DYNCNET: A PROTOCOL FOR FLEXIBLE TASK ASSIGNMENT APPLIED IN AN AGV TRANSPORTATION SYSTEM[1]

Danny Weyns [a] Nelis Boucké [a] Kurt Schelfthout [a] Tom Holvoet [a]

[a] *Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium*

Abstract

The work presented in this paper is part of an ongoing effort to study suitable task assignment mechanisms for decentralized MAS. Our focus is on systems that are characterized by tasks with *delayed commencement*. Such a task requires a preceding effort before the agent can start executing the task. An example is a robot that first has to move to the location of a task before it can start executing that task. Important quality requirements for assigning tasks with delayed commencement are flexibility (enable agents to adapt task assignment with changing circumstances) and openness (enable agents to take into account other agents that come and go during the process of task assignment).

In previous work, we have studied Contract Net (CNET) and a field-based approach for task assignment (FiTA). CNET does not provide the required flexibility and openness. FiTA satisfies the required qualities, however, the field-based approach provides an emergent solution for task assignment. It is well known that emergent solutions are difficult to engineer and reason about. This raises the question whether it is not easier to extend CNET to take dynamics and changes into account.

This paper presents the DynCNET protocol. DynCNET is an extention of CNET, with "Dyn" referring to support for dynamic task assignment. DynCNET provides flexibility and openness for assigning tasks with delayed commencement. We compare the DynCNET protocol with CNET and FiTA in an AGV transportation system. Our experiences in this real-world setting show that: (1) the performance of DynCNET and FiTA are similar, while both outperform CNET; (2) extending CNET to deal with delayed commencement of tasks is not obvious; the complexity to engineer DynCNET is similar to FiTA but much more complex than CNET; (3) whereas task assignment with FiTA is an emergent solution, DynCNET explicitly specifies the interaction among agents allowing engineers to reason on the assignment of tasks. This latter property may be of overriding importance in the selection of an agent-based approach for task assignment in practice.

1 Introduction

Task assignment in multiagent systems (MAS) is complex, especially in systems characterized by highly dynamic and changing operating conditions. The work presented in this paper is part of an ongoing effort of our research group to study suitable task assigning mechanisms for decentralized MAS. Our focus is on MAS with the following characteristics:

- The system is subject to highly dynamic and changing operating conditions. Examples are continuous changes in availability of resources, particular failure modes, variations in network traffic, etc.
- Agents can dynamically join and leave the system. Agents may be removed from the system at runtime, new agents may join, or agents may leave the system temporarily.
- The task stream is dynamic and irregular. Examples are dynamically changing workloads, unpredictable peaks in demand, etc.
- Tasks are characterized by delayed commencement, i.e. the execution of a task requires a preceding effort of an agent before the task can actually be executed.

[1]The research of the first author is supported by the the Research Fund K.U.Leuven. The research of the second author is supported by the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT).

A typical example application in the domain of manufacturing control is described in [4]. In this system, workpieces move on a conveyor belt along a set of machines. Workpieces have to pass through a series of operations. Some of the machines can perform the next operation on the workpieces. [4] proposes a approach where workpieces select the next machine based on a first-price, single-round auction. However, several kinds of dynamics may arise while a workpiece moves towards the assigned machine. A machine may finish an operation on another workpiece, making the machine available for the next workpiece. Such machine may be more suitable to execute the next operation on a workpiece that is on its way to its assigned machine. Machines can be added and removed from the system, or can be temporarily unavailable due to a breakdown or maintenance. Task assignment with a first-price single round auction is not able to deal with such kinds of dynamics.

A second typical domain that complies with the problem characteristics are automated transportation systems. In this paper we, use an industrial AGV transportation system as an example. An AGV transportation system consists of a set of automatic guided vehicles (AGVs) that transport loads in a warehouse. AGV transportation systems have to deal with dynamic operation conditions. Production machines may have variable waiting times, certain routes may be blocked, etc. AGVs can leave and re-enter the system, for example for maintenance. The stream of transport tasks that enter the system is typically irregular and unpredictable. Tasks in an AGV transportation system are characterized by delayed commencement. To execute a task, an AGV first has to drive to a load before it can pick the load and transport it to the destination. While driving towards the load all kinds of changes in the system may occur. New tasks may enter the system that are more suitable for the AGV to execute, new AGVs may become available that are more suitable to perform the task, etc.

Quality Requirements for Dynamic Task Assignment. In order to deal with the problem characteristics mentioned above, two important quality requirements for task assignment are flexibility and openness. [19] denotes flexibility as "the quality of being adaptable, i.e. being able to change to fit changed circumstances." For task assignment, flexibility denotes that agents are able to adapt task assignment with changing circumstances during the process of task assignment. Openness is defined as "the quality of being open, i.e. characterized by an attitude of ready accessibility, affording unobstructed entrance and exit." For task assignment, openness denotes that agents take into account other agents that enter and leave during the process of task assignment.

Previous Work. In a joint R & D project (EMC^2 [6]), the DistriNet research group and Egemin have developed a decentralized architecture for an AGV transportation system. This architecture is structured as a MAS where an agent is associated with each AGV and each task in the system. An important goal of the new architecture is to improve flexibility and openness of the system.

In previous work, we have applied two approaches for task assignment in AGV transportation systems. First, we have applied the well-known Contract-Net protocol (CNET [15]). In CNET, an initiator calls for proposals and participants offer proposals to perform the task. When the initiator has received the proposals from all participants, it evaluates the proposals and assigns the task to the participant with the best offer. CNET is a simple protocol, but does not offer flexibility and openness with respect to delayed commencement, i.e. the protocol does not consider opportunities that occur in the period between the entry of a task in the system and the actual execution of the task.

As a second approach, we have developed a field-based approach for task assignment (FiTA [18]). In FiTA, transport agents emit fields that attract idle AGVs. To avoid multiple AGVs driving to the same transport, AGV agents emit repulsive fields. AGV agents follow the gradient of the combined field that guide the AGVs towards pick locations of loads. The AGV agents continuously reconsider the situation and definitive assignment of a transport is delayed until the load is finally picked, which benefits the flexibility and openness of the system. In [18] is demonstrated that FiTA outperforms CNET; the cost is only a small increase of bandwidth usage. Moreover, FiTA turned out to be highly robust with respect to the loss of messages.

Motivations for DynCNET. Standard CNET does not provide the required flexibility and openness to cope with the system characteristics mentioned above. FiTA satisfies the required quality goals, however, the field-based approach provides an *emergent solution* for task assignment. It is well known that emergent approaches are difficult to engineer and reason about. Egemin engineers as well as colleague researchers and reviewers therefore raised the question whether it is not easier to adapt standard CNET to take dynamics and changes in the system into account.

This paper presents the DynCNET protocol. DynCNET is an extention of CNET, with “Dyn” referring to support for dynamic task assignment, i.e. DynCNET supports flexibility and openness for assigning tasks with delayed commencement.

Overview. In the next section, we give a brief overview of the decentralized architecture of the AGV transportation system. Section 3 introduces the DynCNET protocol. In section 4, we discuss test results obtained from applying DynCNET in an AGV transportation system, and we compare DynCNET with standard CNET and FiTA. Section 5 discusses related work. Finally, in section 6 we draw conclusions.

2 The AGV Transportation System

An AGV transportation system is a fully automated industrial transport system, that uses multiple AGVs to transport loads (raw materials, finished products, etc.) in an industrial environment such as a warehouse. In this section we give a brief overview of the AGV transportation application and the basic software architecture.

2.1 AGVs and Transports

AGVs. An AGV is a unmanned, battery powered vehicle, capable of picking up loads, driving around and dropping them. AGVs can communicate by means of a wireless LAN. The AGVs are restricted to follow a predefined layout in the environment. The layout is defined in a map that consists of a network of stations and segments. AGVs are equipped with low-level control software that provides an interface to command the AGV; example commands are move(segment), pick(load), etc. The low-level control software uses sensors and actuators, and beacons in the environment to stay on track, turn, pick and drop loads, and determine the current position.

Transports. A transport represents a task to move a load from a pick location to a drop location. Transports are generated by an external system, typically a warehouse management system. Each transport has a priority that increases over time, e.g. as a result of long waiting times. The execution of a transport starts when an AGV picks the load and it terminates when that AGV drops the load at the drop location.

2.2 Basic Software Architecture

Fig. 1 shows the deployment view of the AGV transportation system.

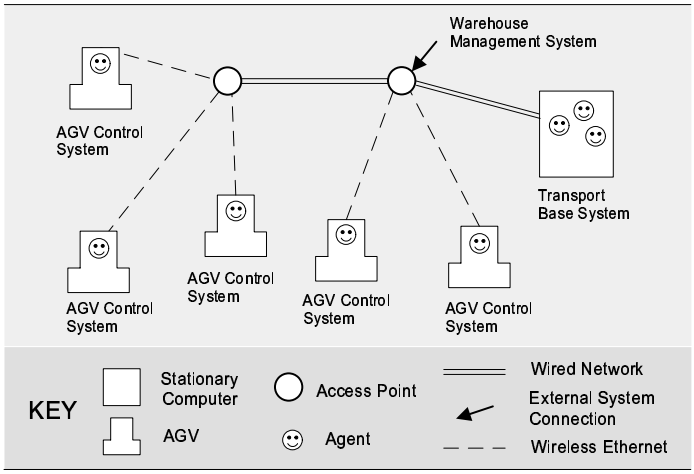

Figure 1: Deployment view of AGV transportation system.

Each AGV is equipped with a computer containing an *AGV Control System*. The AGV control system contains a single *AGV agent* that is responsible for obtaining and executing transports, and ensuring that the AGV gets maintenance on time (such as charging the AGV’s battery). As such, an AGV becomes an autonomous entity that can take advantage of opportunities that occur in its vicinity, and that can enter/leave the system without interrupting the rest of the system.

Each transport in the system is represented by a *transport agent*. Transport agents are part of the *Transport Base System* that is deployed on a stationary computer located in the warehouse. A transport agent is responsible for assigning the transport to an AGV and reporting the status and completion of the transport to the warehouse management system. Transport agents are autonomous entities that interact with AGV agents to find suitable AGVs to execute the transports.

All the subsystems can communicate via a wireless network. The warehouse management system interacts with the AGV transportation system via a wired network.

3 DynCNET Protocol

We start by explaining a number of general properties of the DynCNET protocol. Then we give an overview of the default sequence of the protocol. Next we explain how the agents involved in a protocol can switch the assignment of tasks. We will use the AGV transportation scenario depicted in Fig. 3 to illustrate the steps of the protocol.

General Properties. DynCNET is an $m \times n$ protocol. An initiator that offers a task can interact with m participants, i.e. the candidate agents that can execute the task. On the other hand, each participant can interact with n initiators that offer tasks. As an example, consider the scenario shown in Fig. 3. In the AGV transportation system, an initiator corresponds with a transport agent that represents a task in the system and the participant corresponds with an AGV agent that can execute tasks. We denote the area where an initiator (or participant) searches for participants (or initiators) the area of interest of the initiator (or participant). The dotted circles in Fig.3 show the current areas of interest of AGV A (top) and task x (bottom). For task x, there are currently two candidate AGVs to execute the task: F and G. For AGV A on the other hand, there are three possible tasks to execute: u, v, and w.

Due to the dynamics in the system, the set of candidate tasks (initiators) and agents that can execute a task (participants) can change over time. E.g., in the right part of Fig. 4, AGV E has just dropped its load and becomes a candidate to execute task x.

Default Sequence. Fig. 2 shows an AUML interaction diagram with the default message sequence of DynCNET.

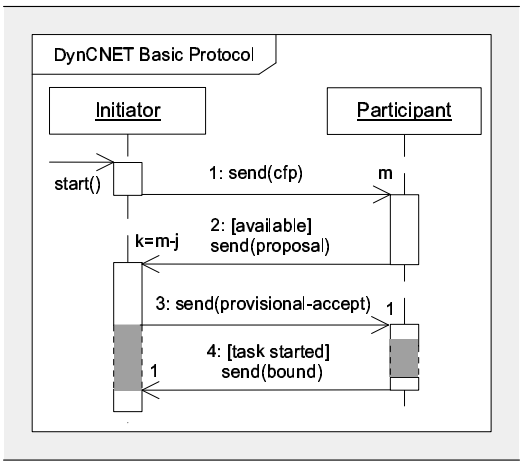

Figure 2: High-level AUML diagram of the DynCNET protocol. Dark zones in the activation boxes represent periods in the protocol when agents can switch the provisional agreement.

The default protocol consists of four steps: (1) the initiator sends a call for proposals; (2) the participants respond with proposals; (3) the initiator notifies the provisional winner; and finally, (4) the selected participant informs the initiator that the task is started. These four steps are basically the same as in the standard CNET protocol. The flexibility of DynCNET is based on the possible revision of the provisional task assignment between the third and fourth step of the protocol.

Switching Initiators and Participants. To explain how agents can switch tasks when the conditions in the environment change, we use the UML state diagram of Fig. 5. This state diagram shows a compact repre-

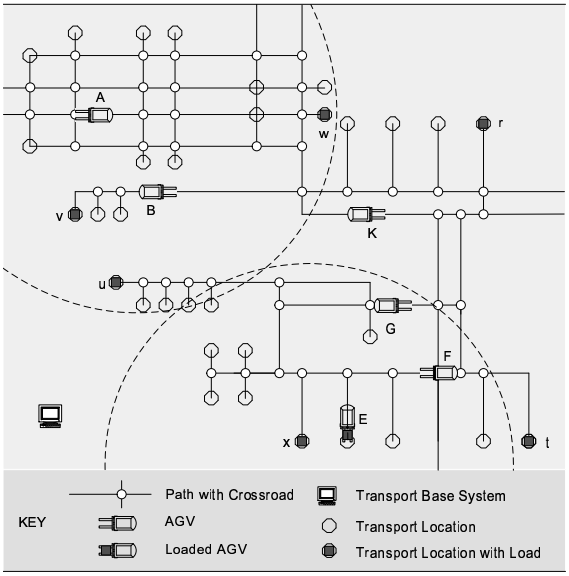

Figure 3: Example scenario in an AGV transportation system to illustrate DynCNET.

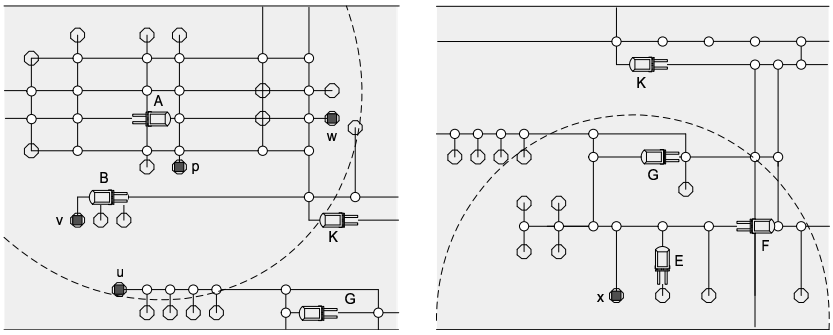

Figure 4: Left: task p provides an opportunity for AGV A. Right: AGV E becomes available for task x.

sentation of the behavior of the initiator and participant agents in the protocol[2]. First we look at the protocol from the perspective of the participant, then we look from the point of view of the initiator.

Switching Initiators. Consider the situation in Fig. 3 where AGV A has a provisional agreement to execute task w. While AGV A drives towards the pick location of task w, a new task p enters the system, see the left part of Fig. 4. This new task is an opportunity for AGV A.

DynCNET enables participants to switch initiators and exploit such opportunities. When a participant is ready to execute a task, it enters the `Voting` state where it answers `cfp`'s with proposals. When the participant receives a `provisional–accept` message (step 3 in Fig. 2), it enters the `Intentional` state, see Fig. 5. As soon as the participant starts the task, it sends a `bound` message to the initiator to inform this latter that the execution of the task is started. The participant is then committed to execute the task. However, if a new opportunity occurs before the task is started, i.e. the participant receives a better offer, the participant changes to the `Switch Initiator` state. In this state the participant retracts from the provisional task assignment and switches to the more suitable task.

Switching Participants. Consider the situation in Fig. 3 where the task x has a provisional agreement with AGV G. While AGV G drives towards the pick location of task x, AGV E becomes available, see the right part of Fig. 4. This new AGV is an opportunity for transport x.

[2]We have made abstraction of a number of synchronization issues in the description of the state diagrams. An example is the situation where an initiator has not yet received the bound message from the provisionally assigned participant that has started a task, while this initiator switches participants. To deal with such synchronization problems, confirmation messages are used. However, to avoid overloaded diagrams, we have omitted these messages. For more details, see [5].

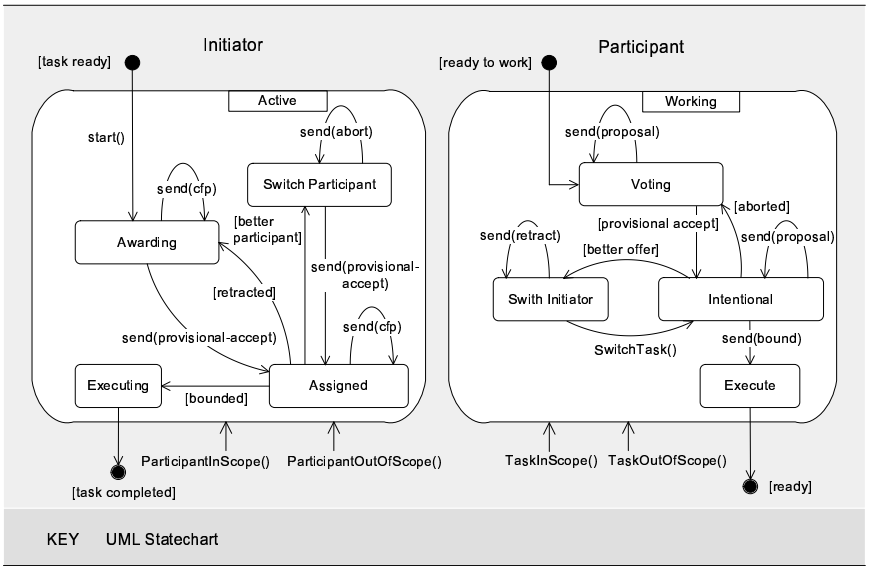

Figure 5: High-level description of the DynCNET protocol

DynCNET enables initiators to switch participants and exploit such opportunities. When an initiator has sent a `cpf` and received the proposals from the participants, it sends a `provisional-accept` message (step 3 in Fig. 2) and enters the `Assigned` state, see Fig. 5. As soon as the initiator receives a `bound` message from the selected participant it enters the state `Executing` in which the task is effectively started. However, if a new opportunity occurs before the task is started, i.e. the initiator receives a better offer from a participant, the initiator changes to the `Switch Participant` state. In this state the initiator sends an `abort` to the provisionally assigned participant and switches to the more suitable participant.

`TaskInScope()` and `TaskOutScope()` are functions that notify the participant when new tasks enter and leave its area of interest. Such functionality can be provided by the perception module of the participant that monitors the area of interest of the agent in the environment, or it can be provided as a notification service by the environment. Similarly, the functions `ParticipantInScope()` and `ParticipantOutOfScope()` notify the initiator when new participants enter and leave its area of interest. [3] elaborates on these functions in the AGV application.

Discussion. Contrary to the suggestions of Egemin engineers and colleague researchers and reviewers, extending standard CNET to take dynamism and change into account is not obvious. Enabling the agents to switch the assignment of tasks increases the complexity of the protocol significantly. Since AUML diagrams do not support the specification of dynamic $m \times n$ protocols, we used state diagrams to describe DynCNET. However, these diagrams are not simple and becomes even more complex when synchronization is taken into account. In addition, DynCNET requires the tuning of various parameters. Examples are the range of interest of both types of agents, the growth rate to extend this range when no suitable candidates are found, the pace to send call for proposals, etc. Parameter tuning is typically associated with emergent solutions such as FiTA. This paper indicates that a flexible agent-interaction protocol that deals with dynamism and change in the system also requires considerable efforts to tune parameters.

Contrary to FiTA, DynCNET explicitly defines the various communicative actions, the points of choice, and the internal state of the agents in the protocol. The protocol precisely specifies which participants are involved in the assignment of a task, when a task is provisionally assigned, when the assignment switches, and when a participant becomes committed to execute the task. The explicit specification of the protocol allows engineers to reason on the assignment of tasks and it helps them to debug the system.

DynCNET (as FiTA) supports openness during delayed commencement. Whereas FiTA inherently supports openness, the DynCNET protocol requires specific functionality for openness. In FiTA, a participant is guided by the combination of sensed fields. A field that disappears or a new field that appears will affect the combination of sensed fields. DynCNET includes specific functionality to notify agents when other agents enter and leave its current area of interest (ParticipantInScope, ParticipantOutOfScope, InitiatorInScope, and InitiatorOutOfScope).

The flexibility of DynCNET is provided by the points of choice for each agent (Switch Participant and

Switch Initiator). The points of choice are abstractly defined in the protocol and need to be instantiated according to the requirements of an application at hand. In the AGV application (discussed in the next section), agents use the priorities of tasks and the distance between AGVs and loads to switch tasks. More advanced approaches can be considered, e.g., participants may (to some extent) favour tasks that are located near to other tasks increasing the change to find a closely located task when the original assignment of tasks for some reason switches.

A final remark concerns the convergence of the protocol. A potential risk of DynCNET is that the assignment of tasks oscillates between participants and no tasks are executed. In the AGV application, oscillations were avoided by: (1) limiting the areas of interest of the agents, and (2) choosing different areas of interest for initiators (task agents) and participants (AGV agents). In particular, the area of interests of AGV agents covered up to 1/10th of the total area of the map and the area of transport agents was 4 times smaller as that of AGV agents.

4 Applying DynCNET in Practice

This section discusses the test results obtained from applying DynCNET in a simulated AGV transportation system. The section starts by explaining the test setting. Subsequently, we present the results of the various tests and we reflect on the test results. Due to space limitation only the main test results are presented. For a detailed discussion of the remaining test results we refer to [5].

4.1 Setting

AGV Transportation System. All tests are performed on the map of an AGV transportation system that is implemented by Egemin at EuroBaltic. The size of the physical layout is 134 m x 134 m. The map has 56 pick and 50 drop locations. We used a standard transport profile that is used by Egemin for testing purposes. This profile generates 140 transports with a random pick location and a random drop location per hour real time. Each transport is assigned a random priority that increases over time. In the simulation, we used 14 AGVs just as in the real application. The average speed of driving AGVs is 0.7 m/s, while pick and drop actions take an average amount of time of 5 s.

Infrastructure. For the tests, we used an AGV simulator [1]. The simulator uses a framework for time management [9] to ensure that the simulation results are independent of performance characteristics of the execution platform, working load of the processor, and memory size. Tests where executed on a cluster of 40 machines: P4 2Ghz, 512MB RAM, Debian Stable 3.0. For a detailed discussion of the AGV simulator, we refer to [1].

Metrics. Every simulation was run for 200.000 timesteps, corresponding to approximately 4 hours real time, i.e. one timestep represents 20 ms in real time. All test results displayed in the paper are average values over 20 simulation runs. Performance is measured in terms of throughput (number of finished transports as a function of simulated timesteps) and reaction time (average waiting time per transport as a function of simulated timesteps). Additionally, the amount of communication needed is measured (number of messages sent per transport).

Reference Algorithms. We compared DynCNET with standard CNET and FiTA. With CNET each transport that enters the system is assigned as soon as possible to the most suitable AGV, i.e., an idle AGV for which the cost to reach the pick location is minimal. When transports can not be assigned immediately, they enter a waiting status. All waiting transports are ordered by priority, and this priority determines the order in which transports are assigned. With FiTA, AGVs combine fields that guide them toward pick locations of transports. Transport agents emit fields that attract idle AGVs. The strength (and thus the range) of the fields of transport agents that attract AGVs depends on the priority of the task and increases over time. The strength of the repulsive fields emitted by the AGVs is fixed.

Parameter Settings. Preceding to the tests, we determined the most suitable set of values for the parameters of the three tested protocols. Parameter tuning is a labour-intensive job. For most of the parameters, for FiTA as well as for DynCNET, we were able to determine a range of values from which we could select one without significantly affecting the performance of the protocol. We suspect that the constrained nature of the problem (such as the restrictions imposed by the layout) accounts for this relaxation. A thorough discussion

of parameter tuning however, is not feasible within the limited space of this paper, for a detailed overview we refer to [5].

4.2 Test Results

We have measured communication load, average waiting time, and the number of finished transports. In addition, we have performed a stress test in which AGVs have to handle as quickly as possible a fixed number of transports from a limited number of locations.

Communication Load. To compare the communication load, we have measured the average number of messages sent per finished transport. The left part of Fig. 6 shows the results of the test.

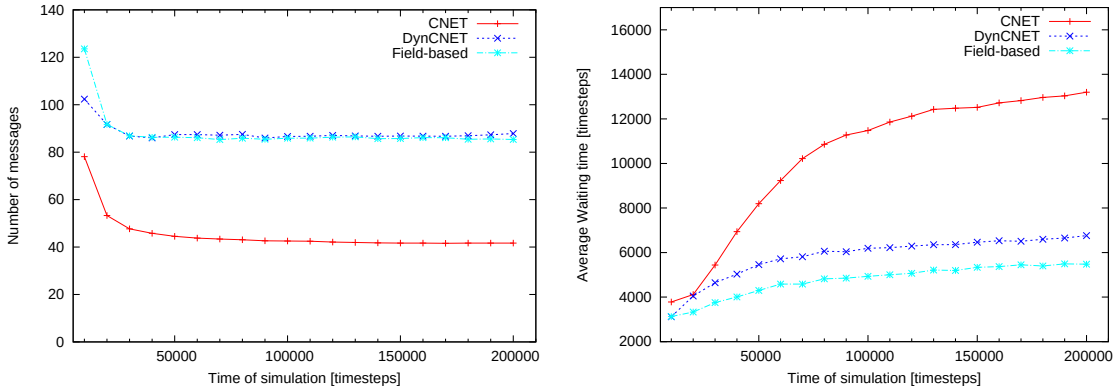

Figure 6: Left: amount of messages being sent per finished transport. Right: average waiting time

The number of messages of DynCNET is approximately the same as for FiTA, while the communication load of CNET is about half of the load of the dynamic mechanisms. However, an important difference exists between the type of messages sent. Fig. 7 summarizes the number of unicast and broadcast messages sent by the three mechanisms.

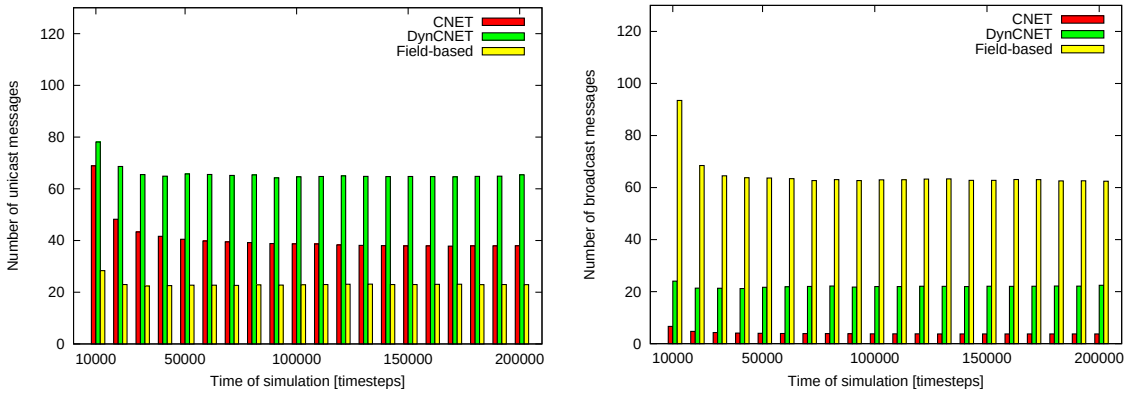

Figure 7: Left: number of unicast messages. Right: number of broadcast messages

For CNET, more than 90 % of the communication are unicast messages. For DynCNET the balance unicast–broadcast messages is about 75–25, yet, for the field-based approach this balance is about 25–75, thus the opposite. This difference is an important factor for selecting appropriate communication infrastructure for a particular task assignment mechanism and vice versa.

Average Waiting Time. The left part of Fig. 6 shows the test results for average waiting time for transports. Average waiting time is expressed as the number of timesteps a transport has to wait before an AGV picks up the load. After a transition period (around 20.000 timesteps), DynCNET outperforms CNET. The difference increases when time elapses. On the other hand, FiTA is slightly better than DynCNET over the full test

range. A possible explanation is that idle AGVs in FiTA immediately start moving when they sense a field of a task, while in DynCNET AGVs only start moving after they are provisionally accepted as a winner. The fact that DynCNET is not able to outperform FiTA came as a surprise to us. Since every aspect in DynCNET is explicitly specified, we expected that if fine tuned well DynCNET could perform better than FiTA.

Number of Finished Transports. The right part of Fig. 8 shows the number of transports finished by each of the protocols during the test run.

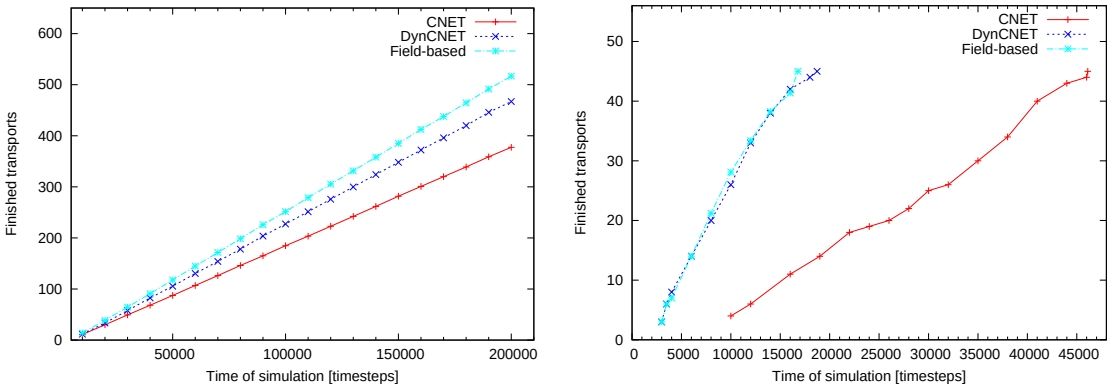

Figure 8: Left: number of finished transports. Right: number of finished transports in the stress test

The results confirm the measures of the average waiting time per finished transport. DynCNET handles more transports than CNET, but less than the field-based protocol. After four hours in real-time, CNET has handled 380 transports, DynCNET has handled 467 transports, and the field-based approach 515 transports. For the 467 executed transports of DynCNET, we measured an average of 414 switches of transport assignments performed by transport agents and AGV agents.

Stress Test. In addition to the standard transport test profile, we have performed a stress test in which 45 transports are created at a limited number of locations in the beginning of the test. These transports have to be dropped at a particular set of destinations. The test simulates for example the arrival of a truck with loads that have to be distributed in a warehouse. The task of the AGVs is to bring the loads as quickly as possible to the right destinations. The transport test profiles for the three mechanisms was identical. The left part of Fig. 8 shows the test results.

The slopes of the curves of FiTA and DynCNET are similar but much steeper than the curve of CNET. The results demonstrate that CNET requires about 2.5 times more time to complete the 45 transports than the dynamic protocols.

In the stress test, the communication load of FiTA is slightly higher as for DynCNET during the full test. In the initial phase of the test, the dynamic protocol requires much more bandwidth as CNET (about factor four), later the difference converges to the difference we have measured for the standard test profiles (about factor two).

Variance. The tests we have discussed are non-deterministic. Orders are generated randomly and priorities are assigned randomly. To verify the statistic significance of the mean values of the test results we have calculated the 95 % confidence interval [17]. The results show confidence intervals between 5 and 10 % variance relative to the mean values, meaning that the variance of the test results is small. For details we refer to [5].

4.3 Summary

DynCNET outperforms CNET on all performance measures, the cost is a doubling of required bandwidth. Contrary to our expectations, DynCNET is in general not able to outperform FiTA. At best DynCNET is able to equal the performance of FiTA. Fig. 9 compares several quality attributes and engineering aspects of the three mechanisms.

Flexibility and Openness. Both DynCNET and FiTA support flexible assigning of tasks with delayed commencement. In FiTA, the choices of the participant agents are implicitly determined by the combination of

	Flexibility during delayed commencement	*Openness during delayed commencement*	*Robustness with respect to message loss*	*Mechanism engineering*	*Parameter tuning*	*Type of communication*
CNET	No: one shot assignment of tasks	Not supported	Requires additional support	Use of common engineering diagrams	Limited number of parameters; tuning is easy.	Mainly unicast
Field-Based Approach	Yes: combination of fields determine the participants' choices	Inherent to the approach	Inherent to the approach	No common engineering approaches available	Various parameters; tuning requires considerable efforts	Mainly broadcast
DynCNET	Yes: explicit switch points for initiator and participants	Explicitly built-in	Requires additional support	Use of common engineering diagrams	Various parameters; tuning requires considerable efforts	Mainly unicast

Figure 9: Summary of quality attributes and engineering aspects of the three mechanisms

the sensed fields. DynCNET provides explicit points of choice for initiators and participants. Both DynCNET and FiTA support openness during delayed commencement, i.e. both mechanisms allow initiators to take into account new participants that become available and participants can participate in the assignment of new tasks that become available. Whereas FiTA inherently supports openness (the combination of fields adapts when fields disappear or new fields appear), the DynCNET protocol includes explicit functions that notify initiators and participants when changes occur. Neither flexibility, nor openness is supported by CNET.

Robustness. In FiTA, the freshness of the received fields is taken into account to determine the attraction and repulsion of fields. When an agent misses an update of a field due to the loss of a message, the previous value of the field is used. Yet, to determine the combined field that guides the agent, less importance is given to older field values. As such, FiTA is (to some degree) robust to message loss. CNET and DynCNET on the other hand fail when a message gets lost and the prescribed sequence of messages is disrupted. As such, the contract-net based mechanisms require additional support for robustness to message loss.

Mechanism Engineering. No common engineering approaches are available for the design and development of FiTA. On the other hand, DynCNET allows to specify the behavior of the agents by means of common engineering diagrams such as interaction diagrams and state charts. We used UniMod [16] to design the DynCNET protocol as a state machine. UniMod enables to draw the state machine and export the diagram to an XML file. This XML file was used to interpret the state machine in the agent program.

Parameter Tuning. CNET does not require much parameter tuning. The designer has only to define a criterion that determines which participants are considered by the initiators. To deal with dynamism and change in the system, DynCNET (as FiTA) requires considerable efforts to tune parameters.

Type of Communication. A significant difference exists in the ratio unicast–broadcast messages that are used in the three task assignment mechanisms. This difference is important for selecting appropriate communication infrastructure for a specific task assignment mechanism and vice versa. Further research is needed to investigate the implications of the type of communication of the different mechanisms.

5 Related Work

Task assignment is an extensive domain of research. Here we focus on mechanisms that are based CNET.

CNET is a widely known and extensively used protocol that uses an auction-like mechanism to achieve task assignment. CNET was originally proposed by Smith and Davis [15] and is included in a FIPA-standard [7]. The DynCNET protocol is an extension to CNET with two distinctive characteristics: (1) it enables $m \times n$ negotiations, i.e. a participant can manage concurrently multiple negotiation processes with the initiators and the initiator can manage negotiation processes with multiple participants; (2) it supports changes on the initiator and the participant side and delays the definitive assignment until the task is effectively started. In this section, we list several variants to CNET and show how they relate to DynCNET.

Several extensions of CNET exist that support $m \times n$ negotiations. [10] describes a protocol that allows a bidder to place bids for multiple unassigned tasks. This is achieved by giving the bidder the option to refuse the task in step 3 of the original protocol (compare Fig. 2). The manager will then award the contract to the second best bidder, who can again accept or refuse it, etc. This protocol does not supports changes at

both sides or delayed commencement.

The protocol in [2] focusses on self-interested agents supporting several negotiation processes in parallel. The approach introduces two levels of bidding, i.e. a pre-bidding phase in which the participants can still change their commitment and a definitive bidding phase in which the task is effectively assigned to a single participant. The protocol allows to reconsider commitments during the pre-bidding phase, but the window in which commitments can be changed is small. As soon as all agents answered in the pre-bidding phase, the definitive bidding phase starts and the participants can no longer change their commitment. An interesting contribution of [2] is that the authors formally prove the converge of the protocol.

FIPA proposes the Iterated CNET protocol, allowing multi-round iterative bidding [8]. The protocol is multi-round in the sense that the initiator can decide to issue a new call for proposals (and thus start a new round) *after* the bidding phase *instead* of accepting a proposal. There is no partial commitment during the iterations and the protocol does not allow to reconsider the situation once a proposal is accepted.

Related to the Iterated CNET protocol are levelled-commitment contracts [13]. The idea of a levelled-commitment contract is that any of the agents in a contract can decommit by paying an agreed decommitting penalty to the other agent(s) in the contract. A leveled-commitment contract is not a protocol on itself, but this type of contracts can used in negotiation protocols between self-interesting agents.

Finally, there are several researchers who use the original CNET protocol in combination with other strategies to provide support for dynamism. Maturana et al. [11] use a mediator pattern for dynamic scheduling, combining mediation and sub-tasking using CNET to produce a schedule. Mediator agents are used to coordinate the resource agents using the CNET protocol. In case of breakdowns, the system is rescheduled by re-running the CNET protocol for the task to find available slots in the schedule. Shen et al. [14] and Ouelhadj et al. [12] also use a combination of the mediator architecture and the CNET protocol, but introduce different re-schedule techniques for different real-time events. For example, the re-scheduling technique for rush orders (try to replace a task by another task) differs from a machine breakdowns (find the nearest free slot in the schedule). These mechanisms using CNET in combination with schedules allow for some flexibility, but rescheduling the tasks takes a lot of time. Thus this type of architectures is more suited for relatively stable schedules and a limited amount of changes.

6 Conclusions

The work presented in this paper is part of an ongoing effort to study suitable task assignment mechanisms for decentralized MAS. Our focus in on the assignment of tasks with delayed commencement. In previous work, we studied two mechanisms for task assignment: standard CNET and FiTA, a field based approach for task assignment. In this paper, we presented the DynCNET protocol, an extension to standard CNET that supports dynamic task assignment.

We compared DynCNET with CNET and FiTA in a real-world test setting. DynCNET (as FiTA) outperforms CNET on all performance measures. The cost is a doubling of required bandwidth. However, contrary to suggestions made by engineers and colleague researchers, extending standard CNET to take dynamism and change into account is not obvious. Enabling the agents to switch the assignment of tasks increases the complexity of the protocol significantly.

Our experiences with DynCNET with FiTA yield the following conclusions:

- Contrary to our expectations, DynCNET was not able to outperform FiTA, both mechanisms have similar performance characteristics.
- Parameter tuning for DynCNET has similar complexity as for FiTA.
- Whereas FiTA is inherently robust to message loss, DynCNET is not and requires additional functionality to deal with message loss.
- FiTA mainly uses broadcast messages, while DynCNET mainly uses unicast message.
- DynCNET explicitly defines the task assignment process among the agents while in FiTA everything is implicitly enclosed in the fields.

Egemin, our partner in the EMC^2 project, currently considers to apply an agent-based approach for task assignment in one of its AGV transportation systems. Due to the explicit specification of the protocol the engineers tend to prefer DynCNET over FiTA. The fact that DynCNET allows engineers to reason on the

assignment of tasks may be of overriding importance in the selection of a mechanism for task assignment in practice. Still, extensive tests are necessary before an eventual decision can be made.

DynCNET is a promising protocol for task assignment in decentralized MAS. Yet, before any claim can be made about its general applicability, the protocol needs further tests in different application domains.

References

[1] DistriNet AGV Simulator, 2005. www.cs.kuleuven.ac.be/∼distrinet/taskforces/agentwise/agvsimulator/.

[2] Samir Aknine, Suzanne Pinson, and Melvin F. Shakun. An Extended Multi-Agent Negotiation Protocol. *Journal on Autonomous Agents and Multi-Agent Systems (JAAMAS)*, 8(4):5–45, 2004.

[3] N. Boucké, D. Weyns, K. Schelfthout, and T. Holvoet. Applying the ATAM to an Architecture for Decentralized Contol of a AGV Transportation System. In *2nd International Conference on Quality of Software Architecture*. Springer, 2006.

[4] S. Bussmann and K. Schild. Self-organizing manufacturing control: An industrial application of agent technology. In *4th International Conference on Multi-Agent Systems*, 2000.

[5] B. Demarsin. *DynCNET: A Protocol for Flexible Transport Assignment in AGV Transportation Systems*. Thesis Master of Artificial Intelligence, Katholieke Universiteit Leuven, Belgium, http://www.cs.kuleuven.be/∼danny/DynCNET.pdf, 2006.

[6] Egemin Modular Controls Concept. EMC ^{2} project, IWT, Belgium. *http://emc2.egemin.com/*, 2005.

[7] FIPA TC Communication. FIPA Contract Net Interaction Protocol Specification, Doc. SC00029, 2002. FIPA-Standard.

[8] FIPA TC Communication. FIPA Iterated Contract Net Interaction Protocol Specification, Doc. SC00030, 2002. FIPA-Standard.

[9] A. Helleboogh, T. Holvoet, D. Weyns, and Y. Berbers. Extending time management support for multi-agent systems. In *Multiagent and Multiagent-based Simulation, New York, USA*, Lecture Notes in Computer Science, Vol. 3415, 2005.

[10] T. Knabe, M. Schillo, and K. Fischer. Improvements to the fipa contract net protocol for performance increase and cascading applications. In *Workshop on Multiagent Interoperability*, 2002.

[11] F. Maturana, W. Shen, and D. Norrie. Metamorph: An adaptive agent-based architecture for intelligent manufacturing. *nternational Journal of Production Research*, 37(10):2159–2174, 1999.

[12] D. Ouelhadj, P. Cowling, and S. Petrovic. *Book of Intelligent Systems Design and Applications*, chapter Contract Net Protocol for Cooperative Optimisation and Dynamic Scheduling of Steel Production, pages 457–470. Springer-Verlag, 2003.

[13] T. Sandholm and V. Lesser. Levelled-commitment contracting: A backtracking instrument for multiagent systems. *AI Magazine*, 23(3):89–100, 2002.

[14] W. Shen and D.H. Norrie. An agent-based approach for dynamic manufacturing scheduling. In *Agent-Based Manufacturing Workshop, Minneapolis*, pages 117–128, 1998.

[15] R. G. Smith. The contract net protocol: High level communication and control in a distributed problem solver. In *IEEE Transactions on Computers, C-29(12)*, 1980.

[16] UNiMod. *http://unimod.sourgeforce.net*, 2006.

[17] E. Weisstein. Confidence Interval, Probability and Statistics, MathWorld. 2006.

[18] D. Weyns, N. Boucké, and T. Holvoet. Gradient Field Based Transport Assignment in AGV Systems. In *5th International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS, Hakodate, Japan*, 2006.

[19] WordNet 2.1. Princeton University Cognitive Science Library. *http://wordnet.princeton.edu/*, 2006.