Applied Computer Systems

doi: 10.2478/v10312-012-0001-7

2012 / 13

Experimental Analysis of Contract NET Protocol in Multi-Robot Task Allocation

Aleksis Liekna[1], Egons Lavendelis[2], Arvids Grabovskis[3], [1-3]*Riga Technical University*

***Abstract* – This paper focuses on the experimental analysts of Contract NET protocol for Multi-Robot task allocation. The problem domain consists of multiple vacuum cleaning robots that need to cooperate for cleaning an area that is beyond the capabilities of a single robot. A robot simulator has been used to experiment with various area and robot locations, and the summary of the effort required to process the tasks has been recorded. Experimental results show that using Contract NET protocol alone is not sufficient to achieve optimal results in task allocation. A more advanced strategy with or without involving the Contract NET protocol is required. Possible strategies are outlined and their analysis is the subject of the future work.**

***Keywords* – Contract NET protocol, multi-agent systems, multi-robot systems, task allocation**

I. Introduction

Autonomous agents have to share tasks and allocate them to the most appropriate agents without help of any third parties in order to be capable to execute tasks that can not be done by a single agent. Agents have to decompose tasks and find other agents that are capable of execution of the subtasks. Corresponding communication mechanisms or high level interaction protocols are used to find appropriate agents and allocate tasks to them. To fit needs of different domains, various interaction protocols have been developed for task allocation in multi-agent systems. Some of them are built for specific domains, like the monotonic concession protocol and the corresponding Zeuthen strategy [1] for task oriented domains where agents have to reallocate a set of tasks. Such protocols can not be used in more general situations. Other mechanisms are designed for more general situations where an agent needs to allocate some task to the most suitable agent. Examples of such protocols are different kinds of auctions like English auction, Dutch auction, first-price sealed-bid auction and Vickrey auctions [2], and a Contract NET interaction protocol. English auction, Dutch auction and Contract NET protocols have been formally specified as interaction protocols by Foundation of Intelligent Physical Agents (FIPA) [3] and used in practical applications.

The Contract NET protocol has several advantages over other protocols. Firstly, it allows finding an agent that is the most suitable for the task. Secondly, it is the only protocol that has been accepted as a standard by FIPA and no longer has experimental status [3]. Thus, it is standardized, widely used and well known to the developers of multi-agent systems. Additionally, it is reliable in the sense that if an individual agent becomes unavailable, the task can be easily reassigned to another agent [4].

Still, it is not clear whether the Contract NET protocol will achieve globally optimal solution in real environments where tasks have long execution times, because if a contract is made about one subtask the corresponding agent is not available for the corresponding time despite that some other subtasks may need it more. There are no practical analyses about usage of Contract NET in multi-robot task allocation. Thus, the aim of the paper is to analyse the results achieved by Contract NET protocol in the multi-robot task allocation in a specific domain of vacuum cleaning robots. A set of vacuum cleaning robots widely used as single robots are joined in a multi-robot system that is capable to clean larger premises than a single robot [5]. The following problem is analysed in the paper. The manager agent (the initiator of the interaction) receives the task to clean some area. It decomposes the whole task into subareas to be cleaned by certain robots. Tasks have to be assigned to robots in a way to minimize the total cleaning effort required. Thus, there is a need for appropriate task allocation strategy. We do experiments with the Contract NET protocol and analyse efficiency of task allocation in various situations.

The analysis is performed in the prototype of the multi-robot management system being built for management of multiple Roomba vacuum cleaning robots [6]. The current version of the system consists of three layers. Firstly, the lower layer is a virtual environment imitating a set of vacuum cleaning robots. Secondly, the middle layer or a mediator is introduced to separate the logical part of the system and physical robots, allowing easy substitution of robots by the virtual environment used in practical experiments. Thirdly, the logical layer is implemented as a multi-agent system. Each robot is managed by a corresponding agent, resulting in autonomous entities that consist of robot together with the corresponding agent. The user gives task to the manager agent and task allocation is done by agents. As a consequence, the experiments are concerned with the third layer and multi-agent system.

The remainder of the paper is organized as follows. The Section 2 introduces the background of the research. It introduces to the multi-robot task allocation problem and the Contract NET interaction protocol. The related work is outlined in the Section 3. The Section 4 describes the setup used for the experimental analysis. The results of the experiments are given in the Section 5 and analysed in the Section 6. The Section 7 concludes the paper and outlines the main directions of the future work.

II. Background

A. Multi-Robot Task Allocation

During the last decades a significant shift of attention has been made from single robot systems to multi-robot systems [5], [7], [8]. There is a need not only to coordinate a few robots, but also large groups of probably heterogeneous robots [9]. Centralized approaches fail in case of large groups. As a result, decentralized multi-robot coordination has become an actual problem. One of the main coordination problems is to find appropriate robots for each task and allocate tasks to them.

Various multi-robot task allocation mechanisms exist. Some of them are adopted from multi-agent system theory, while others have been developed specially for multi-robot systems. Well known examples of multi robot task allocation protocols are TraderBots [10] and Murdoch [9]. The Murdoch concentrates on the message addressing. It proposes to address messages by task that is offered for execution. It simplifies finding appropriate robots to offer tasks to by sending a message to robots that are capable to execute exactly the needed task. Unfortunately, the approach does not deal with optimal task allocation. The choice of the robot to allocate the task is left on the initiator of the protocol. The Murdoch can be considered a solution for finding appropriate robots to start negotiations and allocation not the solution to optimal task allocation.

The TraderBots approach is an example of economics-based task allocation mechanisms. It is based on the idea that each individual robot maximizes its profit. Individual robots have the rights to choose either to execute task by themselves or become contractors and offer their tasks to other robots (bidders) that can submit bids with the reward that they agree to execute the task for. If the reward submitted by any bidder is smaller than the costs for the contractor to execute the task, the task is awarded to the bidder for the amount of its bid. Thus, the task allocation can be done in the following way. Firstly, the tasks are allocated using greedy task allocation. Afterwards, robots may trade tasks to each other enabling the system to change the allocation if it is not optimal. In case of large groups of robots this will result in a need for large amount of communications to find out if there is any agent that is willing to bid for every task.

A specific direction of multi-robot systems is swarm-like systems that use ideas of emergent cooperation (one of the first works is [11]). In such systems group-level cooperative behaviour emerges from their interactions with each other and the world. An emergent system can be effective and is simple and elegant solution to a problem. However, emergent cooperation solutions have traditionally been applied to extremely large, homogeneous robot groups. These techniques are a poor fit for small- to medium-scale systems [9].

Autonomous agents and robots have similarities in the sense that task allocation is usually done in a decentralized manner. Thus, the above described task allocation mechanisms for multi-agent systems are used to allocate tasks in multi-robot systems. Example of the multi-agent task allocation protocol used in multi-robot systems is a Contract NET interaction protocol [13]. The main limitation of the multi-agent mechanism usage is the uncertainty if the protocols designed for software environments and usually tested in determined and discrete environments work well in real environments of multi-robot systems.

B. Contract NET Protocol

The Contract NET protocol is an interaction protocol for task allocation. It is used by one agent, named the initiator, which wishes to have some task performed by one or more other agents, named the participants. The initiator is a manager that is willing to optimise a function that characterizes the task. This characteristics is commonly expressed as the price in some domain specific way. It can also be soonest time of completion, fair distribution of tasks, or any other domain specific characteristics. For a given task, any number of participants may respond to a proposal.

Originally the Contract NET was proposed in 1980 by R.G. Smith [14]. In 2002 the protocol was standardized by FIPA [13]. The standardized and nowadays widely used version is slightly modified by adding rejection and confirmation communicative acts to inform participants about rejection and acceptance of the proposal.

This standard version of the protocol is identified by the token *fipa-contract-net* as the value of the protocol parameter of the Agent Communication Language (ACL) message. The flow of the protocol is the following (See Figure 1 for the UML protocol diagram). The initiator asks for proposals from other agents by issuing a Call for Proposals (CFP) that specifies the tasks and any additional conditions for its execution. Participants receiving the CFP are potential contractors and can submit their proposals to perform task that includes preconditions of the execution of the task like price, time when the task will be done or any other precondition. Alternatively, agents may refuse. As soon as the deadline passes, the initiator evaluates the received proposals and selects agents to perform the task. It can choose one, several or none if there are no satisfactory proposals. The agent(s) of the selected proposal(s) will be sent acceptance notification, and the remaining agents will receive rejections. As soon as the initiator accepts the proposal, the participant has a commitment to carry out the task, i.e. the proposal is binding to the participant. After completing the task the participant sends the completion message to the initiator that can be either information that the task is done or include also results of the task. In case of failure corresponding message is sent. Additionally, not-understood messages can be sent at any moment of the protocol in case the agent did not understand the previous message [13].

In general the Contract NET protocol allows finding the most suitable agent for a single task by comparing proposals submitted by agents. Still it is not clear how to use it optimally for multiple tasks at the same time. Additionally, the Contract NET protocol is built for multi-agent systems. It is not clear how it will work in different domains with robots working in physical environment.

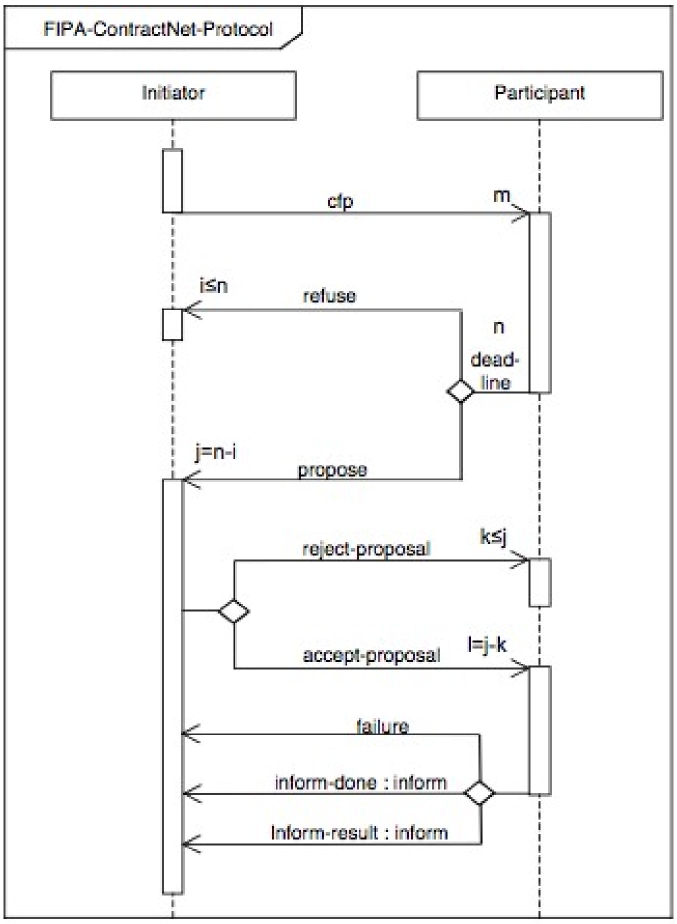


Fig. 1. The interaction diagram of the Contract Net protocol [13]

III. Related Work

The Contract NET protocol and its modifications are widely used in different application domains, for example, manufacturing systems [16], resource allocation in grid and sensor web environments, as well as in hospitals [17], [18], [19], electronic marketplaces [20], vehicle scheduling [21], power distribution network restoration [22], etc. It has become both de facto and formal standard of task allocation protocols. The wide application of the Contract NET protocol has led to the situation where major agent development platforms like JADE offer libraries with implementation of the Contract NET protocol facilitating practical usage of the protocol [15]. Various analyses of the Contract NET protocol applications have been carried out, too.

Dellarocas and Klein have analysed usage of the Contract NET protocol together with electronic social institutions to help guarantee stability and efficiency of interactions [20]. The authors analyse the possibilities to increase quality of service and decrease communication overhead instead of analysing the resulting task allocation. Additionally, it concentrates on tasks done by agents, not physical robots.

Different algorithms for task allocation inside the Contract NET interaction protocol have been analysed in grids. The optimality of task allocation is analysed in virtual environment of resources and users. Criteria like price paid, success rate and load balancing are used to analyse different algorithms that the initiator may use for choosing the appropriate participant to allocate the task to. It has been concluded that methods for choosing the winning bid can improve values of the abovementioned criteria [18].

Contract NET protocol has been applied to systems of physical robots. Mostly it is done with some modifications. A well known modification of the Contract NET is the abovementioned Murdoch protocol that modifies the addressing of the agents by the tasks they are capable to carry out [9]. Another study is done by Golfarell and colleagues. They have analysed a Contract NET based approach for task swapping among robots. The effectiveness has been analysed in terms of utility and complexity of interactions. Still, this analysis concerns only the case when each robot already has some tasks allocated to them and they need to swap tasks so that both robots get more suitable tasks to them [23].

The Contract NET protocol has been analysed in various other aspects. Analysis with respect to the message semantics and possible states of Contract NET based interactions is done by Paurobally and Cunningham [24]. Usage of instance-based learning has been analysed to improve the performance of the multi-agent system. It is analysed how the scalability issues of the Contract NET protocol can be solved by this learning method [19]. Kodama and colleagues have used genetic algorithms for agents to learn parameters to determine what contracts should be preferred [22]. Another solution to limiting the complexity of communications and improving scalability is the MACRO approach [17]. It introduces broker agents that make allocation of the hierarchically decomposable tasks more effective. The effectiveness of the approach is experimentally proven. Nevertheless, this approach is meant for agents whose capabilities have constant geographical location as it is in the sensor web.

Still, to the authors' knowledge, there is no analysis on how the Contract NET will operate in the environment of the multi-robot systems with homogenous robots and different-sized tasks of the same type. Thus the paper analyses the appropriateness of the Contract NET protocol to task allocation in such multi-robot systems; in particular it analyses task allocation to multiple vacuum cleaning robots.

IV. Setup for Experimental Analysis of Task Allocation

A. *Problem Domain and Problem Statement for Multi-Robot Task Allocation*

Problem domain for multi-robot task allocation is vacuum cleaning. Single vacuum cleaning robots are good for cleaning relatively small areas such as hotel-rooms, whereas areas such as warehouses and hangars are beyond the capabilities of a single robot – it may take too much time and/or resources to clean that sort of area by a single vacuum cleaning robot. That is why multiple vacuum cleaning robots must be introduced to complete such vacuum cleaning tasks. The whole area must be split into smaller ones and each of the smaller areas must be assigned to an individual vacuum cleaning robot.

This leads to a situation when there are multiple tasks and multiple robots in the system. As soon as there is more than one task and/or more than one robot capable of completing such a task, one must use a task allocation strategy to assign tasks to robots in a way to maximize the expected performance. Hence, the term *optimal task allocation* is introduced. A solution is optimal if there is no better solution given to all the information available in the system [9]. Obviously an optimal solution gives maximum expected performance. In vacuum cleaning problem domain the optimal solution is the one that minimizes the effort required for cleaning an area. In this particular case the effort can be measured as the sum of distances traveled by vacuum cleaning robots. Since the distance traveled while cleaning may depend on several factors not directly related to the task allocation (such as the amount of dirt on the floor), the only thing that matters is the distance a vacuum cleaning robot travels from the moment a task is allocated to it, till it reaches the assigned area to clean and starts the cleaning process.

This allows the formulation of the *problem statement* for the optimal task allocation in a particular vacuum cleaning domain: given a set of areas (a set may be obtained splitting a larger area into smaller ones) to clean, and a set of robots capable of cleaning an area, allocate each area to a robot in a way that the summary distance traveled by robots among areas to clean is minimized. This formulation respects the Multi-Robot Task Allocation (MRTA) definition given in [9].

B. Multi-Agent System Architecture for Multi-Robot System Management

Each vacuum cleaning robot has its own algorithm for cleaning the area, and it is assumed here that this algorithm fits its purpose. Nevertheless, robots lack social capabilities and it is impossible to assign only a part of area to a specific robot. To overcome this issue and unite the individual robots in a multi-robot system, the authors of the paper introduce multi-agent system architecture for multi-robot system management. Robots use their own built-in algorithms for area cleaning, and the purpose of the multi-agent system is only to specify which area has to be cleaned by which robot.

Multi-agent system architecture consists of three layers – the robot layer, the mediator and the multi-agent system. At the bottom of the layer hierarchy there is the (vacuum cleaning) robot layer. Robots receive commands from the mediator and periodically report their current status (coordinates, sensor data, etc.) to the mediator. Application logic (such as task allocation) is not included in this layer – it is only responsible for communication between the mediator and the robots themselves. Robot-specific logic and data are used in this layer.

Multi-agent system layer is responsible for management of robots and consists of four types of agents namely: robot, manager, gateway and user interface (UI) agents. There is exactly one UI, manager and gateway agent in the multi-agent system, while the robot agent count is equal to the count of the physical robots. The UI agent represents the user in the multi-agent system and is described in detail in the next subsection.

Manager agent is responsible for task decomposition and task allocation to robot agents. It receives an area cleaning request from the UI agent, splits the area into smaller ones if necessary (according to the size of area) and uses the Contract NET protocol to allocate each of the resulting areas as tasks to robot agents. The process of task allocation using Contract NET protocol is described in detail in Subsection D of this section.

The robot agents represent physical robots in the multi-agent system. They participate in contract NET negotiations issued by the manager agent and send appropriate commands to the robot they represent.

Mediator layer provides mapping between the multi-agent system layer and the robot layer. It ensures encapsulation of robot layer specific details from the multi-agent system, as well as the separation of multi-agent system specific logic and data from the robot layer.

The multi-agent system architecture is implemented in the following way. The multi-agent system layer is implemented in JADE. The gateway agent of the multi-agent system layer contains a simulator of the mediator layer which is implemented in Java. The mediator layer simulator also includes a vacuum cleaning robot simulator.

C. Graphical User Interface

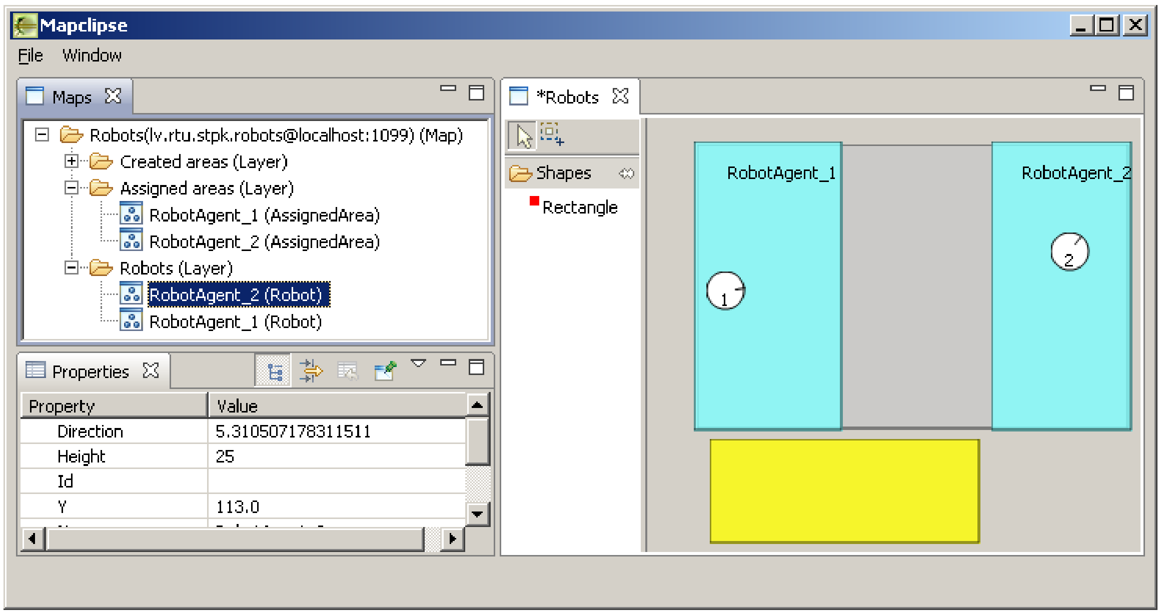
Choosing JADE as an agent development framework, it makes Java-based graphical user interface (GUI) development frameworks more favorable, because another JADE platform could be integrated into GUI in order to connect to the main platform. There is a wide variety of such GUI frameworks like Netbeans, Java's built-in Swing, Eclipse, and many others. The decision was in favor of Eclipse, because of a rich set of ready to use components, its extensibility, rich documentation, project maturity, and developer experience.

In order to build map viewer, Eclipse's Graphical Editing Framework (GEF) [25] has been chosen over Draw2D framework because GEF uses model-view-controller design pattern and provides a means to update the view depending on model changes in an efficient way – only those parts of view are updated, which have been changed.

The concept of map composition basically is the same with some minor differences between different areas of application:

- Map consists of several ordered layers
- Layers contain elements of the same type
- Elements of latter layers overlay the elements of the first ones

These common aspects have led to an initiative to build an extensible map viewing framework, where a robot map viewer will be only one of the implementations. Common application domain allows avoiding implementing the same parts of system (like change event propagation, property viewing, some viewer parts, etc.) over and over, while delegating plug-in to handle specific tasks such as connecting to information source, committing changes to map, updating model, and providing custom figures. As a result GUI project has been split into several parts:



```
Robots(lv.rtu.stpk.robots@localhost:1099) (Map)
  Created areas (Layer)
  Assigned areas (Layer)
    RobotAgent_1 (AssignedArea)
    RobotAgent_2 (AssignedArea)
  Robots (Layer)
    RobotAgent_2 (Robot)
    RobotAgent_1 (Robot)
```

Property	Value
Direction	5.310507178311511
Height	25
Id	
Y	113.0

Fig. 2. Mapclipse GUI with vacuum cleaning robot map plug-in

- Model class definition that would support class property change event propagation
- Generalized map drawing framework core called “Mapclipse” that reflects the state of model in GUI
- Example map plug-in used while developing a core framework and customized figures of robot map plug-in
- Vacuum cleaning robot map plug-in, which connects to main JADE platform, populates the model, and customize the figures to its needs.

This project breakdown turned out to be very convenient, because GUI developer could implement, test and debug a core framework and customized figures using an example map plug-in of which he had a higher degree of control. Other benefits of using the example plug-in include simulating different model states, faster system startups during GUI development (no need to run main JADE platform), and focusing merely on GUI development while leaving JADE specifics aside which improved developer productivity.

Figure 2 shows the Mapclipse with vacuum cleaning robot map plug-in. On the left side there is map information – list of layers and their elements. Below the map list there is the properties view that shows the state of the selected element. In the map viewer (editor area) there can be seen two robots and two areas that are assigned to them for cleaning. The area between the assigned areas (one large area created by a user but split between robots according to an inner algorithm) is going to be assigned for cleaning right after one of the robot finishes the currently assigned area. The lowest area is created by a user, but not yet sent to a multi-agent platform for processing.

The list of possible application areas of Mapclipse is very wide, for example, robot self-localization sensory data merging visualization where different layers represent information gathered from laser sensors, odometer sensors, GPS, etc.

Mapclipse supports simultaneous use of different map providers. For example, using robot map plug-in, more than one connection to different main platforms can be established meanwhile viewing information about robot self-localization status.

D. Contract NET as the Task Allocation Strategy

Contract NET protocol is used in the multi-agent system layer to allocate tasks to agents. Manager agent plays the initiator role and robot agents play the participant roles. When a new request for cleaning an area is received by the manager agent, it is processed in the following way. First of all, the manager agent uses geometric transformations to split the initial area into smaller ones in such a way that each of the resulting areas can be processed by a single vacuum cleaning robot. Next, the manager agent starts the contract NET protocol on each of the resulting areas. Robot agent who proposes the lowest price for cleaning the area wins. It then travels to the area and starts cleaning it. When area cleaning is complete, robot agent informs the manager agent. When all sub-areas of the initial area are cleaned, the manager agent responds to the UI agent’s initial request to indicate the success (or failure) of the operation.

Contract NET on each area is done independently of others and the order of the contract NETs is not defined – each area is assumed to be a separate task. When the robot agent receives a call for proposal on a specific area, it sends the proposal only if it is not currently executing a task, otherwise it sends a refuse message.

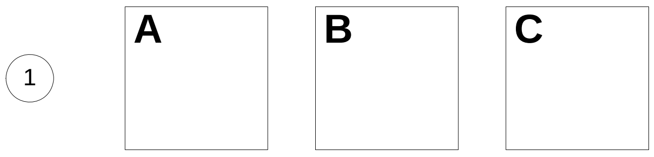


Fig. 3. Setup for the first experiment – one agent and three areas to clean

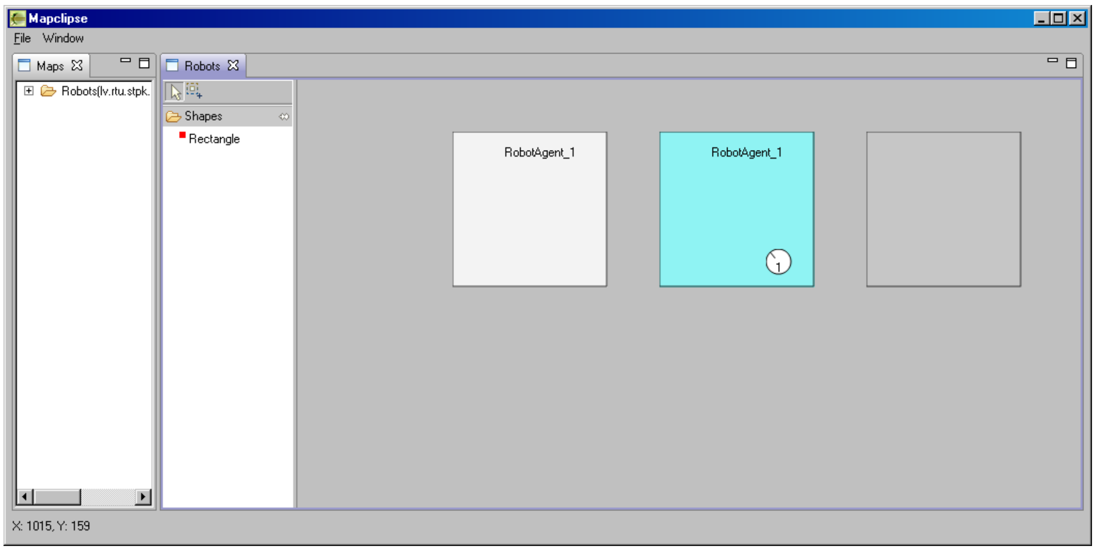


Fig. 4. First experiment in progress. Robot agent is processing the second area. The first area is already processed, while the third area is yet to-be processed

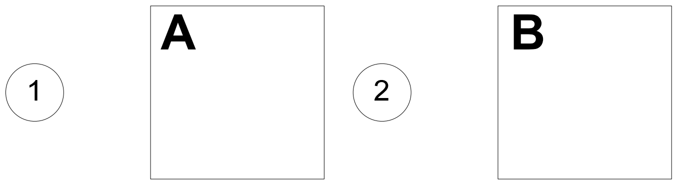


Fig. 5. Setup for the second experiment – two robots and two areas to clean

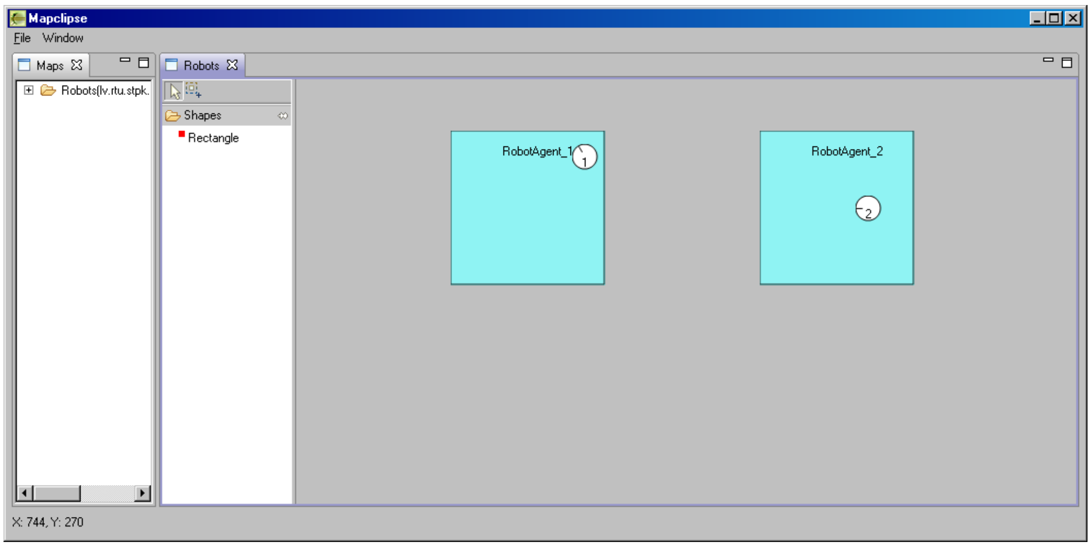


Fig. 6. Second experiment in progress. Robot 1 processing the first area, while robot 2 is processing the second one

V. Experiments and the Results

The purpose of experiments is to empirically test various combinations of robot and area cleaning task placements and to record the distance traveled by robot agents among the tasks. This data can be analyzed to determine how optimal the task allocation is by using contract NET protocol.

It is given that a robot agent has reached an area to clean, when it has reached the center of that area. Distance between the tasks is measured as the distance between the center coordinates of the appropriate areas to clean.

A. One Agent, Multiple Tasks

The first experiment has been conducted with one agent and multiple areas to clean (tasks) (see Figure 3).

It is not hard to see from the Figure 3, that the logical order of cleaning areas (the optimal task allocation) is A → B → C. Given that initial x and y coordinates of the robot are (10, 25) and the center coordinates of areas A, B and C are respectively (45, 25), (85, 25) and (125, 25) then the distance robot has to travel among the areas is 30 + 40 + 40 = 110 units. This calculation is done assuming the robot is in the center of the area, when it is finished to clean it. In reality, the robot can be at any point of the given area and there is no way to predict this point for sure, because no details of the internal cleaning algorithm supplied by the robot manufacturer are given. Therefore the term *predicted distance* is introduced to denote the theoretically calculated distance. Similarly, the term *observed distance* denotes the actual distance measured in practical experiments.

Since the order of the Contract NETs is not defined, any of areas A, B or C could be auctioned and thus processed first. The same is also true for the second and third area, which means there are 6 possible area processing combinations, i.e. A → B → C, A → C → B, B → A → C, B → C → A, C → A → B and C → B → A. Practical experiments have been conducted to measure the observed distance for each of these combinations (see Fig. 4). Ten experiments have been conducted for each of the combinations and the average values of traveled distances have been calculated. The optimal combination A → B → C has showed the average observed distance of 132 units taken as a baseline. The rest of results are shown in Table I.

TABLE I
Observed Distances of Various Processing Combinations

Processing combination	Observed distance	Difference from baseline %
A → C → B	164	24
B → A → C	197	49
B → C → A	214	62
C → A → B	232	76
C → B → A	212	61

B. Two Agents, Two Tasks

The second experiment has been conducted with two agents and two tasks as shown in Figure 6.

Center coordinates of agent 1 are (10, 25), while the center of agent 2 lies at (70, 25). Center coordinates of areas A and B are respectively (45, 25) and (105, 25). The optimal task allocation would be that agent 1 goes A and agent 2 goes B (1 → A, 2 → B). This gives the predicted distance of 35 + 35 = 70 units, and this is what happens when the area B is auctioned first. But, since the order of Contract NETs is not defined, there could easily be a situation when A is auctioned first and B is auctioned second. Agent 2 is closer to A than agent 1, so agent 2 would win a Contract NET for the area A, while agent 1 would win a Contract NET for the area B (since agent 1 is busy with A and would send a refuse message to an auction of the area B). The predicted distance in this case (1 → B, 2 → A) is 95 + 25 = 120 units, which exceeds the predicted distance in the optimal case (1 → A, 2 → B) by 71%. Practical experiments (see Fig. 6) show that the observed distance and predicted distance in both cases are equal since there is no uncertainty involved by the robot internal cleaning algorithm, as in the first experiment.

VI. Analysis of Experimental Results

Experimental results show that the difference in the observed distance between the optimal task allocation and task allocation in a particular vacuum cleaning domain, using separate Contract NETs in undefined order, can vary from 24% to 76%. It means that the applied task allocation strategy is far from optimal. Several problems can be observed here.

First problem lies in the fact that the order of Contract NETs is not defined. For a single robot and multiple task case (the first experiment) it means that a robot will go to the area which is auctioned first independently of the distance between the current location of robot and the area. This leads to situations when the robot goes to the farthest possible area, while there are other areas closer to the current location of the robot. For the second experiment, where the number of areas and the number of robots match, it is possible that only a local maximum in optimality could be reached. If the initiator of a Contract NET does not have a clue about other areas to clean (or behaves like it does not have), it may assign a robot to an area closest to that robot (which gives local optimum), but it does not take into account the other robot who may be further away from that area, but assigning the other robot to that specific area could lead to a global optimum.

The second problem arises from the fact that each of the tasks is considered to be a separate entity not related to others. This problem can be thought of as the cause of the first problem. For example, if the three tasks in the first experiment were considered subtasks of a larger (possibly abstract) task, it might be possible to predefine the order of Contract NETs while looking, as well as the whole picture. Similarly, the order of the Contract NETs could be predefined in the second experiment, if both areas were considered parts of a larger task. This way a global optimum could be achieved instead of local one.

Although experiments may seem trivial at first, they illustrate basic problems faced when allocating tasks to multiple robots executing spatially distributed tasks. Situations described here are considered to be the basic building blocks of which more complex situations consist. Example of such a situation is the case involving multiple robots and multiple tasks.

VII. Conclusions and Future Work

Analysis of experimental results leads to a conclusion that Contract NET protocol works very well if the tasks are interdependent and the order of their execution is negligible. This, however, is not the case with a multi-robot system consisting of vacuum cleaning robots, which process area cleaning tasks. Results show that the order of task execution matters – the distance traveled by robots among the tasks varies up to 76% depending on the order of task execution. In addition, the connection among different tasks can not be ignored, since the areas lie in the physical coordination space and are related to each other. These relations among areas may define the optimal order of task execution and are thereby an important part of task allocation.

This indicates the need for additional task allocation strategies in cooperation with the applied Contract NET protocol. The Contract NET protocol serves its purpose well – it allocates a task to a robot that gives the best bid. The actual problem is to determine the order of task auctions with respect to their relationships. The next paragraph indicates possible solutions which are the subject of our future work.

One of the possible solutions is to sequentially assign tasks using Contract NET protocol as done before. Then, instead of starting the execution tasks immediately, robots perform an additional negotiation among themselves. The purpose of such negotiation is to exchange tasks assigned by the Contract NET protocol to achieve the optimal result. If two robots determine that by exchanging their current tasks, they will get higher expected performance, and they perform the exchange. This way the initial task allocation (before the exchange) does not really matter – the tasks could be simply assigned randomly and then all responsibility is shifted to the negotiations among the robots.

Another possible solution is to modify Contract NET protocol in a way that tasks are assigned iteratively. Each robot is asked for the price of completion of each task. When the initiator receives all possible bids, it then selects a combination of bids that give maximum expected performance and accepts them, while rejecting the rest.

Another alternative is to auction all available tasks at once. The robots then bid for specific combination of tasks and the best bid is then accepted by the auction initiator, who assigns the appropriate tasks to the highest bidder.

All the mentioned alternatives have their positive sides and their downsides. Their identification as well as the analysis is the subject of our future work.

Acknowledgments

This research has been partly supported by the European Social Fund within the project "Support for the Implementation of Doctoral Studies at Riga Technical University" and also supported by ERAF European Regional Development Fund project 2010/0258/2DP/2.1.1.1.0/10/APIA/VIAA/005 Development of Intelligent Multiagent Robotics System Technology.

References

[1] J. S. Rosenschein and G. Zlotkin, *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. MIT Press, Cambridge, MA., 1994.

[2] M. Wooldridge, *An Introduction to MultiAgent Systems - Second Edition*, John Wiley & Sons, May 2009.

[3] "FIPA Interaction Protocol Specifications". [Online] Available: http://www.fipa.org/repository/ips.php3 [Accessed: Sept 26.2011].

[4] C. La Fournie, Cooperation using Negotiations and the Contract-Net Protocol. Available online: http://pages.cpsc.ucalgary.ca/~laf/601.72/Negotiaion&ContractNet.pdf [Accessed Sept 26, 2011].

[5] E. Lavendelis et al., "Multi-agent Robotic System Architecture for Effective Task Allocation and Management." In *Recent Researches in Communications, Electronics, Signal Processing & Automatic*: Proceedings of the 11th WSEAS International Conference on Signal Processing, Robotics and Automation (ISPRA '12), United Kingdom, Cambridge, February 22-24, 2012. - pp 167-174.

[6] M.T., Ribes, "Optimization of Floor Cleaning Coverage Performance of a Random Path-Planning Mobile Robot." Universitat de Lleida. Escola Politècnica Superior. Enginyeria en Informàtica, 2007..

[7] R. C.Arkin, T. Balch, and E. Nitz, "Communication of behavioural state in multi-agent retrieval tasks." In *Proceedings of the IEEE Intl. Conf. on Robotics and Automation (ICRA'*, Atlanta, Georgia, 1993, pp. 588–594.

[8] P. Stone and M. Veloso, "Task Decomposition, Dynamic Role Assignment, and Low-Bandwidth Communication for Real-Time Strategic Teamwork", *Artificial Intelligence 110(2)*, 1999, pp. 241–273.

[9] B. P. Gerkey, "On Multi-Robot Task Allocation." PhD Dissertation. University of Southern California Computer Science Department, August 2003.

[10] M.B. Dias, TraderBots: "A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments." Doctoral Dissertation, Robotics Institute, Carnegie Mellon University, 2004.

[11] J.-L. Deneubourg., G. Theraulaz and R Beckers, "Swarm-made architectures". In *Proceedings of the European Conference on Artificial Life (ECAL)*, 1991, Paris, France, pp. 123–133.

[12] J.A. Kensler and A.Agah, "Neural networks-based adaptive bidding with the contract net protocol in multi-robot systems". *Journal Applied Intelligence archive* Vol. 31 (3), December 2009, pp. 347-362.

[13] "FIPA Contract Net Interaction Protocol Specification." Foundation for Intelligent Physical Agents, 2002. [Online] Available: http://www.fipa.org/specs/fipa00029/ [Accessed: Sept 26, 2011].

[14] R.G. Smith, "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver." *In IEEE Transactions on Computers*, Vol. C-29, No. 12, December 1980. pp. 1104-1113.

[15] Bellifemine F. et al., Developing Multi-Agent Systems With JADE, Wiley, 2004. 286 p.

[16] F.-S. Hsieh and C.Y. Chiang, "Workflow Planning in Holonic Manufacturing Systems with Extended Contract Net Protocol." In *Next-Generation Applied Intelligence, Lecture Notes in Computer Science*, 2009, Vol. 5579/2009, pp. 701-710.

[17] J.S. Kinnebrew and G. Biswas, "Efficient Allocation of Hierarchically-Decomposable Tasks in a Sensor Web Contract Net." In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology* - Volume 02 (WI-IAT '09), Vol. 2. 2009. pp. 225-232.

[18] K. Goswami and A. Gupta, "Resource Selection in Grids Using Contract Net." In *Proceedings of the 6th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2008)*, pp. 105-109, 2008.

[19] U. Deshpande, A Gupta and A. Basu. "Performance Improvement of the Contract Net Protocol Using Instance Based Learning." In *Distributed Computing - IWDC 2003. Lecture Notes in Computer Science*, 2003, Vol. 2918/2003, pp. 290-299.

[20] C. Dellarocas, M. Klein and J.A. Rodriguez-Aguilar. "An exception-handling architecture for open electronic marketplaces of contract net

software agents.” *Proceedings of the 2 ACM Conference on Electronic Commerce*, Minneapolis, MN, October 17-20, 2000.

[21] Sandholm, T. “An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations.” In *Proceedings of Eleventh National Conference on Artificial Intelligence*, pp. 256-262. January 1993.

[22] J. Kodama et al, “Multi-agent-based autonomous power distribution network restoration using contract net protocol.” *In Electrical Engineering in Japan*, 166, 2009. pp. 56–63.

[23] M. Golfarelli, D. Maio and S. Rizzi. “A Task-Swap Negotiation Protocol Based on the Contract Net Paradigm.” Technical Report CSITE, No. 005-97, 1997.

[24] S. Paurobally and J. Cunningham. “Verifying the Contract Net Protocol: A Case Study in Interaction Protocol and Agent Communication Language Semantics.” In *Proceedings of the second international workshop on Logic and Communication in Multi-Agent Systems (LCMAS 2004)*, Nancy, France 16-20 August, 2004, pp. 98-117.

[25] B. Moore, et al., *Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework*, IBM Redbooks, 2004, [Online]. Available: http://www.redbooks.ibm.com/redbooks/pdfs/sg246302.pdf [Accessed April 12, 2012].

A. Liekna received his Bc.sc.ing degree in 2008 and his Mg.sc.ing. degree in 2010 from Riga Technical University. At the moment he is a PhD student at Riga Technical University. His major field of study is computer science.

He works as a Research Assistant at Riga Technical University. His research interests include artificial intelligence and multi-agent systems.

He is awarded by the Latvian Foundation for Education for his bachelor thesis “Development and Implementation of Reinforcement Learning Model”.

E. Lavendelis studied at the Faculty of Computer Science and Information Technology of Riga Technical University (Riga, Latvia) since 2001 and received Dr.sc.ing in 2009. The topic of the doctoral thesis was “Open Multi-Agent Architecture and Methodology for Intelligent Tutoring System Development”.
He started to work as a Researcher in 2005. Since 2010 he is a Senior Researcher at Riga Technical University, the Department of System Theory and Design. His research interests are agent technologies, multi-agent systems, agent-oriented software engineering and intelligent tutoring systems.

A. Grabovskis is currently a PhD student at Riga Technical University. He received his Bc.sc.ing degree in 2008 and his Mg.sc.ing. degree in 2010 by graduating the same university. His major field of study is computer science, particularly computer systems. He currently works at Riga Technical University as a Software Developer and he is the Developer Group Leader. In addition he is a Research Assistant at Riga Technical University. His research interests are modular system design and development, artificial intelligence, knowledge representation using logic, intelligent agents and their development.