selected for the general presentation of the contract net. We can describe an agent functioning in accordance with such a principle as *prudent*, for it minimises the risks of overheating but increases the risks of lay-offs. This technique is particularly interesting if there are not many agents submitting proposals and if the risks connected with local overheating are not very great.

(2) In contrast, by not taking account of proposals already submitted, an agent reserves no resources for the carrying out of tasks for which it submits proposals. It then risks finding itself with several contracts to fulfil, that is, with having too much work in relation to its capacities. The interesting thing about such an agent locally is that it is always more competitive, in relative terms, since it does not take account of the resources it will actually have to commit to other contracts. This method is obviously more advantageous than the previous one when there are a large number of agents likely to submit proposals, for the probability of being awarded a contract is lower. In order to resolve conflicts relating to the awarding of tasks, it is possible to arrange for a bidder which is awarded too many contracts to refuse tasks awarded to it. In this case, the manager must send out a new request for bids or, at the very least, re-evaluate the proposals which have been submitted to it and choose to sign a contract with another bidder. Our algorithm, incidentally, takes this eventuality into account.

(3) Finally, it is possible to find a middle way by using the principles of support for decision making in determining the prospects for each task. To do this, we have to associate two values with the tasks – the probability of obtaining a contract and the estimated profit. The probability of obtaining a contract corresponds to the probability of being appointed to carry out the task. For example, if an agent X estimates that a proposal has a 20% likelihood of being accepted, it associates it with the probability 0.2. This evaluation corresponds to the advantages it can envisage. On the basis of considerations of this order, an entire branch of research into multi-agent systems is working on the use of micro-economic theories based on the concepts of the market and of utility, together with the games theory (Rosenschein and Zlotkin, 1994).

Influence of sub-contractors

A problem which has never been tackled by authors working on contract nets, and which has generally been dealt with by very few authors, is that of sub-contractors, or, more generally, of the enrolment of agents†. When a bidder X carries out a task T under a contract with a manager A it may decide to break this task down into sub-tasks, and it may not be capable of carrying out certain of these sub-tasks itself. It then has to behave like a manager, and issue a new request for bids for the set of

†To our knowledge, the only author to have spoken explicitly about these enrolment problems in the context of the contract net is T. Bouron (1992).

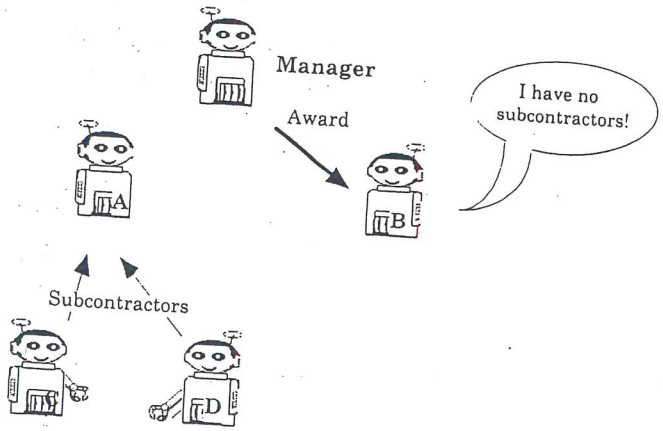

Figure 7.18 Problem of sub-contracting. B has no sub-contractor, whereas it has just been awarded the contract by the manager.

sub-tasks (T_1,..,T_n) stemming from the task T. A bidder confronted with such a situation can issue this request only after having received the contract, or else can ask sub-contractors to commit themselves to carrying out these tasks first, before submitting proposals to manager A. Obviously, this is a recursive problem, and sub-contractors may decide to look for sub-sub-contractors, and so on.

Let us take the example of the furniture removers, where there are two types of agent co-existing – the 'carrier' robots, which know how to carry furniture, and the 'supervisor' robots, which organise the carriers into teams, sign contracts with the managers, coordinate the actions of the carriers, and ensure that the work is of good quality. Here the supervisors are playing the role of bidders in relation to the task of moving house, and the carriers than of sub-contractors, whose tasks are limited to those connected with carrying furniture. If a manager issues a request for moving the contents of a room, the supervisors can either respond directly to the proposal, and then issue the request for bids, or make up their teams first and then respond.

(1) With the first solution, which we shall call *early commitment*, it is possible that the bidder (the supervisor in our example) will have problems if it does not find any sub-contractor (worker) that is free and capable of helping. Figure 7.18 shows the problem that two bidders may be confronted with if they apply to the same sub-contractor. Bidder B, which has just been awarded a contract by the manager, is incapable of carrying it out, for it has no sub-contractor to help it, whereas A, which does have sub-contractors, has not been awarded the contract.
The problem here stems from the fact that B submitted its proposal without knowing if it had sufficient resources to be able to execute the contract. It is

therefore stuck, and it has to break the contract with the manager, which obviously entails a lot of reorganisation. This type of allocation therefore functions well if resources are plentiful and if the risk of being without a sub-contractor is slight.

(2) The second solution obviously seems to be a correct solution to the preceding problem. To avoid not having a sub-contractor, it is sufficient to issue a sub-request (that is, a request for bids to find sub-contractors) before submitting a proposal. Any agent that cannot find sub-contractors will submit a proposal that will not attract much interest. If the opposite is true, a proposal can be submitted in complete confidence. We then say that the bidder is making a *late commitment*, since it first asks its sub-contractors to commit themselves to it, before it commits itself, in its turn, to the manager. But alas, this solution raises even more problems than the first, for it is necessary, first of all, to provide for a mechanism for managing the sub-contractors' commitments. If the proposal is not accepted by the manager, the sub-contractors must be informed that they are freed from their commitments. We are then confronted with problems similar to those we saw in the preceding section in relation to reserving capacity for tasks, and the solutions for resolving them are analogous. In addition, even if we assume that these problems are solved, we nevertheless find ourselves faced with a classic problem in resources management for distributed systems, the appearance of deadlocks when several processes have to access resources simultaneously – for example, if there are only six carrier robots for two supervisors, Bertram and Charles, and if, when the request for bids is received, the task requires four carriers. Each of the two supervisors may have bids from three carriers. In this case, neither Bertram nor Charles can respond positively to this request, whereas the number of workers is actually greater than that required for the task. The solution of such a problem involves the detection and the suppression of the conflicts producing these deadlocks. We shall deal with this problem in Section 7.3.4, which covers the general problem of the commitment of agents.

(3) A third solution exists, which is far from being ideal but which makes it possible to ensure a certain stability for the system and, above all, to avoid both deadlocks and non-execution of contracts. It consists simply in setting up relatively permanent teams of sub-contractors, which, for an agent, comes down to 'hiring' agents that it generally needs, and binding them to itself through a contract of exclusivity. The sub-contractors may respond only to requests from a single agent, their 'boss', refusing all work which does not come from this agent. In exchange, the boss undertakes to supply work for them. The establishment of these teams is therefore a more permanent affair than in the situations analysed above, the agents being part of a team for the implementation of a whole series of tasks. Team management is harder and easier at the same time than simply enrolling someone for a precise task. It is harder, for it is necessary to manage the hiring and firing of the agents, while trying to respond to requests with the teams already in existence. But it is also easier, for all the modifications are local – to a team – and so have no consequences for other teams. In the example of the furniture removers, if the number of workers for carrying out a task is relatively constant, it will be easier to set up a team of furniture removers, rather than to manage the conflicts arising with each contract in relation to the defining of groups of sub-contractors. It is also relatively easy to adapt the hiring pattern to fit the tasks required if the work is fairly similar. It is merely necessary to take on additional agents if the need arises, and to make agents redundant when the work needs fewer sub-contractors, using a fairly simple regulation system.

What can be learned from the contract net

The contract net offers a task allocation mechanism which brings great flexibility to the manner in which the agreement between the clients and the suppliers is managed. However, this is not a model for the resolution of problems, but only for a protocol for the organisation and dynamic distribution of work. Like all protocols, it has no bearing on the content of what is exchanged. In particular, it assumes that a common language exists for the description of the tasks and the data to be sent. If R. Smith was able to give us some indications on how to constitute such a language on the basis of the concept of the description of attributes (cf. Section 7.3.2), other options are open, but at all events remain the responsibility of the designer of a multi-agent system.

Advantages

The advantages of the contract net are obvious:

(1) We are dealing with an allocation mechanism which is simple to create and which can be designed on a distributed architecture. However, some precautions should be taken with regard to access to agents, the influence of multiple managers and the management of commitments.

(2) It can be very dynamic, as there is absolutely no need to call on any acquaintances management. An agent need only be capable of registering with (check-in) and leaving (check-out) the network in order to take account of the arrival of new agents and the disappearance of old ones.

(3) The task and the agent are matched up on the basis of a bilateral agreement between the client and the supplier, which makes it possible to take account of a large number of parameters, such as the agents' skills, the workload, the type of task to be carried out, the description of operations, the type of data to be supplied, the maximum authorised waiting period, the cost, the urgency, etc.

Drawbacks

Unfortunately, the contract net is not free from drawbacks:

(1) If the design is simple, the execution is difficult, and generates a large number of messages. In fact, the number M of messages for a net such as we have given in BRIC is in the order of:

$$M = N + 2\alpha N = (1 + 2\alpha)N$$

where α represents the ratio of the number of bidders interested in a request for bids to the total number of agents in a system. It will be seen that as soon as the ratio approaches 0.5, which means that half of the agents are interested in each request, the total number of messages is in the order of N^2. For this reason, the contract net can be applied only to small societies of agents in which only high-level tasks can be dealt with by requests for bids, in such a way that the transaction times are not too prohibitive in relation to execution times. And indeed, who would be interested in a system for which all the computation time was taken up by distribution procedures and which therefore did not do anything productive?

(2) We have also seen that the simplicity of the implementation is only on the surface and that, if we wish to benefit from a relatively high-performance allocation, either the number of requests for bids must be low or else a complex structure has to be implemented to manage the problems related to the parallelism of the processes. Incidentally, it is a shame that very little study has been done on performance evaluation for this type of allocation and on the theoretical and practical problems encountered in distributed implementation.

(3) Finally, when tasks can be broken down further, it is necessary for the agents to have a decision-making strategy which allows them to choose between early commitment, late commitment and commitment by hiring.

The contract net system can therefore be applied only to small-size nets carrying out tasks broken down into very small elements, and in this case it offers considerable advantages in terms of simplicity and dynamism, as witnessed by their application in industry. We may, in particular, refer to Parunak (1990) for an introduction to the Yams flexible workshop management system and to the Flavors Paint Shop described briefly in Chapter 1.

7.3.3 Variations and hybrid allocations

The mechanisms which we developed before correspond to 'pure' approaches, whether we are speaking of the contacts system approach or that of using a contract net. It is obviously possible either to create variations of one of these approaches or to merge them to try to combine their advantages.

Contract net guided by proposals

The classic form of the contract net is guided by requests. It is the managers that ask the potential bidders to submit proposals concerning the tasks which the managers would like to see carried out. As we have seen, this form of allocation is more particularly adapted to low load situations, in such a way as to diminish conflicts, as well as the large number of messages needed for transactions.

It is possible to invert contract net protocol so that suppliers alert potential clients to their capacity. In this case, the system is guided by the proposals, that is, by the availability of the bidders. Although it is frequently used in the case of task distribution in distributed operation systems, this approach has been studied only to a very limited extent in the context of multi-agent systems. Nevertheless, if the suppliers inform the clients of their skills and their availability, this technique then resembles mechanisms for managing direct contacts, since the manager then knows the set of agents likely to be of interest to it.

Synthesis of contract net and acquaintance network

Managing tasks through acquaintances or through a contract net involves the same problems. What is needed is a system that can operate without the system designer having to say explicitly who must do what. In other words, a system that can ensure that tasks will be carried out by competent agents and at the same time that will arrange and take into account transformations such as addition and deletion of agents or modification of agents' capacities.

In an acquaintance network, the data characterising the skills or the state of an agent can be accessed directly by the clients, which can then make decisions without having to broadcast general advertisements. In a pure contract net, in contrast, it is assumed that the agents have no *a priori* knowledge of other agents. It is therefore necessary to ask all those that are interested to indicate their skills by means of a proposal.

We have seen that the acquaintance network is decidedly faster than the contract net (especially in direct allocation), since all that is needed is to ask the agents whether they know how to do the work, without there being any necessity to issue a request for bids. However, the acquaintance network lacks dynamism, since any modification requires a network reorganisation phase. It is possible to try to merge these two approaches in several ways in order to obtain the best advantage from them. Several ways exist in which acquaintance networks and the contract net can be combined.

(1) The first consists of using the acquaintances system for 'small' tasks and the contract net for 'big' tasks. We have seen that the number one problem for the contract net relates to the number of messages, together with the possible waiting times before the manager can evaluate the proposals and award the contract. When large-scale tasks are involved, these waiting times can be accepted. But the same is not true for elementary tasks, which require only

a limited time to be carried out. In that case, we can use an acquaintance network for all requests that do not call for too much work (and an allocation error is not too serious, since the tasks do not take much time) and reserve the requests for bids for the more large-scale tasks.

(2) In the second an acquaintance network can learn by using requests for bids every time this is necessary. For example, this can happen if there is a problem in the acquaintance network and a client does not find a supplier corresponding to what it wants, or again if new agents wish to enter the network. A contract net protocol can then be used to build and modify the acquaintance relationships.

Finally, a system of acquaintances can also be seen as a kind of cache memory of the contract net, or again as a compiled structure of the latter. The acquaintances act as a cache memory. Accessing them is faster, since it is no longer necessary to question the acquaintances about their capabilities but as a counterpart there must be a suitable way of managing any lack of coherence between acquaintances and agents which can arise, in particular, during reorganisations of the network. This is also a compiled version of the contract net, which makes it possible to avoid the request for bids and the cost of sending the associated messages. In this case, the reorganisation of an acquaintance network can be compared to a recompilation of a previously compiled structure.

7.3.4 Contracts and commitments

Concept of contract

A contract is defined as 'an agreement, by means of which one or more persons undertake to give, to do, or not to do, something'. This definition can just as easily be applied to contracts between agents. The supplier commits itself to a client to do everything possible to carry out the task, the attribution of which it has accepted, and the client commits itself not to propose the same task to another agent (even if sometimes this commitment is not always carried through). As we saw in Chapter 5, a set of commitments forms dependency structures. The fact that A can carry out its contract itself depends on another contract which B has accepted. This set of dependencies constitutes a network which structures collective action and enables cognitive agents to accomplish tasks by assuming that their suppliers will actually carry out the tasks to which they have committed themselves.

End of contract

A contract goes through a series of states, of which the main ones are as follows:

(1) Initially, it is *in the course of allocation*. Whether we are dealing with a contract net or a system of allocation through acquaintances, no one has yet been selected to carry out the task, and the contract has therefore not yet been concluded.

(2) The fact that the supplier agrees to carry out a task commits the latter to doing it, and so the contract is *concluded*.

(3) If all goes well, the task is carried out normally and the supplier reports back that the task has been properly completed, and the contract is then in the *executed* state.

(4) If, by contrast, an unforeseen event occurs during the execution of the contract because the supplier cannot fulfil its commitments, the contract is *broken*. In this case, the agent sends an `Impossible` message to its client.

What can the client do when a contract is broken? It is presented with several alternatives, and choosing one rather than another depends on its capacities, and on the nature of the task, T.

- If we are dealing with a task of small importance, the client for T may decide purely and simply to abandon it.
- If the task is important, and forms part of a more important task T_1, the client for T can (a) try to find another supplier, or else (b) tell the client for T_1 that it cannot execute T_1 either, which risks leading to a veritable cascade of broken contracts.

Breaking contracts can also lead to penalties. Thus, in the case of an acquaintance network, the client for T can delete this supplier from its acquaintances having the skills necessary for carrying out T. In a contract net, these penalties will compel the supplier to submit fewer bids in future for the execution of any tasks like T.

7.4 Integrating tasks and mental states

7.4.1 The SAM system

How can we integrate mental states, such as intentions and commitments, and allocation algorithms such as the contract net or acquaintances management? In order to answer this question, let us study the case of a system, SAM (Social Agent Model), developed by Thierry Bouron (1992). The interesting thing about his work is that it demonstrated the importance of collaboration mechanisms for the performance of the group. He applied his model to the very classical example of the pursuit by predators of prey, in which predators have to surround the prey animals, which flee by moving at random, as set out in Chapter 1. All the difficulties relate to the design of the predators and, in particular, to the way in which they cooperate to catch prey animals. So the idea is to design different cooperation strategies and evaluate them in relation to the performance levels for captures. In particular, his strategies consist of defining ephemeral *associations*, which last only for the time required to carry out the overall task.