DynCNET: A Protocol for Dynamic Task Assignment in Multiagent Systems

Danny Weyns, Nelis Boucké, Tom Holvoet, Bart Demarsin
Katholieke Universiteit Leuven, Celestijnenlaan 200A, Belgium
{Danny.Weyns, Nelis.Boucke, Tom.Holvoet}@cs.kuleuven.be

Abstract

Task assignment in multiagent systems is a complex coordination problem, in particular in systems that are subject to dynamic and changing operating conditions. To enable agents to manage dynamism and change, adaptive task assignment approaches are needed. In this paper, we introduce DynCNET, a protocol for dynamic task assignment that extends standard contract net (CNET). DynCNET allows the agents involved in the protocol to switch the assignment of tasks dynamically. We use an industrial automated transportation system as illustration and present results obtained from a real-world test setting that compare DynCNET with standard CNET and a field-based approach for task assignment.

1 Introduction

Two important factors for the growing complexity of today software systems are: the highly dynamic operating conditions under which systems have to operate such as altering workloads and variations in availability of resources, and the inherent distribution of resources which makes central control practically infeasible. In our research, we study situated multiagent systems (situated MAS) for engineering such systems. A situated MAS structures the software into a number of interacting autonomous entities (agents) that are situated in an environment. Control in situated MAS is decentralized, the system functionality results from the cooperation of the agents via the mediating environment.

One particularly challenging coordination problem in situated MAS is task assignment. Tasks in situated MAS are often 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. During delayed commencement, the conditions in the environment may change. Agents should be able to take into account these changes and dynamically adapt the assignment of tasks. A characteristic example application is a transportation system that uses multiple automatic guided vehicles (AGVs) to transport loads in an industrial environment. The stream of tasks that enter the system is typically irregular and unpredictable. Tasks in an AGV transportation system are characterized by delayed commencement, i.e., an AGV first has to drive to a load before it can pick the load and transport it to the destination. While driving toward 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. Such a problem setting requires a flexible task assignment approach that is able to cope with dynamics in the system. In this paper, we introduce a protocol called DynCNET. DynCNET is an extention of the well-know contract net protocol (CNET [8]), with "Dyn" referring to support for dynamic task assignment.

Overview. In the next section, we explain the DynCNET protocol. In section 3, we discuss test results obtained from applying DynCNET in an AGV transportation system and we compare DynCNET with CNET and a field-based approach for task assignment. Section 4 discusses related work. Finally, in section 5 we draw conclusions.

2 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. 2 to illustrate the steps of the protocol. In the AGV application two types of agents are used: AGV agents and transport agents. Each AGV is controlled by an AGV agent. Transports are represented by transport agents that reside at a transport base system, i.e. a stationary computer located in the warehouse.

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. 2. 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

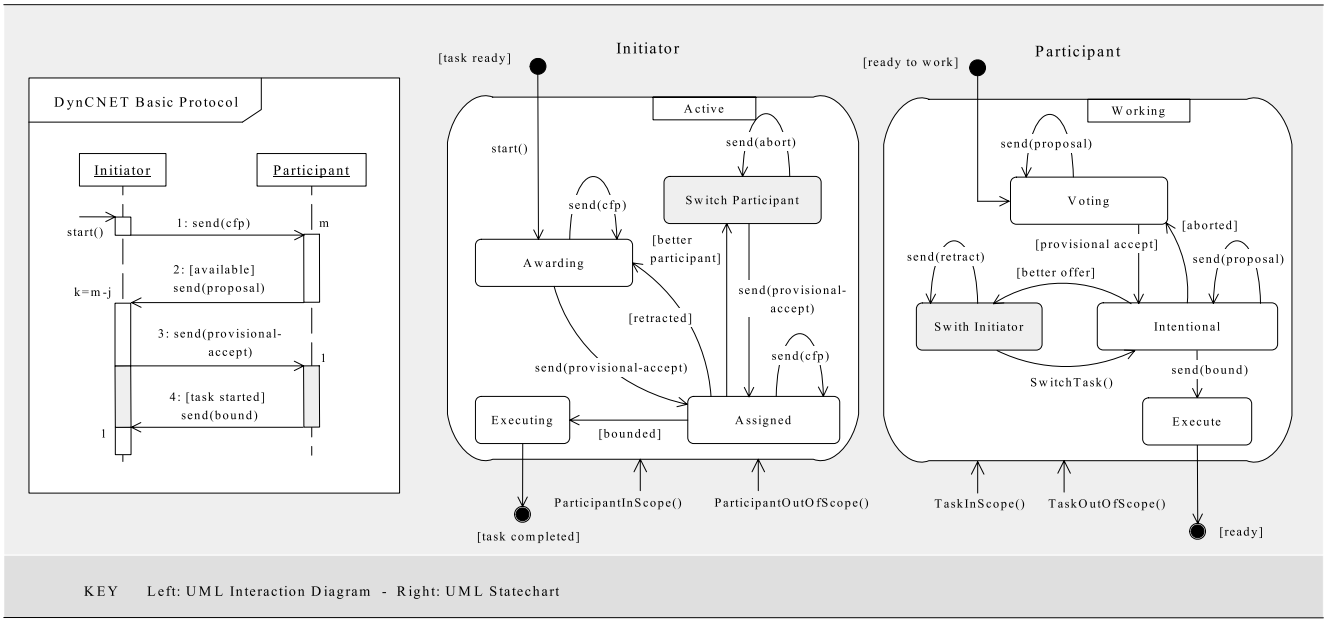

Figure 1. High-level diagrams of DynCNET. Left: interaction diagram; right: state diagram

(or participant) searches for participants (or initiators) the area of interest of the initiator (or participant). The dotted circles in Fig.2 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 (AGV E is delivering a load). 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., when AGV E drops its load at location s, it becomes a candidate to execute task x.

Default Sequence. The AUML interaction diagram of Fig. 1 shows the default message sequence of DynCNET. 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, i.e. the shades zones in Fig. 1.

Switching Initiators and Participants. To explain how agents can switch tasks when the conditions in the environment change, we use the UML state diagram in Fig. 1 that shows a compact representation of the behavior of the agents in the protocol. 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. 2 where AGV A has a provisional agreement to execute task w.

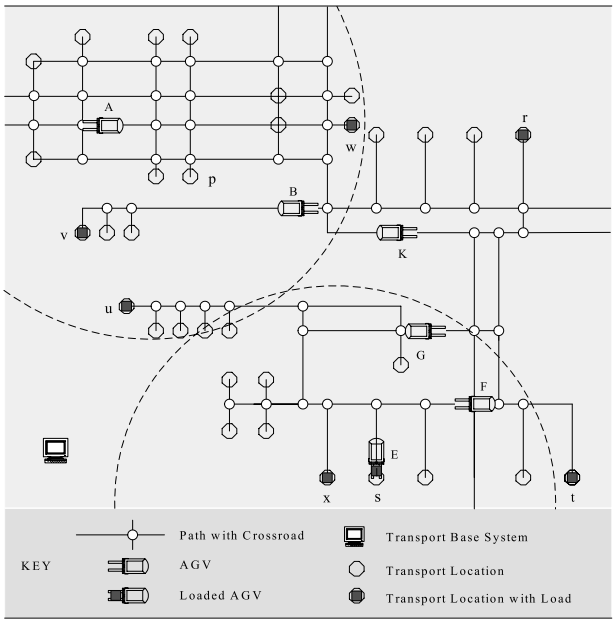

Figure 2. Scenario to illustrate DynCNET

While AGV A drives toward the pick location of task w, a new task may enter at the transport location p. 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 the interaction diagram of Fig. 1), it enters the `Intentional` state. As soon as the participant

starts the task, it sends a `bound` message to inform the initiator. 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 (`SwitchTask()`).

Switching Participants. Consider the situation in Fig. 2 where the task x has a provisional agreement with AGV G. While AGV G drives toward the pick location of task x, AGV E may drop its load at transport location s and becomes available. This new AGV is an opportunity for transport x. DynCNET enables initiators to switch participants and exploit such opportunities. When an initiator has sent a `cfp` and received the `proposals` from the participants, it sends a `provisional-accept` message (step 3 in interaction diagram of Fig. 1) and enters the `Assigned` state. 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`, 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 [11] of the participant that monitors the area of interest of the agent in the environment. Similarly, the functions `ParticipantInScope()` and `ParticipantOutOfScope()` notify the initiator when new participants enter and leave its area of interest.

Convergence. 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.

Synchronization messages. To handle synchronization, confirmation messages are used. For example, when an initiator switches participants it first sends an abort to the participant that has provisionally accepted, see state diagram in Fig. 1. This latter then sends a message to confirm the abort. However, if this participant has already started the task (transition `Intentional` to `Execute`) but the initiator has not yet received the bound message, it refuses the abort. The switch will then canceled. Due to space limitations, we have made abstraction of these synchronization messages in our explanation. For details, see [10].

3 DynCNET Applied in Practice

In this section, we give a number of test results obtained from applying DynCNET in a simulated AGV transportation system. These tests where performed in the context of a joint R&D project between AgentWise and Egemin (http://emc2.egemin.com). The tests are performed on the map of an transportation system with 14 AGVs that is implemented by Egemin at EuroBaltic. We used a standard transport test profile that generates 140 transports with a random pick location and a random drop location per hour real time. Every simulation was run for 200.000 timesteps, corresponding to approximately 4 hours real time. Displayed test results are average values over 30 simulation runs. Error bars in the figures show the 95 % confidence interval for the test results.

We compare DynCNET with standard CNET and a field-based approach for task assignment (FiTA [9]). In FiTA, AGV agents are guided towards pick locations of transports by following the gradient of a field that combines attractive fields emitted by transport agents and repulsing fields emitted by other AGV agents.

Test Results. We discuss the test results of communication load and reaction time. To compare the communication load, we have measured the average number of messages sent per finished transport. The left part of Fig. 3 shows the results of the test. DynCNET requires (as FiTA) about twice the bandwidth of standard CNET. Average waiting time is expressed as the number of timesteps a transport has to wait before an AGV picks up the load. The right part of Fig. 3 shows the test results for average waiting time for transports. After four hours in real-time, this resulted in CNET having handled 380 transports, DynCNET having handled 467 transports, and FiTA 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.

Discussion. DynCNET has similar performance characteristics as FiTA and both outperform CNET, the cost is a doubling of required bandwidth. DynCNET allows to specify and reason about the behavior of the agents by means of common engineering diagrams such as interaction and state diagrams, such common engineering approaches are currently not available for FiTA. On the other hand, DynCNET requires explicit support to deal with message loss. In FiTA, the freshness of the received fields is taken into account toe determine the attraction an repulsion of fields giving older field values less importance.

4 Related Work

CNET was originally proposed by Smith and Davis [8] and is included in a FIPA-standard [2]. [3] describes a protocol that allows a bidder to place bids for multiple tasks.

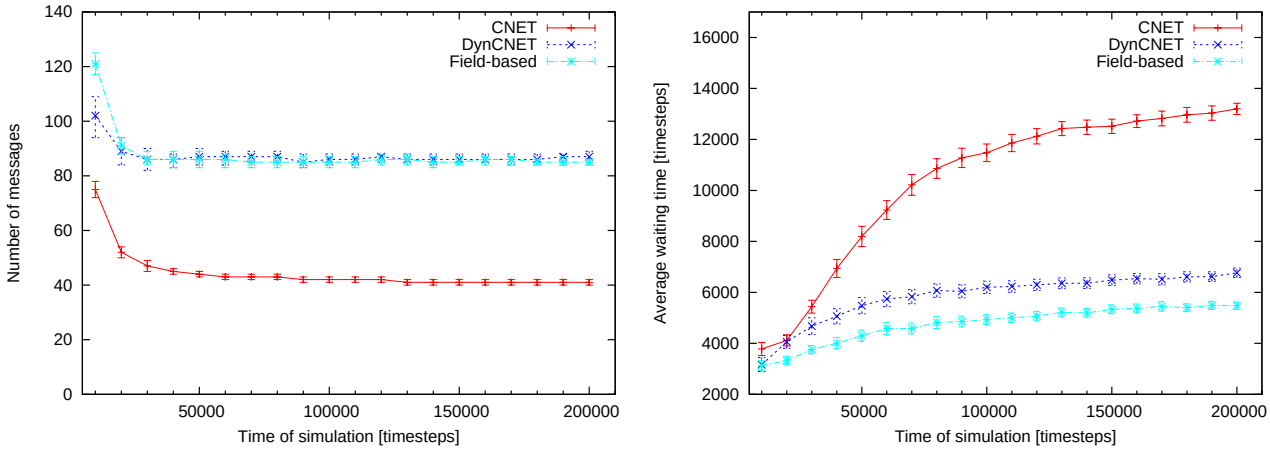

Figure 3. Left: amount of messages being sent per finished transport. Right: average waiting time

This protocol does not supports changes at both sides or delayed commencement.

The protocol introduced in [1] 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 FIPA Iterated CNET protocol allows multi-round iterative bidding [2]. Related are leveled-commitment contracts [6]. The idea of a leveled-commitment contract is that any of the agents in a contract can de-commit by paying an agreed penalty to the other agent(s) in the contract. Leveled-commitment contracts are typically intended for negotiation protocols between self-interesting agents.

Finally, several researchers combine CNET with other strategies to provide support for dynamism. [7, 4, 5] are examples that use a mediator pattern for dynamic scheduling and CNET to produce a schedule. These mechanisms allow for some flexibility, but task rescheduling is time consuming making the approaches more suitable for systems with limited dynamics.

5 Conclusions

In this paper, we presented DynCNET, a protocol for dynamic task assignment in situated MAS and we showed test results from a real-world test setting that compare the approach with standard CNET and a field based approach. DynCNET allows agents to switch the assignment of tasks dynamically. Our experiences show that DynCNET outperforms standard CNET, while it has similar performance characteristics as FiTA. There is an important difference between DynCNET and FiTA with respect to engineering support. DynCNET allows to use common engineering diagrams for design and reasoning, wheras no common engineering support is currently available for FiTA. On the other hand, DynCNET requires explicit support to deal with message loss while this is naturally supported in FiTA.

References

[1] S. Aknine, S. Pinson, and M. F. Shakun. An Extended Multi-Agent Negotiation Protocol. *Journal on Autonomous Agents and Multi-Agent Systems*, 8(4), 2004.

[2] FIPA TC Communication. FIPA Contract Net Protocol and Iterated Contract Net Protocol Specification, Doc. SC00029, Doc. SC00030, 2002.

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

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

[5] D. Ouelhadj, P. Cowling, and S. Petrovic. Intelligent systems design and applications. Springer-Verlag, 2003.

[6] T. Sandholm and V. Lesser. Levelled-commitment contracting: A backtracking instrument for multiagent systems. *AI Magazine*, 23(3):89100, 2002.

[7] W. Shen and D. Norrie. An agent-based approach for dynamic manufacturing scheduling. In *Agent-Based Manufacturing, Minneapolis*, 1998.

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

[9] 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.

[10] D. Weyns, N. Boucké, T. Holvoet, and B. Demarsin. *DynCNET: A Protocol for Flexible Transport Assignment in AGV Transportation Systems*. Technical Report CW 478, Katholieke Universiteit Leuven, Belgium, 2007. http://www.cs.kuleuven.ac.be/publicaties/rapporten/CW/2007/.

[11] D. Weyns, E. Steegmans, and T. Holvoet. Towards Active Perception in Situated Multi-Agent Systems. *Applied Artificial Intelligence*, 18(9-10):867–883, 2004.