Chapter 6

Distributed TDMA Based Task Allocation Framework for SRS and MRS

6.1 Introduction

The problem of distributing and scheduling a set of tasks among a group of robots to achieve the system goals, taking into account the operational constraints is described as Multi-Robot Task Allocation (MRTA) problem. 'Task allocation' is done to decide "which robot executes, which task, at what time". Time synchronization and localization are pre-requisites for the task allocation process.

The first reported and the widely accepted work on formal classification of MRTA problem was by Gerkey and Matarić in 2004 [31]. The proposed taxonomy classifies MRTA problem along three axes as follows.

- Single-Task robots (ST) Versus Multi-Task robots (MT)
- Single-Robot tasks (SR) Versus Multi-Robot tasks (MR)
- Instantaneous Assignment (IA) Versus Time-extended Assignment (TA)

ST/MT classification is based on the number of tasks that can be assigned to a robot during each task assignment cycle. ST robots can be assigned (or can complete) only a single task whereas MT robots can be assigned multiple tasks during each task assignment cycle. SR/MR classification is based on the number of robots required to complete a task. SR tasks can be completed by a single robot whereas MR tasks require multiple robots to complete the task. IA consider only instantaneous conditions of system such as the availability of the robot, resources or characteristics of the robots required for task allocation, etc., whereas TA performs task allocation by taking into account, the current and future requirements of the system. Thus, Gerkey and Matarić classify the MRTA problem into 8 categories- ST-SR-IA, ST-SR-TA, MT-SR-IA, MT-SR-TA, ST-MR-IA, ST-MR-TA, MT-MR-IA and MT-MR-TA. The ST-SR-IA is the simplest task allocation problem to solve, in which each robot is assigned a single task and each task is attended by only one robot. MT-MR-TA is the most complex among all MRTA problems, in which each robot is assigned multiple tasks considering the current and future requirements and each task requires multiple co-operating robots for its completion.

A new taxonomy for MRTA classification, iTax was proposed in 2013 by G.Korsah et al. which is an extension to the taxonomy proposed by Gerkey and Matarić [79]. iTax is based on the concept of utility functions, which represents the ability of the robots to measure their interest in specific tasks. In MRTA systems, the utility functions indicate how the robot skills can match the tasks requirements. The iTax taxonomy takes into account the inter-related utilities and constraints among tasks. Based on the degree of interdependency, four possible classifications were proposed as follows. 1) no dependencies: the utility of robot-task does not depend on task of any other robot or its own schedule, 2) in-schedule dependencies: the utility of robot–task depends on its own schedule, 3) cross-schedule dependencies: the utility of robot–task depends on its own schedule and the schedules of other robots, 4) complex dependencies: the utility of robot–task depends on the schedules of other robots and their decomposition.

From the detailed literature review provided in Section 2.3, research gaps were identified. Based on the analysis it is concluded that a framework for task allocation which can be used to solve any of the eight types of MRTA problem needs to be developed. The swarm robotic systems are adaptive and decentralized. The information about the whole system will not be available with the robots and hence, the optimization based task allocations schemes are not suitable for SRS. The framework should be reactive to changes in environmental conditions or for scaling population of

robots. The task allocation scheme should be independent of path planning or navigation of robots, so that the scheme can be used for task allocation of land/air-borne robots or for co-operating networks in land and air. Most importantly the framework should be computationally efficient with lesser communication overhead. A novel, task allocation framework DTTA- a distributed, TDMA based task allocation framework for swarm of robots which can be utilized to solve any of the 8 different types of task allocation problem identified by Gerkey and Matarić is presented in the chapter.

6.2 DTTA Task Allocation Framework

A novel, distributed, task allocation framework which can be utilized to solve 8 different types of MRTA problem identified by Gerkey and Matarić is presented in this section. The framework can be utilized for different types of MRTA problems i.e. SR-ST-IA, SR-MT-IA, MR-ST-IA, MR-MT-IA and their time-extended assignment (TA) counterparts. To this account, the MRTA problem can be defined as follows. Given a set of available robots- $R = \{R_1, R_2, ...R_n\}$ and available tasks- $T = \{T_1, T_2, ...T_n\}$, completion of each task may require multiple robots and each robot may be assigned to multiple tasks, the framework should provide a feasible allocation of tasks such that all the tasks are completed as per the performance requirements of the system. The proposed task allocation framework is henceforth referred as DTTA (Distributed TDMA based Task Allocation) framework. Important characteristics of the DTTA task allocation framework are as follows.

- The task allocation framework can solve any of the 8 different types of MRTA problem identified by Gerkey and Matarić. DTTA is suitable for systems utilizing swarm of robots in which the system may require different (8 different types identified by Gerkey and Matarić) type of task allocations depending on the emerging requirement.
- The task allocation scheme is reactive, which considers task allocation problem as a dynamic decision problem, that is reconsidered over time rather than as a static assignment problem.
- The framework is distributed in nature. In addition to this, the number of computations required and the communication messages required for task allocation do not proportionally

increase with respect to the number of robots in the system and hence the framework is scalable.

- Unlike most of the solutions suggested for MRTA problems in literature [80, 118], DTTA framework isolates path planning and navigation of robot from the task allocation problem and hence may be utilized for any kind of robot in land or for co-operative systems comprising of land robots and air-borne robots (drones).
- DTTA framework can be effectively utilized for the task allocation in clustered or non-clustered robot deployment scenarios.

Following assumptions have been made in the design of DTTA framework:

1. The robots are equipped with wireless communication devices and members of the swarm can communicate via the wireless network. The type of wireless network (Wi-Fi, Wireless LAN, ZigBee, etc.) can vary according to the requirements of the application.
2. Robots in the system are time synchronized such that the maximum synchronization error among members of the swarm is bounded. The "Swarm-Sync" time synchronization framework presented in Chapter 3, can be utilized for synchronization.
3. Each robot should be capable of localizing or determining its position in the area of interest. The localization scheme proposed in Chapter 4 and 5 may be utilized for localization in indoor environments.
4. A central robot/node in the system referred as 'auctioneer' has the information about the tasks required to be serviced collectively by the robots in the system within a given period of time.

The task allocation strategy employed in DTTA is inspired by the market based task allocation schemes which are reactive when compared to the optimization based approaches [119, 120]. In traditional market/auction based task allocation schemes, the robots competitively bid for task(s) in response to the task announcement by the auctioneer. For example, consider an SR-ST problem in which each robot can be assigned only a single task. If a system has 'm' robots and if the auctioneer announces 'n' tasks simultaneously, each robot must inform other robots about its own utility (bid)

of the tasks. Based on how the winner of bidding is selected, the market based approaches can be centralized (winner selected by auctioneer) or distributed (winner selected by individual robot) [82, 83]. In case of distributed market based approaches, for e.g. in Broadcast of Local Eligibility (BLE) task allocation scheme, the number of communication messages transmitted for bidding in the system is $O(m)$, asssuming that the bid for all tasks are transmitted in the same message by a robot. If the task announcement and bidding of each task is performed, one task at a time, then for 'n' tasks, the number of bidding messages required is $O(mn)$. BLE algorithm requires each robot in the system to compare, for each task, its own utility to that of every other robot, leading to a computational complexity of $O(mn)$ per robot. [82]. The major challenge in implementing competitive bidding is the containment of bidding messages in the network, as the size (the number of robots) of the network increases. DTTA replaces competitive bidding with cooperative, self-task assignment in which individual members of the swarm assign tasks to itself based on the task announcement from the auctioneer without the bidding operation.
The different communication message types defined for task allocation in DTTA are -

1. **Task Announcement Message:** Task announcement messages are transmitted to announce the list of tasks which are to be completed by the robots and additional information pertaining to the tasks to be completed. The task announcement messages are transmitted by the auctioneer in non-clustered robot deployments. For clustered deployments, cluster heads can also transmit task announcement messages.

2. **Extended Task Announcement Message:** These messages are transmitted by a robot, when a robot identifies that it will not be able to complete the accepted task according to the performance requirements, unless the cooperation from other robots in system is obtained. The robot can request for transfer of task to other robot(s) or cooperation from other robots to complete the task, through extended task announcement messages.

3. **Task Acceptance Message:** These messages are transmitted by a robot in response to a task announcement or extended task announcement message, when it selects a task(s) for itself. The robots announce the accepted task(s) via task acceptance messages.

4. **Extended Task Acceptance Message:** These messages are transmitted by a robot which is already assigned a task, in response to an extended task announcement message. The

extended task acceptance messages are transmitted when the robot accepts additional task(s) to be serviced.

During task allocation, each robot in the system assumes one of the three possible roles:

1. **Relay Node:** The relay node/robot monitors the progress of other robots in the system towards the completion of tasks and relays (forward) the wireless messages between the robots which are not in coverage range.
2. **Cluster Head:** In a system with large number of robots, the robots may be divided into groups or clusters. Each cluster will be controlled and managed by a cluster head robot.
3. **Worker Node:** The robot which performs the announced tasks. Cluster head and relay nodes manages and monitors the worker nodes.

The various steps involved in task allocation for clustered and non-clustered deployments are mentioned in 6.2.1 and 6.2.2 respectively.

6.2.1 DTTA for Non-clustered Deployments

The steps involved in the task allocation are represented as a state flow diagram in Figure 6.1. The different steps involved in task allocation in non-clustered deployments are as follows:

1. **Task Announcement:** The central auctioneer broadcasts the task announcement message which marks the beginning of the 'hyperperiod'. Each robot which receives the task announcement message may assign task(s) for itself. In this thesis, 'hyperperiod' is defined as the duration of time for which the task assignment/allocation is performed by robots. The task announcement messages are of the format - $SourceID : Type : TTL : CC : CI_i : RC : RI_i : TI_n : MTR$ (Refer Table 6.1). The task announcement message incorporates the ID of the robot which performs the task announcement ($SourceID$), type or purpose of message ($Type$), information regarding the tasks to be serviced such as- TTL (deadline of the tasks), Cluster control (CC) field indicating if clustering is required or not, whether clusters should be of equal size, number of clusters required, Cluster Information fields (CI)

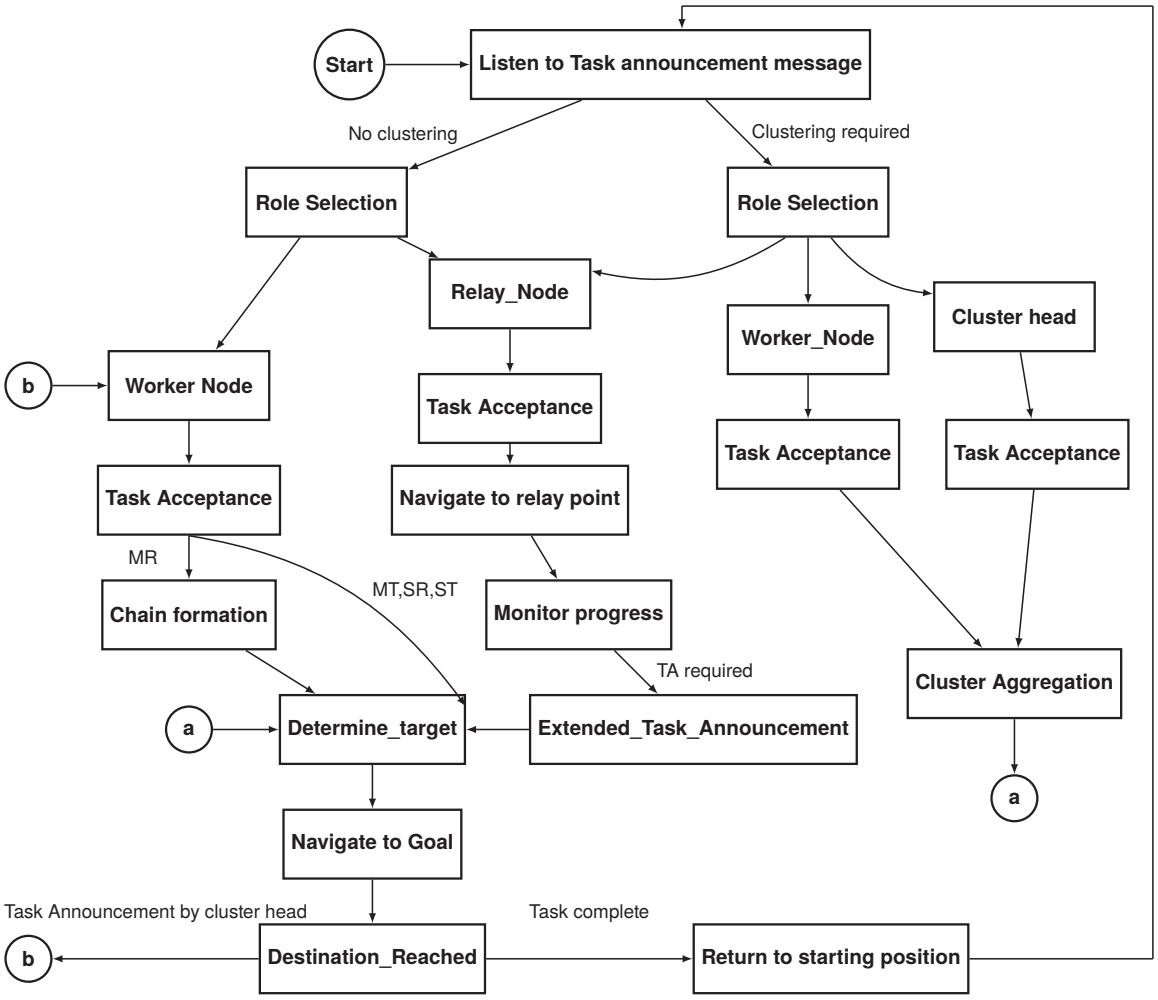

Figure 6.1. DTTA- State Flow diagram

indicating the characteristics of each cluster to be formed, Relay node control information (RC) indicating whether relay nodes are required for the task completion, Relay Information field (RI) including information about each relay node, Task information field (TI) including the information about the each task- which includes Task ID, number of robots required to complete the task, target location of task, energy budget and other characteristics of the robot required to service the task.

2. **Role Selection and Task Acceptance:** After the broadcast of task announcement message by the auctioneer, the hyperperiod starts and the robots participate in role selection process as indicated in Figure 6.1. The robots which receive the task announcement message will determine the number of relay nodes, cluster head nodes and worker nodes required for task completion from the task announcement message. The robots then assume role of either

relay node, worker node or cluster head node. The preference order of the roles is in the following order - relay node (highest) and worker node (lowest) respectively. If all the roles in higher preference are already filled by other robots, then the robots will assume the next available role. In case of small-scale robot deployment, clustering is not required and hence the available roles will be either as relay or worker node.

Each robot in the system is assigned a Time Division Multiple Access (TDMA) slot within the hyperperiod based on the ID number of the robot. Each robot utilizes its TDMA slot to assign role and task for itself and to announce the same via task acceptance message. The TDMA slots for a particular hyperperiod are self-assigned by robots based on the priority of robots. The priority assignment is based on the ID number of the robot. The robot with the highest priority is assigned the first TDMA slot. Each robot maintains a 'Task queue' to store the task acceptance messages from the other robots. To assign the task to itself, the robots inspect its task queue for available task acceptance messages from other robots during the current hyperperiod. The auctioneer announces the characteristics of the robot, including the energy budget required for completing a particular task in the 'TI' field. Robots available

Table 6.1. List of Abbreviations

No	Symbol	Item	Comments
1	$SourceID$	ID of the robot originating the message	
2	$Type$	Type of message being transmitted	Different type of messages includes- a) Task announcement message, b) Extended task announcement messages, c) Task acceptance messages, d) Extended task acceptance messages.
3	TTL	Time to Live	Dead line of tasks. Requires multiple fields if TTL is different for each task
4	CC	Cluster Control information	This field includes following information- Indicates if clustering is required ($C = 1$), Clusters are of equal size ($E = 1$) and number of clusters (NC) required.
5	CI_i	Cluster Information	Number of CI fields $i = 0, ..NC$. Includes required features of cluster members, number of robots/cluster (RC).
6	W	No of tasks announced/ task announcement message	
7	RC	Relay node Control information	Indicates if relay nodes are required, and the relay node count (RN)
8	RI_i	Relay node Information	Relay node location and characteristics; Number of RI fields $i = 0, ..RN$
9	TI_n	Task Information	Number of TI fields, n=1..W Information about task (being announced/accepted) such as Task ID, number of robots required/task (CH), location of the task, characteristics of the robot, Energy budget of the task (EB), Extended Task Assignment allowed ($TA = 1$)
10	MTR	Maximum number of tasks a robot can service/task announcement message	
11	L_x, L_y	Current 2D position of the robot	
12	Count	Count of tasks which the robot will service for a given task announcement message	
13	Role	Role selected by the robot	Possible roles are worker, relay and cluster head.

for the task, which comply with the requirements will assign a task to itself during the task assignment process. The robots obey the following rules during the assignment process.

- The robot of a certain priority will attempt to service the task with lowest ID first.
- If Task with lowest ID (e.g. Task1) is already accepted by higher priority robot and the 'count' of robots required to service the task is met then the robot attempts to accept the next task (e.g. Task2)
- If the count of robots required to service a task is 'greater than unity' then the robot services the task along with the other robots servicing the task.
- If a robot accepts a single robot (SR) task, then the robot can accept to service 'MTR' (Refer Table 6.1) number of single robot tasks if the targets are close to each other.

Each robot broadcasts the count and the ID of the tasks which it will service via task acceptance messages. The task acceptance messages (Refer Table 6.1) are broadcast messages of the format - $SourceID : Type : Role : CI_i : RI_i : Count : TI_n : L_x : L_y$. The number of messages required for task assignment/allocation in DTTA is equal to the number of robots required to service the tasks in the task announcement message.

3. **Priority Rotation:** After the completion of hyperperiod, or after certain number of hyperperiods based on the application, the priority of robots is rotated in order to ensure a fair distribution of load among all the robots, thus extending the lifetime of the system. It has to be noted that message transmissions are not required for TDMA slot assignment or rotation of priority, as the same is based on the unique ID of robots.

4. **Task Servicing:** Once the hyperperiod is complete, the robots start servicing the tasks. In SRS, the swarm of robots which will service the same task (robots in MR-ST mission) can form a chain (if the robot structure supports chaining) and move towards the destination. Chaining reduces the communication and computational overheads in path planning.

5. **Task Monitoring and Extended Task Announcements:** If any of the tasks auctioned by the auctioneer is not accepted by robots in the system, then the announcement of the specific task is repeated in subsequent task announcement messages. Once the robots start servicing the tasks, the worker robots monitor their progress of work based on the deadline assigned to them for completing the task. The relay nodes which has the task acceptance messages from

the worker nodes will also monitor the progress of the worker robots. If the worker/relay robot estimate that worker node will fail to meet the deadline, the robot broadcasts extended task announcement messages. The robots which are free can respond to the extended task announcement message via the task acceptance messages.

Robots which are already on a mission may also assist the robot transmitting the extended task announcement if the deadline of the task which is already accepted by the robot can be met, even if the robot services the task announced via the extended task announcement message. If the robot already on a mission chose to serve the extended task announcement message it will broadcast its acceptance of task via the extended task acceptance message. For eg: Robot 'A' which is not able to cross an obstacle in its path on its own, broadcasts the extended task announcement message when it estimates that it may not be able to meet the deadline of the assigned task. Through extended task announcement message, Robot 'A' requests for assistance from two robots to cross the hurdle. Robots 'B' and 'C' already on a mission and navigating towards their destination, if they receive extended task annoncement messages may respond to it via extended task acceptance messages. The robots 'B' and 'C' can assist robot 'A' to cross the hurdle by forming suitable formations following which 'B' and 'C' may continue performing their own tasks. Robot assigned to an MR task are also allowed to respond to extended task announcement messages if this will not affect the performance of tasks already accepted by them. If the robots which are in proximity to the robot requesting for assistance, do not respond within a particular time interval (based on application), then relay nodes will further transmit the extended task announcement message. The extended task announcement messages are broadcast via relay nodes with the '$SourceID$' and location of the original node which transmitted the extended task announcement message. Task acceptance and extended task acceptance messages are of the same format and can be distinguished by the robots based on their 'Type' field as indicated in Table 6.1.

6.2.2 DTTA for Clustered Deployments

DTTA can be extended for large scale robotic networks which require clustering as follows. The steps mentioned in the Section 6.2.1 are followed with the modifications as mentioned below.

The task announcement messages are of the format - $SourceID : Type : TTL : CC : CI_i : RC : RI_i : TI_n : MTR$ (Refer Table 6.1). The number of clusters required, the number of robots required in each cluster, etc will be announced via the CC and CI_i fields. The task to be completed by the cluster is provided in the TI_n field. After the task announcement by the auctioneer, the robots perform role selection. The preference order of the roles is in the following order - cluster head (highest), relay node and worker node (lowest) respectively. The cluster heads are selected first, which in-turn will announce their selected tasks via task acceptance messages. This will be followed by relay and worker node selection. The worker nodes in their task acceptance messages will indicate the cluster which they would join in the CI_n field. Once task assignment is complete, the nodes in a cluster may optionally aggregate or chain together, before moving towards the target location assigned to the cluster members. On reaching the target the cluster head broadcasts task announcement message to delegate the work among the cluster members as explained in Section 6.2.1. The worker nodes in the cluster will assign tasks for themselves and broadcast the same via task acceptance messages. Once the mission is accomplished, the cluster head nodes request the members of the cluster to aggregate together via task announcement messages. The cluster member nodes will again aggregate together before returning to their starting point.

Relay nodes ensure that communication path is established between the cluster heads and the central auctioneer throughout the mission irrespective of the target locations of robots. Once the worker robots start moving to their destination they monitor their own progress towards the mission. If the robots are unable to progress further due to any unforeseen situations (e.g. a hurdle on the path difficult to cross), the same will be conveyed to the peer robots and cluster heads via extended task announcement messages. If robots in SR-ST mission are available nearby, then the robots may choose to assist the trapped robot to progress further by crossing hurdles through suitable formations as supported by the robot structure. Since robots in MR-MT or MR-ST robots have in schedule dependencies they will not respond to extended task announcements. The relay node will ensure that the extended task announcement messages are attended by either the robots which are already on a mission or by idle robots so that the mission can be continued without interruption. Extended task announcement messages are of the same format as that of task announcement messages with unique ‘Type’ field.

6.3 Introduction to ARGoS Simulator

Simulators are indispensable for prototyping multi-robot systems. Use of simulators makes it possible to validate a proposed idea in a safe and controllable environment, with lesser human effort without causing any damage to the robot and other objects. Autonomous Robots Go Swarming (ARGoS) simulator, an open source simulator which was released in 2012, during the European Union (EU) funded project "Swarmanoid" was utilized for the simulation of DTTA framework [2]. ARGoS was also the official simulator of three other EU-funded projects, ASCENS 2, H2SWARM 3 and E-SWARM 4 [2]. Key features of ARGoS simulator are described in this section.

Core Architecture: ARGoS is designed to support swarm of robots. The core elements of the architecture is depicted in Figure 6.2. ARGoS simulator implements a modular architecture at every level. The main architectural components are implemented as software modules which can be selected at run-time as plug-ins. White boxes in Figure 6.2 indicate the user definable plug-ins. Plug-ins include robot models, sensors, physics engines, actuators, visualizations, etc. Programmers can choose the required plug-ins for each aspect of the simulation and instantiate only the resources or features which are required for the particular application thereby improving the simulation speed of the simulator. Users can override or extend the existing plug-ins to fully

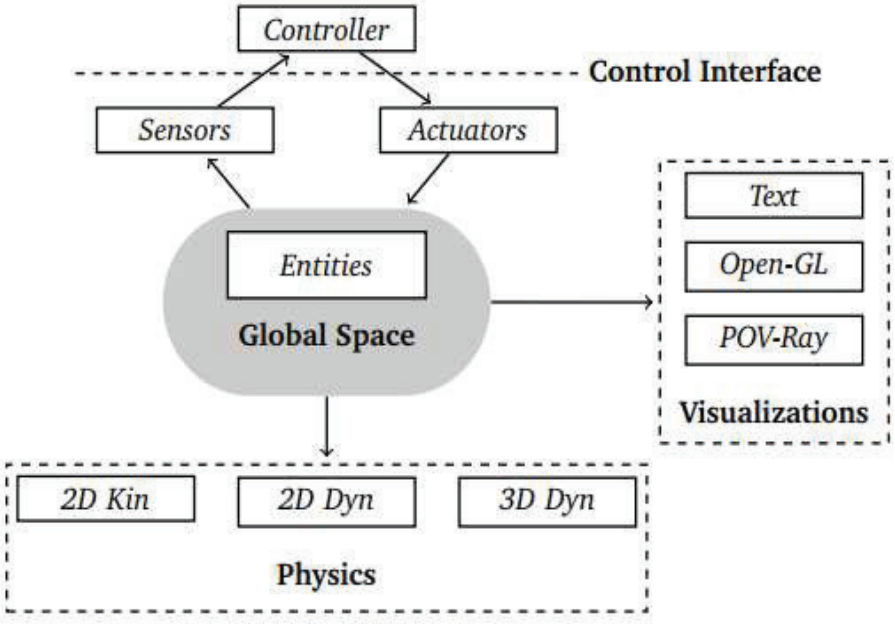

Figure 6.2. The Architecture of ARGoS Simulator [2]

customize the simulator.

The 'global space' in Figure 6.2 is a central repository containing all the relevant information about the state of the simulation. Information is organized as basic architectural elements referred to as entities. Each type of entity stores information about a specific aspect of the simulation. ARGoS natively offers several types of entities and the user can define and integrate new types depending on requirement. A robot is typically represented in the simulated 3D space as a composable entity, that is, an entity that contains other entities. Entities that can compose a robot include the controllable entity, which stores a reference to an instance of the user-defined robot controller, its sensors, actuators, and the embodied entity, which stores spatial information about the robot and the way it occupies space such as the position, orientation of the robot etc. Each type of entity is indexed in data structures optimized for speed.

Most distinctive feature of ARGoS simulator when compared to other multi-robot simulators is that the 3D simulated world can be divided into regions, and each region can be assigned to a different physics engine. Each physics engine manages a subset of the embodied entities in the simulated space. The engines can be executed in parallel and the robots, as they navigate the environment, migrate seamlessly from engine to engine. In addition, the architecture of ARGoS is designed to maximize the usage of modern multi-core CPUs. On a desktop personal computer which supports quad-core processor, ARGoS can simulate up to 4000 robots in real-time [2]. Visualization modules in ARGoS read the state of the simulated 3D space and output a representation of it. ARGoS offers three types of visualization for analysis and debugging- an interactive graphical user interface based on Qt46 and OpenGL7, a text-based visualization designed for interaction with plotting programs such as GNUPlot9, a rendering engine based on the popular ray-tracing software POV-Ray8.

Robots, Sensors and Actuators, Wireless Communication: The ARGoS simulator support different types of robots utilized in EU-funded Swarmanoid project. The three types robots supported by ARGoS are

- Foot-bots-robots which can navigate on land.
- Hand-bots-robots which can climb walls.
- Eye-bots-robots which can fly.

Hence the simulator can be used for different applications which require cooperating robots in land or air. As shown in Figure 6.2, the ARGoS provides an abstract control interface to actuators and sensors. Robot controller interacts with the simulated space via sensors and actuators. Foot-bots navigate using wheels and tracks which allows it to move on complex terrains. The foot-bots and eye-bots support range and bearing device and Wi-Fi for wireless communication. Range and bearing device allow robots in line-of-sight to exchange messages wirelessly. Range and bearing device, upon receipt of a message, can also calculate the relative position (distance and angle) of the sender. The implementation of this device is divided into two parts: the range and bearing sensor and the range and bearing actuator. The former manages the receipt of messages from other robots and the latter sets the message to send. Foot-bot is equipped with two cameras- an omnidirectional camera, and camera which can be mounted looking upwards or frontally. The cameras can be used for path planning of robot. Eye-bot supports a pan and tilt camera with actuators to control the attitude of camera. The foot-bot also support IR proximity sensors, around the robot for near obstacle detection. Another notable feature of foot-bots is the gripper, using which it can hold onto other robots or other objects for cooperative task completion. By gripping on to other foot-bots, the robots can help each other to cross obstacles or drag/push an object from one location to another. Compared to other multi-robot simulators like Player/Stage, ARGoS provides a 3D simulation environment and supports self-assembly of robots [121]. Performance of ARGoS is found to be superior to the performance of stage simulator [122]. ARGoS also provide interface to support network simulators like Network Simulator-2 (NS-2) and Network Simulator-3 (NS-3) enabling simulation of networked robots. Researchers are working on developing the fully functional network simulator interface (NS-3 and NS-2) for ARGoS [122].

ARGoS Programming: ARGoS simulator is written in C++ programming language. The robots can be programmed in C++. In addition to this, ASEBA and Lua scripting languages are integrated with ARGoS. ARGoS simulator can run on Linux and Mac OS. ARGoS requires three essential files for simulations:

- The robot controller (either C++ code or Lua code);
- The code (only in C++), either to update the information regarding the simulation like adding objects to environments or to add visualizations;
- The .argos (XML) file describing the experimental conditions.

The control interface is the same for simulated and real robots, thus the code developed in simulator can be recompiled and download onto the different robots supported by ARGoS.

6.4 Implementation of DTTA Framework on ARGoS Simulator

The feasibility of proposed DTTA framework was verified by modeling two diverse application scenarios on ARGoS simulator as explained in this section.

6.4.1 Application Scenario 1 - Small Scale Swarm Task Allocation Scenario.

A small-scale (typically including robots lesser than 30 in number) swarm robotic system is modeled for analysing the feasibility of DTTA framework. An example of an equivalent real-world application is a warehouse transportation task in which robots are required to navigate to the location where the goods are stored and carry the required object from the place of storage to the place of dispatch of goods. Another example of a practical application is a scenario in which robots are deployed to perform transportation of goods during a defense mission or during search and rescue missions. It is assumed that the auctioneer has information about the tasks to be performed or the objects to be transported by the robots in the system within a given period of time. In the case of outdoor applications such as search and rescue, in which the information about the deployment scenario is not known a priori, the tasks to be performed by the robots in the system can be identified through surveillance and then announced by the central auctioneer. If the area to be covered by the robot is large, the central auctioneer can be a land robot or a drone, based on the type of the terrain. To evaluate the DTTA framework, the robots are assigned a transportation task in which they are required to transport objects from one known location to another.

The application scenario is modeled in ARGoS simulator as follows. An arena of 10m x10m is modeled in ARGoS simulator and three objects are placed at three known locations in the arena. The task(s) to be completed by the robots is to transport objects (tasks) from the known location to another location. New tasks will appear at periodic intervals of time before which the already announced tasks are to be completed. Maximum workload or number of tasks that can be announced

Table 6.2. Modelling of Simulation Scenario 1 in ARGoS.

Parameter	**Configuration**
Arena Size	10m x10m
No of robots assigned to mission	8
Maximum tasks announced by auctioneer in a task announcement message (W)	3
Maximum number of task that can be serviced by a robot (MTR)	2
Type of Robot	Foot-bot
Obstacle sensing	Camera and twelve IR sensors placed around robot
Communication module	Range and bearing
Communication range of Foot-bot	5m
Coopertative task completion	Using gripper
Time reference	Ideal clock/timer provided by ARGoS
Localization	Using range and bearing sensor/ wheel encoders

by the auctioneer (W) at a time (during a round) is '3'. Eight rounds of task announcements were conducted to check the feasibility of DTTA framework. One round includes 8 task announcement sequences as shown in Table 6.3. Each task announcement sequence includes announcement of 3 tasks. The maximum number of tasks that can be serviced by a robot (MTR) is '2'. The gripper on foot-bot is used to hold the object. The robots are programmed such that, If a single robot has to service two tasks, then the robot will grip on to the first object and then drag both the objects towards the destination. The number of robots required to pick up an object is variable for every round as shown in Table 6.3. For example, the number of robots required to serve 'Task1' may be '1' or '2' for a given round. Eight robots are required to complete the mission, out of which 6 are worker nodes, one is auctioneer and the other one is relay node. Since the tasks involve only

small number of robots, clustering is not implemented. One relay node (Figure 6.3) is utilized to monitor the progress of the worker nodes.

The robots are programmed in C++ and Lua based scripting is used for the initialization of the 3D simulation space and the objects in the simulation space including the robots. The robots used in the mission are foot-bots. The foot-bots are programmed to navigate to the perform obstacle sensing using IR proximity sensors (12 sensors placed around the robot). The movement of robot is determined using the wheel encoders on the robot. The range and bearing sensor is used to localize the robot in the area. The wireless communication is also achieved using range and bearing sensors. The communication range of the foot-bots in this case is restricted to 5 meters. Thus 6 different types of MRTA problems are covered in the simulation scenario as indicated in Table 6.3. The combinations which is not tested are MR-MT-IA and MR-MT-TA. Although the DTTA framework allows the allocation of these problems as well, the robot is not designed to collaboratively grip and hold two or more objects. The priority of robots is rotated only after each round.

In case the worker nodes could not progress as expected, the worker node broadcasts extended task announcement messages. The relay nodes may further announce extended task announcement

Table 6.3. Task Announcement sequences/round for Application Scenario 1.

Task Announcement Sequence	**Task1 : No of Robots required**	**Task2: No of Robots required**	**Task3: No of Robots required**	**MRTA type**
1	1	1	1	SR-MT, SR-ST
2	2	2	2	MR-ST
3	2	2	1	MR-ST, SR-ST
4	2	1	2	MR-ST, SR-ST
5	1	2	2	MR-ST, SR-ST
6	1	2	1	MR-ST, SR-MT
7	1	1	2	SR-MT, MR-ST
8	2	1	1	MR-ST, SR-MT

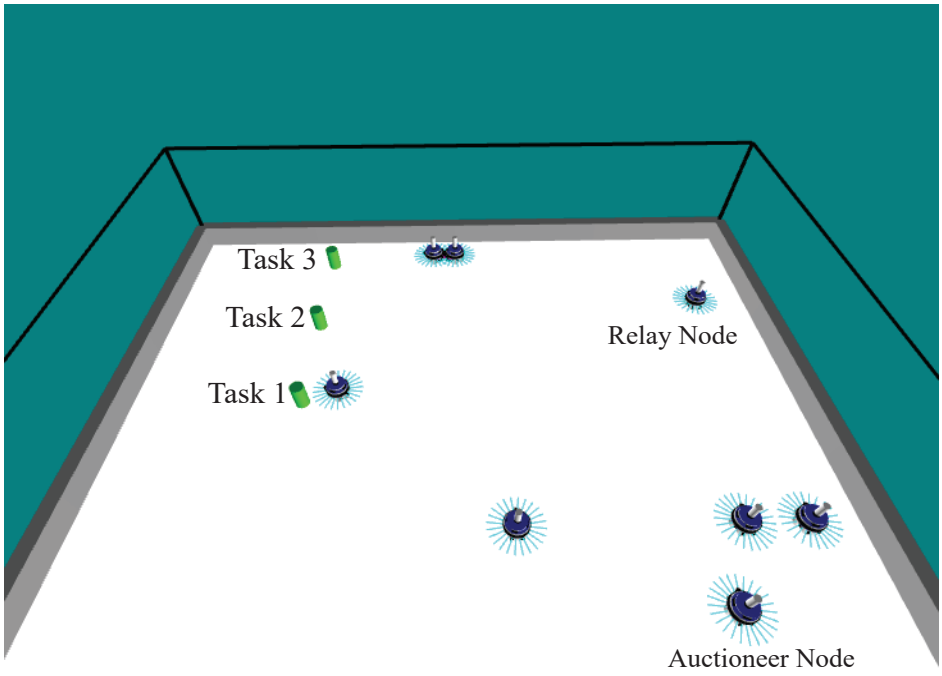

Figure 6.3. Application scenario 1- Foot-bots performing object pick-up task.

messages to ensure successful completeness of the work by other nodes which are free. This guarantees fault tolerance in the system. In 'MR' scenario, when two robots are required to perform a task, chaining of robots is implemented so that, the multiple robots involved in the task proceed to the target location as a chain (Figure 6.3). Figure 6.3, depicts the task scenario where the task announcement is as follows- Number of robots required for Task 1 =1, Number of robots required for Task 2=1 and Number of robots required for Task 3=2. According to DTTA, Task 1 and 2 will be serviced by one robot whereas Task 3 will be serviced by 2 robots which move towards the target as a chain. The computational and communication overhead can be reduced if the robots proceed as chain towards the destination rather than as individual robot as only the leading robot will have to perform path planning and obstacle detection. The test case in Figure 6.3 also represent the scenario where robot serving Task 1 and Task 2 exhibit in-schedule dependency and robots serving Task 3 exhibit cross-schedule dependency. The robots utilized in the experiments are not capable of holding two objects collaboratively with other robots. Hence a robot assigned for a multi-robot (MR) task will not commit to service other tasks.

Table 6.4 provides the number of robots assigned and number of task acceptance messages transmitted per task announcement message. Figure 6.4 indicates the number of tasks served by robots during each round of experiment. Figure 6.5 provides the total number of tasks served by each

Table 6.4. Robot Allocation for Application Scenario 1.

Task1 : No of Robots required	**Task2: No of Robots required**	**Task3: No of Robots required**	**MRTA type**	**No of Robots Assigned**	**No of task acceptance messages**
1	1	1	SR-MT, SR-ST	2	2
2	2	2	MR-ST	6	6
2	2	1	MR-ST, SR-ST	5	5
2	1	2	MR-ST, SR-ST	5	5
1	2	2	MR-ST, SR-ST	5	5
1	2	1	MR-ST, SR-MT	3	3
1	1	2	SR-MT, MR-ST	3	3
2	1	1	MR-ST, SR-MT	3	3

robot for 8 rounds. It can be observed from Figure 6.5 that DTTA protocol is a fair protocol which ensures balanced distribution of tasks among the set of robots.

6.4.2 Application scenario 2 - Medium Scale Swarm Task Allocation Scenario.

To validate the feasibility of using of DTTA for a medium scale swarm network and to demonstrate the suitability of DTTA for diverse robotic structures, another simulation scenario was modelled for an application with 30 worker robots, one relay robot and one task auctioneer robot. The robots are assigned the mission to cover an area of 10m x 30 m searching for an object of interest. An analogy to the real-world scenario is the problem of detection of landmines in an area of interest. Since large area has to be covered, the eye-bot, an air-borne robot is chosen as the auctioneer and relay node. The foot-bots serve as worker nodes. The auctioneer surveys the area to be covered and performs task announcement.

The auctioneer announces 3 tasks to be accomplished, i.e. the formation of 3 clusters with 5, 6 and

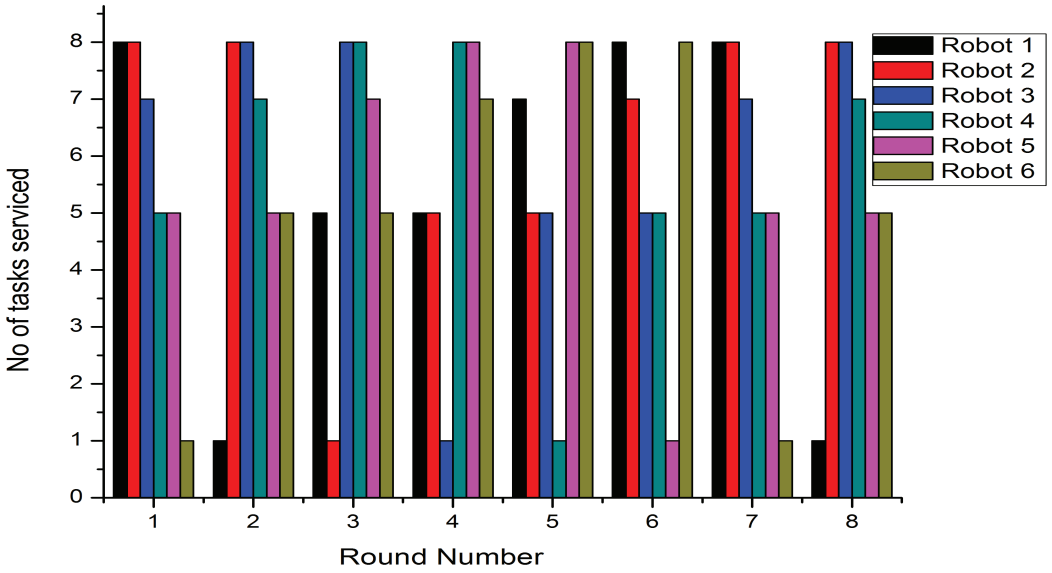

Figure 6.4. No of tasks serviced by robots/round for Application scenario 1.

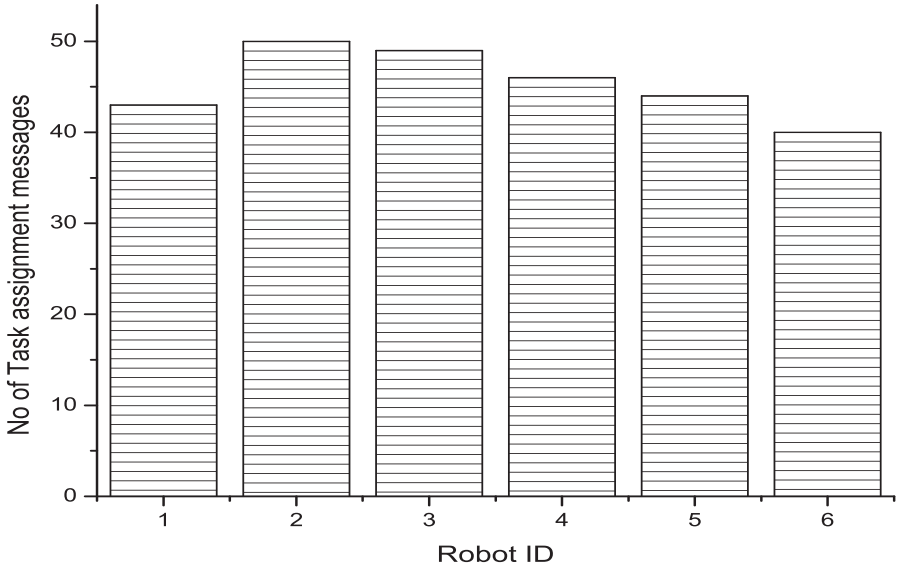

Figure 6.5. Total number of task assignment messages/robot for 8 rounds for Application scenario 1.

19 foot bots respectively through the CC and CI_i fields. The auctioneer in the task announcement message will also announce the requirement of a relay node through 'RC' and 'RI' fields of task announcement message. The communication range between the foot-bots is restricted to 5 meters and the communication range of eyebot is configured to be 10 meters. The target position of the cluster head and relay node is conveyed to robots through the 'CI' field of task announcement message. Once the robots receive the task announcement messages, they self-assign tasks and broadcast task acceptance messages. Once hyperperiod is complete the robots move towards the target location. Members in the cluster are assigned the same task, i.e the task is assigned as a MR-ST problem. The robots are programmed to co-operatively move in such a way that at any point in time the distance between its neighbouring nodes in a cluster is equal. The distance between the nodes is maintained using range and bearing sensor. After the clusters reach the desired area of interest, cluster head performs the required task announcements for its cluster members to ensure coverage of area under each cluster. Cluster members accept the new tasks via task acceptance messages, following which the members of the swarm disperse to complete the mission as in Application scenario 1. Figure 6.6 depicts the scenario where three clusters have reached the target successfully using DTTA framework. The rays between the foot-bots and the eye-bot indicate that the corresponding robots are in communication range. It can be observed that auctioneer is in communication range of cluster 3, whereas the auctioneer can communicate with cluster 1 and cluster 2 only via the relay node. Thus, the feasibility of DTTA for medium scale network is also validated using ARGoS simulator.

6.4.3 Complexity Analysis of Task Allocation using DTTA Framework

Table 6.5 provides a comparison of the computational and communicational complexity of DTTA with the popular market based approaches for SR-ST task allocation scenario. The analysis is provided for a SR-ST scenario with 'm' robots and 'n' tasks. Traditional market based approaches are competitive in nature, such that all robots will bid for the task by sending bidding messages. Hence their commnication complexity is 'm'. In traditional market based approaches, for task assignment comparison of bids from 'm' robots for the 'n' tasks is required. However, the communication complexity of DTTA is '1' if the robot assigns a task for itself. In traditional market based approaches, the communication cost is incurred even if the robot is not assigned a

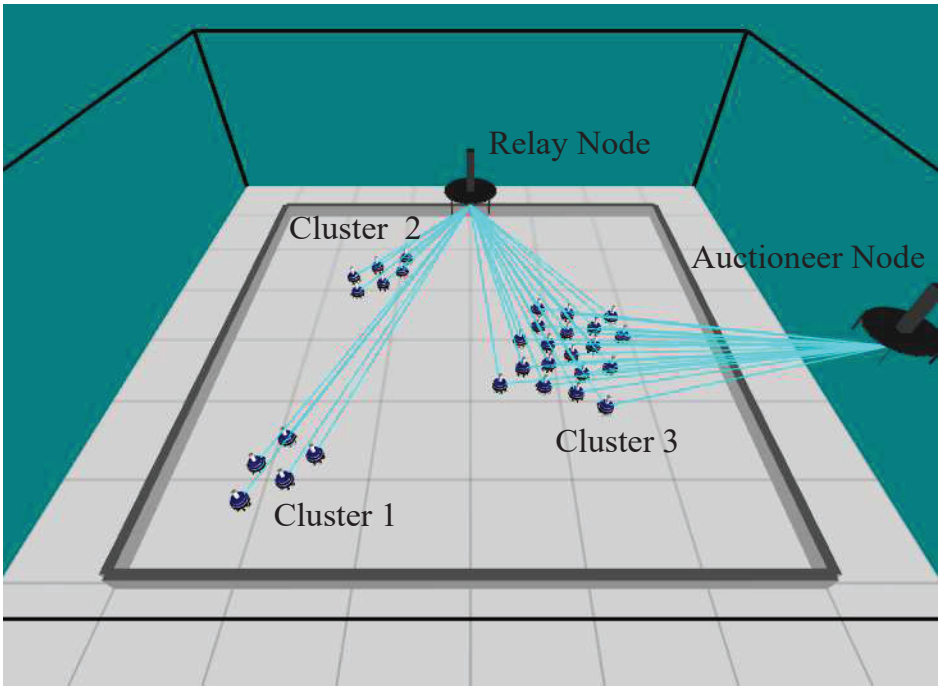

Figure 6.6. Foot-bot clusters at their target location-Application Scenario 2.

task. Also, the computational complexity of DTTA for SR-ST is equal to the number of robots required to service the tasks in the task announcement message.

For a generic task allocation problem considering all 8 types of MRTA problems, the worst case computational complexity is as follows.

$$Worst\ case\ Computational\ complexity = 3 * W * M \tag{6.1}$$

where 'M' is the number of robots required to perform the tasks in the task announcement message and 'W' is the maximum number of tasks announced/task announcement message. The multiplication factor of '3' in the complexity equation is owing to the fact that for any situation other than SR-ST, the count of the tasks to be serviced and the count of the robots required/task should also be taken into consideration before assigning the task. DTTA recommends batch processing of tasks, in which a smaller number of tasks are considered at a time for task allocation, thus reducing the number of computations required for task assignment.

Table 6.5. Comparison of complexity of different task allocation schemes

Task Allocation Scheme	**Computational Complexity**	**Communication Complexity**	**MRTA Type**
Alliance[119]	mn	m	SR-ST
M+[120]	mn	mn	SR-ST
Murdoch [123]	1-bidder, n-auctioneer	n	SR-ST
BLE [124]	mn	mn	SR-ST
First price auctions [125]	1-bidder	n	SR-ST
Dynamic role assignment [126]	1-bidder n-auctioneer	n	SR-ST
DTTA	No of tasks to be serviced/task announcement	1 (if robot accepts task)	SR-ST

6.5 Conclusions

The major conclusions based on the work presented in this chapter are summarized as follows.

- DTTA- a novel, distributed, TDMA based task allocation framework for a swarm of robots which can be utilized to solve any of the 8 different types of MRTA problem identified by Gerkey and Matarić is presented in this chapter.
- The proposed task allocation framework is also suitable for the task allocation of clustered scalable networks.
- The feasibility of utilizing the DTTA framework for robots is tested for two different application scenarios tested on ARGoS simulator.
- Proposed framework is a market based approach which considers task allocation problem as a dynamic decision problem, that needs to be iteratively reconsidered over time rather than as a static assignment problem.

- In DTTA, the computational complexity of task allocation of an SR-ST problem is proportional to the number of robots required to service a given task. In other market based approaches the computational complexity is equal to the total number of robots in the system.
- Similarly, in DTTA, the communication overhead is involved for a robot only if it will service task(s). Traditional market based approaches incur communication overhead even if task is not assigned to a robot.
- The computational complexity of generic MRTA problem, considering all 8 possible types of MRTA problem is 3*W*M, where 'M' is the number of robots required to perform the announced set of tasks and 'W' is the maximum number of tasks announced/task announcement message.