# A Survey of Extensions to the Contract Net Protocol

Engin Bozdag (1119869)

Delft University of Technology,
EWI, Software Technology

**Abstract.** The Contract Net Protocol (CNP) is a widely used task allocation protocol in Multi-Agent Systems. It is fast, flexible and has low communication costs. However it is limited in some issues and has shortcomings if the setting for task assignment is more complicated. This paper addresses those limitations and summarizes extensions that has been published over the years. It also evaluates the protocol and its extensions with the criteria based on [1, 2].

## 1 Introduction

Coordination theory is defined as "a body of principles about how activities can be coordinated, and how actors can work together". The task assignment process forms an important part of the coordination and assigning tasks in a heterogeneous multi agent system (MAS) architecture is not simple. When agents have different capabilities and different subproblems require different capabilities, a protocol is needed in order to find the right agent for the right task in an efficient way. The assignment of tasks to agents and the (re-) allocation of tasks in a MAS is one of the key features of automated negotiation systems [3].

One possibility of solving this task assignment problem is using a centralized approach, where all *initiators* (agents with the tasks) and *participants* (agents that compete for acquiring tasks) send their information to a central decision maker. This decision maker will decide which participant will perform which task and then notifies it accordingly. According to Malone and Smith [4, 5], this centralized scheme has lower coordination costs, but is very vulnerable to agent failures.

Contract Net (CNet) is a well known high-level protocol that uses a decentralized scheme. In CNet, there is no central decision maker. Initiators (also known as managers) are responsible for monitoring the execution of a task and processing the results of its execution. Participants (also known as contractors) are responsible for the actual execution of the tasks. Agents in a MAS are not designated a priori as managers or contractors. These are only roles, and any node can take on either role dynamically during the course of problem solving.

The negotiation process of CNet begins by initiators announcing available tasks. Participants evaluate these tasks and submit bids on those for which they are suited. The initiators then evaluate the bids and award contracts to the nodes they determine to be most appropriate. Note that a participant may further partition a task and award contracts to other nodes, by acting as the initiator of that task. Control of the whole task assignment is distributed, because processing and communication are not focused at particular nodes [6].

CNet was proposed as early as 1980 and still widely used, but it has a lot of weak points. This paper discusses the limits of the original protocol and mentions several extensions that has been made through the years. It also evaluates the performance of the basic protocol and the extensions, based on a set of quality attributes which are taken from [1, 2].

The remainder of this paper is structured as follows: Section 2 describes the extensions Smith himself mentioned in the original paper [6]. Section 3 addresses the limitation of concurrent bids in the original protocol and describes a solution. Section 4 criticizes the need of agents committing to tasks before actually getting the task and introduces an improvement. Section 5 describes how CNet agents behave in case of exceptions and presents the alternative "Citizen" approach. Section 6 introduces a set of quality attributes which are given in [1, 2]. It then compares the basic protocol and the extensions based on the introduced attributes. Related work is discussed in Section 7, followed by the conclusions in Section 8.

## 2 Extensions suggested by Smith

Smith himself acknowledges that the basic negotiation mechanism has some limitations and mentions the following improvements:

- *Immediate Response Bids*: In the basic CNet protocol, a node cannot submit a bid until it goes idle and is actually ready to process a new contract. However with this strategy,an initiator cannot receive any bids if:
  - there are no idle nodes
  - no node is eligible for the task
  - or if all nodes rank the task too low.

  With a class of bid which is called **immediate response**, participants can notify the initiator why they are not responding. (BUSY, INELIGIBLE, or LOW RANKING). The initiator then can take the necessary action. This approach however does not seem very viable in open systems, where agents might be developed by different parties. Some agents might not comply with these immediate responses, or violate some of their underlying assumptions.

- *Directed contracts*: If the initiator knows a participant which is eligible for the task, he might award it to him directly. This is called **directed contract**. The initiator then sends a **directed award** message without making a task announcement. The participant can accept or refuse the task. This scheme enhances the efficiency of the protocol.

- *Request and Information Messages*: If a node requires a simple information, then a request-response sequence is used, instead of task announcement and bidding.

- *Node Available Messages*: If the net is heavily loaded, the unlimited use of task announcements can saturate the available communication channel. In this case, idle nodes can send **node available** messages, making themselves available for tasks. If an initiator has a task for this node (that is not yet awarded and eligibility requirement is met), then it might award the task. Smith does not define how a "heavy load" on the net is determined. This

might require a more or less global view of the system state, which is difficult to create without heavy bandwidth requirements.

The following sections present improvements made by other literature.

## 3 New conversational primitives

Aknine et al.[7] note that several initiators can concurrently call for bids, so a participant may have to manage several negotiation processes in parallel. Some applications of the CNet protocol force the participant to sequence their negotiation process (for example by preventing the agent to respond to multiple bids). This might cause the participants to miss some contracts. If there are delay failure detectors, it may force the managers to consider these participants as failures. There is also chance that participants may break their commitments, since the basic CNet protocol allows this. This might cause further delay, since initiators has to call again for bids to find new participants.

The solution they suggest for these problems is to introduce new conversational primitives. They replace basic CNet primitives (Bid,Accept and Reject) with new ones (*PreBid, PreAccept, PreReject, DefinitiveBid,DefinitiveAccept, DefinitiveReject*). In this scheme, an initiator announces a task, and participants submit their temporary bids (PreBid's). The initiator sends the best participant a temporary positive answer (PreAccept) and all the other participants a temporary negative answer (PreReject). The receiver of PreAccept message can send its DefinitiveBid. The receivers of PreReject can submit a better bid, they can send new PreBid's until they receive a DefinitiveReject from the initiator. When an initiator receives a DefinitiveAccept, it will evaluate this bid and compare it with new PreBids received by previously rejected participants. If one of those new PreBids is higher than the potential contractor's DefinitiveBid, or if this DefinitiveBid is less than the PreBid of one of the pre-rejected participants, then a new PreAccept message will be sent (to the participant with the better bid). Otherwise a DefinitiveAccept will be sent to the potential contractor and a DefinitiveReject to all the other participants.

This new scheme gives the participants the possibility to simultaneously bid to multiple tasks. It reduces the global negotiation processes between agents and the situations of decommitment of the contractors. According to the experiments carried by Aknine et al., time performance of this extension performs better than Smith's basic CNet protocol. For detailed information please see [7].

## 4 Contract Net with Confirmation Protocol

Knabe et al.[8] note that in every execution of the protocol all participating agents need to commit themselves to do the job, although only one of them will actually get the task awarded. Therefore, this procedure is not complete in that it will not compute solutions that could be found with better approaches.

Their solution Contract Net with Confirmation Protocol (CNCP) is similar to the one mentioned in Section 3. After receiving all the bids, the initiator arranges them in a sorted list and sends requests to participants starting with the best bid to find out if they can actually do the job. If the participant sends a refuse or a deadline has passed, the next participant is tried. This iteration goes on until a participant sends an *agree* message. Thus, instead of making the commitment in the bidding stage, the participant only commits at the time of sending the agree message. Even though this approach needs some overhead to repeatedly find the next best bid, it only requires one agent to make a single commitment, which is achieved by using the confirmation stage in the protocol. This way a participant can reply to all incoming calls from different initiators without the need of already allocating resources, before actually getting the task.[8]

## 5 Citizen Approach

According to Dellarocas et al. [9, 10], CNet protocol is prone to a wide range of potential exceptions. These include:

- *Agent Death:* If a node acting as a participant dies, its initiator will not receive the results it is expecting. In addition, if the agent has itself acted as an initiator (and subdivided the task), its participants will be "orphaned", their results for the task do not matter anymore. This cascading effect can go in deeper levels, if the participants also act as initiators. Finally, the matchmaker (an agent which indexes other agents by the skills they claim to have) will continue to offer the dead participant as a candidate, which will result in a wasted traffic.
- *Fraudulent participant/initiator:* A buggy or intentionally malicious node can weak havoc through fraudulent task announcement, bidding or assigning tasks.
- *Resource poaching:* Long-duration tasks might tie up the participants, thereby freezing out resourced needed for high-priority tasks that might come later.

Dellarocas et al. concentrate on the first exception, agent death. They note that the traditional way CNet deals with agent death is a "'survivalist"' approach. If no results are received by the deadline the participant promised, the initiator will re-start the task announcement process. This means that on every timeout, it will retry. This approach however will not eliminate "orphaned" tasks, and is prone to a "timeout cascade" effect, as it is mentioned before. It still does not solve the problem with the matchmaker.

Dellarocas offers a "citizen" solution,by analogy to the way exceptions are handled in human society. In such environments, citizens rely on a whole host of social institutions (the police, lawyers, etc) to handle most exceptions. This usually gives a good result, because such institutions are able, by virtue of specialized expertise and accepted legitimacy, to deal with exceptions more effectively than individual citizens. In the citizen approach, An Exception Handling (EH) service takes the role of the social institutions. When an agent joins the MAS, the EH

service begins periodic polling of the agent. If an agent dies or does not respond before the deadline, EH service takes a series of actions to resolve the problem:

- It notifies the matchmaker, so that the dead agent can be removed from the list of available participants
- If the dead agent was a participant and was performing task for an initiator, EH service notifies the initiator so that it can re-allocate the task
- If the dead agent was acting as an initiator, the EH service acts as a proxy and tries to find new initiators for the dead agent's participants. The proxy waits for a task announcement for the orphaned tasks, and submits a bid that is likely to be highly competitive, since the tasks were already in progress, so a nearby completion time can be given as bid. If the proxy wins the bid, it forwards the results to the new initiator. This scheme might not be suitable in some environments since task results might get obsolete quickly. In such cases,the EH service simply kills the orphaned tasks.

The EH service can also help avoiding the agent death problem by transparently filtering out the bids that come from the most failure-prone of the bidders, thereby reducing the probability that a task will be assigned to an agent that dies during the task execution.

Dellarocas ran a series of experiments to see if the citizen approach performs better than the survivalist approach. They tested different agent configurations by varying the number of tasks and participants. They ran the simulations for three cases:

- Failure-free environment (baseline case)
- Failure-prone environment, "survivalist" agents using timeout/retry.
- Failure-prone environment, "citizen" agents, supported by EH services

According to the results, citizen agents produced faster task completion times and a much smaller variation in task completion times than the survivalist agents. The difference in performance is significant for longer task chains. Dellarocas explains this by the increase in probability of multiple agent deaths in the same task tree. In the survivalist case, each death may trigger the "timeout cascade" effect. According to the results, introducing the proxy service has a positive effect in overall efficiency, as does the bid filtering. For a more detailed explanation, see [9, 10]

## 6 Comparison

There is few literature that defines attributes for comparison of different MAS. Bass et al.[2] described a couple of quality attributes, but they are given for general software systems, not all the elements there apply to MAS. Davidsson and Johansson on the other hand defined several abstract attributes in their case study concerning dynamic resource allocation. We have taken the following to be used in our comparison [1]:

1. *Responsiveness*: How long an initiator has to wait until it finds a suitable participator?
2. *Load balancing:* How evenly is the load balanced between the participants?
3. *Fairness:* Are all the tasks treated equally?
4. *Utilization of resources:* Are the available resources utilized as much as possible?
5. *Communication overhead:* How much extra communication is needed for the resource allocation?

Next to these attributes, we will consider the following, taken from [2], also mentioned in [1]:

- *Robustness:* How vulnerable is the system to node or link failures and how does it react to them? Can the system go into a deadlock?
- *Modifiability:* How easy is it to change the system after it is implemented (and often deployed)?
- *Scalability:* How good is the system at handling large numbers of users (initiators and participants)?

Now we will discuss all those attributes in the following subsections

### 6.1 Responsiveness

The task assignment process begins with the initiator sending an announcement to the participants. In the basic protocol, CNet might force participants to sequence their bids. Let us take the following example: We have two participants P1 and P2 and two initiators I1 and I2. P1 and P2 has both submitted a bid to I1. CNet does not enable P1 and P2 to bid for I2, until they receive an answer from I1. This is not optimal, since only one of them will receive the task. So, I2 has to wait for a while, even though there is an available participant out there. The problem gets worse if the number of initiators gets bigger. Of course the initiator might learn the reason of no replies by the use of "immediate response" suggested in Section 2, but that does not change the responsiveness criterion we have determined.

The improvements suggested in sections 3 and 4 however allow the participants to work with several initiators concurrently, which increases the responsiveness.

### 6.2 Load Balancing

In an heterogeneous MAS with low load, agents with better capability (thus lower execution time) will get the task. Agents with worse capability will not get it, unless better agent is already busy. This is certainly a problem with the basic CNet protocol, since agents have to wait the reply of their bid before bidding for another task.

### 6.3 Fairness

In the basic CNet protocol, an agent can break its commitments when itt receives an offer for a better task in comparison with those for which the agent is committed. This is not fair to the task which the agent was committed, the initiator now has to re-announce the task, so the waiting time will increase. With the improvements, commitment is postponed to a later stage, so it seems more fair.

### 6.4 Utilization of resources

As it's argued in [8], basic CNet protocol forces the participant to commit resources before it receives the task. If it does not receive the task, those resources are wasted. With the improvements, agents can reply to all incoming call for proposals without needing to already allocate the resources at this early stage of interaction.

### 6.5 Communication Overhead

As well as the original CNet protocol, the improvements mentioned on Section 3 and 4 require O(n) messages, where n denotes the number of participants. The improvements require more messages to be sent (definitely with Pre/Definitive scheme), but these do not cause serious consequences. These messages are sent in packages since agents have concurrent negotiations. There are no messages pending, since agents can interact with different initiators concurrently. This results in a shorter negotiation process, which is showed in [7]

There is a minor disadvantage of both improvements. In the original CNet protocol, the initiator has to find the maximum bid only once. With CNCP, some overhead is required to repeatedly find the next best bid. This is also the case in Pre/Definitive scheme, since PreBids and DefinitiveBids can change. However, this additional computational effort is several orders of magnitude lower than the effort for sending the messages. According to Aknine et al.[7], even though less messages is sent using the original protocol, the negotiation phase is significantly larger. This occurs due to sequential bidding scheme and pending messages.

### 6.6 Robustness

As it is mentioned in Section 5, the original protocol uses the timeout/retry scheme to deal with failures. This approach does not eliminate "orphaned" tasks, and is prone to a "timeout cascade" effect. According to the experimental results of Dellarocas et al. [9, 10], the citizen approach mentioned in Section 5 performs a lot better than the original CNet protocol, in a failure-prone environment.

According to Aknine [7], if there are no deadlines, deadlocks might occur with the original CNet protocol, which will block the system. Because of sequential bidding scheme, two initiators might wait bids from two different participants, until they decide who wins the task. However participants won't send any bid

to the other initiator until they know if they won the current task. This will clearly block the system if no deadline is introduced. This does not occur with the improved versions, since agents can bid concurrently for different tasks.

### 6.7 Modifiability

Dellarocas [9, 10] argues that modifying agents, for example in order to add exception handling, will greatly increase the burden on agent developers. This is difficult, because the agent environment may be difficult to anticipate. Making change in exception handling behavior is also difficult, since it requires coordinated changes in all agents, which might be developed by different parties.

Other than that, we could not locate an experiment questioning modifiability for CNet or another MAS protocol. Therefore we cannot analyze it any further.

### 6.8 Scalability

According to Knabe [8], in the basic CNet protocol with MAS of 100 initiators (each having one task) and 100 participants (each capable of performing one task), the chance of getting a bid accepted is circa 0.64. If other agents make more than one bid, the probability is even lower. So in more than one third of all cases, the available resources of the participant will be idle due to the basic CNet strategy. This is justified by the experiment performed by Aknine et al. [7]. If the number of agents increase, average waiting time for agents also increases.

Juhasz and Paul implemented a Java-based multi-agent task allocation testbed for the analysis of the CNET protocol. Their conclusion is that without deadline the protocol cannot be used for task assignment, and with deadline, the performance depends on the number of agents and the load of the agents. Under light agent load, the delay is linear in the number of agents. Under heavy load, the protocol often has to be performed repeatedly [11]. However this research is very primitive, and more research on this area is needed.

## 7 Related Work - Coalition Formation

Durfee mentions the possibility of an initiator decomposing the overall problem differently, in case it cannot find a suitable participant. According to Durfee, the relationship between problem decomposition and subproblem allocation is extremely complex and has not received sufficient attention [3].

Shehory and Kraus [12, 13] mentions a similar problem. They argue that there are cases in CNet where single participants cannot performs tasks by themselves and tasks cannot be partitioned, or the partition is computationally too complex. Their solution for such problems is assigning tasks to groups of agents. In some cases, this approach also improves the efficiency of the task execution when the performance of single agent is worse than the performance of a group.

Sandholm et al. [14] divides the activities of coalition formation into three categories:

– *Coalition structure generation:* Partitioning the set of agents into exhaustive and disjoint coalitions
– *Solving the optimization problem of each coalition:* Pooling the tasks and resources of the agents in the coalition, and solving their joint problem
– *Dividing the value of the generated solution among agents:*

The work of the authors above [12–15] all focuses on coalition structure generation. Shehory and Kraus shows that the general task allocation problem is computationally exponential (they are at least NP-complete). They approach the coalition formation problem as a Set Partitioning Problem (SPP) and note that the exact solutions for SPP are NP-Hard. They present polynomial-complexity algorithms that yield results which are close to the optimal results and bounded by a logarithmic ratio bound. The algorithms also use a distribution method which prevents most of the possibly overlapping calculations in a MAS.

## 8 Conclusions

Contract Net is a relatively old but widely used protocol in Multi Agent Systems.It is simple and fast, but is prone to deadlocks if no deadline is introduced, and definitely not optimal because of sequential bidding scheme. Luckily there has been improvements that removed some of the limitations of the original protocol. The mentioned improvements offer a more reliable and optimal performance. Since we did not have the necessary time, we could not test the algorithms using a framework such as SWARM [16] or JADE [17]. We tried to compare them in the light of available literature. An experiment should be performed in order to gain quantitative and comparable results.

## References


1. P. Davidsson, S. Johansson, Evaluating multi-agent system architectures: A case study concerning dynamic resource allocation (2002).
2. L. Bass, P. Clements, R. Kazman, Software Architecture in Practice, Addison Wesley, 2003.
3. E. H. Durfee, Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence, MIT Press, 2000, Ch. Distributed Problem Solving and Planning.
4. T. W. Malone, Modeling coordination in organizations and markets, Management Science 33 (10) (1987) 1317–1332, (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 151–158, Morgan Kaufmann, 1988.).
5. T. W. Malone, S. A. Smith, Modeling the performance of organizational structures, Operations Research 36 (1988) 484–497.
6. R. G. Smith, The contract net protocol: High level communication and control in a distributed problem solver, IEEE Transactions on Computers C-29 (1980) 1104–1113.
7. S. Aknine, S. Pinson, M. F. Shakun, An extended multi-agent negotiation protocol, Autonomous Agents and Multi-Agent Systems 8 (1) (2004) 5–45.
8. T. Knabe, M. Schillo, K. Fischer, Improvements to the fipa contract net protocol for performance increase and cascading applications (2002).
9. D. Klein, Rodriguez-Aguilar, Using domain-independent exception handling services to enable robust open multi-agent systems: The case of agent death, Autonomous Agents and Multi-Agent Systems 7.
10. C. Dellarocas, M. Klein, J. A. Rodriguez-Aguilar, An exception-handling architecture for open electronic marketplaces of contract net software agents, in: ACM Conference on Electronic Commerce, 2000, pp. 225–232.
11. Z. Juhasz, P. Paul, Scalability analysis of the contract net protocol (2002).
12. O. Shehory, S. Kraus, Task allocation via coalition formation among autonomous agents, in: IJCAI (1), 1995, pp. 655–661.
13. O. Shehory, S. Kraus, Methods for task allocation via agent coalition formation, Artificial Intelligence 101 (1–2) (1998) 165–200.
14. T. Sandholm, K. Larson, M. Andersson, O. Shehory, F. Tohme, Coalition structure generation with worst case guarantees, Artificial Intelligence 111 (1-2) (1999) 209–238.
15. T. W. Sandholm, V. R. Lesser, Coalition formation among bounded rational agents, in: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95), Montréal, Québec, Canada, 1995, pp. 662–671.
16. Swarm conceptual framework for agent-based models.
    URL http://www.swarm.org
17. Jade framework.
    URL http://sharon.cselt.it/projects/jade