Matchmaking within Multi-Agent Systems

B. Pour Ebrahimi   K. Bertels   S. Vassiliadis   K. Sigdel

Computer Engineering Laboratory, ITS, TU Delft, The Netherlands

{behnaz, koen, stamatis ,kamana}@ce.et.tudelft.nl

Abstract— **A distributed system is one in which components located at networked computers communicate and coordinate their actions only by passing messages. The research areas of multi-agent systems and distributed systems coincide, and form the research area of distributed agent computing. The study of multi-agent systems (MAS) focuses on systems in which many intelligent agents interact with each other. In many situations, agents have to locate resources involving some kind of matchmaking. In this paper, we will give an overview of different approaches of matchmaking processes. We are interested in large-scale networks where certain nodes, called consumers, have to locate available resources and other nodes, called producers, are looking for tasks to execute. For matchmaking to occur, some issues must be considered, such as how to enable agents to decompose their tasks and goals, how to enable agents to communicate and how to enable agents to represent and reason about the actions, plans, and knowledge of other agents in order to interact with them. Criteria such as the maximum number of matched tasks and resources, task allocation efficiency, resource usage efficiency and matching time must be considered. The choice of the matchmaking architecture will evidently impact important aspects such as task allocation, load balancing and routing as well as interaction between agents. In this paper we provide an overview of existing approaches such as centralized matchmaking and peer-to-peer communications and propose a framework that compares the different approaches in terms of performance, efficiency, communication cost and task complexity.**

Keywords: Match-making, resource allocation, peer-to-peer, centralized matchmaking.

I. Introduction

Computation in networks of processing nodes, each holding a part of the inputs and/or resources initially, can roughly be classified into centralized, duplicated and distributed computations. A centralized solution relies on one node being designated as the computer node and processing the resources to process the entire application locally. All input data and relevant resources are sent to this node, and after local processing the computer sends the relevant output data to each of the other nodes. A duplication solution sends all input data to each node, after which each node processes the entire application and throws away all output data except those it needs itself. Duplicated computation is used to compute routing tables in the internet. In distributed computation, the processing steps of the application are divided among the participating nodes. The

goal is to minimize communication and computation cost. The distributed model is characterized by a collection of autonomous processing elements, called nodes. In addition to some computing and storage resources, each node has the possibility to exchange information with some of the other nodes. These are referred to as its neighbors and the communication place through a link.

The study of multi-agent systems (MAS) focuses on systems in which many intelligent agents interact with each other. The agents are considered to be autonomous entities, such as software programs or robots. The agents can share a common goal or they can pursue their own interests. Multi-agent systems are often distributed systems, and distributed systems are platforms to support multi-agent systems. The basic idea in multiagent system approach is to have a collection of agents that have to solve a small subproblem. Agents may be affected by other agents (including humans) in pursuing their goals and executing their tasks. Interaction can take place indirectly through the environment in which they are embedded or directly through a shared language. Whenever an agent needs additional resources to perform its assigned task, the agent needs to locate the available resources and make use of it. This assumes some kind of matchmaking as a precursor to possibly collaboration [25].

In this paper first we review the issues that are important in distributed systems like resource allocation, load balancing, routing and synchronization and we survey different approaches of their implementation. Then we discuss matchmaking process and different mechanisms of matchmaking. In next section we present a comparison of matchmaking mechanisms. At last we will have the summary and discuss future research directions.

II. Issues of Distributed Systems

A distributed system can be viewed as a collection of computing and communication resources shared by active users. In distributed and dynamic domains, control is distributed and the multiple agents must collaborate to accomplish the tasks at hand. To achieve a maximum efficiency of large distributed computing systems, the workload has to be distributed equally throughout the network. Delivering messages and also synchronization are other aspects to be considered in distributed systems. In following we investigate some approaches to perform these issues.

A. Resource Allocation

The sharing of resources is a main motivation for constructing distributed systems. Good resource allocation decision in a dynamic and unpredictable environment must consider overall system optimization across tasks, and the sustainability of the agent society for the future tasks and usage of the resources. In agent coalition formation method [4], agents can form a coalition and execute a task together. In this scheme each agent bids the maximum affordable cost for each task. Based on the bidding information and the cost curves of the tasks, the agents are splitted into groups, one for each task, and cost division among the group members for each task is calculated. It is always efficient to include an agent in a coalition if the agent can afford the marginal cost. The objectives of this method include: to optimize the total performance of the tasks and to divide the cost among agents in a fair way to achieve good sustainability.

In distributed systems the multiple agents each obtain only local information and face global ambiguity - an agent may know the results of its local operations but it may not know which other collaborators must be involved to fulfill the global task and which operations these collaborators must perform for success. Also the situation is dynamic so that a solution to the resource allocation problem at one time may become obsolete when the underlying tasks have changed. This means that once a solution is obtained, the agents must continuously monitor it for changes and must have a way to express such changes in the problem. Modi [10] proposed a formalization of distributed resource allocation that represents both dynamic and distributed aspects of the problem and a general solution strategy that uses distributed constraint satisfaction techniques.

Microeconomic mechanism can also be considered as resource allocation mechanisms in distributed systems because the economic activities of human beings can be viewed as a form of large-scale resource allocation without centralized control. As distributed systems grow in size and the nodes become more powerful and complex, the complexity of making resource allocation decisions grows dramatically. Microeconomic approaches limit this complexity by partitioning the complex global problem of allocating many resources, supplied by many sources and used by many types of consumers into a set of simple, independent sub-problems. There are two main ways to allocate resources among the competing agents. One of them is the exchange based economy and the other is the price based economy. In the exchange based economy, each agent is initially endowed with some amounts of the resources. They exchange resources till the marginal rate of substitution of the resources is the same for all the agents. Kurose,

in his paper [15] , provided decentralized algorithms to allocate resources (such as files or file fragments) in a cooperative and non-competitive manner among agents (computer system). The optimization criteria included communication cost and average processing delay. Unlike the selfish model where each user had its own utility function, this model has a global utility function which is known to all the users in the distributed system.

In a price based system, the resources are priced based on the demand, supply, and the wealth in the economic system. The allocations are done based on reaching an equilibrium price where demand equals the supply. Bidding and auctioning is another form of resource allocation based on prices. There are several auctioning mechanisms such as the Sealed Bid Auction, Dutch Auction, and English Auction.The basic philosophy behind auctions and biddings is that the highest bidder always gets the resources, and the current price for a resource is determined by the bid prices [8].

Natural models have also used by researchers to design multiagent systems to solve different kinds of resource allocation or optimization problems. A common natural model used is that of ant behaviors [2]. Although each ant undertakes very simple activities, the global outcome of the collection accomplishes goals that are very difficult without the ants behaving as a team. Ant behaviors are an example of what is known as Complex Adaptive Systems (CAS). These systems have a large number of members with simple functions and limited communication among them. Camorlinga [23] proposed an approach that uses the CAS framework to develop multiagent systems for storage resource allocation based on squirrels' hoarding behaviors in peer-to-peer systems. From a computational perspective, squirrels are allocating resources to storage demands in such a way that resources are balanced.

B. Load Balancing

When the demand for computing power increases the load balancing problem becomes important. We can state the load balancing problem as follows. Given the initial job arrival rates at each computer in the system find an allocation of jobs among the computers so that the response time of the entire system over all jobs is minimized.

In general we can distinguish static and dynamic load balancing algorithms. In the case of a static load balancing policy, a fixed process graph which represents the distributed computation mapped onto the interconnection network. In this case the aim is to minimize edge dilation, processor load differences and edge congestion. If the load situation changes in an unpredictable way, as it is the case for many applications, it is necessary to use a dynamic load

balancing strategy which is adaptive to this changing load situation. To be efficient for large distributed systems the load balancing algorithm itself should be distributed. Any dynamic distributed load balancing strategy can be separated into a decision part and a migration part. In the decision part of the algorithm a decision to migrate or to keep a load unit is made. This decision can be based on the local load situation and that of the neighboring processes or it depends on the load situation of any subset of the whole network. In the first case it is called a 'local decision base' whereas the second case is called 'global decision base' [16].

The load balancing problem is to design algorithms that minimizes the mean job waiting time by migrating the jobs to balance the workloads of the processor nodes. This can also be done using the bidding and auctioning mechanisms, and price controls. In the load balancing economy, jobs migrate in search of suppliers based on the job preference model. Each job independently computes the best place (node) to be served based on its preferences and wealth, and the resources prices. Ferguson[7] proposed a microeconomic algorithm for load balancing in distributed computer systems. In this approach the system is partitioned into sets of completely independent agents that compete for the resources in the system. Through the laws of supply and demand,this competition dynamically sets prices that are charged for purchasing the available resources. A globally effective allocation of the resources is achieved indirectly through the selfish optimization and competitive behavior of the economic agents. In this approach a job attempts to purchase resources and be serviced. To do so, it computes a budget set and a preference relation on elements of the budget set based on price and service time and then generate a bid for the most preferred budget set element. A job is allowed to migrate through the system in search of CPU service, but it must pay each time it crosses a link. Job must return to the processor at which it entered the system after receiving CPU service.

C. Routing

Delivering messages between pairs of processors is a basic and primary activity of any distributed communication network. This task is performed using a routing scheme, which is a mechanism working in a distributed fashion for routing messages in the network. The routing mechanism can be invoked at any source node and be required to deliver a message to some destination node. A routing algorithm is a (computable) function that for each message arriving at a node determines the link on which the message has to be transmitted, and this is as function of its destination or any other information contained in the header of

the message. The term efficient groups a set of desirable quality factors like: the routes generated by the algorithm are (near) shortest paths in the graph; the time to compute the function is low; the number of routes using a same link is low; the size of the data structure required by the algorithm is small; the routing scheme is fault tolerant, and so on[9].

D. Synchronization

One of the issues must be considered along with task allocation is synchronization problem. In distributed systems synchronization can be performed in centralized and distributed mechanisms. In MAS, centralized synchronization does not fit with autonomy of agents and scalability. Besides, centralized control conflicts with the distributed nature of MAS. Distributed synchronization can be implemented as regional synchronization[6] that enables agents to synchronize with only colleagues inside their perceptual range. With this model agents form synchronized groups on the basic of their actual locality. Different groups act asynchronously, while agents act synchronously in their group. With regional synchronization, each agent is responsible to setup synchronization with other agents. Synchronization between two agents then comes down to reach a mutual agreement about synchronization. To reach such an agreement agents negotiate with one another by means of exchanging messages.

III. MATCHMAKING APPROACHES

In the field of matchmaking, many approaches have been proposed. We can classify them in 3 different categories:

• First is to use an market bidding mechanisms [24] [5][13]. In the last few years, there has been increase in research that combines the disciplines of multi-agent systems and economics. When the agents need to decide on task assignments, then a decision should be made on which agent will carry out a given task. Most of these questions can be resolved by providing the agents with a monetary system, modeling them as buyers and sellers of tasks and resources. Each buyer and seller will set a price and quantity of how much it is willing to buy or sell. The market mechanism then computes the equilibrium price at which supply and demand converge and the market clears. An alternative but related mechanism is the auction. The most common auction protocols are one-to-many and many-to-many auctions. In one-to-many auctions one agent initiates an auction and a number of other agents can make a bid . The English auction, Dutch auction, first-price sealed-bid auction, second-price sealed-bid belong to this category. The basic philosophy behind these auctions is

that the highest bidder always gets the resource, and the current price for a resource is determined by the bid prices. In many-to-many auction, several agents initiate an auction and several other agents can bid in the auction. The double auction is the most known auction protocol for many-to-many auctions. In these auctions, buyers and sellers are treated symmetrically with buyers submitting bids and sellers submitting offers.

- The second common form of matchmaking within multi-agent systems is through a broker or middleagent [14] [12] [3]. One of the major problems facing multi-agent systems is finding the services and information that the agent needs and connecting to the agent that is providing this service. There are two types of information used in the agent interaction process: preferences and capabilities. To protect the privacy of each agent, preferences flow from a requester to a provider agent, and capability information flows from the provider to the requester. Agents that deal with both types of information that are neither requesters nor providers are middle-agents. There are many different forms of such agents, differing in whether consumer or provider information is stored and if the facilitator directly connects consumers and providers or acts as an intermediary for transactions, but they all act as a directory for available services in the system. The three most commonly used types are : blackboards, brokers and matchmakers. In these systems there are three general agent categories: service providers, service requesters, and middle agents. Matchmaking is the process of finding an appropriate provider for a requester through a middle agent. Provider agents advertise their capabilities via middle agents who store these advertisements. The requester asks the middle agent whether it knows of a provider with the desired capabilities. The middle agent matches the request against the advertisements and returns the result[see fig.1].

- The third approach is to allow peer-to-peer communications inducing lower communication cost but using only local information [20] [17]. In local matchmaking providers and consumers are distributed randomly around the system and consumers look only for an acceptable match within their neighborhood not necessarily yielding the best possible match. In the studies on peer-to-peer communication mechanisms, in [20] and [19], agents are considered connected by a random peer-to-peer communication networks which search partners for tasks by forming small groups that locally modify their network connections. In [20] as agents find compatible neighbors, clusters form and (in a successful system) gradually get larger. A successful trail of the system is one that ends up with most of the agents connected to each other, usually in a single large cluster. So in this model cluster size is not limited
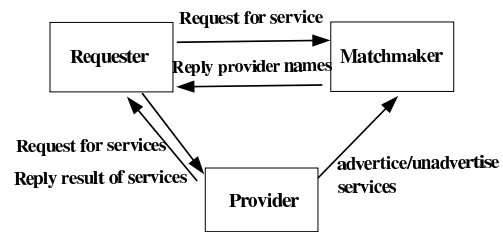

Fig. 1. Matchmaking using matchmaker

and thus the design goal of avoiding centralized control is not fully met. In [19] by limiting cluster size distribution in the system as a whole is maintained. In [21] a modified version of the centralized auction and the agent bidding update algorithm presented in [22] is used. In this approach, an auction is made up of agents representing individual traders each with a single good to buy or sell, and a fixed reservation price. In the paper has been shown that for the peer-to-peer auction the message round cost remains constant as the number of agents increases while for the centralized case it increases linearly. But the cost of number of bidding rounds to find equilibrium in peer-to-peer auction is about two times higher than centralized auction. In the remains of this paper, we consider market based matching and middle layer matching similar to each other even though the algorithms underlying both approaches may differ.

IV. COMPARISON OF MATCHMAKING MECHANISMS

On the continuum from centralized to peer-to-peer matchmaking, in peer-to-peer mechanisms, the agents are grouped into clusters which are controlled centrally[see fig.2]. Clusters could grow as large including all the agents in the system and a cluster controller would then become a global central controller. To avoid the use of a central controller, the clusters must be decentralized or cluster size must be limited. Decentralizing the clusters is not feasible, however, combining or splitting up clusters without some global cluster information is difficult and put decentralize clusters completely to fail. Another approach to maintaining locality is to keep clusters centralized, but to limit the size of the clusters. Given the variety of approaches for matchmaking, it is important to be able to determine the conditions under which particular mechanisms are more performant than others. A set of criteria that can be used to assess the usefulness of a particular mechanism can be:

- **Scalability**: in large distributed systems, this issue is very important. Scalability requires that an increase in the number of new agents and resources has no noticeable effect on performance nor on administrative tasks.
- **Dynamic and flexible behavior**: multi agent systems need to manage problems such as increases and fluctuations of the number of agents, changes of task profiles and
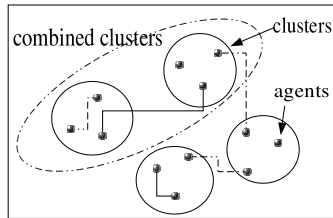
Fig. 2. Matchmaking using peer-to-peer communication

drop-outs of agents. They should be able to determine the most appropriate organizational structure by themselves at run-time (self-building) and to change this structure as their environment changes (adaptivity).

- **Throughput**: an important element is the throughput, especially when considering very large scale systems. Throughput is measured as the amount of useful work carried out by the system in a unit time. In MAS throughput is defined as the number of tasks that can be allocated to available resources in a period of time.

- **Robustness**: is the ability of the system to remain operational even when minor or major disrupting events occur. This is a vital characteristic of a distributed environment and the matching mechanism should therefore also have this property.

- **Communication cost**: The number of messages that must be processed and passed to find matches is another criterion that could be taken into account. This cost affects on matching time and communication overhead.

In the next section, we compare two main matchmaking mechanisms, centralized and localized, based on these criteria, as given in Table I.

A. Centralized Matchmaking

All facilitator architectures involve offered services or requests being stored in a central location. Agents wishing to find matches then submit a request to this location. Facilitator architectures are good for finding optimal matches since the facilitator is in a position to compare all available possibilities.

- **Scalability**: A centralized controller limits the scaling capability of the system. As the system grows, a central controller that stores data on all agents will need more and more memory. It will also need an increasing amount of processing time to search for matches among its stored advertisements and it will need more and more bandwidth to handle requests from the agents. Centralized matchmakings are very efficient in certain population sizes. Going below this boundary will generate lower matching rate and going beyond the boundary will increase the matching time but not the allocation efficiency[1].

- **Dynamic and flexible behavior**:In centralized matchmaking (using one facilitator), middle agents must provide a directory of the entire system. For large and dynamic system such a directory can be expensive to maintain, and if there are many requests for matches the middle agents can become a bottleneck since all matches are made through requests to the facilitator. Facilitators which provide complete directories of the consumers or providers in a system, can consume a large amount of memory and all advertisements and requests must be made in a way the facilitator can understand. Such requirements would be a form of global knowledge that would make it more difficult to add new agents which must be initialized somehow[20].

- **Throughput**: the centralized matchmaking guarantees that an agent finds a match if it exists or allows an agent to find the best match system wide. But as the system grows, the matching time (the time is needed for a consumer to find a appropriate resource), will increase.

- **Robustness**: one central middle-agent represents a single point of failure and communication bottleneck in the system. If the system loses the middle-agent, it is not possible to search for matches. Robustness can be designed into the system by alllowing other nodes to assume the matchmaking function whenever required. Mullender and Vitanyi [18] present a general model of a distributed directory service and its memory and messages costs, and Jha et al [11] discuss splitting a single facilitator's function among a number of agents.

- **Communication cost**: Depending on which centralized mechanism is used, communication cost may differ. Centralized matchmaking based on one matchmaker, requires only 2 messages per agent where tasks are requested and resources are allocated. Considering a system with N agents if we compute the number of messages that must be processed and passed in each trail, this cost will be $c = 2N$. In centralized auction, each round auctioneer must send a message to each trader and then receive a message from them, but reaching an equilibrium price takes several rounds. So for this case the former cost must be multiplied by the number of rounds. For example consider a system with 50k agents, this cost will be 100k for a system with central matchmaker. For centralized auction case that takes about 50 rounds, the cost of message passing will be 5000k. As we can observe this cost increases linearly as the system grows. In large scale systems multiple matchmakers can be introduced to reduce substantially the communication overhead.

B. Localized Matchmaking

In local matchmaking agents can only interact within a local neighborhood, that is a small subset of other agents whose addresses are known to that agent. Any particular consumer can find the provider he is looking for within a local area.

- **Scalability**: In this method, agents search locally for

	centralized	**Localized**
Scalability	scalable in a particular range of population size	scalable in case of interaction between clusters
Flexibility	flexible in case of using multiple matchmakers	flexible
Robustness	robust in case of using multiple matchmakers	robust
Throughput	efficient in low scale systems higher matching rate	efficient due to potential concurrency and if there is interaction between clusters lower matching rate
Communication cost	less communication cost in low scale systems for large scale systems can become a bottleneck	less communication cost in large scale systems

TABLE I

COMPARISON OF MATCHMAKING MECHANISMS

matches and the size of an agent's neighborhood is based upon the resources of the agent, so as the number of agents in the system grows, the size of the neighborhood of any agent in the system remains constant. Even though such an approach guarantees scalability , it is not certain that it will result in a more efficient matchmaking. This restriction creates a more scalable agent system.[19]. As research [1] shows that the matching rate is largly influenced by the size of the population in which the matching must occur. So if there is no task migration between the clusters, increasing the number of agents will increase the number of clusters but will not improve matching efficiency.

• **Dynamic and flexible behavior**: Matchmaking without a facilitator using peer-to-peer communication minimize the set up cost for the system and the memory, processing, and communication resource requirements of individual agents. This method allows to check a large number of possible matches in parallel. By avoiding a central facilitator that must understand advertisements for services, the need for a single common system wide capability is reduced[20]. In a dynamic system where tasks end and clusters change, system decay client-server matches because of an eventual client server distribution. To overcome this problem, matchmaking can continue indefinitely organizing into new sets of clusters, as long as some agents are willing to be flexible and abandon tasks they can not find matches for[19].

• **Throughput**: In this approach agents search among a local set of neighbor agents for matches, and form cluster partnership to expand their search space. Such a system either quickly form a single large cluster where almost all links are matched, or remained almost completely unconnected. This behavior varies with the number of task categories[20]. localized matchmaking reduces communication overhead but the time required to find a match in the local neighborhood increases substantially.

• **Robustness**: Peer-to-peer mechanisms don't use one central middle-agent, so they avoid single point of failures.

• **Communication cost**: If we compute the number of messages that must be processed and passed in each trail in peer-to-peer agent auction[21], considering a system with N agents and s clusters and g agents per cluster, the maximum message round cost calculated for a bidding round is $c = 5s/2 + 3g - 1$. For example in a system with 50k agents, reaching to equilibrium price takes about 100 rounds in peer-to-peer, so the cost of message passing will be 5000k. With increasing the number of agents in the system, this cost almost remains constant.

V. SUMMARY AND FUTURE WORK

In this paper we looked at different matchmaking mechanisms and the most important issues of matchmaking process. Comparing different approaches shows that each mechanism has advantages and disadvantages. In terms of flexibility and robustness, localized matchmaking seems to have some advantages over a more centralized approach. However, as far as the matching efficiency is concerned localized matchmaking is less performing especially when there is little or no interaction between clusters. In addition, peer-to-peer mechanisms reduce communication overhead but they increase matching time. In terms of the number of messages that must be passed in the system in each trail, centralized matchmaking acts better but we should take into account that for large scale systems this cost increases linearly in centralized case and also large scale systems can result in a prohibitive amount of traffic for message passing.

Future work will focus on the idea of a dynamic view of matchmaking where the conditions under which either approaches is more efficient, will be researched. This will allow us to design an architecture capable of modifying itself given a particular context. For instance based on this idea the entire system can be partitioned into segments such that every segment has a matchmaker. Agents can then interact with each other in a local neighborhood, and matchmaking in different segments can take place in parallel. Splitting

and combining segments in case of entering new agents and ending existing tasks must be implemented. Migration tasks between segments to balance workload and task category distribution among segments and to reach a fully matched configuration should also be taken into account.

REFERENCES

[1] K. Bertels, N. Panchanathan, S. Vassiliadis, and B. Pour Ebrahimi. Centralized matchmaking for minimal agents. In *Proceedings of the Conference on Parallel and Distributed Computer Systems*, page 9, November 2004.

[2] Dorigo M et al Bonabeau E. Inspiration for optimization from social insect behaviour. *Nature*, 406, July 2000.

[3] Epema D. Bucher A. Local versus global queues with processor co-allocation in multicluster systems. In *Eighth Workshop on Job Scheduling Strategies for Parallel Processing (in conjunction with HPDC-11)*, pages 184–204. 2002.

[4] Katia Sycara Cuihong Li. A stable and efficient scheme for task allocation via agent colition formation. In *Algorithms for Cooperative Systems*. World Scientific, 2004.

[5] Ch. Weinhardt D. Veit, J.P. Muller. Multidimensional matchmaking for electronic markets. *Journal of Applied Artificial Intelligence*, 16(9-10):853–869, 2002.

[6] T. Holvoet D. Weyns. Model for situated multi-agent systems with regional synchronization. In *Proceedings of Concurrent Engineering, Enhanced Interoperable Systems*, pages 177–188, 2003.

[7] Y. Yemini D.F. Ferguson and C. Nikolaou. Microeconomic algorithms for load balancing in distributed computer systems. In *Proc. of International Conference on Distributed Systems*, pages 491–499, 1988.

[8] Y. Yemini D.F. Ferguson, J. Sairamesh and C. Nikolaou. Economic models for allocating resources in computer systems. In Scott H. Clearwater, editor, *Market-Based Control*, pages 156–183. World Scientific Publishing Co. Pte. Ltd., 1996.

[9] Gavoille. Routing in distributed networks: Overview and open problems. *SIGACTN: SIGACT News (ACM Special Interest Group on Automata and Computability Theory)*, 32, 2001.

[10] Modi J., Jung H., Tambe M., Shen W., and Kulkarni S. A dynamic distributed constraint satisfaction approach to resource allocation. In *Proceedings of Autonomous Agents and Multi-Agent Systems Workshop on Distributed Constraint Reasoning*, 2002.

[11] S.and P. Chalasani Jha, O. Shehory, and K. Sycara. A formal treatment of distributed matchmaking. In *Proc. of International Conference on Autonomous Agents*, pages 457–458, May 1998.

[12] Czajkowski K., Foster I., Karonis N., Kesselman C, Martin S., Smith W., and Tuecke S. A resource management architecture for metacomputing systems. In *The 4th workshop on Job Scheduling Strategies for Parallel Processing*, pages 62–82. 1998.

[13] Klusch M. K. Sycara, Lu J. and Widoff S. Matchmaking among heterogeneous agents on the internet. In *Proceedings. AAAI Spring Symposium on Intelligent Agents in Cyberspace*. 1999.

[14] Harada L. Kuokka D. Matchmaking for information agents. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 672–678, 1995.

[15] J. F. Kurose and R. Simha. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Transactions on computers*, pages 705–717, May 1989.

[16] R. Lling, B. Monien, and F. Ramme. A study of dynamic load balancing algorithms, 1991.

[17] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. Asynchronous complete method for general distributed constraint optimization. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems Part I*, July 2002.

[18] S. J. Mullender and P. M. B. Vitanyi. Distributed match-making. *Algorithmica*, pages 367–391, 1988.

[19] E. Ogston and S. Vassiliadis. Local distributed agent matchmaking. In *Proceedings of the 9th International Conference on Cooperative Information Systems*, pages 67–79, 2001.

[20] E. Ogston and S. Vassiliadis. Matchmaking among minimal agents without a facilitator. In *Proceedings. 5th International Conference on Autonomous Agents*, pages 608–615, May 2001.

[21] E. Ogston and S. Vassiliadis. A peer-to-peer agent auction. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems Part I*, pages 151–159, July 2002.

[22] M. Preist C., Van Tol. Adaptive agents in a persistent shout double auction. In *Proc. of 1st International Conference on the Internet Computing and Economics*, pages 11–17, 1998.

[23] J. Anderson S. Camorlinga, K. Barker. Multiagent systems for resource allocation in peer-to-peer systems. In *Proceedings of the winter international synposium on Information and communication technologies*, pages 1–6, 2004.

[24] Jennings N.R. Vulkan, N. Efficient mechanisms for the supply of services in multi-agent environments. *Journal of Decision Support Systems*, 28(1-2):5–19, 2000.

[25] Gerhald Weiss. *Multiagent Systems*. The MIT Press, 1999.