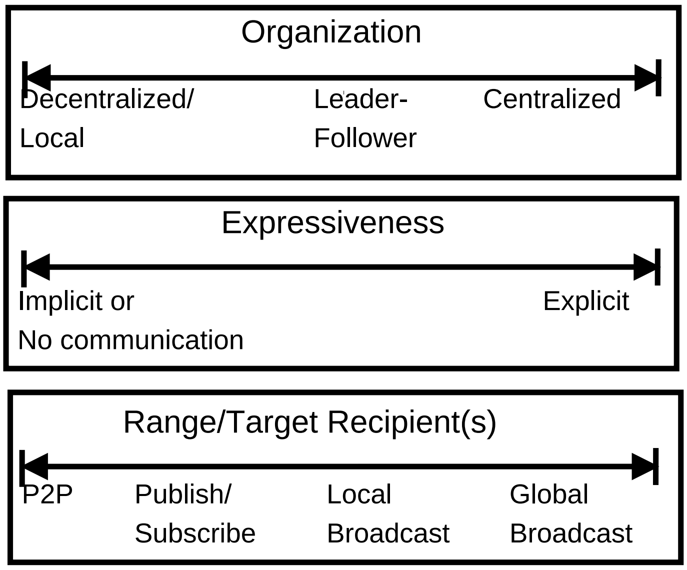


Figure 2.46: Common communication strategies found in MRS

waves can be feasible. Most of the mobile robots come with a series of LEDs, and tiny camera that can emit light signals and detect it with camera. Fig. **??** shows the robot-robot communication through the red and green coloured LEDs. Many robots can also detect blobs of colours and can recognize peers of other objects through the use of a color-coded markers. Fig. **??** shows a team of robots with colour-coded markers attached on it that can be detected by other robots. Although a lot of researches have been carried out to design robot skin and tactile communication system, we do not know any instance of tactile communication system in MRS. Similarly Lochmatter et al. (2007) showed a limited success in odor-source localization, a form of detetcing chemical signals, we do not know about any full-fledged chemical communication in any MRS. Some researchers also tried to establish communication among robots relying mainly on vision (Kuniyoshi et al. 1994).

2.5.4 Communication strategies

whatever be the communication need or modalities in a MRS, suitable strategies are required to disseminate information in a timely manner to a target audience that will maximize the effective task-completion minimizing delays and conflicts. In order to discuss the complexities of communication strategies we have selected three independent scales: organization, expressiveness and range of communication, by which we can measure the level of complexities and classify a MRS according to its communication characteristics. Fig. 4.3 outlines these scales and they are described in the following paragraphs.

Centralized and decentralized communications

Similar to the organization of MRTA, communication in a MRS can be organized using an external/internal central entity (e.g. a server PC, or a leader robot) or, a few leader robots, or by using decentralized or local schemes where every robot has the option to communicate with every other robot of the team. From a recent study of multi-robot flocking Çelikkanat et al. (n.d.) shows that a mobile robot flock can be steered toward a desired direction through externally guiding some of its members, i.e. the flock relies on multiple leaders or information repositories. Communication in a MRS can also be characterized its expressiveness or the degree of explicity. In one extreme it can be fully implicit, e.g. stigmergic, or on the other end, it can be fully explicit where communication is done by a rich vocabulary of symbols and meanings. Researchers generally tend to stay in either end based on the robotic architecture and task-allocation mechasim used. However, both of these approaches can be tied together under any specific application. Below these two major categories are described.

Explicit or direct communication

This is also known as intentional communication. This is done purposefully by usually using suitable modality e.g. wireless radio, sound, LEDs. Because explicit communication is costly in terms of both hardware and software, robotic researchers always put extra attention to design such a system by analysing strict requirements such as communication necessity, range, content, reliability of communication channel (loose of message) etc.

Implicit or indirect communication

This is also known as indirect stigmergic communication. This is a powerful way of communication where individuals leave information in the environment. This method was adopted from the social insect behaviour, such as stigmergy of ants (leaving of small amount of pheromone or chemicals behind while moving in a trail).

Local and Broadcast communications

The target recipient selection or determining the communication range (or sometimes called radius of communication) is an interesting issue of research. Researchers generally tries to maximize the information gain by using larger range. However, transmission power and communication interference among robots play a major role to limit this range. In this case, the minimum nummber of peers is 1 (or just the closest neighbor) and it can vary based on the selection of a suitable targeting strategy and available bandwidth. Based on the number of recipients of message, the communication strategy is termed differently. Such as,
Global broadcast communication: where all other robot can receive the message.
Local broadcast communication: where a few robots in local neighborhood can receive the message.
Publish-subscribe communication: where only a selected (previously subscribed) number of robots can receive the message.
Peer-to-peer communication: where only a single peer robot can receive the message.

2.5.5 Key issues in MRS communication

In multi-robot communication researchers have identified several issues. Some of the major issues are discussed here. Determination of local neighborhood:
Most swarm-robotic researchers, who use algorithms based on local-neighborhood of communication, face this problem of defining the range of local neighborhood. Agah & Bekey (1995) presented that larger communication range is not always optimum for some types of tasks e.g. exploration where a large number of recipient robots decreased the performance of exploration task. Yoshida & Arai (2000) provided a design of optimal communication range of homogenous robots based on their saptial and temporal analyses of information defusion within the context of cooperative tasks in a manufacturing shop-floor. Spatial design tried to minimize the time for information transmission and temporal design tried to minimize

the information announcing time to avoid excessive information diffusion. Eq. **??** descibes their optimal range $\chi_{optimal}$ as a function of information acquisition capacity of robots (c) and the probability of information output of a robot (p). Here c is an integer representing the upper-limit of number of robots that can be the target recipents at any time without the loss of information and $\chi_{optimal}$ gives the average number of robots within the output range.

$$\chi_{optimal} = \frac{sqrt[c]c!}{} \tag{2.6}$$

Kin Recognition
Kin recognition refers to the ability of a robot to recognize immediate family members by implicit or explicit communication or sensing. In case of MRS, this can be as simple as identifying other robots from objects and environment or as finding team-mates in a robotic soccer. This is an useful ability that helps interaction, such as cooperation among team members.
Representation of Languages
In case of effective communication several researchers also focused on representation of languages and grounding of these languages in physical world.
Fault-tolerance, Reliability and Adaptation
Since every communication channel is not free from noise and corruption of messages significant attention has been also given to manage these no communication situations, such as by setting up and maintaining communication network, managing reliability and adaptation rules when there is no communication link available. In terms of guaranteeing communication, researchers also tried to find ways for a deadlock free communication methods (Arkin 1998), such as signboard communication method (Wang 1989).

2.6 Application of MRS in automation industry

In order to examine the feasibility of our approach of emergent DoL, we have selected the distributed automated manufacturing application domain. Most of the research in this area is inspired by intelligent multi-agent technology (Shen et al. 2001). A few other researchers also tried to apply the concepts of biological self-organization (Ueda 2006, Lazinica & Katalinic 2007). In this section we

Figure 2.47: KIVA systems revolutionary material handling system

have reviewed these concepts and technologies mainly focusing on physical embodiment of agents, i.e., the use of multiple mobile robots or automated guided vehicles (AGV).

2.6.1 Multi-agent based approaches

Since early 80s researchers have been applying agent technology to manufacturing enterprise integration, manufacturing process planning, scheduling and shop floor control, material handing and so on(Shen et al. 2006). An agent as a software system that communicates and cooperates with other software systems to solve a complex problem that is beyond the capability of each individual software system (Shen et al. 2001). Most notable capabilities of agents are autonomous, adaptive, cooperative and proactive. There exists many different extensions of agent-based technologies such as Holonic Manufacturing System (HMS) (Bussmann et al. 2004). A holon is an autonomous and cooperative unit of manufacturing system for transporting, transforming, sorting and/or validating information and physical objects. Agent based technologies have addressed many of the problems encountered by the traditional centralized method. It can respond to the dynamic changes and disturbances through local decision making. The autonomy of individual resource agents and loosely coupled network architecture provide better fault-tolerance. The inter agent distributed communication and negotiation also eliminate the problem of having a single point of failure of a centralized system. These facilitate a manufacturing enterprise to reduce their response time to market demands in globally competitive market. Despite having so many advantages, agent-based systems are still not widely implemented in the manufacturing industry comparing to the other similar technologies, such as distributed objects and web-based technologies due to the lack of integration of this systems with other existing systems particularly real-time data collection system, e.g., RFID (radio frequency identification), SCADA (supervisory control and data acquisition) etc (Shen et al. 2006). Another barrier is the increased cost of investment in exchange of some additional flexibility and throughput (Schild & Bussmann 2007).

2.6.2 Biology-inspired approaches

The insightful findings from biological studies on insects and organisms have directly inspired many researchers to solve problems of manufacturing industries in a biological way. These can be categorized into two groups: one that allocates task with explicit potential fields (PF) and another that allocate tasks without specifying any PF. Below we have discussed both types of BMS.

Explicit potential filed based BMS

The biological evidences of the existence of PF between a task and an individual worker such as, a flower and a bee, a food source and an ant, inspired some researchers to conceptualize the assigning of artificial PF between two manufacturing resources. For example, PF is assumed between a machine that produce a material part and a worker robot (or AGV) that manipulates the raw materials and finished products. (Ueda 2006) conceptualized this PF as the attractive and repulsive forces based on machine capabilities and product requirements. Task allocation is carried out based on the local matching between machine capabilities and product requirements. Each machine generates an attractive field based on its capabilities and each robot can sense and matches this attractive filed according to the requirements of a product. PF is a function of distance between entities. Here, self-organization of manufacturing resources occurred by the process of matching the machine capabilities and requirements of moving robots. Through computer simulations and a prototype implementation of a line-less car chassis welding (Ueda 2006) found that this system was providing higher productivity and cost-effectiveness of manufacturing process where frequent reconfiguration of factory layout was a major requirement. This approach, was also extended and implemented in a supply chain network and in a simulated ant system model where individual agents were rational agents who selected tasks based on their imposed limitations on sensing.

BMS without explicit potential fields

Several other researchers did not express the above PF for task allocation among manufacturing resources explicitly, rather they stressed on task selection of robots based on the task-capability broadcasts from the machines to the worker robots. In case of (Lazinica & Katalinic 2007), task capabilities are expressed as the required time to finish a task in a specific machine. They used assigned priority levels to accomplish the assembly of different kinds of products in the computer simulation of their bionic manufacturing system. In another earlier computer simulated implementation of swarm robotic material handing of a manufacturing work-cell, (Doty & Van Aken 1993) pointed out several pitfalls of such a BMS system, such as dead-lock in manufacturing in inter-dependant product parts, unpredictability of task completion, energy wastage of robots wandering for tasks etc. Although most of these problems remain unsolved researchers are still exploring the concepts BMS in order to achieve a higher level of robustness, flexibility and operational efficiency in a highly decentralized, flexible, and globally competent next generation automated manufacturing system.

CHAPTER 3

Experimental Tools

3.1 General methodological issues

3.1.1 Design of MRS

Before setting up a MRS platform for doing any practical research, one needs to decide the answers of a few basic questions. Firstly, what is the target application domain of this MRS ? Some MRS researches may try to implement and verify the performance of an algorithm inspired from biological social systems, while others may not. Some MRS may focus to solve real-life problems like emergency search and rescue in a disaster site, while others may concentrate on increasing productivity of a manufacturing shop-floor. The selection of this application domain will most likely determine the tasks to be done by individual or group of robots. This will lead to select suitable robots for doing that particular tasks. These tasks and robot group characteristics can be described by using any existing MRS taxonomies, e.g., taxonomies provided by (Gerkey & Mataric 2004) and (Dudek et al. 1996) are widely used for this purpose.

Secondly, what are the organizing principles (i.e., control architecture) of the robot group in question ? From the task requirements and robot capabilities, one need to fit the MRS into a suitable paradigm of architecture and control. In Sec. **??** we reviewed three most common architectural paradigms of MRSs: classic knowledge-based, traditional market-based/role-based and bio-inspired swarm robotic paradigm. Upon selecting a paradigm, most important characteristics of target MRS: e.g., design of individual robot controller, robot-robot and robot-environment communication and coordination patterns etc. will be revealed.

Thirdly, what enabling tools and technologies (hardware and software) are required to function the whole robot group autonomously ? Whatever is the architectural design of a MRS, we need to ensure that whole group can maintain the necessary level of task performance through the desired interaction and communication strategies. For reducing cost and other practical reasons, individuals robots may not have the capabilities to localize itself without the help of an external GPS or camera etc. The enabling technologies will make-up this individual robot's shortcomings. Moreover, based-on a selected communication technology we need to set-up necessary robot-robot and robot-environment inter-networking infrastructure, e.g., network switches, gateways etc.

Finally, what extra hardware and software are necessary to observe and record experiments and its data for further analysis and improvement ? This extra system becomes an essential part of the target MRS.

We intend to design our target MRS for emulating multi-robot manufacturing scenario where robots do some shop-tasks in different machines. The notion of tasks has been kept very simple as our robots are not capable of doing many high-level practical tasks e.g. gripping or recognizing objects, carrying loads etc. Many researchers use additional hardware modules with their robots (e.g., gripper to collect pucks or any small objects from floors). We have not put any effort for emulating that kind of trivial activities, rather we concentrated on the performance of our algorithm, e.g. in task-allocation, from high-level perspectives. Doing real manufacturing tasks has been kept as a future research issue. Thus by "doing a task" our robots usually perform two functions: 1) navigate to a fixed task-location in the experiment arena (hereafter called *navigation*), and 2) they do so by avoiding any dynamic obstacle hereafter called *obstacle avoidance*. Depending on the time-out value of doing a task, a robot can wait at task-location if it arrives earlier or may switch to a different task and change direction on-the-fly. These symbolic tasks can be mapped to any suitable real task in multi-robot manufacturing domain, such as, material handling or attending a machine for various production or maintenance jobs e.g., welding different machine parts, cleaning or doing maintenance work of a machine etc.

Based on the task-requirements we have selected a simple miniature mobile robot, Epuck, that can do the above navigation tasks avoiding any dynamic or static obstacle. According to the classification of (Dudek et al. 1996) our system can be described as:

`SIZE-INF:` The robot team size is larger than 2 robots and the number of the tasks. Actually we have kept the robot team size as multiple times larger than the number of tasks.

`COM-INF:` Robots can communicate with any other robot.

`TOP-ADD:` Every robot can communicate with any other robot by name or address (link path). Naming of robot's communication link path is assumed to follow any simple convention, e.g., /robot1, /robot2 etc.

`BAND-INF:` Every robot can communicate with other robot with as much bandwidth as necessary. Since our robots need to exchange simple messages we ignore this bandwidth issue.

`ARR-DYN:` Robot can change the arrangement dynamically.

`CMP-HOM:` Each robot is initially identical in both hardware and software, but they can become different gradually by learning different tasks by different degrees (in software).

We have closely followed the swarm robotic principles for controlling the group. Although we do not impose any necessity for local communication and interaction. The details of our control architecture has been described in Sec. expt-tools:arch. Since our robots can not localize themselves by their own hardware we provided them their instant position and orientation (hereafter called *pose*) data information from a multi-robot tracking system (MRTS). Sec. 3.3.1 describes about our tracking system. This system also helps us in recording and logging individual robot's task performance and pose information. We use Bluetooth communication technology, built-in with our E-puck robots, for host PC to robot communication. Robot can also communicate with each-other physically over Bluetooth. However, we do not use that mode of communication. Instead, inter-robot communication is done in host PC's virtual inter-process communication channel. This has been illustrated in Sec. 3.3.1.

3.1.2 Real robotic experiments vs. simulations

Traditionally robotic researchers use software simulation to validate their model before stepping into real-robot experiments. Simulating a model by software code is easier and much faster compared to real-world experiments. It does not require any sophisticated hardware setup or time-consuming debugging. However, in modern times the abundance of real hardware systems and tools encourage researchers to test their work in real systems from the inception of their models. Here we briefly summarize the reasons why we do not follow the traditional "simulation first" approach. Instead of comparing both approaches extensively, we present our rationale behind doing all our experiments in real hardware.

Firstly, contemporary state-of-the-art agent-based simulation packages are essentially discrete-event simulators that execute models serially in a computer's CPU (**?**). However in real-world systems agents act in parallel and give us the "what you see is what you act upon" environment. In simulations that might not be case.

Secondly, the robot-robot and robot-environment interactions are complex and completely unpredictable than their simulations. Unexpected failures and inter-agent interactions will not occur in simulations that can either cause positive or negative effects in experimental results (Krieger & Billeter 2000).

Thirdly, it is not easy to faithfully model communication behaviours of agents in simulations. In our case, we use short-range Bluetooth communication system which is subject to dynamic noise and limited bandwidth conditions.

Fourthly, the dynamic environment conditions, e.g., increased physical interferences of larger team of robots, can also influence the experiment's outcome which is obvious in simulations.

Finally, we believe that the algorithm tested in real-robots can give us strong confidence for implementing in real-systems and later on, this can also be extended or verified in simulations. However, conversely speaking, algorithm implemented in simulation has no warranty that this will work practically with robots.

3.2 Hardware

3.2.1 E-puck robots

We use E-puck [1] robots developed by Swiss Federal Institute of Technology at Lausanne (EPFL) and now produced by Cyberbotics [2] and some other companies. The upside of using E-puck is: it is equipped with most common sensing hardware, relatively simple in design, low cost, desktop-sized and offered under open hardware/software licensing terms. So any further modification in hardware/software is not limited to any proprietary restriction. However, the downside of using E-puck robot is: it's processor is based on dsPIC micro-controller (lack of standard programming tool-chains), limited amount of memory (lack of on-board camera image processing option) and default communication module is based-on Bluetooth (limited bandwidth and manual link configuration). So the programming of E-puck can be done through C language and uploaded from PC to robot through wire: I^2C and RS232 channel or, through Bluetooth wireless communication channel. This can be tedious and time-consuming if one needs to change the robot controller frequently. However, we intend to keep the robot's functionalities very simple and limited to two main tasks: avoiding obstacles and navigating from one place to another. Thus the default hardware of E-puck seems enough for our experiments.

Table 3.1 lists the interesting hardware information about an E-puck robot. The 7 cm diameter desktop-sized robot is easy to handle. It's speed and power autonomy is also reasonable compared with similar miniature robots such as Khepera and its peers (**?**). The IR sensors provide an excellent capabilities for obstacle avoidance task. Although we do not make use of the tiny camera of E-puck, the combination of sound and LEDs can be very effective to detect low-battery power or any other interesting event. By default, E-puck is shipped with a basic firmware that is capable of demonstrating a set of it's basic functionalities. Using the supplied Bluetooth serial communication protocol (hereafter *BTCom protocol*), it is possible to establish serial communication link between host PC and robot firmware at a maximum possible speed of 115 kbps. Using this protocol, one can remotely send command to robot, e.g., set the speed of the motors, turn on/off

[1] www.e-puck.org
[2] http://www.cyberbotics.com

Table 3.1: E-puck robot hardware

Feature	**Description**
Diameter	about 7 cm
Motion	max. 15 cm/s speed (2 stepper motors)
Battery power	about 3 hours (5Wh LiION rechargeable battery)
Processor	16 bits micro-controller with DSP core, Microchip dsPIC 30F6014A at 60MHz (about 15 MIPS)
Memory	RAM: 8 KB; FLASH: 144 KB
IR sensors	8 IR sensors measuring ambient light and proximity of obstacles in a range of 4 cm
LEDs	8 red LEDs on a ring and 1 green LED in the body
Camera	colour camera (max. resolution of 640x480)
Sound	3 omni-directional microphones and on-board speaker capable of playing WAV or tone sounds
Bluetooth	for robot-computer and robot-robot wireless communication

LEDs and read the sensor values, e.g., read the IR values or capture image of the camera etc.

3.2.2 Overhead GigE camera

In order to set-up a multi-robot tracking system (MRTS), we have selected a state-of-the-art GE4900C colour camera from Prosilica [3]. The Prosilica GE-Series camera, are very compact, high-performance machine vision cameras with Gigabit Ethernet interface. This has following main features:

CCD technology convert light into electric charge and process it into electronic signals. Unlike in a complementary metal oxide semiconductor (CMOS) sensor, CCD provides a very sophisticated image capturing mechanism that gives high uniformity in image pixels. The high resolution enables us to track a relatively large area e.g., 4m X 3m. In this case, 1 pixel dot in image roughly can represent approximately 1mm X 1mm area. Although the frame rate may seem low initially, but this small frame-rate gives optimum image processing performance with large image sizes, e.g. 16 MB/frame. Prosilica offers both Windows and Linux SDK for image capture and other necessary operations. Using this SDK, we have converted

[3] http://www.prosilica.com

Table 3.2: Features of Prosilica GigE Camera GE4900C

Feature	**Description**
Type	Charge-coupled device (CCD) Progressive
CCD Sensor	Kodak KAI-16000
Size (L x W x H)	66x66x110 (in mm)
Resolution	16 Megapixels (4872x3248)
Frame rate	Max. 3 frames per second at full resolution
Interface	Gigabit Ethernet (cable length up to 100 meters)
Image output	Bayer 8/12 bit

Table 3.3: Server PC Configuration

Processor	Quad-Core Intel Xeon Processor up to 3.33GHz (1333MHz FSB, 64-bit, 2X 6MB L2 cache)
RAM	32GB (4GB ECC DIMMS x 8 slots)
Graphics Card	NVIDIA Quadro FX 570 (Memory: 256MB)
Hard-disk	SATA 3.0Gb/s 7200RPM 2 x 250 GB
OS	Ubuntu Linux 9.10 64bit

default Bayer8 format image into RGB format image and used that in our tracking software.

3.2.3 Server PC configuration

We use Dell Precision T5400 server-grade PC with the following main technical specifications: This high performance PC has supported us implementing our algorithms without having any fear of running out of RAM. Since the maximum supported RAM of a 32 bit PC architecture is limited to 2 GB we select Ubuntu Linux 9.10 64bit OS. As an open-source OS, Ubuntu offers excellent reliability, performance and community support. In order to enable Bluetooth communication in our host PC, we have added 8 USB-Bluetooth adapters (Belkin F8T017) through a suitable USB-Bluetooth hub.

3.3 Enabling software tools and frameworks

3.3.1 SwisTrack: a multi-robot tracking system

In almost all types of robotic experiments, vision-based tracking becomes the standard feasible solution for tracking robot positions, orientations and trajectories. This is due to the low cost of camera hardware and availability of plenty of standard image-processing algorithms from computer vision and robotic research community. However, setting-up a real-time multi-agent tracking platform using existing software solutions are not a trivial job. Commercial systems tend to provide submillimetre level high precision 3D tracking solutions with a very high price ranging from tens of thousands of pounds which is typically greater than the annual research budget of a medium sized research lab in UK! Besides robotics researchers prefer open-source solution to closed-source proprietary one due to the need for improving certain algorithms and applications continuously. Another line of solution, namely borrowing certain open-source tracking code from XYZ lab, typically ends up with a lot of frustration while tuning parameters manually, fixing the lab lighting conditions, seeing bad performance of programs that frequently leak memory or show segmentation fault and so forth. GUI or camera calibration can hardly be found in those so-called open-source applications. The third option for solving this tracking issue becomes "re-inventing the wheel" or hiring some research students to build a system from scratch. Certainly this is also not feasible due to the limited time, resource and skill in this field. Another big issue is the expiration of research fellowship before doing any practical research!

In the beginning of our research we met the all three types of scenario stated above. We got very high price quotes from several commercial motion capture solution providers including, Vicon [4] and some others. We tested some well-known and some not-so-well-known open-source object tracking systems including ARTag [5], ARToolKitPlus [6]. We also developed our own versions of Open-CV [7] algorithms for tracking colour blobs based on a GNOME application Cynbe's vison-app [8]. However, our algorithms failed to scale well due to the fluctuations

[4] http://www.vicon.com
[5] http://www.artag.net
[6] http://studierstube.icg.tu-graz.ac.at/handheld_ar/artoolkitplus.php
[7] http://opencv.willowgarage.com
[8] http://muq.org/ cynbe/vision-apps

in lighting conditions, lack of proper integration of all related components and some other issues (Sarker 2008). Finally, we settled with SwisTrack (Lochmatter et al. 2008) , a state-of-the-art open-source multi-agent tracking platform developed at Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland . Thanks to the hard working developers and generous sponsors of EPFL who offer this excellent tool to scientific research community and empower many research labs to track multi-agent systems out-of-the-box.

With the improved version 4 released in February 2008, SwisTrack is now becoming the *de-facto* standard tool for multi-robot tracking. Being open-source, flexible, modular and customizable, Swistrack provides a clean development and deployment path for tracking marked or marker-less objects in real-time. SwisTrack is written in C++ using common C++ libraries and frameworks. The component-based modular development style is very powerful for developing a custom algorithm and wrap it in a custom component. In Sec. 3.3.2 we show that how we append our custom communication components in the image processing pipeline. This pipeline can be compared with Unix command processing where output of one command becomes the input of another subsequent command and many commands form a chain or pipeline. Here in SwisTrack, at first an image capturing component grab camera image using USB, IEEE1394/Firewire, GigE or other supported interfaces (see Fig. **??**). Then subsequent SwisTrack components work on this image and do various processings e.g., background subtractions, colour conversions, blob-detection, tracking etc. These components follow standard computer-vision algorithms and can be used without any code modification. But if necessary they can also be modified or optimized though changing source code and/or tuning parameters on-the-fly. As shown in Fig. **??** SwisTrack GUI can take parameters in real-time and update images. Final output from images, e.g., object position, orientation, trajectory etc. can be sent over standard communication interfaces, e.g. TCP/IP, NMEA etc. SwisTrack has a lot of other features, such as multi-camera tracking, remote-control of SwisTrack over TCP/IP etc. which are documented in SwisTrack Wiki-book documentation [9].

We have set-up SwisTrack with our Prosilica GigE camera GE4900C and configured it for tracking about 40 E-puck robots with their on-top markers . These

[9]http://en.wikibooks.org/wiki/Swistrack

markers are binary-coded numbers (aka em circular bar-codes) that have certain binary bits or chip-lengths. We have used 20 bits and 15 bits chip-lengths. In order to uniquely identify the position and orientation of these markers these numbers are encoded with a fixed hamming distance, i.e. differences in bits of any two numbers. We have used a fixed hamming distance of 6 bits. As seen in Fig. **??** these markers are 8cm diameter and clearly identified and tracked by SwisTarck from a camera image resolution of 4872x3248. SwisTrack has no component for grabbing our Prosilica GigE camera and we have developed a Prosilica GigE input component using Prosilica SDK and Open-CV library. The version of SwisTrack that we have used (late May 2008 version, SVN no.) has worked pretty well except a few minor things, such as real-time configuration changing was very unstable due to our large camera image size (16MB/frame). In order to avoid that we use static configuration files that is loaded by SwisTrack in the beginning of our experiment runs.

Fig. **??** shows the components that we have used throughout our experiments. Along with the standard blob detection and circular bar-code reading components, we use our custom D-Bus server communication component that send pose information to D-Bus IPC channels (see Sec. 3.3.2). These components require a little tuning of few parameters, e.g. blob size, blob counts etc. They can be done once and saved in component configuration files for loading them in next runs. Although SwisTrack provides a wide range of components for object trajectory tracking we have not used them yet. we have not saved grabbed camera images as video from within SwisTrack due to heavy CPU load and memory usage. Besides, the video output component of our version of SwisTrack can not produce smooth video files in our set-up. So, we occasionally save image frames in files and most of our experiments, we use Ubuntu Linux's standard desktop tool, *recordmydesktop* [10] for capturing screen as video. The standard fluorescent lightings of our lab seem sufficient for our overhead GigE camera and we have prevented interferences of outside sun-lights by putting black blinds in the window. Our camera has been configured automatically through standard configuration files while program start-up.

[10] http://recordmydesktop.sourceforge.net/

3.3.2 D-Bus: an inter-process communication protocol

Inter-process communications (IPC) among various desktop software components enable them to talk to each other and exchange data, messages or request of services. Technological advancements in computer and communication systems now allow robotic researchers to set-up and conduct experiments on multi-robot systems (MRS) from desktop PCs. Many compelling reasons, including open licensing model, availability of open-source tools for almost free of cost, community support etc., make Linux as an ideal operating system for MRS research. However the integration of heterogeneous software components in Linux desktop becomes a challenging issue, particularly when each robot-control software needs sensory and other data input from various other software components (e.g. pose data from a tracker server, task information from a task server etc).

Traditional IPC solutions in a standard Linux desktop, e.g. pipes, sockets, X atoms, shared memory, temporary files etc. (hereafter called *traditional IPCs*), are too static and rigid to meet the demand of a dynamic software system ((Wittenburg 2005)). On the other hand, complex and heavy IPC like CORBA fails to integrate into a development tool-chain efficiently. They also require a steep learning curve due to their complex implementations. Besides, the failure of Desktop Communication Protocol (DCOP) in system-wide integration and interoperability issues encouraged the development of the D-Bus message bus system, D-Bus for short ((Pennington et al. 2010)). This message bus system provides simple mechanisms for applications to talk to one another (see details in Section 2). In this paper we describe how we exploit the simplicity and power of D-Bus for running a large MRS.

In pursuing a suitable multi-robot control architecture for our large number of robots, we have found that traditional IPCs are inadequate to support the important requirements of IPC among several heterogeneous software components of a large MRS. Firstly, real-time support in IPC is critical for connecting time-critical control applications. For example, a multi-robot tracking system (MRTS) can share robot pose information with a robot-controller client (RCC) though shared memory (SHM). This pose information can be used to help navigating a robot in real-time. However if MRTS crashes and stops writing new pose information into the SHM, RCC has no default mechanism to know that SHM data is outdated. Some form of reference counting mechanism can be used to overcome this issue, but that makes the implementation of RCC complicated and error-prone.

Secondly, IPC must be scalable so that adding more software components (thus more robots, sensors, etc.) in the information sharing game does not affect the overall system performance. But clearly this can not be achieved through traditional IPCs, e.g. SHM or temporary files, as the access to computer memory and disk space is costly and time consuming. Thirdly, IPC should be flexible and compatible enough to allow existing software components to join with newly developed components in the information sharing without much difficulties. Again existing IPC mechanisms are too static and rigid to be integrated with existing software components. Besides, incompatibility often arises among different applications written in different programming languages with different semantics of IPC. Fourthly, IPC should be robust, fault-tolerant and loosely coupled so that if one ceases to work others can still continue to work without strange runtime exceptions. Finally, IPC should be implemented simply and efficiently in any modern high level programming languages, e.g., C/C++, Java, Python etc. Practically this is very important since IPC will be required in many places of code and application programmers have little time to look inside the detail implementation of any IPC.

Here we present a scalable and distributed multi-robot control architecture built upon D-Bus IPC. D-Bus IPC works asynchronously in real-time. It has virtually no limit how many software components participate in information sharing. In fact, to the best of our knowledge, the performance of D-Bus daemon does not vary if the number of participating software components varies. In this paper we have shown that by using only the signalling interfaces, SwisTrack ((Lochmatter et al. 2008)), an open-source multi-robot tracking tool can be integrated with our multi-robot control framework. All software components are loosely coupled and unlike traditional IPCs, one does not depend on another for setting up and shutting down IPC infrastructure. For example, in case of SHM one software component explicitly needs to set-up and clean-up SHM spaces. In case of D-Bus any software component can join and leave in the information sharing process at any time. Each component implements its own fall-back strategy if desired information from another component is unavailable at any time. Based on a thin C API, D-Bus also provides many binding in common programming languages. In this work, we use

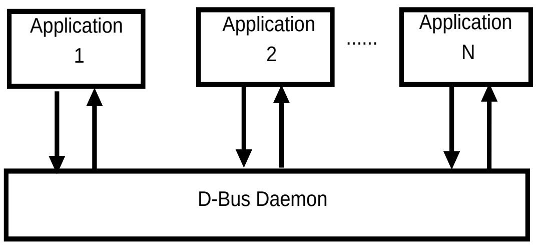


Figure 3.1: A typical view of D-Bus message bus system

dbus-python, a Python binding for D-Bus, that provide us a very clean and efficient IPC mechanism.

D-Bus Overview

D-BUS was designed from scratch to replace CORBA and DCOP to fulfil the needs of a modern Linux system. D-BUS can perform basic application IPC as well as it can facilitate sending events, or signals, through the system, allowing different components in the system to communicate. D-BUS is unique from other IPC mechanisms in several ways, e.g. 1) the basic unit of IPC in D-BUS is a message, not a byte stream, 2) D-BUS is bus-based and 3) It has separate system-wide and user/session-wide bus ((Love 2005)) . The simplest form of D-Bus communication is process to process. However, it provides a daemon, known as the message bus daemon, that routes messages between processes on a specific bus. In this fashion, a bus topology is formed (see Fig. 3.1), allowing processes to speak to one or more applications at the same time. Applications can send to or listen for various events on the bus.

D-Bus specification ((Pennington et al. 2010)) provides full details of D-Bus message protocols, message and data types, implementation guidelines and so forth. Here we discuss some relevant part of this specification. As we have already mentioned D-Bus provides a system-wide system bus and user-level session bus. Fig. 3.1 an example of this bus structure. In this paper we have limited our discussion to the latter one and skipped some advance topics like D-Bus security and so on. Here a few basic D-Bus terminologies have been introduced from D-Bus literature.

D-Bus Connection: *DBusConnection* is the structure that a program first uses to

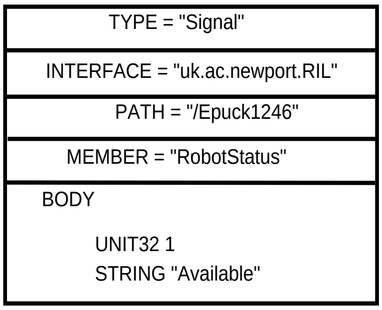


Figure 3.2: A typical structure of a D-Bus signal message

initiate talking to the D-Bus daemon, Programs can either use DBUS_BUS_SYSTEM or DBUS_BUS_SESSION to talk to the respective daemons.

DBus Message: It is simply a message between two process. All the DBus intercommunication are done using *DBusMessage*. These messages can have the following four types: method calls, method returns, signals, and errors. The DBusMessage structure can carry data payload, by appending boolean integers, real numbers, string, arrays, etc. to the message body.

D-Bus Path: This is the path of a remote *Object* (hereafter, this is captitalized to avoid ambiguity) of target process, e.g. orgfreedesktopDBus.

D-Bus Interface: This is the interface on a given Object to talk with, e.g. org.freedesktop.DBus.

D-Bus Method Call: This is a type of DBus message that used to invoke a method on a remote Object.

D-Bus Signal: This is a type of DBus message to make a signal emission. As stated in D-Bus specification, Signal messages must have three header fields: PATH giving the object the signal was emitted from, plus INTERFACE and MEMBER giving the fully-qualified name of the signal. The INTERFACE header is required for signals, though it is optional for method calls. The structure of a signal is shown Fig. 3.2 and it shows the design of robot-status signal that emits over specified interfaces and paths with a data payload of an integer and a string containing robot-status message.

D-Bus Error: This is the structure that holds the error code which occurs by calling a DBus method.

Strategies for Application Integration

Under D-Bus, there are two basic mechanisms for applications to interact with each other: by calling a remote Object of target application and by emitting a signal for interested applications. To perform a method call on a D-BUS Object, a method call message must be sent to that Object. It will do some processing and return either a method return message or an error message. Signals are different in that they cannot return anything: there is neither a ”signal return” message, nor any other type of error message see this [11] for some example use-cases. Thus on D-Bus everything can be done asynchronously without the need of polling.
D-Bus provides several language bindings for integrating D-Bus to any native application. The core D-BUS API, written in C, is rather low-level and large. On top of this API, bindings integrate with programming languages and environments, including Glib, Python, Qt and Mono. On top of providing language wrappers, the bindings provide environment-specific features. For example, the Glib bindings treat D-BUS connections as GObjects and allow messaging to integrate into the Glib mainloop. The preferred use of D-BUS is definitely using language and environment-specific bindings, both for ease of use and improved functionality ((Love 2005)).

3.3.3 BTCom/Myro: E-puck robot control programs

E-puck robot comes with a set of software tools and libraries to program and to monitor low-level sensor and actuator values. The low-level C library with driver code of Epuck robot can be downloaded from E-puck website [12]. In order to modify and recompile this library E-puck developers recommend both Windows and Linux cross-compling tool-chains. Under Windows, the MPLAB environment from Microchip [13] can be used with their C30 compiler. We have used this commercial tool for programming E-puck robot (dsPIC micro-controller), since the Linux counterpart, piklab [14] has been found unstable and still under-development. A wide variety of boot-loaders, both under Windows and Linux, can be used to upload the *.hex* firmware files to E-puck robot over Bluetooth. We have found a most reliable

[11] http://www.ibm.com/developerworks/linux/library/l-dbus.html
[12] www.e-puck.org
[13] http://www.microchip.com
[14] http://piklab.sourceforge.net/

E-puck boot-loader built-in with the trial version of Webots [15] simulator. E-puck website also provides various other tools, e.g. Player robot control framework driver [16] Matlab interfacing program, Epuck-monitor (under Windows) etc. for monitoring (or setting) sensors (or actuator) values. The default firmware running in E-puck robot is *BTCom*. It initializes Epuck robot hardware and waits for user command over Bluetooth serial port. A set of well-defined BTCom commands can be found in Epuck-library documentation.

For high-level control of E-puck robot, we have used Myro robot-control framework [17] that is developed by Institute for Personal Robots in Education. While BTCom provides a set of user commands for controlling the robot it does not take care the setting up the Bluetooth serial connection. Moreover the supplied user commands are very primitive in nature. For example, from a high-level perspectives it is more desirable to command a robot for moving at a specific speed for a given time. Under BTCom, a user can not achieve this without interactively giving low-level motor commands, e.g. set left/right motor speed. On the other hand, by setting up Myro framework, the Bluetooth communication with host PC and E-puck robot can easily be set-up under Python's pySerial [18] module. This provides an elegant solution for controlling E-puck from software code. Besides, Myro provides a thin wrapper code for E-puck's BTCom for defining high-level user commands. For example, instead of setting left/right motor speed a user can send a forward command with speed and time-out as its parameters. With the simplicity and interactivity of Python programming, this wrapper makes E-puck programming and debugging very simple and easy.

The default BTCom has another limitation that it's detection of low battery voltage is almost unnoticeable by naked eye. For running a long time experiment this is critical since we would like to continue our experiments even if a few robots' batteries run out. By the default code of BTCom, a tiny red LED, located near the poer LED in E-puck body, turns on when battery voltage becomes low. This LED light is not visible from a crowd of robots. In order to overcome this issue we modified BTCom so that it can turn of all LEDs when battery voltage becomes

[15] http://www.cyberbotics.com/products/webots/
[16] http://code.google.com/p/epuck-player-driver/
[17] http://wiki.roboteducation.org/
[18] http://pyserial.sourceforge.net/

critical. The exploited the hardware interrupt signal from Low-Voltage-Detection (LVD) module of E-puck hardware.

Finally we developed our custom navigation and obstacle avoidance algorithms in Python. The navigation function is based on our camera pose information. In each time-step, robot gets it current pose information from our multi-robot tracking system. It then determines its current coordinate (location) relative to the target object and calculates the differences in pose and orientation. To advance forward, it at first corrects its heading based on the difference and then moves forward for a small fixed distance towards the target. Of course, in every time-step, it also checks that if it is located within the target object's boundary and if that is the case it ceases its motion. Obstacle avoidance algorithm works under the navigation code. While a robot tries to move forward if an obstacle is sensed by its IR sensor it makes a random turn and tries to avoid it. Due to the noisy sensor values, it takes two or a few time-steps to completely get rid of that obstacle.

3.3.4 BlueZ: Linux's Bluetooth communication stack

The physical communication between the host PC and E-puck robot occurs over Bluetooth wireless radio communication channel. As defined by the Bluetooth Special Interest Group's official technology info site [19]

> *Bluetooth* technology is a wireless communications technology intended to replace the cables connecting portable and/or fixed devices while maintaining high levels of security. The key features of Bluetooth technology are robustness, low power, and low cost.

The obvious reason for selecting Bluetooth as the communication technology of E-puck is perhaps due to it's low cost, low battery usage and universality of hardware and software. Each E-puck robot has a Bluetooth radio link to connect to a host PC or nearby other E-puck robots. Under the hood, this Bluetooth chip, LMX9820A [20], is interfaced with a UART (Universal Asynchronous Receiver/Transmitter) microchip. The bluetooth chip can be used to access to the UART ”transparently” using a bluetooth rfcomm channel. Using this mode, one can access the e-puck as

[19] www.bluetooth.com
[20] http://www.national.com/opf/LM/LMX9820A.html

if it is connected to a serial port. According to the specification of LMX9820A, it supports Bluetooth version 1.1 qualification that means the maximum supported data transfer speed is 1Mbit/s. But typically it is configured to use a serial port's 115200 bits/s speed.

Before starting to use an E-puck robot one needs to set-up the Bluetooth connection with the robot. Typical Bluetooth connection set-up from a Bluetooth-enabled host PC includes a few manual steps: detecting the remote Bluetooth device, securely bonding the device (e.g. exchanging secret keys) and setting-up the target *rfcomm* or serial connection (over radio) channel. Various Bluetooth software stacks are available under different OSes. Under Linux, BlueZ [21] becomes the *de-facto* standard software platform. The BlueZ stack was initially developed by Max Krasnyansky at Qualcomm [22] and in 2001 they decided to release it under the GPL. The BlueZ kernel modules, libraries and utilities are known to be working prefect on many architectures supported by Linux. It offers full support for Bluetooth device scanning, securely pairing with devices, automatic rfcomm or serial link configuration, monitoring and so forth. Initial scanning and secure bonding of E-puck devices can be done by a set of BlueZ tools, namely, *hcitool, l2ping, hciconfig, rfcomm* etc. While initializing, BlueZ's core daemon, *bluetoothd*, reads the necessary configuration files (e.g. rfcomm.conf) and dynamically sets up or binds all Bluetooth devices' links and thereafter, routes all low-level communications to them. BlueZ's supplementary package Hcidump offers logging raw data of all Bluetooth communications over a host PC's Bluetooth adapter. In our host PC, we have used USB dongle type Bluetooth adapter. We have also used various Linux serial connectivity tools, e.g *minicom, picocom etc.* to test link configurations and to send BTCom commands to E-puck robots.

From the above points, we can see that setting up and maintaining connectivity to E-puck robots through Bluetooth links is not a trivial task. Thus, one needs to consider automating the process of Bluetooth link set-up and verification in order to save time in initializing real experiments. Moreover, comparing with other common wireless technologies, e.g. Wifi, Bluetooth is a relatively low-bandwidth technology. In case of almost all wireless technologies, presence of lots of wireless

[21] www.bluez.org

[22] http://www.qualcomm.com/

devices causes significant noises and interferences. Thus, one also needs to consider the channel capacity or total available bandwidth for communications. Within the context of our experiments, we have got an interesting open question: *What is the maximum number of E-puck robots that can talk to host PC simultaneously*. However, finding the answer of this question is beyond the scope of this thesis and here we would only like to stick with the feasible configuration of Bluetooth links without modifying any low-level protocols or technical implementation.

3.3.5 Python's Multiprocessing: process-based multi-threading

The real-time interactions among multiple software applications often require concurrency and synchronization, to some degrees, in their functions. Although a common inter-process communication (IPC) protocol, e.g. D-Bus, solves the problem of data-sharing among different application processes, synchronization of data in various processes remains a challenging issue. The idea of simultaneous and parallel execution of different part of application codes without any IPC, typically on multiple CPU cores, introduces the notion of multi-threading programming. Both process-based and thread-based approach of program execution has pros and cons. Threads are light weight and they can share memory and state with the parent process without dealing with the complexity of IPC. Threads can be useful to the algorithms which rely on shared data/state. They can increase throughput by processing more information faster. They can also reduce latency and improve the responsiveness of an application, such as GUI actions. However, since threads implicitly "share everything" programmers have to protect (lock) anything which will be shared between threads. Thus thread-based programs are subject to face race conditions or deadlocks among multiple threads.

On the other hand, processes are independent process-of-control and they are isolated from each other by the OS. In order to do any data/state sharing they must use some form of IPC to communicate/coordinate. Comparing with threads, processes are big and heavy since process creation takes time and these processes also tends to be large in program size and memory footprint. Since processes "share nothing" – programmers must explicitly share any data/state with suitable mechanism. From a high-level robotic programmer's point of view, both thread-based and process-based application design approaches are disadvantageous. Since thread-based approach requires careful attention in data-sharing it becomes very difficult to design bug-free program in short time-scale. On the other hand process-based approach requires to set-up IPC mechanisms and manage them. However, the latter approach is less likely to produce bugs as data sharing is explicit.

Almost all modern computer OSes and *high-level* programming languages, e.g. C/C++, Java etc. offer multi-threading support. However implementation of multi-threading programming involves lots of low-level thread management activities. In this respect *very high-level* programming languages e.g. Python, Ruby etc. offer more efficient and elegant solutions for dealing with multi-theaded programs. Along with this multi-threading issue, various other factors influence us to use Python for coding our high level robotic programs. For example, Python programs are *interpreted* by various Python interpreters. Unlike dealing with compiling issues in most of the high-level programming languages, Python allows programmers to focus on their algorithms more quickly and integrate their systems more effectively. We have found Python's interactive program development process more productive and flexible than non-interactive programming approach found in some other languages.

Starting from from version 2.6 Python offers an integration of thread-based programming with process-based programming through it Multiprocessing module [23]. Traditionally, Python offers threads that are real, OS/Kernel level POSIX *pthreads*. But, older Python programs can have only a single thread to be executing within the interpreter at once. This restriction is enforced by the so-called "Global Interpreter Lock" (GIL). This is a lock which must be acquired for a thread to enter the interpreters space. This limits only one thread to be executing within the Python interpreter at once. This is enforced in order to keep interpreter maintenance easier. But this can also be sidestepped if the application is I/O (e.g. file, socket) bound. A threaded application which makes heavy use of sockets, wont see a a huge GIL penalty. After doing a lot of research on various alternatives of this approach, Python community has offered Multiprocessing as a feasible solution to side-step GIL by the CPU-bound applications that require seamless data/state sharing. It follows the threading API closely but uses processes and IPC under the hood . It also offers distributed-computing facilities as well, e.g. remote data-

[23] http://docs.python.org/library/multiprocessing.html

sharing and synchronization. Thus, we have exploited the power and efficiency of Multiprocessing that enables us to make a modular and flexible implementation of multi-robotic software system. Sec 3.4 explains some of our implementations of Python Multiprocessing module.

Python Multiprocessing offers various mechanisms for sharing data among processes or, more precisely speaking, among sub-processes. We have used a separate *Manager* process that handles all the data storage tasks and event-based process synchronizations. Managers are responsible for network and process-based sharing of data between processes (and machines). The primary manager type is the BaseManager - this is the basic Manager object, and can easily be subclassed to share data remotely. We use this Multiprocessing Manager object that runs a server process in one machine and offers data objects through proxies in parallel to many client processes over network interfaces.

3.4 Multi-robot control architecture

Controlling a robot in a well-organized manner involves following a control architecture. As defined by (Mataric 2007), a robot control architecture is a set of guiding principles and constraints for organizing a robot's control system. Since last few decades, robot control architectures has been evolving from deliberative to reactive and hybrid (combination of deliberative and reactive), behaviour-based and to some other forms. It has been well established that hybrid control can bring together the best aspects of both reactive and deliberative control by combining the real-time low-level device control and high-level deliberative action control. Only reactive (or deliberative) control approach is not enough for enabling robots to do complex tasks in a dynamic environments ((Gat et al. 1997)).

As shown in Fig. 3.3, this is usually achieved by a three-layer architecture composed of deliberator, sequencer and controller . Controller usually works under real-time reactive feedback control loops to do simple tasks by producing primitive robot behaviours, e.g. obstacle avoidance, wall following etc. Deliberator performs time-consuming computations, e.g. running exponential search or computer vision processing algorithms. In order to achieve specific task goals, the middle component, sequencer, typically integrates both deliberator and controller

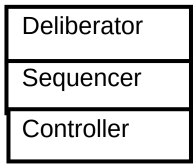


Figure 3.3: Classical three layer robot control architecture

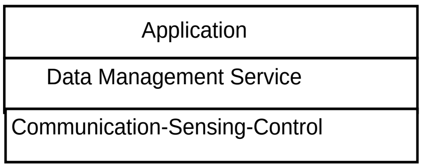


Figure 3.4: Our abstract multi-robot control architecture

maintaining consistent, robust and timely robot behaviours.

3.4.1 Hybrid event-driven architecture on D-Bus (*HEAD*)

We organized our multi-robot control architecture, HEAD into three layers as shown in Fig. 3.4. Although HEAD has been designed by adopting the principles of hybrid architecture it has many distinct features that are absent in a classical hybrid architecture. Firstly, with respect to controller layer, HEAD broadly views sensing and control as communication with external entities. Communication as sensing is not new, e.g. it has been reported in multi-agent learning ((Mataric 1998)) and many other places. When robots' on-board computing resources are limited communication can effectively make up their required sensing capabilities. On the other hand, low-level device control is also a series of communication act where actuator commands are typically transmitted over a radio or physical link. Thus, at the bottom layer of HEAD is the communication layer where all external communication takes place over any suitable medium. Components sitting in this layer either act as sensors that can receive environmental state, task information, self pose data etc. via suitable communication link or do the real-time control of devices by sending actuator commands over a target communication channel.

Secondly, the apparent tight coupling with sensors to actuators has been reduced by introducing a data and event management (DEM) layer. DEM acts as a short-term storage of sensed data and various events posted by both controller and deliberator components. Task sequencing has been simplified by automated event triggering mechanism. DEM simply creates new event channels and interested components subscribe to this event for reading or writing. If one components updates an event DEM updates subscribed components about this event. Controller and deliberator components synchronize their tasks based on this event signals. DEM efficiently serves newly arrived data to controller and deliberator components by this event mechanism. Thus neither specialized languages are needed to program a sequencer nor cumbersome if/else checks are present in this layer.
Finally, deliberator layer of HEAD has been described as an application layer that runs real-application code based on high-level user algorithms as well as low-level sensor data and device states. In classic hybrid architecture the role of this layer has been described mainly in two folds: 1) producing task plan and sending it to sequencer and 2) answering queries made by sequencer. Application layer of HEAD follows the former one by generating plan and queuing it to DEM layer, but it does not support the latter one. DEM layer never makes a query to an application since it acts only as a passive information gateway. Thus this reduced coupling between DEM and application layer has enhanced HEAD with additional robustness and scalability. Additional applications can be added with DEM layer's existing or new event interfaces. Any malfunction or failure in application layer or even in communication layer can be isolated without affecting others.

3.4.2 Robot controller clients

3.4.3 D-Bus signal interfaces

D-Bus IPC allows us to completely decouple the interaction of different parts of software components of HEAD. Here we use the term *software component* or application to denote the logical groupings of several sub-processes or threads that works under a mother process or main thread (here we use the term thread and process interchangeably). Software components that follows our three-layer architecture for grouping its processes are called *native component* whereas existing software applications are called *external component*. As shown in Fig. 3.5,

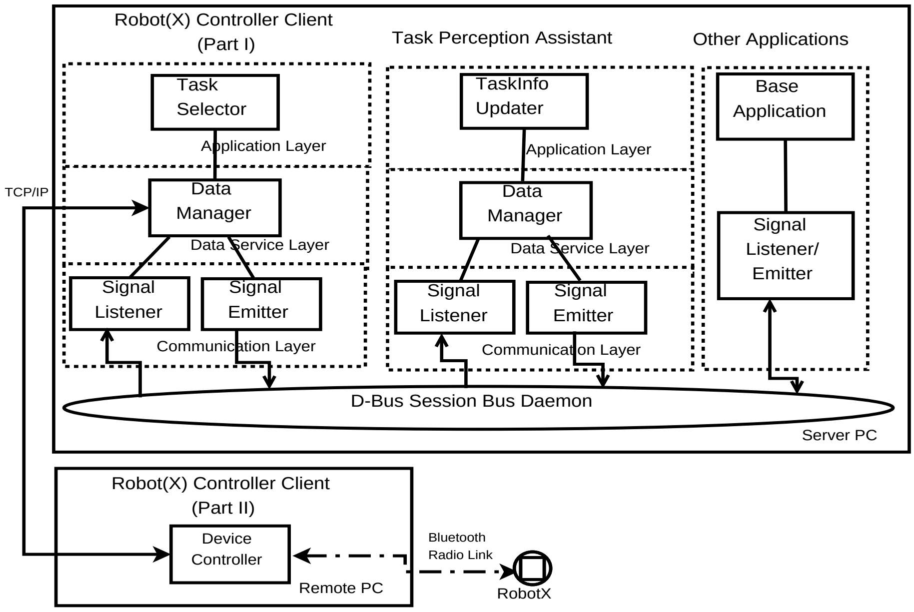


Figure 3.5: General outline of *HEAD*.

RCC and TPA are native software components of HEAD whereas SwisTrack, a multi-robot tracking tool ((Lochmatter et al. 2008)) used with HEAD, is called an external component.

3.4.4 Software integrations

In order to integrate both native and external components with HEAD. We have designed two separate communication process: a D-Bus signal reception process, SignalListener, and a D-Bus signal emission process, SignalEmitter. Inside a native component both of this process can communicate with data and event management process, DataManager, by using any suitable mechanisms, such as, multi-threading, multi-processing (offered in Python multiprocessing), TCP or any other networking protocol.

Any external component that intend to act as a sensing (actuating) element of HEAD need to implement a SignalEmitter (SignalListener). For example, we extend SwisTrack with D-Bus signal emitting code (aka SignalEmitter) so that it can emit robot pose messages to individual robots D-Bus path under a common inter-

[23] http://docs.python.org/library/multiprocessing.html

face (uk.ac.newport.SwisTrack). This emitted signal is then caught by SignalListener of individual robot's RCC. Thus the tight-coupling between SwisTrack and RCC has been removed. During run-time SwisTrack can flexibly track variable number of robots and broadcast their corresponding pose messages without any re-compilation of code. Moreover, in worse cases, if SwisTrack or RCC crashes it does not affect any other component at run-time.

Expanding SignalEmitter and SignalListener for more D-Bus signals does not require to make any change the IPC implementation code. Let us first look at how to setup a signal emission process.

Steps for setting up signal emission:

Step 1: Connect to a D-Bus daemon. Sample C code:

```
DBusError  error;
DBusConnection *conn;
dbus_error_init (&error);
conn = dbus_bus_get (DBUS_BUS_SESSION, &error);
```

Step 2: Optionally reserve a D-Bus path or service name (this is not required if the same path is not used by any other process).

Step 3: Send signal to a specified path. Sample C code:

```
DBusMessage *message;
message = dbus_message_new_signal (
``/target/dbus/path'',``target.dbus.interface'',
``Config'');
/* Send the signal */
dbus_connection_send (connection, message, NULL);
dbus_message_unref (message);
```

In order to add more signals we just need to repeat step 3 as many times as we need. On the other hand, signal listening can be done by setting up a suitable event loop under any supported language bindings. A basic implementation of both of these processes in Python language can be found in this tutorial [24].

[24] http://dbus.freedesktop.org/doc/dbus-python/doc/tutorial.html

CHAPTER 4

Validation of Attractive Field Model for Self-regulated MRTA

4.1 Motivations

In this chapter we have discussed about the attractive field model (AFM) (Arcaute et al. 2008) an interdisciplinary model of self-regulated task-allocation or division of labour (DOL) in social systems. The model has been originated from the EPSRC collaborative project "Defying the rules how self-organizing systems work". Under this project, we have studied three different social systems: ants, humans and robots in order to identify generic mechanisms that lead sustainability of social systems through self-regulation [1]. The idea of finding a generic model of DOL, by collaboratively studying these social systems, has many fascinating advantages. Firstly, this interdisciplinary study makes it possible to develop a generic model of self-regulatory DOL by combining the strengths of different disciplines overcoming their individual shortcomings. For example, ant colonies are the ideal example for studying the self-regulatory social systems, however it is very difficult to pinpoint the exact mechanism leading to a specific behaviour (Arcaute et al. 2008). Artificial systems e.g. MRS can be used to explore and verify biological hypotheses using totally controlled experiments. Similarly observational data from human

[1]The partners of this project were from University of West of England, University of Hull, University of Wales, Newport and Imperial College London and they researched on ants, humans, robots and mathematical models respectively

social systems can be combined with the data from the biological experiments to enhance our understanding of social self-regulatory mechanisms.

Secondly, synergy of different methods, experimental and observational data from disparate disciplines gives us the ability to construct a more abstract, yet powerful, model which may not be available through independent studies. The higher-level abstraction can be very helpful to increase the usefulness of this model since we can easily separate generic and the domain-specific components of the model. Generic part guides us to design a core-framework that can be used to create basic characteristics of a system, whereas domain-specific part can be implemented independent of other disciplines. For example,different social systems use different communication mechanisms, yet almost all of them share many common aspects in their self-regulatory behaviours (see Sec. 2.2 for an example).

Thirdly, the scope of application of our generic models has been extended in many folds by tackling the common challenges of different disciplines. For example, both human social organizations and multi-robot systems suffer from the scalability issue for large organization. The properties of local sensing and local communication with relatively incapable sensory organs/hardware are present in both ants and swarm robotic systems. Thus, the integration of solutions from three major disciplines gives us a highly flexible and extensible model of DOL.

The construction of AFM has been achieved through a series of collaborative interactions among all project partners. From the biological experiments of ants colonies *Temnothorax albipennis* we inferred the bottom-level rules and roles of feedback in collective performance of ants brood-sorting and nest construction after emigration to a new site. We also analysed the observational data from the self-organized infrastructural development of an *eco-village* by an open community of volunteers resided in Ireland. These helped us to identify the generic rules of DOL in social systems. Later on, we formalized these generic rules into AFM (Arcaute et al. 2008) and validated them by putting AFM into our robot controllers within the context of a virtual manufacturing shop-floor scenario. In this chapter, we have described mainly the later part of our study, i.e. robotic validation of AFM, with a brief presentation on interpretations of AFM from different social perspectives.

Why communication-based DOL ?

For the following reasons, we rely on communication among robot group to exploit our generic mechanism of division of labour.
Firstly, from our understanding of different kinds of communication strategies of biological social systems (Sec. 2.2), this is obvious that the role of communication can not be ignored for achieving division of labour in MRS. Instead of being too much addicted to communication-less algorithms, perhaps due to the limitation of current communication technology (e.g. mimicking biological stigmergy), we need to exploit the existing state of the art in communication technology for developing functional and robust division of labour mechanisms for future MRS. By selecting a suitable communication strategy and enabling robots to self-regulate their certain behaviours, we can significantly reduce the communication load of a MRS. Secondly, Whatever be the objectives of a target MRS under any research approach, e.g., maximizing robustness, scalability and/or task performance, the issue of task-allocation of an individual robot remains same, i.e., what task should it select at a particular time point considering dynamically changing task requirements, choices of peer robots and environmental conditions. In this issue, unlike traditional MRS approach, I emphasize on maintaining the overall group level performance and robustness, not just focusing on the instantaneous maximum benefit (or minimum cost) of a robot by performing a particular task. Finally, by combining the above two points, we propose to achieve division of labour in MRS as a self-regulated task allocation process of a group of robots, where robots can dynamically select suitable tasks, or can switch from one task to another based on continuously sensed and communicated information of tasks, states etc. through their respective sensory and communication channels. Thus, by adopting this self-regulated task selection and communication strategies, some of the robots can have chances to specialize on some particular tasks, and as a whole, the system can maintain a level of plasticity without producing unnecessary communication burden on the system.

4.2 The Attractive Field Model (AFM)

4.2.1 Generic framework

Inspired from the DOL in ants, humans and robots, we have proposed the following four rules that are the necessary ingredients to obtain self-regulation in any social system. In this dissertation, these rules are mentioned as *generic rules of division of labour*.

Rule 1: Continuous flow of information. Self-regulatory systems need to establish the continuous minimum flow of information over period of time, where self-regulation can be defined, for at least two states of an agent: 1) receiving information about task(s) and 2) ignoring information or doing no task. The updated information will reflect the changes of the system i.e. it will encode the necessary feedback for the agents. Thus, this property will act as the basis of the smooth switching of states, between these two minimum states or, among multiple states (e.g. in case of multiple tasks or many sub-states of a single task) .

Rule 2: Sensitization. Self-regulatory systems allow the differentiation in use of or access to information, e.g. through sensitization or learning tasks. This differentiation is regulated by the characteristics of the system, e.g. the ability of the agents to learn tasks that are repeatedly performed.

Rule 3: Concurrence. Self-regulatory systems include concurrent access to information from different spatial positions with certain preferences. This preference is not fixed and can changes with the dynamics of the system.

Rule 4: Forgetting. Self-regulatory systems include forgetting, e.g. the ability of agents to diminish information over time, if not used. The system determines the amount of info being released, and this changes with time. For example, specialists might have to attend an emergency situation and switch tasks that contributes to the forgetting of old task experiences. This is considered as crucial to allow flexibility in the system.

Having this general framework of self-regulation, we can now formalize AFM that will describe the properties of individual agents and the system as a whole. In terms of networks, the model is a bipartite network, meaning that there are two different types of nodes. One set of nodes describes the sources of the attractive fields and the other set describes the agents. Links only exist between different types of nodes and they encode the flow of information so that, even if there is no direct link between two agents, their interaction is taken into account in the information flow (i.e. existence of *weak* interaction). The strength of the field depends on the distance between the task and the agent. This relationship is represented using weighted links. In addition, there is a permanent field that represents the no-task or option for ignoring information. The model can be mapped to a network as shown in Fig. 4.1. The correspondence is given below:

1. Source nodes (o) are tasks that can be divided between a number of agents.

2. Agent nodes (x) an be ants, human, robots etc.

3. The attractive fields correspond to stimuli to perform a task, and these are given by the black solid lines.

4. When an agent performs a task, the link is of a different sort, and this is denoted in the figure by a dashed line. Agents linked to a source by a red line are the agents currently doing that task.

5. The field of ignoring the information (w) corresponds to the stimulus to random walk, i.e. the no-task option, and this is denoted by the dotted lines in the graph.

6. Each of the links is weighted. The value of this weight describes the strength of the stimulus that the agent experiences. In a spatial representation of the model, it is easy to see that the strength of the field depends on the physical distance of the agent to the source. In addition, the strength can be increased through sensitisation of the agent via experience (learning). This distance is not depicted in the network, it is represented through the weights of the links . In the figure of the network, the nodes have an arbitrary place. Note that even though the distance is physical in this case, the distance in the model applied to other systems, needs not to be physical, it can represent the accessibility to the information, the time the information takes to reach the receiver, etc.

In summary, looking at the network, we see that each of the agents is connected to each of the fields. This means that even if an agent is currently involved in a task,

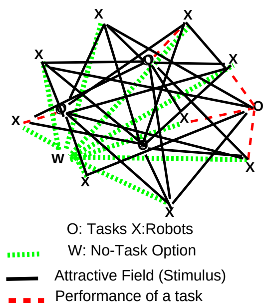


Figure 4.1: Attractive Filed Model (AFM)

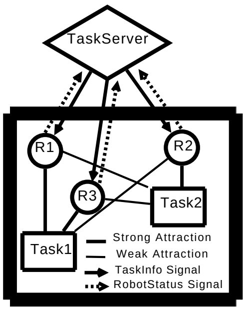


Figure 4.2: A centralized communication scheme

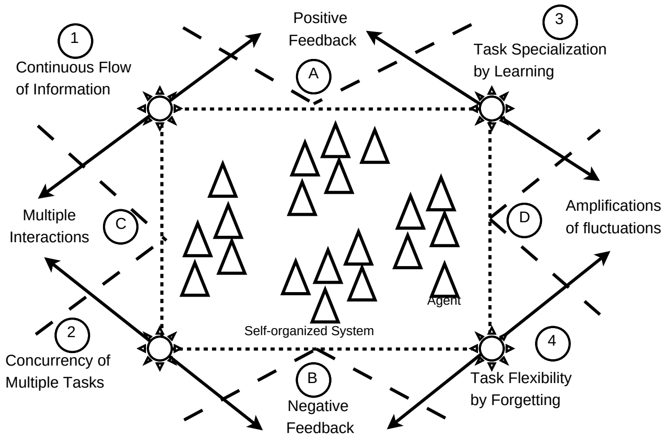


Figure 4.3: Four generic rules establish the self-regulated DOL in social systems

the probability that it stops doing it in order to pursue a different task, or to random walk, is always non-zero. The weighted links express the probability of an agent to be attracted to each of the fields.

4.2.2 Relationship of AFM with self-organization

It is interesting to note that our proposed four generic rules has become the four major corners of the foundation of a self-organized system (Fig. **??**). As discussed in Sec. 2.1.1, self-organized systems exhibit four distinct perspectives known as so-called ingredients or properties of self-organization. However, it is not clear how those properties can come into existence. Here, we describe the four underlying mechanisms that explains how self-organization can be realized in different social systems. From our understanding, we can explain it in the following ways.

Firstly, multiple interaction becomes meaningful when *continuous flow of information* occurs by exchanging signals or cues among agents or their environment that regulates their behaviours. This, in turn, contribute to the task-allocation and switching in the social level. In the biological and swarm-intelligence (SI) literature, multiple interactions are often described as an essential ingredient of self-organization. However, random interaction without any definite purpose may not contribute to self-organization at all.

Secondly, in SI positive feedback has been attributed as another mechanism of self-organization. But it is not easy to understand what creates positive feedback in a social system. Possible answers might be the characteristic of the environment (e.g., ants select shorter path since density of pheromones becomes higher and thus more ants becomes attracted in that path), the decrease of response-threshold of individuals (thus increase of probability of selecting a task) etc. To make the answer more concrete, we have explicitly attributed *sensitisation* or learning as a mechanism of positive feedback. There might exist other mechanisms too. But clearly sensitisation will be one of the reliable mechanisms for achieving positive feedback.

Thirdly, similar to positive feedback, we have proposed *forgetting* that contributes to provide negative feedback about a task or decreasing the probability to select it. Other negative feedback mechanisms can be implemented by assigning a saturation level to each task which is also present in our model (Arcaute et al. 2008).

Finally, creating artificial amplification of fluctuations or stochastic events is not a straight-forward issue. It throws many open questions. Does a system designer intentionally impose irregularity in task-performance of agents ? Is random movement enough for simulating randomness in a system ? Since emergencies do not always pop-up on request, we provide the rule of *concurrency* that enables agents to maintain even a small amount of probability of selecting a low-priority or less sensitized or distant task. This concurrency mechanism provides a high-degree of robustness in the system such that all tasks can be attended even if specialization of agents delays them in switching to some tasks.

4.2.3 Interpretation of AFM in an ant colony

The interpretation of AFM in an ant colony almost exactly follows the above generic interpretation. Moreover it also reveals a few additional characteristics of AFM. For example, at the individual level, information is processed differently by each individual, and is certainly not constant nor continuous. In addition, there are lower and upper thresholds on the amount of info necessary for DOL to take place. The time scale at the individual level is very small compared to the system level's time scale, and since it is this latter time scale that we are interested in, we can approximate the propagation of information at this macro time scale to a continuous. In the model emphasize is given to whether the information is used e.g. stimulation to perform a task, or unused e.g. random walk (RW). In case of sensitization or learning, there is no encoding of individual performance, the specialists are more attracted towards the task but do not perform quicker the task. This could be implemented, although it does not appear to be a necessary condition for DOL, since the performance of colony increases as a result of sensitisation. Simultaneity (concurrence) of tasks over period of time of self-regulation is achieved through spatial dependence of strength of stimulus. Flexibility in task switching is achieved through forgetting. A concrete interpretation of AFM in an ant colony, along with simulation results can be found in (Arcaute et al. 2008).

4.2.4 Interpretation of AFM in a human society

The interpretation of AFM in a human society can be made using many different approaches. For example the following can be mapped to different nodes and characteristics of AFM in a human society. Source nodes (o) can be resources
Agents (x) can be people
The links correspond to the flow of information, resources, etc.
The distance dependence can be introduced here in the same way as with the ants. People working in a certain area are more likely to receive info/resources with respect to that area, than another person doing something else. How the person uses that info/resources, corresponds to the people's ability to contribute towards a given goal, defined by the informal network.
The weight of the link is the amount of info a person gets. The person can use it or dismiss it. Random walking in the ants would correspond to dismiss it. AFM does

not differentiate between good and bad info, it is only considered as good.

4.2.5 Interpretation of AFM in a MRS

The interpretation of AFM in a MRS also follows almost exactly as in generic interpretation. However, in order to make the interpretation more concrete, let us consider a manufacturing shop floor scenario where N number of mobile robots are required to attend to M number of shop tasks spread over a fixed area A. Let these tasks be represented by a set of small rectangular boxes resembling to manufacturing machines. Let R be the set of robots $r_1, r_2, ..., r_i$ and J the set of tasks $t_1, t_2, ..., t_j$. Each task t_j has an associated task-urgency ϕ_j indicating its relative importance over time. If a robot attends a task t_j in the xth time-step, the value of ϕ_j will decrease by an amount δ_ϕ in the (x+1)th time-step. On the other hand, if a task has not been served by any robot in the xth time-step, ϕ_j will increase by another amount in (x+1)th time-step. In order to complete a task t_1, a robot r_i needs to be within a fixed boundary D_{j1} of t_1. If a robot completes a task t_j it gets sensitised to it and this will increase the robot's likelihood of selecting that task in the future. We call this variable the affinity of a robot r_i to task t_j its *sensitization* k^i_j. If a robot does not do a task t_j for some time, it forgets about t_j and k_j is decreased.

According to AFM, all robots will establish attractive fields to all tasks due to the presence of a system-wide continuous flow of information. The strength of these attractive fields will vary according to the distances between robots and tasks, task-urgencies and corresponding sensitizations, $S_{i,j}$ of robots. This is encoded in Eq. 4.1.

$$S^i_j = tanh\{\frac{k^i_j}{d_{ij} + \delta}\phi_j\} \qquad (4.1)$$

$$P^i_j = \frac{S^i_j}{\sum_j S^i_j} \qquad (4.2)$$

Eq. 4.1 states that the stimuli of a robot i to a particular task j, S^i_j depends on i's spatial distance to j (d_{ij}), level of sensitization to j (k^i_j), and perceived urgency of that task (ϕ_j). We use a very small constant value δ to avoid division by zero when a robot has reached a task. Since S^i_j is a probability function, it is chosen as a $tanh$ in order to keep the values between 0 and 1. The probability of selecting each task

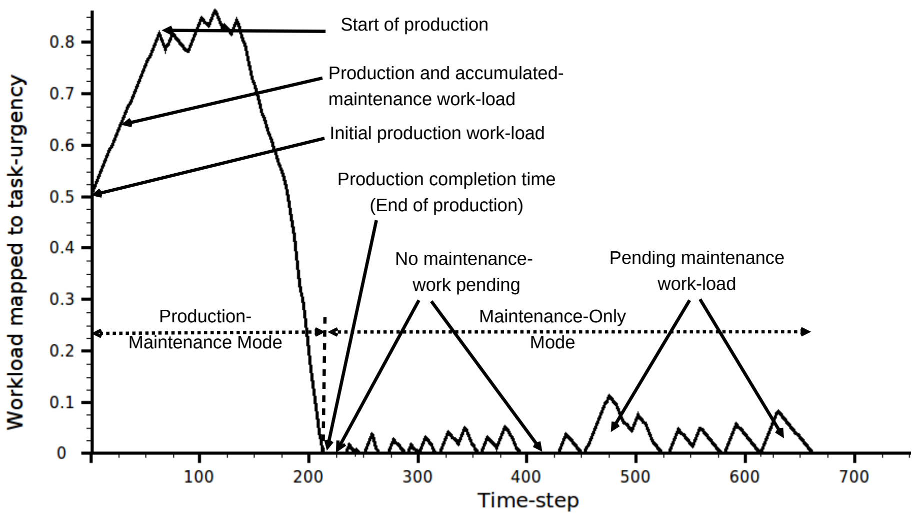


Figure 4.4: Virtual Shop-floor production and maintenance cycle

has been determined by a probabilistic method outlined in Eq. 4.2. AFM suggests that concurrency of a self-regulatory system can be maintained by specifying at least two task options: doing a task and not doing a task. In robots, the latter can be be treated as random walking. So in any time-step a robot will choose from M+1 tasks. Let T_a be the allocated time to accomplish a task. If R_1 can enter inside the task boundary within T_a time it waits there until T_a elapsed. Otherwise it will select a different task.

4.2.6 A case study: virtual manufacturing shop-floor

In our virtual manufacturing shop-floor (VMS) scenario, each task represents a manufacturing machine. These machines are capable of producing goods from raw materials, but they also require constant maintenance works for stable operations. Let W_j be a finite number of material parts that can be loaded into a machine j in the beginning of its production process and in each time-step, ω_j units of material parts can be processed ($\omega_j \ll W_j$). So let Ω_j^p be the initial production workload of j which is simply: W_j/ω_j unit. We assume that all machines are identical. In each time step, each machine always requires a minimum threshold number of robots, called hereafter as *minimum robots per machine (μ)*, to meet its constant maintenance work-load, Ω_j^m unit. However, if μ or more robots are present in a machine for production purpose, we assume that, no extra robot is required to do its maintenance work separately. These robots, along with their production jobs, can do necessary maintenance works concurrently. For the sake of simplicity, in this paper we consider μ = 1. Now let us fit the above production and maintenance work-loads and task performance of robots into a unit task-urgency scale. Let us divide our manufacturing operation into two subsequent stages: 1) *production and maintenance mode (PMM)*, and 2) *maintenance only mode (MOM)*. Initially a machine starts working in PMM and does production and maintenance works concurrently. When there is no production work left, it then enters into MOM. Fig. 4.4 illustrates this for a single machine. Under both modes, let α_j be the amount of workload occurs in a unit time-step if no robot serves a task and it corresponds to a fixed task-urgency $\Delta\phi_{INC}$. On the other hand, let us assume that in each time-step, a robot, i, can decrease a constant workload β_i by doing some maintenance work along with doing any available production work. This corresponds to a negative task urgency: $-\Delta\phi_{DEC}$. So, at the beginning of production process, task-urgency, occurred in a machine due to its production work-loads, can be encoded by Eq. 4.3.

$$\Phi_{j,INIT}^{PMM} = \Omega_j^p \times \Delta\phi_{INC} + \phi_j^{m0} \tag{4.3}$$

where ϕ_j^{m0} represents the task-urgency due to any initial maintenance work-load of j. Now if no robot attends to serve a machine, each time-step a constant maintenance workload of α_j^m will be added to j and that will increase its task-urgency by $\Delta\phi_{INC}$. So, if k time steps passes without any production work being done, task urgency at k^{th} time-step will follow Eq. 4.4.

$$\Phi_{j,k}^{PMM} = \Phi_{j,INIT}^{PMM} + k \times \Delta\phi_{INC} \tag{4.4}$$

However, if a robot attends to a machine and does some production works from it, there would be no extra maintenance work as we assumed that μ = 1. Rather, the task-urgency on this machine will decrease by $\Delta\phi_{DEC}$ amount. If ν_k robots work on a machine simultaneously at time-step k, this decrease will be: $\nu_k \times \Delta\phi_{DEC}$. So in such cases, task-urgency in $(k+1)^{th}$ time-step can be represented by:

$$\Phi_{j,k+1}^{PMM} = \Phi_{j,k}^{PMM} - \nu_k \times \Delta\phi_{DEC} \tag{4.5}$$

At a particular machine j, once $\Phi_{j,k}^{PMM}$ reaches to zero, we can say that there is no more production work left and this time-step k can give us the *production* *completion time* of j, T_j^{PMM}. Average production time-steps of a shop-floor with M machines can be calculated by the following simple equation.

$$T_{avg}^{PMM} = \frac{1}{M}\sum_{j=0}^{M} T_j^{PMM} \tag{4.6}$$

T_{avg}^{PMM} can be compared with the minimum number of time-steps necessary to finish production works, T_{min}^{PMM}. This can only happen in an ideal case where all robots work for production without any random walking or failure. We can get T_{min}^{PMM} from the total amount of work load and maximum possible inputs from all robots. If there are M machines and N robots, each machine has Φ_{INIT}^{PMM} task-urgency, and each time-step robots can decrease N $\times$ $\Delta\phi_{DEC}$ task-urgencies, then the theoretical T_{min}^{PMM} can be found from the following Eq. 4.7.

$$T_{min}^{PMM} = \frac{M \times \Phi_{INIT}^{PMM}}{N \times \Delta\phi_{DEC}} \tag{4.7}$$

$$\zeta_{avg}^{PMM} = \frac{T_{avg}^{PMM} - T_{min}^{PMM}}{T_{min}^{PMM}} \tag{4.8}$$

Thus we can define ζ_{avg}^{PMM}, *average production completion delay* (APCD) by following Eq. 4.8: When a machine enters into MOM, only μ robots are required to do its maintenance works in each time step. So, in such cases, if no robot serves a machine, the growth of task-urgency will follow Eq. 4.4. However, if ν_k robots are serving this machine at a particular time-step k^{th} , task-urgency at $(k+1)^{th}$ time-step can be represented by:

$$\Phi_{j,k+1}^{MOM} = \Phi_{j,k}^{MOM} - (\nu_k - \mu) \times \Delta\phi_{DEC} \tag{4.9}$$

By considering $\mu = 1$ Eq. 4.9 will reduces to Eq. 4.5. Here, $\Phi_{j,k+1}^{MOM}$ will correspond to the *pending maintenance work-load (PMW)* of a particular machine at a given time. This happens due to the random task switching of robots with a no-task option (random-walking). Interestingly PMW will indicate the robustness of this system since higher PMW value will indicate the delay in attending maintenance works by robots. We can find the average PMW (APMW) per time-step per machine, χ_j^{MOM} (Eq. 4.10) and average PMW per machine per time-step, χ_{avg}^{MOM} (Eq. 4.11).

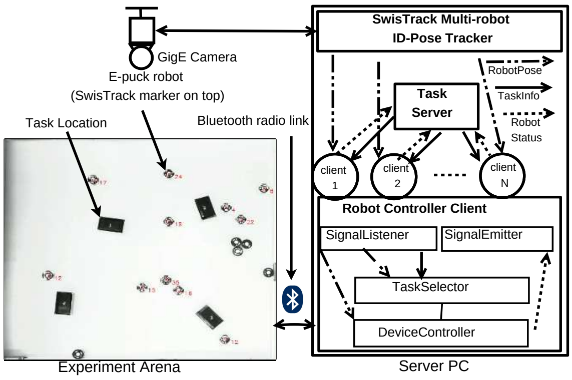


Figure 4.5: Hardware and software setup

$$\chi_j^{MOM} = \frac{1}{K}\sum_{k=1}^{K}\Phi_{j,k}^{MOM} \qquad (4.10) \qquad\qquad \chi_{avg}^{MOM} = \frac{1}{M}\sum_{j=1}^{M}\chi_j^{MOM} \qquad (4.11)$$

4.3 Implementation

We have developed a system a multi-robot tracking system can track at least 40 E-puck robots [2] and these robots can operate together according to the generic rules of the AFM. As shown in Fig. 5.2, our software system consists of a multi-robot tracking system, a centralized task server and robot controller clients. Here at first we have presented the design of our communication system. Then we have discussed about our specific implementation.

4.3.1 Design of communication system

In order to establish a system-wide continuous flow of information, we need to implement a suitable communication system for our robots. Here we have presented a centralized communication system for our manufacturing shop-floor scenario. As shown in Fig. 4.2, in this model there exists a centralized *TaskServer* that is responsible for disseminating task information to robots. The contents of task information can be physical locations of tasks, their urgencies and so on. TaskServer delivers this information by emitting *TaskInfo* signals periodically. The method of signal emission depends on a particular communication technology. For example,

[2] www.e-puck.org

in a wireless network it can be a message broadcast. Task-Server has another interface for catching feedback signals from robots. The *RobotStatus* signal can be used to inform TaskServer about a robot's current task id, its device status and so on. TaskServer uses this information to update relevant part of task information such as, task-urgency. This up-to-date information is sent in next TaskInfo signal.
In Fig. 4.2 an initial configuration of this model has been presented. Upon receiving an initial TaskInfo signal robot R_1 has shown strong attraction towards $Task1$ and robot R_3 has shown strong attraction toward $Task2$. This can be inferred from Eq. 4.1 that says if the initial task urgencies and sensitizations for all tasks are same, a robot will strongly be attracted towards a task that is relatively closer to it.

4.3.2 MRS implementation

The major components of our implementation are a multi-robot tracking system, robot controller clients and a centralized task-server. In order to track all robots real-time we have used SwisTrack (**?**), a state of the art open-source, multi-agent tracking system, with a16-megapixel overhead GigE camera. This set-up gives us the position, heading and id of each of the robots at a frequency of 1. The interaction of the hardware and software of our system is illustrated in Fig. 5.2.
For inter-process communication (IPC), we have used D-Bus technology [3]. We have developed an IPC component for SwisTrack (hereafter called as *SwisTrack D-Bus Server*) that can broadcast id and pose of all robots in real-time over our server's D-Bus interface.
Apart from SwisTrack, we have implemented two major software modules: *TaskServer* and *Robot Controller Client (RCC)*. They are developed in Python with its state of the art *Multiprocessing* [4] module. This python module simplifies our need to manage data sharing and synchronization among different sub-processes. As shown in Fig. 5.2, RCC consists of four sub-processes. *SignalListener* and *SignalEmitter*, interface with SwisTrack D-Bus Server and TaskServer respectively. *TaskSelector* implements AFM guidelines for task selection . *DeviceController* moves a robot to a target task. Bluetooth radio link is used as a communication medium between a RCC and a corresponding E-puck robot.

[3] http://dbus.freedesktop.org/doc/dbus-specification.html
[4] http://docs.python.org/library/multiprocessing.html

Table 4.1: Experimental parameters

Parameter	Value
Total number of robots (N)	16
Total number of tasks (M)	4
Experiment area (A)	4 m^2
Initial production work-load/machine (Ω_j^p)	100 unit
Task urgency increase rate ($\Delta\phi_{INC}$)	0.005
Task urgency decrease rate ($\Delta\phi_{DEC}$)	0.0025
Initial sensitization (K_{INIT})	0.1
Sensitization increase rate (Δk_{INC})	0.03
Sensitization decrease rate (Δk_{DEC})	0.01

4.4 Experiment design

In this section, we have described the design of parameters and observables of our experiments within the context of our virtual manufacturing shop-floor scenario. These experiments are designed to validate AFM by testing the presence of division of labour, such task specialization, dynamic task-switching or plasticity etc.

4.4.1 Parameters

Table 5.1 lists a set of essential parameters of our experiments. We intend to have a setup that is relatively complex, i.e., with a high number of robots and tasks in a large area. The diameter of the marker of our e-puck robot is 0.08m. So, if we put 4 robots in an area of one square meter, this will give us a robot-occupied-space to free-space ratio of about 1:49 per square meter. This ratio reasonable in order to allow the robots to move at a speed of 5 cm/sec without much interference to each other. We have fixed the number of tasks to 4. Robots also have an additional option for random walking that corresponds to the ignoring task information for ensuring flexibility of our system.

The initial values of task urgencies correspond to 100 units of production work-load without any maintenance work-load as outlined in Eq. 4.3. We choose a limit of 0 and 1, where 0 means no urgency and 1 means maximum urgency. Same applies to sensitisation as well, where 0 means no sensitisation and 1 means maximum sensitisation. This also implies that if sensitization is 0, task has been forgotten completely. On the other hand, if sensitization is 1, the task has been learnt completely. We choose a default sensitization value of 0.1 for all tasks. The following relationships are maintained for selecting task-urgency and sensitization parameters.

$$\Delta\phi_{INC} = \frac{\Delta\phi_{DEC} \times N}{2 \times M} \tag{4.12}$$

$$\Delta k_{DEC} = \frac{\Delta k_{INC}}{M - 1} \tag{4.13}$$

Eq. 4.12 establishes the fact that task urgency will increase at a higher rate than that of its decrease. As we do not like to keep a task left unattended for a long time we choose a higher rate of increase of task urgency. This difference is set on the basis of our assumption that at least half of the expected number of robots (ratio of number of robots to tasks) would be available to work on a task. So they would produce similar types of increase and decrease behaviours in task urgencies. Eq. 4.13 suggests that the learning will happen much faster than the forgetting. The difference in these two rates is based on the fact that faster leaning gives a robot more chances to select a task in next time-step and thus it becomes more specialized on it. Task-Server updates task-info messages in the interval of ΔTS_u=5s and robots stick on to a particular task for a maximum of ΔRT_{to}=10s.

4.4.2 Observables

We have defined a set of observables to evaluate our implementation. The first two observables, the changes in task-urgencies and the changes in active worker ratios, can give us an overall view of plasticity of division labour. Our third observable is to find changes in robot task specialization which is also an important measure of division of labour. Our last measurement is the communication load which is specific to this particular implementation and corresponds to the continuous flow of information. Within the context of our VMS, we measure the average production completion delay (APCD) and average pending maintenance work (APMW) as the metrics of VMS performance.

4.5 Results and discussions

In this section we have presented our experimental results. We ran those experiments for about 40 minutes and averaged them over five iterations. Fig. 5.3 shows the dynamic changes in task urgencies. In order to describe our system's dynamic behaviour holistically we analyse the changes in task urgencies over time. Let $\phi_{j,q}$ be the urgency of a task j at q^{th} step and $\phi_{j,q+1}$ be the task urgency of $(q+1)^{th}$ step. We can calculate the sum of changes in urgencies of all tasks at $(q+1)^{th}$ step:

$$\Delta\Phi_{j,q+1} = \sum_{j=1}^{M} (\phi_{j,q+1} - \phi_{j,q}) \tag{4.14}$$

From Fig. 4.7 we can see that initially the sum of changes of task urgencies are towards negative direction. This implies that tasks are being served by a high number of robots. Fig. 4.9 shows that in production stage, when work-load is high, many robots are active in tasks and this ratio varies according to task urgency changes.

Fig. 5.16 gives us the task specialization of five robots on Task3 in a particular run of our experiment. This shows us how our robots can specialize and de-specialize on tasks over time. The de-specialization or forgetting of tasks is calculated similar to Eq. 5.2. we have calculated the absolute sum of changes in sensitizations by all robots in the following equation.

$$\Delta K_{j,q+1} = \sum_{j=1}^{M} \mid (k_{j,q+1} - k_{j,q}) \mid \tag{4.15}$$

This values of ΔK are plotted in Fig. 4.11. It shows that the overall rate of learning decreases and forgetting increases over time. It is a consequence of the gradually increased task specialization of robots and reduced task-urgencies over time. Fig. 4.8 presents the frequency of signalling task information by TaskServer. Since the duration of each time step is 50s long and TaskServer emits signal in every 2.5s, there should be an average of 20 signals in each time-step.

Within VMS scenario, we have got average production completion time 165 time-steps (825s) where sample size is (5 x 4) = 20 tasks, SD = 72 time-steps (360s). According to Eq. 4.7, our theoretical minimum production completion time is 50 time-steps (250s) assuming the non-stop task performance of all 16 robots with an initial task urgency of 0.5 for all 4 tasks and task urgency decrease rate $\Delta\Phi_{DEC}$ = 0.0025 per robot per time-step. Hence, Eq. 4.8 gives us APCD, ζ = 2.3 which means that our system has taken 2.3 times more time (575s) than the minimum estimated time.

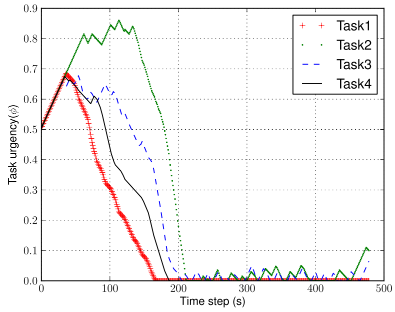


Figure 4.6: Task urgencies observed at TaskServer

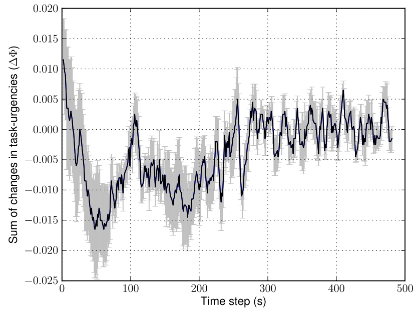


Figure 4.7: Shop-floor workload change history

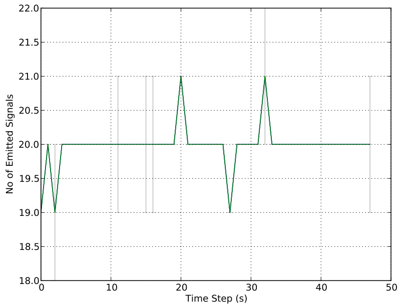


Figure 4.8: Task server's frequency of task information signalling

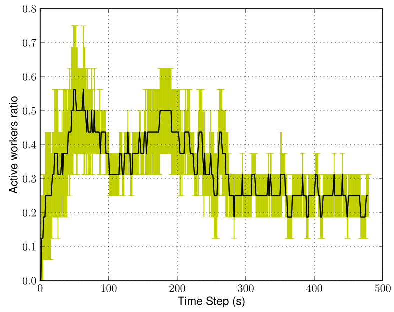


Figure 4.9: Self-organized allocation of workers

Besides, from the average 315 time-steps (1575s) maintenance activity of our robots per experiment run, we have got APMW, χ = 0.012756 which corresponds to the pending work of 3 time-steps (15s) with sample-size = 20 tasks, SD = 13 time-steps (65s), $\Delta\Phi_{INC}$ = 0.005 per task per time-step. This tells us the robust task performance of our robots which can return to an abandoned task within a minute or so.

4.6 Summary and conclusion

In this paper we have validated an inter-disciplinary generic model of self-regulated division of labour (DOL) or or multi-robot task allocation (MRTA) by incorporating it in our multi-robot system (MRS) that has emulated a virtual manufacturing shop-floor activities. A centralized communication system has been instantiated to

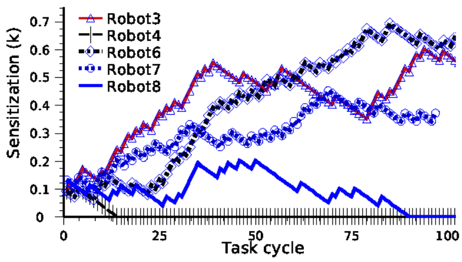


Figure 4.10: Task specialization on Task3

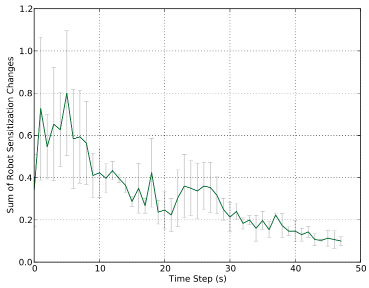


Figure 4.11: Changes in sensitizations of all robots

realize this model. We have evaluated various aspects of this model, such as ability to meet dynamic task demands, individual task specializations, communication loads and flexibility in concurrent task completions. A set of metrics has been proposed to observe the DOL in this system. From our experimental results, we have found that AFM can meet the requirements of dynamic DOL by the virtue of its self-regulatory behaviours. Our centralised communication system broadcasts information to all the robots from a central server. This has the advantage of minimising the communication load and the disadvantage of a single point of failure. In the future, we will explore local peer-to-peer communication models in a MRS having about 40 E-puck robots.

CHAPTER 5

Locality-based Peer-to-Peer Communication Model (LPCM)

5.1 Motivations

5.2 General characteristics of LPCM

Our communication model relies on the local P2P communications among robots. Here there is no centralized server to disseminate information but each robot can communicate to its nearby peers within a certain communication radius, r_{comm}. Here by r_{comm}, we assume that within this distance robots can exchange communication signals reliably without any significant loss of information. A robot R_1 is a *peer* of robot R_2, if spatial distance between R_1 and R_2 is less than its r_{comm}. As shown in Fig. **??**, local communication can also give robots similar task information as in centralized communication mode. It shows that it is not necessary for each robot to communicate with every other robot to get information on all tasks. Since robots can random walk and explore the environment we assume that for a reasonably high robot to space density, all task will be known to all robots after an initial exploration period. In order to update the urgency of a task, robots can estimate the number of robots working on a task in many ways: such as, by using their sensory perception (e.g., camera), by doing local P2P communication and so on. In Fig. **??** we have shown that robots exchange both task information and self status signals to peers.

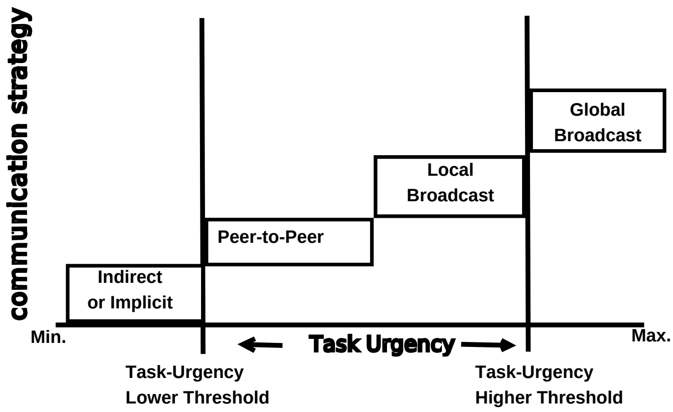


Figure 5.1: Self-regulated communication strategy

We characterize our communication model in terms of three fundamental issues: 1) message content (*what to communicate*) 2) communication frequency (*when to communicate*) and 3) target recipients (*with whom to communicate*) (**?**). In a typical MRS, message content can be categorized into two types: 1) state of each individual robot and 2) target task (goal) information (**?**). The latter can also be subdivided into two types: 1) an individual robot's target task information and 2) information of all available tasks found in the system. Regarding the first issue, our communication model is open. Robots can communicate with their peers with any kind of message. Our model addresses the last two issues very specifically. Robots communicate only when they meet their peers within a certain communication radius (r_{comm}). Although in case of an environment where robots move relatively faster the peer relationships can also be changed dynamically. But this can be manipulated by setting the signal frequency and robot to space density to somewhat reasonably higher value. In terms of target recipients, our model differs from a traditional publish/subscribe communication (PSC) model by introducing the concept of dynamic subscription. In a traditional PSC model, subscription of messages happens prior to the actual message transmission. In that case prior knowledge about the subjects of a system is necessary. But in our model this is not necessary as long as all robots uses a common addressing convention for naming their incoming signal channels. In this way, when a robot meets with another robot it can infer the address of this peer robot's channel name by using a shared rule. A robot is thus always listening to its own channel for receiving messages from its potential peers or message publishers. On the other side, upon recognizing a peer a robot sends a message to this particular peer. So here neither it is necessary to create any custom subject namespace (e.g., (**?**)) nor we need to hard-code information in each robot controller about the knowledge of their potential peers *a priori*. Subscription is done automatically based on their respective r_{comm}.

5.3 Implementation algorithm

Our local communication model has three major aspects: 1) local sensing of peers (and optionally tasks), 2) listening to peer signals and 3) emitting signals for peers. Here we present a typical implementation. Let N be the set of robots. At time step q, a robot i that can receive $h_{i,q}$ information by listening to its incoming channel L_i. Let M be the set of tasks. Each task j has an associated information H_j. It encodes the necessary properties of tasks, such as their locations, urgencies etc. Each task j also has a task perception radius r_{task} such that if a robot comes within this radius at time step q it can perceive current value of $H_{j,q}$. Let at time step q, robot i has its own task information $G_{i,q}$ that has been perceived and listened from its peers. Let r_{comm} be the communication radius of each robot. Let at time step q, $P^i_{p,q}$ be a set of peers of i that are within r^i_{comm}. Let $E^i_{p,q}$ be its active signal emission channels. Algorithm 1 implements our proposed dynamic P2P communication.

Algorithm 1: Locality based Dynamic P2P Communication

Initialization:
$id \leftarrow robotid$
$r_{comm} \leftarrow r_1$
$r_{task} \leftarrow r_2$
$pose[id] \leftarrow (0, 0, 0)$
$G[id],\ P[id],\ L[id],\ E[\,] \leftarrow 0$
Loop:
$pose[id] \leftarrow (x, y, \theta)$
if $pose[id] \in U(pose[k],\ r^k_{task}), (k = 0, 1, ., M - 1)$ **then**
    $G[id] \leftarrow G[id]\ \cup\ H_k$

end if
if $pose[id] \in V(pose[k],\ r^k_{comm}), (k = 0, 1, ., N-1, k \neq id)$ **then**
  $P[id] \leftarrow P[id]\ \cup\ k$
  $h_k \leftarrow W(E[k],\ L[id])$
  $G[id] \leftarrow G[id]\ \cup\ h_k$
end if
for all $k \in P[id], (k = 0, 1, ., N-1, k \neq id)$ **do**
  $W(E[id],\ L[k]) \leftarrow G[id]$
end for
$P[id] \leftarrow 0$
Loop again

From Algorithm 1, we see that a robot controller is initialized with its specific robot-id and default values of r_{comm} and r_{task}. We assumed that these values are same for all robots and for all tasks respectively. Initially a robot has no information about tasks. It has neither listened nor transmitted any information yet. Upon initialization, robot determines its current pose and evaluates a function $U(pose,\ r_{task})$ that helps it to perceive information of a nearby task. This is not strictly necessary as this information can be available from alternate sources. In second step, robot senses its nearby peers by evaluating $V(pose,\ r_{comm})$ and start filling the list of peers P by their id. The signal exchange with a peer is denoted by a communication function $W(emitter,\ listener)$. So by listening to a peer signal, it receives task information h and aggregates this h with its own task info G. In last step, robot emits its task information to its peers stored in P. Finally it erases all values of P and repeat this loop.

5.4 Implementation

We have developed a system where up to 40 E-puck robots (**?**) can operate together according to the generic rules of the AFM. As shown in Fig. 5.2 (right), our software system consists of a multi-robot tracking system, a task perception assistant (TPA) and robot controller clients (RCC). Here at first we have presented the design of our communication system. Then we have discussed about our specific MRS implementation.

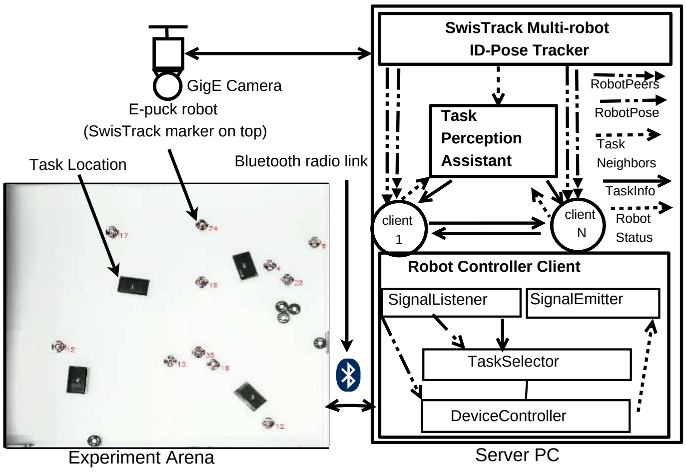


Figure 5.2: Hardware and software setup

5.4.1 Communication system under LPCM

As shown in Fig. 5.2, RCCs disseminate task information to each other by *TaskInfo* D-Bus signal. The contents of task information are physical locations of tasks and their urgencies. However as our robots are incapable of sensing task directly. So it relies on TPA for actual task location and urgency. When a robot i comes within r_{task} of a task j, SwisTrack reports it to TPA (by *TaskNeighbors* signal) and TPA then gives it current task information H_j to i (by *TaskInfo* signal). TPA has another interface for catching feedback signals from robots. The *RobotStatus* signal can be used to inform TPA about a robot's current task id, its device status and so on. TPA uses this information to update relevant part of task information such as, task-urgency. This up-to-date information is encoded in next TaskInfo signal.

5.4.2 MRS implementation

In order to track all robots real-time we have used SwisTrack (**?**), a state of the art open-source, multi-agent tracking system, with a16-megapixel overhead GigE camera. This set-up gives us the position, heading and id of each of the robots at a frequency of 1. The interaction of the hardware and software of our system is illustrated in Fig. 5.2.

For IPC, we have used D-Bus technology. We have developed an IPC component for SwisTrack (hereafter called as *SwisTrack D-Bus Server*) that can broadcast id and pose of all robots in real-time over our server's D-Bus interface.

Apart from SwisTrack, we have implemented two major software modules: *Task Perception Assistant (TPA)* and *Robot Controller Client (RCC)*. They are developed in Python with its state of the art *Multiprocessing* [1] module. This python module simplifies our need to manage data sharing and synchronization among different sub-processes. As shown in Fig. 5.2, RCC consists of four sub-processes. *SignalListener* and *SignalEmitter*, interface with SwisTrack D-Bus Server and TPA. *TaskSelector* implements AFM guidelines for task selection . *DeviceController* moves a robot to a target task. Bluetooth radio link is used as a communication medium between a RCC and a corresponding E-puck robot.

5.5 Experiment Design

In this section, we have described the design of parameters and observables of our experiments. These experiments are designed to validate AFM by testing the occurrence of convergent MRTA. Table 5.1 lists a set of essential parameters of our experiments.

5.5.1 Parameters

We intend to have a setup that is relatively complex, i.e., a high number of robots and tasks in a large area, but with a high probability of convergence. The following criteria shows our rationale behind our selected parameters.

- Robots should be capable of moving at a reasonably high speed (e.g., $\geq 4cm/s$) without interfering to each other very much.
- Tasks density should be as least one task per square meter.
- Host PC to robot communication should be as dedicated and stable as possible.
- No task should be left unattended completely for a long time (e.g., $\geq 300s$).

[1] http://docs.python.org/library/multiprocessing.html

Table 5.1: Experimental parameters

Parameter	Value
Total number of robots (N)	16
Total number of tasks (M)	4
Experiment area (A)	4 m^2
Intial task urgency (Φ_{INIT})	0.5
Task urgency increase rate ($\Delta\phi_{INC}$)	0.005
Task urgency decrease rate ($\Delta\phi_{DEC}$)	0.0025
Intial sensitization (K_{INIT})	0.1
Sensitization increase rate (Δk_{INC})	0.03
Sensitization decrease rate (Δk_{DEC})	0.01
A very small distance (δ)	0.000001
Task info update interval (ΔTS_u)	5s
Task info signal emission interval (ΔTS_e)	2.5s
Robot's task time-out interval (ΔRT_{to})	10s

When many Bluetooth devices talk to a single Bluetooth adapter, communication delays become very frequent due to the fact that each device gets a guaranteed turn to communicate (**?**). After some initial testing, we found a stable server configuration with 8 Bluetooth adapters, i.e., one Bluetooth adapter is used to communicate with two robots. This limits us to set the total number of robots to 16. Also we found that after about 35-40 minutes from the start of our experiments some of robots fail to get the access to their designated Bluetooth adapters. So we limit the length of our experiments to 40 minutes. We expect that this limitation would be removed by distributing Bluetooth adapters among multiple server PCs.

The diameter of the marker of our E-puck robot is 8cm. So, if we put 4 robots in an area of one square meter, this will give us a robot-occupied-space to free-space ratio of about 1:49 per square meter. We have found that this ratio is reasonable to allow the robots to move at a speed of 7.5 cm/sec without much interference to each other. We randomly placed four 18.5 cm x 11.5 cm rectangular boxes as a shop task in our experiment arena. They were about one meter apart from each other.

The initial values of task urgencies can be set to any value as long as they are same for all tasks. We choose a limit of 0 and 1, where 0 means no urgency and

1 means maximum urgency. Same applies to sensitisation as well, where 0 means no sensitisation and 1 means maximum sensitisation. We choose a default sensitization value of 0.1 for all tasks. Our rationale behind selecting task urgency and sensitisation change rates can be found in (**?**).

5.5.2 Observables

We have defined a set of observables to benchmark our implementation. They are briefly explained here.

Changes in task-urgencies ($\Delta\Phi$): In our experiments, urgency of each task in each step has been logged. From the above design of task urgency, we can see that if a task is not served by any robot for 100 consecutive steps (500s), urgency of that task will reach from 0.5 to its maximum value 1.0. On the other hand, if a task is served by only one robot for 200 consecutive steps (1000s) urgency of that task will be 0. But in real experiment, it is more likely that more than one robot will serve a task. So urgency of a task will decrease $\Delta\phi_{DEC}$ times number of working robots on that task (based on AFM guidelines (**?**)). The overall changes in task urgencies will show the convergence behaviour of our system.

Changes in robot sensitizations (ΔK): According AFM, as robots will do tasks they will specialize on each task by increasing or decreasing sensitizations (learning and forgetting). From our above design, we can see that if a robot starts doing a task with an initial sensitization of 0.1 and it repeatedly does it for 30 consecutive steps, sensitization will reach to 1. Thus by logging the sensitization data of each robots we will be able to comment on task specializations of robots.

Changes in robot motions (ΔU): As we might guess that initially the task urgencies will be relatively higher for all tasks so robots will need to do a lot of movements by switching from one task to another. But as the system approaches to converge overall robot motions will be decreased. In order to observe this phenomena we log the pose of robots in every time step.

D-Bus Signals emitted by Task server (S_f): In order to measure the communication load on our communication system we are also interested to log P2P TaskInfo D-Bus signals. Since the emission of signals happens asynchronously it is more likely that the overall communication load on the system will vary over time. This is contrary to the centralized communication where communication load is almost

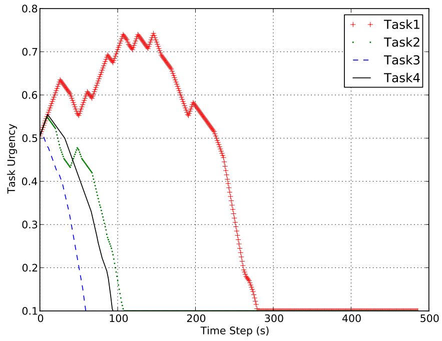


Figure 5.3: Task urgencies observed at TaskServer in local mode r_{comm}=0.5m

constant over time.

5.6 Results and discussions

In this section we have presented our experimental results. We ran those experiments for about 40 minutes and averaged them from three iterations. For comparison purposes, here we present some of the results of our baseline experiments in centralized communication mode. Details can be found in (**?**).

Fig. 5.3 shows the dynamic changes in task urgencies. In order to describe our system's dynamic behaviour holistically we analyse the changes in task urgencies over time. Let $\phi_{j,q}$ be the urgency of a task j at q^{th} step. In $(q+1)^{th}$ step, we can find the change of urgency of task j :

$$\delta\phi_{j,q+1} = (\phi_{j,q+1} - \phi_{j,q}) \tag{5.1}$$

So we can calculate the sum of changes in urgencies of all tasks at $(q+1)^{th}$ step:

$$\Delta\Phi_{j,q+1} = \sum_{j=1}^{M} \delta\phi_{j,q+1} \tag{5.2}$$

Fig. 5.4 plots this sum of changes of task urgencies by a dashed line. If we consider the absolute change over a window of time w in the following equation, we can describe the overall changes of our systems in both positive and negative directions.

$$\Delta\Phi_{jw,q+1} = \sum_{j=0}^{w-1} \mid \Delta\Phi_{q+j} \mid \tag{5.3}$$

Figure 5.4: Convergence of task urgencies in centralized mode

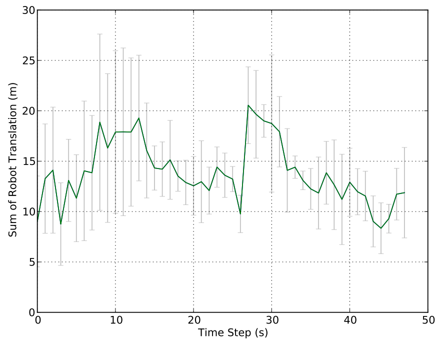


Figure 5.5: Sum of translations of all robots in centralized mode

In order to find convergence in MRTA we have calculated the sum of absolute changes in task urgencies over a window of 2 consecutive steps (100s). This is plotted in solid line in Fig. 5.4. Note that we scale down the time steps of this plot by aggregating the values of 10 consecutive steps (50s) of Fig. 5.3 into a single step value. From Fig. 5.4 we can see that initially the sum of changes of task urgencies are towards negative direction. This implies that tasks are being served by a high number of robots. When the task urgencies stabilize near zero the fluctuations in urgencies become minimum. Since robots chose tasks stochastically, there will always be a small changes in task urgencies. A potential convergence point is located by considering the persistence existence of the value of $\Delta\Phi_{jw,q+1}$ below a threshold 0.1. Tn Fig. 5.4 this convergence happens near step 23 or after 1150s from the beginning of our experiments. This implies that from this point of time and onwards, changes of our system's behaviour remain under a small threshold value.

Using this same criteria, we can find that in local communication experiment convergence happens after step 14 in case of r_{comm}=0.5m (Fig. 5.6) and after step 29 in case of r_{comm}=1m (Fig. 5.8).

We have aggregated the changes in translation motion of all robots over time. Let $u_{i,q}$ and $u_{i,q+1}$ be the translations of a robot i in two consecutive steps. If the difference between these two translations be δu_i, we can find the sum of changes

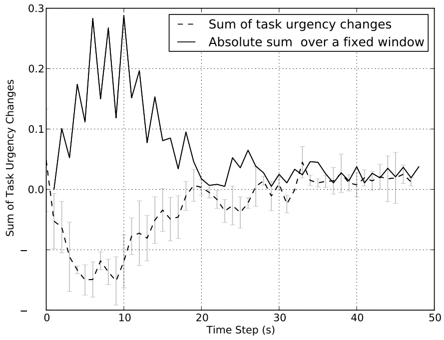


Figure 5.6: Convergence of task urgencies in local mode r_{comm}=0.5m

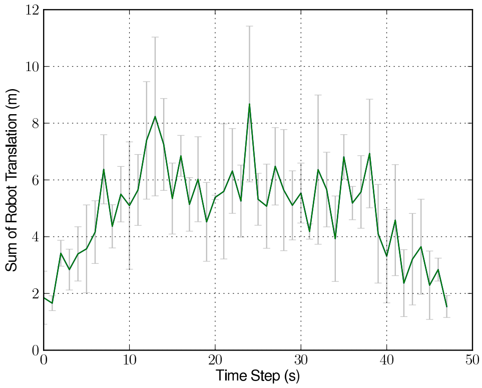


Figure 5.7: Sum of translations of all robots in local mode r_{comm}=0.5m

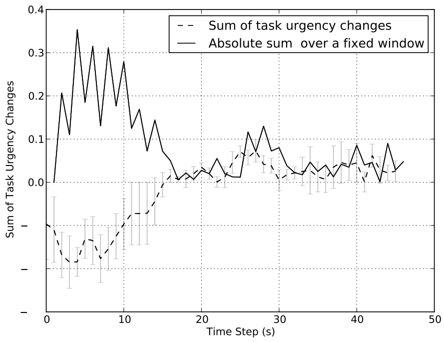


Figure 5.8: Convergence of task urgencies in local mode r_{comm}=1m

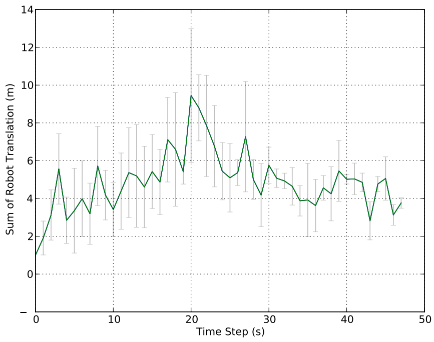


Figure 5.9: Sum of translations of all robots in local mode r_{comm}=1m

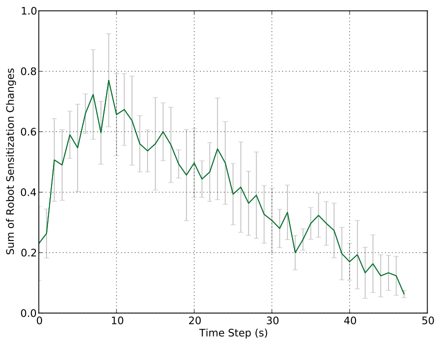


Figure 5.10: Changes in sensitizations in local mode r_{comm}=0.5m

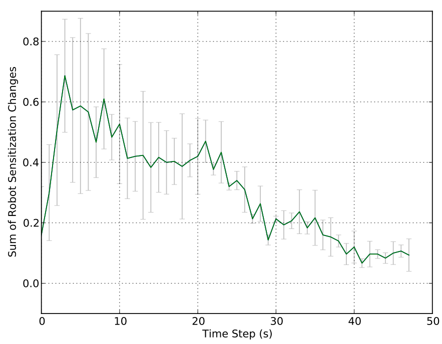


Figure 5.11: Changes in sensitizations in local mode r_{comm}=1m

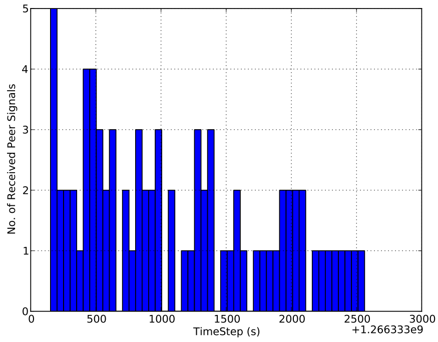


Figure 5.12: Peer signals caught by Robot12 in local mode r_{comm}=0.5m

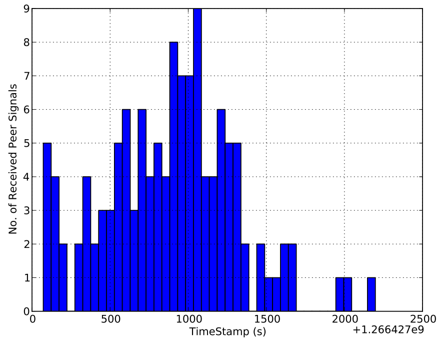


Figure 5.13: Peer signals caught by Robot12 in local mode r_{comm}=1m

of translations of all robots in $(q+1)^{th}$ step using the following equation.

$$\Delta U_{q+1} = \sum_{i=1}^{N} \delta u_{i,q+1} \tag{5.4}$$

The result from centralized communication experiment is plotted in Fig. 5.5. In this plot we can see that robot translations also vary over varying task requirements of tasks. But it fails to show a consistence behaviour like previous plots. The robot translation results from local mode experiments are plotted in Fig. 5.7 and Fig. 5.9. The reduction of robot translation is significant in both local communication experiments.

Similar to Eq. 5.2, we can calculate the absolute sum of changes in sensitizations by all robots in the following equation.

$$\Delta K_{j,q+1} = \sum_{j=1}^{M} \mid \Delta k_{j,q+1} \mid \tag{5.5}$$

This values of ΔK from local experiments are plotted in Fig. 5.10 and Fig. 5.11. They show that the overall rate of learning and forgetting decrease over time. It is a consequence of the gradually increased task specializations of robots.

As an example of P2P signal reception of a robot, Fig. 5.12 and Fig. 5.13 show the number of received signals by Robot12 in two local experiments. It states the relative difference of peers over time in two local cases. The overall P2P task information signals of both of these local modes are plotted in Fig. 5.14 and Fig. 5.15. Note that in centralized communication mode the task info signal frequency

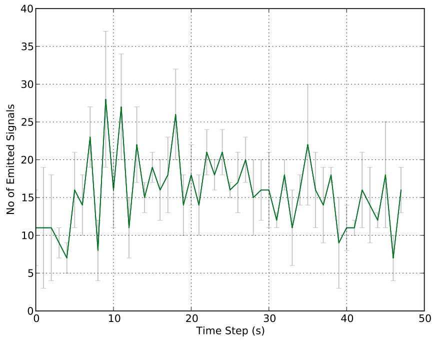


Figure 5.14: Frequency of P2P signalling in local mode r_{comm}=0.5m

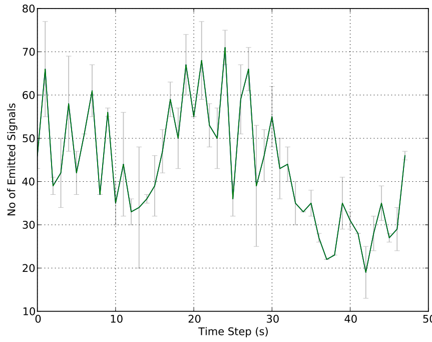


Figure 5.15: Frequency of P2P signalling in local mode r_{comm}=1m

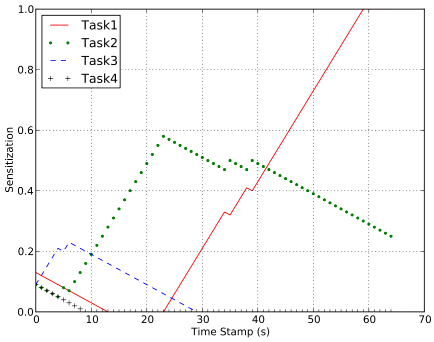


Figure 5.16: Task specialization of Robot12 in local mode r_{comm}=0.5m

Figure 5.17: Changes in translation of Robot12 in local mode r_{comm}=0.5m

was kept 20 signals per time step (**?**).

As an example of task specialization of a robot we plotted sensitization of Robot12 in Fig. 5.16. It shows that this robot has specialized in Task1. The continuous learning of Task1 happens from step 23 to step 59 where it has learned this task completely. This behaviour was found common in all robots with varying level of sensitizations. Hence we get the linear decrease of ΔK in sensitization plots. However, the changes in motion of this robot plotted in Fig. 5.17 was not stable due to the fact that robots frequently avoid dynamic obstacles and select random-walking. Krieger and Billeter presented MRTA in a team of 12 robots in ants' foraging scenario where a central server sent broadcast message containing energy level of the colony to all robots (**?**). They also applied a form of indirect communication in order to share information about discovered food sources among robots. Although this P2P communication did not happen in real time, it improved the robustness of peer recruitment when compared with centralized communication only. Agassounon and Martinoli compared three task-allocation algorithms using simulations and reported that algorithm that used information sharing among local peers became robust in worker allocation (**?**). However, environment condition should be known *a priori* to optimize some parameters. By modulating the transmission power of local on-board wireless device of E-puck robots, Cianci *et al.* enabled them to communicate in different local radii (**?**). In a self-organized decision making experiment, decision of 15 robots converged faster when local communication radius was relatively higher. Since the number of tasks was only two (left or right wall following), robots got global view of the system with increased communication radius. Thus they agreed quickly in one task.

5.7 Summary and conclusion

We presented a locality based dynamic P2P communication model that achieves similar or better self-regulated multi-robot task allocation (MRTA) than its centralized counterpart. Particularly the reduction in robot movement states that local model is preferable when we need to minimize robot energy usage. Our model assumes no prior knowledge of the environment and it also does not depend on the number of the robots. We presented an abstract algorithm that can be used with various P2P communication technologies. We reported our implementation of this algorithm by using D-Bus technology in Linux. Comparative results from local mode experiments shows us that robots relatively perform better when the radius of communication is such small that most of the robots only communicate with their closest peers. This is contrary to the findings from (**?**, **?**) where increased amount of information help to get desired task-allocation quickly. In our case, more information exchange with larger communication radius or with centralized communication increased task switching among robots. In case of local information exchange, robots have more chances to select local tasks by the virtue of self-regulating principles. Thus in local communication mode our system converged better and significantly reduced robot motions.

Our future work include performing more experiments with increased number of robots (up to 40) and tasks. Since our existing Bluetooth communication setup with one server fail to scale we are looking forward to distribute communication loads over a network cluster of tens of servers.

CHAPTER 6

Conclusions

6.1 Summary of contributions

6.2 Future work

Bibliography

Agah A & Bekey G 1995 *in* 'IEEE International Conference on Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century.' Vol. 4.

Agassounon W & Martinoli A 2002 *in* 'Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3' ACM p. 1097.

Agassounon W, Martinoli A & Easton K 2004 *Autonomous Robots* **17**(2), 163–192.

Ali M S 1995 *Scientfic Indications in the Holy Quran* Islamic Foundation Bangladesh.

Andriani P & Passiante G 2004 *Complexity theory and the management of networks : proceedings of the Workshop on Organisational Networks as Distributed Systems of Knowledge, University of Lecce, Italy 2001* Imperial College Press London.

Arai T, Ogata H & Suzuki T 1989 *Intelligent Robots and Systems' 89.(IROS'89)'The Autonomous Mobile Robots and Its Applications'., Proceedings. IEEE/RSJ International Workshop on* pp. 479–485.

Arcaute E, Christensen K, Sendova-Franks A, Dahl T, Espinosa A & Jensen H J 2008 *in* 'Ecol. Complex'.

Arkin R C 1990 *ROBOT. AUTONOMOUS SYST.* **6**, 105–122.

Arkin R C 1998 *Behavior-based robotics* Cambridge, Mass. : MIT Press, c1998. Includes bibliographical references (p. [445]-476) and indexes.

Arkin R & Diaz J 2002 *Advanced Motion Control, 2002. 7th International Workshop on* pp. 455–461.

Asada M, Kitano H, Noda I & Veloso M 1999 *Artificial Intelligence* **110**, 193–214.

Asama H, Matsumoto A & Ishida Y 1989 *Intelligent Robots and Systems' 89.(IROS'89)'The Autonomous Mobile Robots and Its Applications'., Proceedings. IEEE/RSJ International Workshop on* pp. 283–290.

Balch T 2002 *Robot teams: From diversity to polymorphism* pp. 23–35.

Balch T 2005 *in* E Sahin & W Spears, eds, 'Swarm Robotics Workshop: State-of-the-art Survey' number 3342 *in* 'Lecture Notes in Computer Science' Springer-Verlag Berlin Heidelberg pp. 21–30.

Balch T & Arkin R C 1994 *Autonomous Robots* **1**, 27–52.

Balch T & Arkin R C 1998 *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION* **14**, 1.

Baumeister R F & Vohs K D 2007 *Handbook of Self-Regulation: Research, Theory, and Applications* The Guilford Press.

Beckers R, Holland O E & Deneubourg J L 1994 *Artificial Life IV* **181**, 189.

Beer S 1981 *Brain of the firm* J. Wiley New York.

Belta C & Kumar V 2004 *IEEE Transactions on Robotics* **20**(5), 865–875.

Beni G 1988 *Intelligent Control, 1988. Proceedings., IEEE International Symposium on* pp. 57–62.

Beni G 2005 *Swarm Robotics: SAB 2004 International Workshop, Santa Monica, CA, USA, July 17, 2004: Revised Selected Papers* .

Beni G & Liang P 1996 *IEEE Transactions on Robotics and Automation* **12**, 485 – 490.

Bonabeau E, Dorigo M & Theraulaz G 1999 *Swarm intelligence: from natural to artificial systems* Oxford University Press.

Botelho S, Alami R & LAAS-CNRS T 1999 *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on* **2**.

Braitenberg V 1984 *Vehicles, Experiments in Synthetic Psychology* Bradford Book.

Brooks R 1986 *IEEE J. ROBOTICS AUTOM.* **2**, 14–23.

Burgard W, Moors M, Fox D, Simmons R & Thrun S 2000 *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on* **1**.

Bussmann S, Jennings N & Wooldridge M 2004 *Multiagent Systems for Manufacturing Control: A Design Methodology* Springer.

Camazine S, Franks N, Sneyd J, Bonabeau E, Deneubourg J & Theraula G 2001 *Self-organization in biological systems* Princeton, N.J. : Princeton University Press, c2001. What is self-organization? – How self-organization works – Characteristics of self-organizing systems – Alternatives to self-organization –.

Cao Y, Fukunaga A & Kahng A 1997 *Autonomous Robots* **4**(1), 7–27.

Çelikkanat H, Turgut A & Şahin E n.d. *Distributed Autonomous Robotic Systems 8* pp. 215–225.

Chaimowicz L, Campos M F M & Kumar V 2002 *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on* **1**.

Cianci C M, Raemy X, Pugh J & Martinoli A 2007 *Swarm Robotics: Second Sab 2006 International Workshop, Rome, Italy, September 30-October 1, 2006 Revised Selected Papers* .

Cianci, C. R X P J & Martinoli A 2004 Springer chapter Communication in a swarm of miniature robots: The e-puck as an educational tool for swarm robotics Swarm robotics.

Davis R & Smith R 1988 *Distributed Artificial Intelligence* pp. 333–356.

Dias M B, Zlot R M, Kalra N & Stentz A 2006 *Proceedings of the IEEE* **94**, 1257–1270.

Doty K & Van Aken R 1993 *in* 'Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on' pp. 778–782.

Dudek G, Jenkin M & Milios E 2002 *Robot Teams: From Diversity to Polymorphism* pp. 3–22.

Dudek G, Jenkin M, Milios E & Wilkes D 1996 *Autonomous Robots* **3**, 375–397.

Dunbar W & Klein B 2002 *CIM Bulletin* **95**(1057).

Farinelli A, Iocchi L & Nardi D 2004 *IEEE Transactions on Systems, Man, and Cybernetics, Part B* **34**(5), 2015–2028.

Feldman D 1997 *Department of Physics, University of California, July* .

Franklin D, Kahng A & Lewis M 1995 *Proceedings of SPIE* **2496**, 698.

Frings H 1977 *Animal Communication* University of Oklahoma.

Fukuda T & Nakagawa S 1987 *in* 'Proceedings of IECON' p. pages 588595.

Garnier S, Gautrais J & Theraulaz G 2007 *Swarm Intelligence* **1**(1), 3–31.

Gat E, Bonnasso R, Murphy R & Press A 1997 *Artificial intelligence and mobile robots* .

Gazi V & Fidan B 2006 **4433**, 71–102.

Gerkey B & Mataric M 2003 *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on* **3**.

Gerkey B P & Mataric M J 2002 *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION* **18**.

Gerkey B P & Mataric M J 2004 *The International Journal of Robotics Research* **23**, 939.

Hamann H & Worn H 2006 *in* E Sahin, W. M Spears & A. F. T Winfield, eds, ‘Second International Workshop on Swarm Robotics at SAB 2006’ Vol. 4433 Springer Verlag Berlin, Germany pp. 43–55.

Holldobler B & Wilson E 1990 *The ants* Belknap Press.

Huntsberger T L, Trebi-Ollennu A, Aghazarian H, Schenker P S, Pirjanian P & Nayar H D 2004 *Autonomous Robots* **17**, 79–92.

Jeanne R 1999 *Information processing in social insects* pp. 3–30.

Jones C & Mataric M 2003 *in* ‘2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings’ Vol. 2.

Kalra N & Martinoli A n.d. *Distributed Autonomous Robotic Systems 7* pp. 91–101.

Kogut B 2000 *Strategic Management Journal* **21**(3), 405–425.

Konolige K, Fox D, Limketkai B, Ko J & Stewart B 2003 *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on* **1**.

Kppers G, Kppers G, Krohn W & Nowotny H 1990 *Selforganization: portrait of a scientific revolution* Dordrecht ; Kluwer Academic Publishers, c1990. Includes bibliographical references and indexes.

Krieger M & Billeter J 2000 *Robotics and Autonomous Systems* **30**(1-2), 65–84.

Kube C R 1997 *Autonomous Robots* **4**, 53–72.

Kube C R & Zhang H 1993 *Adaptive Behavior* **2**, 189.

Kuniyoshi Y, Kita N, Rougeaux S, Sakane S, Ishii M & Kakikua M 1994 *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on* pp. 767–774.

Labella T 2007 Division of labour in groups of robots PhD thesis Universite Libre de Bruxelles.

Lazinica A & Katalinic B 2007 *International Journal of Automation and Control* **1**(1), 16–27.

Lerman K, Jones C, Galstyan A & Mataric M J 2006 *The International Journal of Robotics Research* **25**, 225.

Lerman K, Martinoli A & Galstyan A 2005 (3342), 143–152.

Liu W, Winfield A F T, Sa J, Chen J & Dou L 2007 *Adaptive Behavior* **15**(3), 289–305.

Lochmatter T, Raemy X & Martinoli A 2007 *Bulletin of the Swiss Society for Automatic Control* (46), 11–14.

Lochmatter T, Roduit P, Cianci C, Correll N, Jacot J & Martinoli A 2008 *in* 'IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008' pp. 4004–4010.

Love R 2005 *Linux Journal* **2005**(130), 3.

Martinoli A 1999 Swarm intelligence in autonomous collective robotics: From tools to the analysis and synthesis of distributed control strategies PhD thesis.

Martinoli A, Lausanne S & Mondada F 1996 *Experimental Robotics IV: The 4th International Symposium, Stanford, California, June 30-July 2, 1995* .

Mataric M 2007 *The Robotics Primer* Mit Press.

Mataric M J 1994 Interaction and Intelligent Behavior PhD thesis Massachusetts institute of Technology.

Mataric M J 1995 *Robotics and autonomous systems* **16**, 321–331.

Mataric M J 1998 *Journal of Experimental & Theoretical Artificial Intelligence* **10**, 357–369.

Mataric M J 2001 *Cognitive Systems Research* **2**, 81–93.

Mataric M J, Nilsson M & Simsarian K T 1995 *in* 'IEEE/RSJ International Conference on Robotics and Intelligent Systems' Pittsburgh, PA.

McFarland D 1994 *From Animals to Animats* **3**, 440–444.

McGregor P & Peake T 2000 *Acta Ethologica* **2**(2), 71–81.

Melhuish C, Holland O & Hoddell S 1998 *From Animals to Animats 5: Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior* pp. 465–470.

Murphy R 2000 *Introduction to AI Robotics* MIT Press.

Ortiz C, Vincent R & Morisset B 2005 *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems* pp. 860–867.

Parker L 1994 *Proceedings of the ASCE Specialty Conference on Robotics for Challenging Environments, Albuquerque, NM, February* .

Parker L 1995 L-alliance: a mechanism for adaptive action selection in heterogeneous multi-robot teams Technical report ORNL/TM–13000, Oak Ridge National Lab., TN (United States).

Parker L E 1998 *Robotics and Automation, IEEE Transactions on* **14**, 220–240.

Parker L E 2008 *Journal of Physical Agents, special issue on multi-robot systems* **vol. 2**(no. 2), 5–14.

Parker L E & Tang F 2006 *Proceedings of the IEEE* **94**, 1289–1305.

Payton D, Estkowski R & Howard M 2005 *in* E Sahin & W Spears, eds, 'Swarm Robotics Workshop: State-of-the-art Survey' number 3342 Springer-Verlag Berlin Heidelberg pp. 45–57.

Pennington H, Carlsson A & Larsson A 2010 'D-bus specification'.

Pratt S, Mallon E, Sumpter D & Franks N 2002 *Behavioral Ecology and Sociobiology* **52**(2), 117–127.

Premvuti S & Yuta S 1990 *Intelligent Robots and Systems' 90.'Towards a New Frontier of Applications', Proceedings. IROS'90. IEEE International Workshop on* pp. 59–63.

Reynolds C W 1987 *Computer Graphics* **21**.

Rothermich J, Ecemis I & Gaudiano P 2005 *in* E Sahin & W Spears, eds, 'From Swarm Intelligence to Swarm Robotics' number 3342 *in* 'Lecture Notes in Computer Science' Springer-Verlag Berlin Heidelberg pp. 58–69.

Rutishauser S, Correll N & Martinoli A 2009 *Robotics and Autonomous Systems* **57**(5), 517–525.

Sahin E & Spears W 2005 *Swarm Robotics: SAB 2004 International Workshop Santa Monica, CS, July 2004 Revised Selected Papers* Berlin, Germany: Springer.

Sahin E, William M S & Winfield A F T 2007 *Swarm Robotics, Second International Workshop, SAB 2006, Rome, Italy, September 30-October 1, 2006, Revised Selected Papers.* International Conference on Simulation of Adaptive Behavior rome, italy edn Springer Berlin.

Sarker M 2008 ‘Emergent self-regulation in social robotic systems’ PhD Transfer Report, University of Wales, Newport, 2008.

Sarker M & Dahl T n.d.

Sayer A & Walker R 1992 *The new social economy : reworking the division of labor* Blackwell Oxford. 19920311.

Schild K & Bussmann S 2007.

Schmickl T, Moslinger C & Crailsheim K 2006 *in* E Sahin, W. M Spears & A. F. T Winfield, eds, ‘Second International Workshop on Swarm Robotics at SAB 2006’ Vol. 4433 Springer Verlag Berlin, Germany pp. 144–157.

Sendova-Franks A B & Franks N R 1999 *Philosophical Transactions of the Royal Society of London B* **354**, 1395–1405.

Shen W, Hao Q, Yoon H & Norrie D 2006 *Advanced Engineering Informatics* **20**(4), 415–431.

Shen W, Norrie D H & Barthes J P 2001 *Multi-agent systems for concurrent intelligent design and manufacturing* Taylor & Francis London.

Simmons R, Smith T, Dias M B, Goldberg D, Hershberger D, Stentz A & Zlot R M 2002 *Multi-Robot Systems: From Swarms to Intelligent Automata, Proceedings from the 2002 NRL Workshop on Multi-Robot Systems* .

Slater P J B 1986 *Animal behaviour* Leisure Circle Wembley.

Spears W & Gordon D 1999 *IEEE International Conference on Information, Intelligence, and Systems* pp. 281–288.

Spears W M, Hamann J C, Maxim P M, Kunkel T, Heil R, Zarzhitsky D, Spears D F & Karlsson C 2006 *in* E Sahin, W. M Spears & A. F. T Winfield, eds, 'Second International Workshop on Swarm Robotics at SAB 2006' Vol. 4433 Springer Verlag Berlin, Germany pp. 129–143.

Stone P & Veloso M 1999 *Artificial Intelligence* **110**, 241–273.

Thurn S, Burgard W & Fox D 2000 *Proc. of the IEEE Int. Conf. on Robotics and Automation* .

Ueda K 2006 *DET, Setubal, Portugal* .

Von Frisch K 1967 *The dance language and orientation of bees* Belknap Press of Harvard University Press.

Wang P K C 1989 *Intelligent Robots and Systems' 89.(IROS'89)'The Autonomous Mobile Robots and Its Applications'., Proceedings. IEEE/RSJ International Workshop on* pp. 486–493.

Werger B B & Mataric M J 2001 *Distributed Autonomous Robotic Systems* **4**, 347356.

West, Richard T L H 2003 *Introducing communication theory : analysis and application* McGraw-Hill.

Wittenburg G 2005 'Desktop ipc'.

Wurman P, D'Andrea R & Mountz M 2008 *AI Magazine* **29**(1), 9–20.

Yahya H 2000 *The miracle in the ants* http://www.harunyahya.com.
URL: *http://www.harunyahya.com*

Yang E & Gu D 2004 *University of Essex Technical Report CSM-404, Department of Computer Science* .

Yoshida E & Arai T 2000 *IEICE Transactions on Communications* **83**(5), 1048–1059.

Zlot R, Stentz A, Dias M & Thayer S 2002 *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on* **3**.