Chapter 6
Dynamic Resource Allocation

The previous chapter described the protocol for allocation of tasks to agents within an organization. When computational load increases, existing agents in an organization may not be able to complete all the goals in time. Hence additional agents may be needed. On the other hand, a drop in the computational load will result in idle agents and lead to inefficient utilization of resources. These two events can occur simultaneously in different organizations of the multi-agent system and cause degradation in its performance.

In order to overcome this situation, the idle agents need to be allocated to the organizations that need them most. What is therefore required is a mechanism for dynamic allocation of agents across the organizations. Section 6.1 highlights the issues not addressed by the existing methods that deal with computational overloads, and shows how they are handled in TRACE. Section 6.2 describes the resource allocation problem. Section 6.3 presents an introduction to market oriented agents. Section 6.4 describes two approaches (price oriented and resource oriented) to market based resource allocation. Section 6.5 gives the details of the proposed method for resource allocation. Section 6.6 reports the results of our simulation experiments. Finally Section 6.7 presents the conclusions.

6.1 Related Work

Execution-time adaptation has been reported earlier in literature [76,77,99,137]. This was described in Chapter 3. For example Ishida et al [137] propose two reorganization primitives, decomposition and composition for reorganization of multi-agent production systems. Decomposition divides one agent into two and composition combines two agents into one. Decomposition is triggered when the problem solving demand on the
system exceeds its ability to respond. Composition is performed when under-utilized resources can be released for use by other systems. Triggering of these primitives changes the population of agents and the distribution of knowledge in the multi-agent system.

Our initial work on handling load variations and efficient utilization of resources is presented in [121,123]. This framework, called AASMAn, integrated the contract net protocol with the decomposition and composition primitives described above.

In RETSINA (Reusable Task Structure-based Intelligent Network Agents), developed by Decker and Sycara [72,76,77,99] execution-time adaptation is handled by means of agent cloning. When an agent becomes overloaded, it creates a new agent that is a clone of itself. The clone is set up to use the resources of another processor.

Another solution suggested by researchers is the mobile agent paradigm [146]. A mobile agent is a program that acts on behalf of a user or another program and is able to migrate from host to host on a network under its own control. The agent chooses when and where it will migrate, and may interrupt its own execution and continue elsewhere on the network. The agent returns results and messages in an asynchronous fashion.

The aim of these methods is to balance load in order to improve system performance. However, these solutions firstly do not address the details of how resources required to perform decomposition or cloning or agent migration are made available if there are multiple requests for a single resource. A fair allocation is defined as one in which resources are allocated to organizations in direct proportion to their need. This need is reflected in the priority of tasks, for the execution of which these resources are required. This issue of fairness of resource allocation is crucial, especially in the case of time constrained domains which require allocation of resources to the most critical tasks.

The second limitation of these approaches is that they require every agent in the MAS to individually carry out the necessary reasoning for performing decomposition,
cloning, or agent migration. The result is inefficient utilization of resources. This is because in situations where a group of agents in the multi-agent system cooperatively solve problems, the need for additional resources needs to be determined for the entire group and not for individual agents of the group.

The proposed resource allocation protocol aims at overcoming these limitations. The first issue is addressed by making use of the economic approach for resource allocation. This simplifies the task of guaranteeing fair allocation of resources. The second limitation is overcome by using one resource manager agent per organization, which determines the resource needs of its entire organization and correspondingly allocates resources.

6.2 Reosurce Allocation Problem

The protocol described in the previous chapter is designed only to perform task allocation given a set of tasks and agents. A mechanism is needed for dynamically allocating agents to organizations based on their problem solving demand. We develop such a mechanism (TRACE-RAP) and combine it with the task allocation protocol in order to obtain a truly adaptive multi-agent system.

Basically, what we have is a market like situation where there is a demand for services (problem solving requests) and agents in the organization supply the required services. Supply and demand are used as the two economic forces to determine the amount of a resource or the supply of services that are provided and its price. Under normal conditions the demand can be met by the organizations. However, occasionally there may be unpredictable changes in demand, requiring either additional supply of resources or resulting in surplus resources. This increase or decrease in the supply of services/resources, is provided by a proportionate increase or decrease in the number of agents. Our approach is to change the number of agents in the organizations and the knowledge possessed by these agents and thereby reorganize the multi-agent system.
As the number of agents required is not known a priori, each organization initially starts with a minimum number of permanent agents and a resource manager. Whenever additional agents are required, they are obtained by buying them from markets set up by the resource managers; one resource manager per organization. These resource managers have a set of marketable agents that they wish to sell to the organizations in need of them. The resource managers keep track of the requirements of their respective organizations and dynamically determine how to sell the marketable agents.

The main objective of the resource manager is to sell agents to organizations that require them most (the ones that are executing higher priority tasks). Section 6.4 describes in detail how these markets operate. The agents that are bought from a market are allocated tasks by permanent agents of the organization using the task allocation protocol. The marketable agents serve as contractors or team members and share some of the computational load of the organizations.

### 6.3 Market Oriented Agents

Various approaches to resource allocation in distributed systems were described in Chapter 3. Of these methods, the one gaining increasing currency is that of a collection of distributed agents as an economic system. Projects that have applied market mechanisms to problems in distributed resource allocation include [14,25,68]. In this approach, agents are participants in a distributed computational economy, interacting in the market to further their own interests. Behaviors are described in standard economic terms of production, consumption, bidding and exchange.

Wellman suggests that in order to take an economic approach, typically invokes three premises [90]. His first premise is that the fundamental problem to be solved is one of *resource allocation*. Second, that it is useful to model behavior in terms of *rationality abstraction*. And third, that it is essential to consider how authority and activity may be *decentralized*. Each of these is described below.
6.3.1. Resource Allocation

A computer can be viewed as a decision machine, where a decision is about choosing from among potential courses of action to solve a given problem. Every decision - hence every computation - is about resource allocation. Choosing to do something entails an allocation of attention and other activity resources to do that thing in lieu of others. Conversely, an allocation of resources defines the activities done and not done. Making such choices appropriately involves weighing the benefits of the activity done against the opportunity cost of foregoing those not done.

Every problem, including ours, can therefore be cast as one of resource allocation. The advantage being that, without considering resources explicitly, it is difficult to express the range of courses of action available, as defined by configurations of resources devoted to the various activities. More importantly, without acknowledging gradations in value, it is impossible to account for tradeoffs among alternate activities [90].

6.3.2 Rationality Abstraction

Economic theory assumes that individual agents are rational, acting so as to achieve their most preferred outcome, subject to their knowledge and capabilities. This approach is similar to much work in AI. Newell [2] proposed that a central characteristic of AI practice is a particular abstraction level at which we interpret the behavior of computing machines. Viewing a system at Newell’s knowledge level entails attributing to the system, knowledge, goals, and available actions and predicting its behavior based on a principle of rationality that specifies how these elements dictate action selection. Newell’s rationality principle is:

*If an agent has knowledge that one of its actions will lead to one of its goals, then the agent will select that action.*

This formulation relegates all matters of resource allocation and graded preferences to some ad hoc auxiliary principles. From the economic perspective, a satisfactory
comprehensive rationality principle should address choice among alternate activities and resource allocations that accomplish goals to varying degrees. Thus, some coherence based rationality principle is required to make sense of the sorts of agent attitudes - knowledge, belief, preference, intention - commonly used in multi-agent system research.

### 6.3.3 Decentralization

Within economics, the problem of synthesizing an interaction protocol via which rational agents achieve a socially desirable end is called *mechanism design*. This is exactly the problem we face in designing multi-agent systems.

As all the three premises hold good for our problem we exploit existing economic ideas. The advantage of using economic principles for resource allocation is that many ideas and results from economics can be directly applied instead of developing new theories. These methods have also been found to possess the properties of feasibility, monotonicity (i.e., its quality of solutions improves with time) and fast convergence and can therefore be used in the development of real time distributed systems [8,51,68].

### 6.4 Market Based Approaches to Resource Allocation

In our human society, resource allocations are in most cases performed through markets. This occurs on many different levels and in many different scales, from our daily grocery shopping to large trades between big companies and / or nations. The market approach to resource allocation in human society has inspired the multi-agent system community to construct similar concepts for multi-agent systems, where trade is performed between computational agents on *computational markets*. Wellman refers to this as *market oriented programming* [89].

In computational markets, a common approach is to use a mechanism that obtains *general equilibrium*. General equilibrium is obtained when a set of prices (one price for
each commodity) is found such that supply meets demand for each commodity and where the agents optimize their use of resource at the current price level. In virtually all multi-agent systems there exist some scarce resources. Thus the issue of resource allocation is of fundamental importance. Two basic microeconomic approaches towards developing distributed resource allocation mechanisms based on general equilibrium theory are price directed approach and resource directed approach [8,38].

### 6.4.1 Price Directed Approach

In the price directed approach [8], an initial allocation of resources is made and an arbitrary set of system wide initial prices is chosen. Prices are then iteratively changed to accommodate the demands for resources until the total demand for a resource exactly equals the amount available. At this point, the resulting final allocation of resources is pareto optimal. Pareto optimum condition is one in which no one agent can be made better off without making someone else worse off.

The market equilibrium is given by [8]

$$ z(p) = 0 \quad (1) $$

where $z(p) = [z(p_1), z(p_2), \ldots, z(p_k)]$ being the aggregate excess demand for commodity $i$ $p = [p_1, p_2, \ldots, p_k]$ where $p_i$ is the price for commodity $i$, and $k$ is the number of commodities. The aggregate excess demand for commodity $i$, at price $p_i$, is the sum of the supply and demand of all agents, i.e.

$$ z(p_i) = \sum z_a(p_i) $$

where $z_a(p_i)$ is the demand of agent $a$ for resource $i$ at price $p_i$. The demand of an agent describes how much an agent is willing to buy (or sell - a negative demand) at a specific price level. In the price-oriented scheme the price vector is updated iteratively, until equation 1 is fulfilled. Since prices are only relative, $p_k$ is set equal to 1 and only $k$-
Chapter 6. Dynamic Resource Allocation

1 elements are searched in the price vector. Inputs to this scheme are the respective net demands of each agent, \( z_a(p) \), where \( a \) denotes an agent. WALRAS, developed by Wellman [89], is a prototype environment for specifying and simulating computational markets.

6.4.2 Resource Directed Approach

An alternative way to express the general equilibrium is to define it as an allocation such that, for each commodity, each agent's marginal utility is the same for an additional amount of resource. In this approach [37,38,68], during each iteration, each agent computes the marginal value of each resource it requires given its current allocation of resources (i.e. computes the partial derivative of its utility function - performance) with respect to that resource, evaluated at the current allocation level. These marginal values are then sent to other agents requiring use of this resource. The allocation of the resource is then changed such that agents with an above average marginal utility receive more of this resource and agents with a below average marginal utility are allocated less of the resource. When analytic formulas are used to compute the performance realized by a given resource allocation, an actual reallocation need not (but may) take place immediately after each iteration; an agent may simply compute its new allocation at each iteration and the resources may then be allocated whenever the algorithm is terminated. In the case that actual measurements are used, however, resources must be immediately reallocated in order for each agent to measure its performance under the new allocation. Applications like the distributed file allocation problem [68], and power load management [38] use the resource directed approach to develop computational markets.

6.4.3 Discussion

The differences between these two approaches are that firstly, their inputs are different. In the price directed scheme, demand is the input and in the resource directed case the inputs are some derivatives of the utility function. In standard micro-economic theory,
the utility function is the primary concept and the demand is derived from the utility function. Another important difference between the price directed algorithm and resource directed algorithm is the \textit{fairness versus feasibility} of the solution they produce [8]. For the price oriented algorithm to converge in reasonable time, the termination condition is \((z(p) < \varepsilon)\) instead of \((z(p) = 0)\). For the resource oriented algorithm to converge, the termination condition is (marginal utility < \(\varepsilon\)) for all agents. In the resource oriented case this means that the allocation is not perfectly \textit{fair}, i.e., some agents pay less than they would have done on a perfect market (marginal utility = 0 for all agents), while others will pay more. On the other hand, in the price-oriented case, the allocation is not perfectly \textit{feasible} (the total amount of resources allocated equals the amount available). In this thesis, we focus on price oriented algorithm in order to obtain a fair allocation.

\section*{6.5 Resource Allocation Protocol}

The multi-agent system organization for TRACE was described in Chapter 4. The system consists of a collection of organizations that in turn consist of a set of agents that cooperate with each other to achieve goals. The agents within an organization exhibit team rationality by obeying the joint responsibility code of conduct for joint actions [96].

Initially it is only the permanent agents that comprise an organization. As problem-solving activity progresses, organizations go through variations in load. In the event of a computational overload, the TAP accommodates high priority tasks by decommitting low priority ones. In order to minimize these lost requests, marketable agents need to be allocated dynamically to the organizations in accordance with their computational loads. This allocation of resources, which results in reorganization of the multi-agent system, is done by the resource allocation protocol (RAP). We assume that requests that are once decommited will be requested again. The RAP reorganizes the multi-agent system so that these decommited requests can be honored when they arrive again.
Chapter 6. Dynamic Resource Allocation

The resource manager obtains the resource needs of an organization from its permanent agents, and on the basis of this information, arrives at a suitable allocation of marketable agents. As the number of marketable agents is fixed, and multiple organizations could be contending for these agents, an allocation is arrived at on the basis of the criticality of decommited tasks, for the execution of which these agents are required. The permanent agents of an organization convey information about the criticality of decommited tasks indirectly by contributing some funds to the resource manager; the more the criticality of decommitments, the higher the contribution of funds. The permanent agents also specify how many additional agents ($\mu$) would be required by the organization. The method used for obtaining $\mu$ is explained in Section 6.5.1. Thus the contribution of funds made by an organization indicates the maximum price that the organization is willing to pay in order to buy $\mu$ marketable agents. The contribution of funds varies from organization to organization and reflects their relative needs for additional resources. The organizations that offer more funds per agent are considered to be more in need of resources than the ones offering less.

The resource managers periodically determine the resource needs of their respective organizations and accordingly conduct reorganization. Each such period is called a reorganization cycle. Thus it is not necessary for every agent of an organization to participate in the process of determining a suitable allocation.

We therefore have the multi agent system organized as a market economy composed of interacting buyers and sellers. The commodities in this economy are processing resources (marketable agents) required to achieve goals. Buyers are organizations that wish to purchase new agents in order to perform some computation. Sellers are the resource managers that wish to sell the marketable agents for the duration of one reorganization cycle. In this economy, monetary funds encapsulate resource rights, and price equates the supply and demand of processing resources. The buyers and sellers execute a resource allocation protocol to arrive at an optimal allocation of resources. Reallocation of resources is done at the beginning of every reorganization cycle and
results in a reorganization (change in the number of agents in the organizations, their communication structure and the distribution of knowledge) of the multi-agent system. For reallocation to be completed, each resource manager goes through the following steps:

1. Collects statistics from agents of its organization.

2. Computes the equilibrium allocation.

3. Notifies permanent agents of its organization about the new allocation.

There are two possibilities with regard to the kind of resources to be sold. The resources could all be of the same kind (homogenous) or there could be resources of different kinds (heterogeneous). In terms of the agent architecture described in Chapter 4, agents are said to be homogenous if all of them have the same kind of goal processor. In this case every agent has the capability to execute every task required by its organization (if it has the recipe). On the other hand, agents are considered heterogeneous if they possess different kinds of goal processors. The capabilities of all the agents are therefore not the same in this case.

The allocation of homogenous resources is described first and then this is extended to handle heterogeneous resources.

6.5.1 Allocation of Homogenous Resources

In order for proper allocation to take place, the requirement for resources in any reorganization cycle is determined on the basis of the information about the previous one. All agents convey the following four items of information (about the previous reorganization cycle) to their respective resource managers at the beginning of every reorganization cycle:

1. Information about the number of decommitments (D)
Chapter 6. Dynamic Resource Allocation

This information is sent by the permanent agents because only they act as organizers and keep track of the number of decommitments. It is assumed that the demand for new agents, \( \mu \), in an organization can be easily computed from the number of decommitments (D) made by the organization. If an agent is capable of completing, on an average, G goals per reorganization cycle, the number of new agents that are required is D/G. As the requests that are decommited by the task allocation protocol are the ones that have low priority, they are very much likely to occur again. Consider the example where agents are involved in information handling for users. User requests that are decommited are the low priority ones and there is a high probability of the user resubmitting his previously rejected request. Thus the number of decommitments can be used as a reasonable measure of resource requirements, and the type of requests decommited provide information about the required capabilities. Based on this information about decommitments, new agents are introduced into an organization that have the capability to take on the decommited goals when they are requested again.

2. Information about the decommited goals.

This information is also sent by the permanent agents and is used for dynamic distribution of domain knowledge to agents. The type of requests decommited provide information about the required capabilities. New agents are introduced into an organization after transferring the domain knowledge (i.e. the recipes required for executing the decommited goals) to them. The transfer is made by the resource manager which has the complete domain knowledge for executing all the goals required by its organization. These new agents can therefore take on the previously decommited goals when they are requested again.

3. Information about the idle time.

This information is conveyed by all marketable agents (because only these agents can be reallocated, not the permanent ones) and helps in identifying idle agents.
Marketable agents that remain idle for more than fifty percent of the reorganization cycle time can be treated as superfluous and considered for allocation to some other organization in need of them. Thus if an organization is allocated $X$ marketable agents in a cycle, has $D$ (from item 1) equal to zero for that cycle, and has $Y$ agents that remain idle most of the time, then the number of agents it requires for the next cycle, $\mu$, is taken to be $X-Y$.

4. Information about the contribution of funds ($F$).

Permanent agents in an organization contribute funds to their resource manager in every reorganization cycle. The sum of these values for all permanent agents indicates the maximum the organization is willing to pay for buying $\mu$ (obtained from item 1 or 3) additional agents.

The funds contributed by different organizations reflect their relative needs. The organizations that contribute more are deemed to be more in need of additional agents than the ones contributing less. The allocation of resources is done on the basis of the amount of funds contributed. Thus the organization that makes the highest contribution is allocated resources first.

We assume that the amount of funds to be contributed is determined by the application. The monetary funding units are used as an abstract form of priority in the multi-agent system. It is the applications burden to ensure that important computations are well funded.

Each resource manager conducts markets on behalf of high level applications and intimates the equilibrium price of a marketable agent to the permanent agents of its organization. On the basis of its funding, the equilibrium price and the number of decommitments in any cycle, the application can determine how much to contribute for the next cycle. We feel that a high level application should not be encumbered with decision making at the low level market mechanisms that locate and purchase the resources necessary for its execution. At the same time, however, it should be possible
for an application to exert some control over the general allocation of funds. The proposed method provides a uniform mechanism with these capabilities.

Every resource manager encapsulates details about the funds and demand for new agents for its organization and communicates this information to every other resource manager of the multi-agent system. The protocol consists of this communication step followed by a local computation by each resource manager. Each resource manager locally computes the equilibrium price of an agent. The demand of organization \( a \) at price \( p \), \( z_a(p) \), indicates how much it is willing to buy at price \( p \). The total demand for agents across all the organizations is \( \sum_{a=1}^{n} z_a(p) \), where \( n \) is the number of organizations in the multi-agent system. Let \( s(p) \) be the supply of marketable agents at price \( p \). The market will be in equilibrium when \( p \) has a value such that

\[
\sum_{a=1}^{n} z_a(p) = s(p)
\]  

In order to find this price, the resource managers initialize \( p \) to the maximum of prices offered by all the organizations. This price is then iteratively changed till equation 2 is satisfied. The number of iterations can however be reduced by using an approximation condition of the form

\[
\left| \sum_{a=1}^{n} z_a(p) - s(p) \right| \leq \varepsilon
\]

where \( \sum_{a=1}^{n} z_a(p) - s(p) \) denotes the aggregate excess demand

### 6.5.2 Equilibrium Price Computation

A more precise statement of the computation is now given. We use the following notation
Chapter 6. Dynamic Resource Allocation

\( F_i \) - denotes the contribution of funds made by the organization \( / \).

\( \text{minprice} \) - indicates the minimum price at which the resource managers can sell the marketable agents.

The demand for marketable agents at organization \( a \) at price \( p \) is given by the function \( z_a(p) \), which is defined as

\[
  z_a(p) = \begin{cases} 
    \mu_a & \text{if } p < \frac{F_a}{\mu_a} \\
    \frac{F_a}{p} & \text{otherwise.}
  \end{cases}
\]

The total supply of marketable agents at price \( p \) is given by the function \( s(p) \), which is defined as

\[
  s(p) = \begin{cases} 
    \text{total number of marketable agents in the MAS} & \text{if } p \geq \text{minprice} \\
    0 & \text{otherwise.}
  \end{cases}
\]

The market is in equilibrium when

\[
  \sum_{a=1}^{n} z_a(p) - s(p) \leq \varepsilon.
\]

Each resource manager goes through the following computations:

1. Communicates \( F \) and \( m \) for its organization to every other resource manager.
2. Initialization
   - The equilibrium price \( p \) is initialized to the maximum of prices offered by all the organizations and the excess demand \( z(p) \) at price \( p \) is evaluated.
3. Iteration
   a) while \( (z(p) > \varepsilon) \) and \( (p > \text{minprice}) \) do
   b) decrement price by a small amount \( p' \). this is referred to as the step size parameter, \( p' = p - p' \)

Once the equilibrium price is determined, an optimal allocation of resources is found. The resource managers then notify all permanent agents of their organization about the new allocation. The permanent agents accordingly update their organizational knowledge.
6.5.3 Allocation or Heterogeneous Resources

The above algorithm can be easily extended to perform allocation of resources that are heterogeneous. Let there be \( k \) types of resources, and \( p = \{ p_1, p_2, ..., p_k \} \) be the price vector, where \( p_i \) denotes the price of resource \( i \). The \( a^{th} \) organizations demand for resource \( i \) at price \( p_i \), \( z_a(p_i) \), describes how much of resource \( i \) the organization will buy at price \( p_i \). The total demand for resource \( i \) across all the organizations is \( \sum_{a=1}^{n} z_a(p_i) \), where \( n \) is the number of organizations in the multi-agent system. The price of resource \( i \) should be fixed such that the total demand equals the supply, i.e.,

\[
\sum_{a=1}^{n} z_a(p_i) = S_i(p_i) \quad (3)
\]

The market will be in equilibrium when a price vector is found for which the above equation is satisfied for all types of resources, i.e.,

\[
\sum_{a=1}^{n} z_a(p_i) = S_i(p_i) \quad \text{for } i = 1..k
\]

Thus the prices in the price vector need to be iteratively changed till equation 3 is satisfied for all types of resources. However an approximation of the form

\[
|\sum_{a=1}^{n} z_a(p_i) - S_i(p_i)| \leq \varepsilon
\]

can be used to terminate the iterations in reasonable time.

In order to achieve equilibrium, every agent of an organization conveys the same information (items 1, 2, 3 and 4) as in the case of homogeneous resources to the resource manager.
From items 1 and 3 the resource manager determines the number of agents of each type required by its organization. The amount of funds indicated in item 4 is the total contribution made for buying all types of agents. This total contribution now needs to be split among the different types of agents. The funds are split in the combined ratio of the minimum prices for these agents and the number of agents required of each type.

If $T$ is the total contribution of funds and two types of agents $A_1$ and $A_2$ are required, then,

 allocation of funds for buying $A_1$, $F_{A1} = \frac{T \cdot p_a \cdot \mu_{A1}}{p_a \cdot \mu_{A1} + p_b \cdot \mu_{A2}}$ and

 allocation of funds for buying $A_2$, $F_{A2} = \frac{T \cdot p_b \cdot \mu_{A2}}{p_a \cdot \mu_{A1} + p_b \cdot \mu_{A2}}$.

where $p_a$ and $p_b$ are the minimum prices at which the resource managers can sell $A_1$ and $A_2$ and $\mu_{A1}$ and $\mu_{A2}$ are the number of agents of type $A_1$ and $A_2$ respectively.

In this way the resource managers obtain the information about the requirement for different types of agents, the number of agents of each type, and the funds associated with each type of agent. The above algorithm (for homogenous resources) can now be applied to each type of resource.

### 6.5.4 Dynamic Distribution of Knowledge

The protocol described above allocates marketable agents to organizations. These agents can however lack the domain knowledge required to take on goals of an organization. This means that such agents need to be first endowed with the required knowledge before they are allocated to an organization. In addition to managing resource allocation, the resource manager also does the job of allocating this knowledge (recipes) to the new agents. We assume the knowledge to be available in a form that facilitates this kind of distribution.

The resource manager possesses all the knowledge required by its organization. Out of this entire knowledge only a selected portion is allocated to the new agent. In order to
determine this portion the resource manager obtains information about the decommited goals from all agents of its organization. The domain knowledge required to execute these goals is then transferred to the incoming agent.

This kind of dynamic distribution of knowledge enables effective use of available computational resources; an agent that is idle but lacks knowledge required to execute goals can acquire that information as indicated above. This also means that agents do not have to be preloaded with extensive amounts of knowledge which may or may not prove useful.

6.6 Experiments

In order to evaluate the effectiveness of the proposed mechanism a number of experiments were carried out. These serve to quantify its ability to

- *Reduce the number of decommitments.*
- *Fairly distribute resources among competing goals.*
- *Adapt to changes in computational load by reorganizing the multi-agent system.*
- *Make effective use of resources.*

In addition to this the MAS possesses the property of openness. New agents can be added to an organization by advertising it with the resource manager of that organization. Entire organizations can also be added by advertising the availability of the resource manager of the new organization with all the existing resource managers.

The system was simulated in C language and the behavior of the system was studied by randomly varying the computational load at different organizations of the multi-agent system. These studies were done assuming that the resources are homogenous.
6.6.1. Reduction in Decommitments

The first experiment was done to measure the reduction in the number of decommitments made by the system. Each organization of the multi-agent system was assumed to have 10 permanent agents and the number of marketable agents was 10 times the number of organizations.

The system was allowed to run for 100 reorganization cycles by randomly varying the computational load in every reorganization cycle. Different organizations contributed different amounts of funds but the amount contributed by an organization was held constant over all the 100 reorganization cycles. The total number of decommitments over the entire run was found. The experiment was then repeated for a multi-agent

Fig 6.1. Variation in decommitments over 100 cycles in 4 organizations
system without using TRACE-RAP (i.e. by equally dividing the marketable agents among the organizations and keeping the number of agents in each organization always constant). From these two results the percentage reduction in the number of decommitments using the reorganization method was determined. The results of this study are summarized in Table 6.1. The graph in Figure 6.1 shows the variations in the requirement for agents in each of the four organizations, which produced the results given in Table 6.1.

Scaling to Larger Systems

A desirable characteristic of open multi-agent systems is the ability to scale well to large systems. The simulation results as shown in Table 6.1 were obtained by increasing the number of organizations from 4 to 64. Note that in this experiment the number of agents increases with the number of organizations (the number of permanent agents in each organization was 10 and the total number of marketable agents was taken as 10 times the number of organizations). The number of requests has also been increased in the same proportion. Basically, the conditions to which an organization is subjected are the same. But the number of such organizations has been scaled up. This increase however did not effect the percentage reduction in decommitments. This indicates that with respect to reduction in decommitments, the proposed reorganization approach scales well to large systems.

### Table 6.1 Percentage reduction in decommitments

<table>
<thead>
<tr>
<th>Number of Organizations</th>
<th>Percentage reduction in decommitments in the MAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>74</td>
</tr>
<tr>
<td>8</td>
<td>80.8</td>
</tr>
<tr>
<td>16</td>
<td>80.2</td>
</tr>
<tr>
<td>32</td>
<td>76.2</td>
</tr>
<tr>
<td>64</td>
<td>76.2</td>
</tr>
</tbody>
</table>
These results were obtained in the absence of a funding strategy. The performance of the system can however be improved by having the application make use of effective funding strategies.

### 6.6.2 Fairness of Resource Allocation

<table>
<thead>
<tr>
<th>Funding Ratio</th>
<th>Org1</th>
<th>Org2</th>
<th>Org3</th>
<th>Org4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ratio of agents allocated</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Org1</td>
<td>33.23</td>
<td>23.33</td>
<td>20.00</td>
<td>23.22</td>
</tr>
<tr>
<td>Org2</td>
<td>34.21</td>
<td>23.68</td>
<td>18.42</td>
<td>23.68</td>
</tr>
</tbody>
</table>

Table 6.2 Fairness of resource allocation for a MAS with 4 organizations

<table>
<thead>
<tr>
<th>Funding ratio</th>
<th>Org1</th>
<th>Org2</th>
<th>Org3</th>
<th>Org4</th>
<th>Org5</th>
<th>Org6</th>
<th>Org7</th>
<th>Org8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ratio of agents allocated</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Org2</td>
<td>18.75</td>
<td>1.25</td>
<td>11.25</td>
<td>18.75</td>
<td>18.75</td>
<td>1.25</td>
<td>11.25</td>
<td>18.75</td>
</tr>
</tbody>
</table>

Table 6.3 Fairness of resource allocation for a MAS with 8 organizations

<table>
<thead>
<tr>
<th>Organization</th>
<th>Funding ratio</th>
<th>Ratio of agents allocated</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9.78</td>
<td>9.38</td>
</tr>
<tr>
<td>2</td>
<td>0.54</td>
<td>0.62</td>
</tr>
<tr>
<td>3</td>
<td>4.89</td>
<td>5.62</td>
</tr>
<tr>
<td>4</td>
<td>9.78</td>
<td>9.38</td>
</tr>
<tr>
<td>5</td>
<td>9.78</td>
<td>9.38</td>
</tr>
<tr>
<td>6</td>
<td>0.54</td>
<td>0.62</td>
</tr>
<tr>
<td>7</td>
<td>4.89</td>
<td>5.62</td>
</tr>
<tr>
<td>8</td>
<td>9.78</td>
<td>9.38</td>
</tr>
<tr>
<td>9</td>
<td>9.78</td>
<td>9.38</td>
</tr>
<tr>
<td>10</td>
<td>0.54</td>
<td>0.62</td>
</tr>
<tr>
<td>11</td>
<td>4.89</td>
<td>5.62</td>
</tr>
<tr>
<td>12</td>
<td>9.78</td>
<td>9.38</td>
</tr>
<tr>
<td>13</td>
<td>9.78</td>
<td>9.38</td>
</tr>
<tr>
<td>14</td>
<td>0.54</td>
<td>0.62</td>
</tr>
<tr>
<td>15</td>
<td>4.89</td>
<td>5.62</td>
</tr>
<tr>
<td>16</td>
<td>9.78</td>
<td>9.38</td>
</tr>
</tbody>
</table>

Table 6.4 Fairness of resource allocation for a MAS with 16 organizations
Chapter 6. Dynamic Resource Allocation

In TRACE, funds abstractly encapsulate relative resource rights, and are analogous to priority. The funding units are abstract since they are completely independent of resource details. They are also relative since the amount of resource to which an organization with a given amount of funding is entitled, varies dynamically in proportion to the contention for that resource.

In order to test the fairness of resource distribution a set of experiments was done for different funding ratios among organizations. The results of these experiments are summarized in Table 6.2. The first row specifies the funding ratio of organizations. The second row indicates the relative number of agents obtained by each organization. A fair distribution is one in which each organization is able to obtain a share of resources that is close to its share of total system funding. As we can see from the Table 6.2, TRACE allocates resources in a manner that is reasonably close to the funding ratio in all the runs. Tables 6.3 and 6.4 present representative simulation runs which demonstrate that a reasonable degree of fairness is continuously maintained even in large systems.

6.6.3 Adaptiveness of the Multi-agent System

The main objective of TRACE is to make the multi-agent system adaptive to variations in computational load. Figure 6.2 shows the variation in number of decommitments over 100 reorganization cycles and Figure 6.3 shows new agents acquired by an organization over 100 reorganization cycles. Initially the number of agents is 10 in all organizations (the number of permanent agents). As the number of decommitments in a reorganization cycle increases, the number of new agents in the next reorganization cycle increases correspondingly. For instance in reorganization cycle 25 the number of decommitments increased to 300. As a consequence of this, the number of marketable agents in the next cycle increased to 10 (for G=30). Similarly it can be seen that a decrease in the number of decommitments results in a decrease in the number of agents. These results demonstrate the ability of the multi-agent system to adapt to
computational load variations, thereby making it applicable to time constrained domains.

Figure 6.2 Variation in the number of decommitments

Fig 6.3. Variation in the number of marketable agents allocated
6.6.4 Efficient use of Resources

As shown in Figure 6.3, the average number of agents over 100 reorganization cycles is 17. In order to achieve the same level of performance as in TRACE, a multi-agent system with a constant number of agents would require 20 permanent agents (the maximum number of agents required by the organization in 100 cycles). Thus the proposed approach which requires around 17 agents on an average is more economical in terms of resource usage.

6.6.5 Reorganization Overhead

One source of overhead is the set of agents called resource managers that were introduced into the multi-agent system to perform reallocation of agents. The sole function of these agents is to perform reallocation at the beginning of every reorganization cycle. The resource managers however remain idle for the remaining part of the cycle. In order to make effective use of these resource managers, they can be allocated tasks just like other agents of the organization.

The second source of overhead is the communication cost that is incurred as a result of transfer of information from the agents of an organization to their resource manager and vice versa. The number of messages that are sent to a resource manager is equal to the number of agents in its organization (N). Thus there would be N transfers of information to the resource manager at the beginning of the reorganization cycle. After arriving at the equilibrium price, the resource manager broadcasts information about the new agents and the equilibrium price to all agents of its organization. This takes another N message transfers. In all there would be 2*N message transfers per reorganization cycle. In addition to this some communication takes place among resource managers. This is the communication regarding the funds and the required number of agents that is broadcast by every resource manager to every other resource manager. However this communication is not considerable since the number of resource
managers is very small compared to the total number of agents in the multi-agent system.

The third factor that needs to be considered is the time required by resource managers to arrive at the equilibrium price i.e., when

\[ |Z(p)| < \varepsilon. \]

The convergence time in general depends on two values: the number of iterations required to reach equilibrium, and \( n \) (the number of buyers and sellers). Let us consider the number of iterations first. The number of iterations required for convergence depends on \( p' \) (the step size parameter), and \( \varepsilon \). The smaller the values of \( p' \) and \( \varepsilon \), the more feasible is the allocation of agents, but larger is the number of iterations. Larger values of \( p' \) and \( \varepsilon \) reduce the number of iterations but may not result in an allocation with the same degree of feasibility. The price of an agent that resource managers arrive at may not be sufficiently close to the actual equilibrium price. This results in an infeasible allocation of resources where the amount of resources allocated may not be equal to the amount of resources available.

The second value that determines the convergence time is \( n \). In general, in multi-agent systems that use economic approach \( n \) equals the total number of buyers and sellers. However in TRACE, \( n \) is equal to the number of resource managers since they represent the resource needs of the entire organization. Thus \( n \) is significantly reduced and therefore results in faster convergence.

6.7 Conclusions

Techniques for building multi-agent systems that can adapt to changing environmental conditions are of great interest. This chapter described a protocol, TRACE-RAP, for resource reallocation and presented the simulation results. The main objective of the proposed mechanism is to obtain a multi-agent system that adapts to varying
computational loads and can therefore be used for time constrained applications. Our objective is achieved by using the economic approach for reallocation of resources. This method is used in conjunction with the task allocation protocol, TRACE-TAP, described in chapter 5.

There are many advantages of using the microeconomic approach for controlling resource usage in a distributed system. Firstly, it allows direct application of many ideas and results instead of developing new theories. Secondly, it is simple to implement. Thirdly, these methods possess the properties of monotonicity, feasibility, and fast convergence. Such an approach therefore holds great promise in that it provides a single decentralized framework, which reduces the complexity of designing large, distributed systems.