Centralized Matchmaking - An Empirical Study

K. Sigdel S. Li B. Pourebrahimi K. Bertels S. Vassiliadis
Computer Engineering Laboratory, ITS, TU Delft, The Netherlands
{kamana, psamuri, behnaz, koen, stamatis} @ce.et.tudelft.nl

Abstract—**Advances in networking technology and computational infrastructure make it possible to construct large-scale high performance distributed computing environment that provides dependable, consistent and pervasive access to high end computational and heterogeneous resources despite geographical distribution of both resources and users. An important research problem for such computing infrastructure is how to assign computation and communication resources to tasks and to schedule the order of their execution in order to maximize some performance criterion. The overall aim of the resource management is to efficiently schedule applications that need to utilize the available resources in such environments. It has been shown that resource heterogeneity impacts the resource allocation in quite significant way in terms of performance, reliability, robustness, scalability and fault tolerance. It has been realized that, in order to support resource allocation in widely distributed heterogeneous environment, it is useful to develop some brokering approaches based on intelligent agent technique for service discovery, performance management and data selection. One such approach for resource allocation involves matchmaking which is the process of finding an appropriate producer for a consumer. Consumer here means any processing node wanting some resources (CPU processing, memory) to solve its job and producer here is an entity who can provide these resources. Using a centralized matchmaking mechanism, there is a central matchmaker receiving both resource offers from producer and resource request from consumer and match them based on certain criterion. In this paper we will evaluate centralized matchmaking mechanism and identify the conditions under which it can perform efficiently. For this, we define different matchmaking functions for central matchmaker and evaluate its performance based on different evaluation criterions such as matchmaking efficiency, throughput, matching time, task execution and resource utilization.**

I. Introduction and Related research

The overall aim of the resource management is to efficiently schedule applications that need to utilize the available resources in distributed environments. The problem in such system is how resource allocation should be done in the case where some resources are lying idle and could be linked with other overloaded nodes in the network. The study of multi-agent systems (MAS) focuses on systems in which many intelligent agents interact with each other[13]. The agents are considered to be autonomous entities, such as software programs or robots. A Multi-Agent System (MAS) is a system composed of a population of autonomous agents, which cooperate with each other to reach common objectives in order to solve common problem [7][4] The agents can share a common goal or they can pursue their ow interests. The basic idea in multi-agent system approach is to have a collection of agents that have to solve a small sub problems. Agents may be affected by other agents in pursuing their goals and executing their tasks. Interaction can take place indirectly through a shared language. Whenever an agent needs additional resources to perform its assigned task, the agent need to located the available resources in make use of it.

Multi-agent systems have received significant attention for its potential contribution towards many diverse domains such as information retrieval and performance management as a system of autonomous agents which cooperate with each other in order to solve common problem [11]. Recently agents are being used in developing distributed systems for planning, scheduling and allocating jobs. Multi agent system are considered as an interesting paradigm for managing and using such grid system. Distributed computing focuses of large-scale resource sharing and high performance computing[3]. To obtain this goal, the interconnection between widely distributed heterogeneous resources has become a necessity for distributed computing. As a result, managing the access to computing and heterogeneous data resources in distributed computing has become a complex and time consuming task. It has been shown that resource heterogeneity impacts the resource allocation in quite significant way in terms of performance, reliability, robustness, scalability and fault tolerance. To support this it is useful to develop brokering approaches based on intelligent agent techniques for service discovery, performance management and data selection. Intelligent agents provide an important means to achieve these objectives to share and disseminate data and software more effectively. For this it is necessary to use *distributed agents*. Using *robust entities*, the system can be made to tolerate failure and recover from them. Management includes various aspects, such as complexity, resource management, fault tolerance and performance analysis, for manageability it is necessary to use *intelligent entities*. In this way, agents can be used to solve the resource allocation challenges in large distributed systems.

On the other hand, multi-agent system can find distributed environment useful testbeds to deploy its agents on a large scale. To support distributed computing, agents can offer different roles, be organized into regional or national dynamic "groups", and be able to migrate between groups to support load balancing. Therefore agents play an important role in distributed computing, and distributed computing can offer useful testbeds for investigating agent services.

There are various approaches for allocating resources in heterogeneous and distributed networks such as matchmaking and brokering, market based approach and peer-to-peer resource allocation. In brokering approach, servers and customers advertise their presence to a common advertising service or brokers by describing their characteristics in advertisements. A broker discovers compatible providers and customers with a generic matching operation and notifies the matched agents, which then employ a protocol to connect to each other and enable exchange of service. In this kind of resource allocation approach, there are three different kinds of agent categories involved: service providers, service requester and middle agents. Match making is the process of finding an appropriate provider for the requester through a middle agent. Each entity (producer or consumer) send their attributes and requirements in special format to the matchmaker [6][11]. Consumer here means any processing node wanting some resources (CPU processing, memory) to solve its job and producer here is an entity who can provide these resources. The matchmaker matches these resource requests from consumer to the appropriate service provided by the producer. Market based approach is based on real world markets where there exist various economic models for setting the price of services based on supply-and-demand and their value to users. These real world economy models such as commodity market model, market bidding, auction model, bargaining model etc can also be applied to allocate resources and tasks in distributed computing[2]. 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[5]. In contrast to the client/server architecture, there is no single central system (server) in a P2P network that provides all data and services to the consumers(clients). Information is directly exchanged between providers and consumers. Most of these systems align their peers in a overlay network to existing network interfaces. P2P network is a distributed network composed of a large number of distributed, heterogeneous, autonomous and high dynamic peers in which participants share a part of their own resources such as processing power, storage capacity, software and files content.

Using a centralized matchmaking mechanism, there is a central matchmaker receiving both resource offers from producer and resource request from consumer and match them based on certain criteria [1][8]. In P2P system, computing nodes distribute resources via direct exchange between computers without having centralized control or hierarchical organization [10][9]. Learnt from past researches, it has been shown that, the completely centralized or completely localized matchmaking mechanism each have their efficiencies and deficiencies. Centralized matchmaking simplifies the management but has low scalability because of bottleneck associated with the central matchmaker when population grows. Localized matchmaking has low throughput as agents only interact within a local neighborhood to find their proper matches. So, there is a need for an adaptive system which can automatically adapt between centralized and localized approaches of matchmaking. In our paper[12], we described the framework for such an adaptive system that can adapt between centralized and peer-to-peer matchmaking mechanism. As a part of our research, in this paper we will investigate the issues involved in centralized matchmaking and present an empirical study. We evaluate the performance of centralized matchmaking algorithm and identify the conditions under which it can perform efficiently. Its efficiency given varying conditions e.g resource intensive or task intensive environment is also studied in the paper.

In the remainder of this paper, in section II, we discuss the experimental model, matchmaking functions and evaluation criterion. Section III presents the experimentation carried out for different matchmaking algorithms and compare them based on different evaluation criterion. Finally, section IV of the paper presents the analysis and conclusion of the experiments and also presents future research directions.

II. The Experimental Model

The goal of this experiment is to evaluate the performance of different matchmaking functions in centralized matchmaking mechanism. In this experiment we assume the number of tasks and resources are more or less equal in the network. For the experiment purpose, we setup grid like environment in local LAN that comprises both windows and Linux machines. The system was implemented on Java platform with TCP/IP as a standard suite of communication protocol. The implementation model was based on standard client-server architecture. Matchmaker was implemented as a dedicated multi-client server that communicates with all the clients (producers or consumers) in the

network using *Java socket* and *Java thread*. In this experiment, we assume tasks are atomic by nature and can not be divided. Tasks and resources are generated randomly following Gaussian distribution. Also, we assume that each node in the network is directly and with the same average latency connected to the matchmaker.

In our environment, we have N agents: A = {a1, a2 ... an }. Some of these agents, called consumers have tasks to perform Ta = {t1, t2 ... tn } for which they are looking for additional resources and others, called producers, have resources to sell, Ra = {r1, r2, ..., rn}. In this case, we consider tasks and resources as function of four different types of resources: CPU, RAM, Bandwidth and Disk. Consumers can make a request for a bundle of resources. In our simulation, tasks and resources are generated considering the assigned hardware configuration. If all the node resources are not used, the node offer them through the matchmaker and becomes producer, otherwise if it needs more resources than it has, it becomes consumer. These resource constraints make matchmaking little bit more complicated.

The basic matchmaking goes as follows:

$$f : C_{(CPU,Mem,BW,Disk)} x P_{(CPU,Mem,BW,Disk)} \longrightarrow [0,1] with :$$

$$f(C_{(CPU,Mem,BW,Disk)}, P_{(CPU,Mem,BW,Disk)}) =$$

$$\begin{cases} 1 \quad iff(C_{(CPU,Mem,BW,Disk)}, P_{(CPU,Mem,BW,Disk)}) \\ \qquad\qquad isamatchingpair \\ 0 \quad otherwise \end{cases}$$

And, match between $C_{(CPU,Mem,BW,Disk)i}$ and $P_{(CPU,Mem,BW,Disk)i}$ is possible if and only if $C_{(CPU,Mem,BW,Disk)} \leqq P_{(CPU,Mem,BW,Disk)}$

In this case, for a consumer to find a match there are two criteria:

- Producer should have enough amount of all resources i.e. CPU, memory, bandwidth and disk storage. Failure to provide any of these resources in sufficient amount results un-match condition.
- Producer should be able to provide resources in specified amount of time that is specified by TTL(time to live) for each task. TTL of resource specifies resources are available only during that period of time and TTL of task determines consumer needs resources within that time. So, matching can only occur if a producer has enough resources within specified TTL of consumer.

A. Matchmaking Functions:

In order to investigate the performance of centralized matchmaking, we define various matchmaking algorithms to find matches between producer and consumer.There are three matching functions that vary in complexity and information used for matching. Evidently, one can define any number of matching functions taking for instance the quality of service into account or any other relevant factor. The choices of these functions are justified in the sense that they vary in complexity. The chosen functions are defined as:

A.1 First Match:

In this matchmaking function, for each consumer request, the matchmaker matches the consumer to the first producer that has enough resources to execute the consumer's request

A.2 Min Difference:

In min difference, the consumer is matched with that producer that has enough resources but also that yields the lowest difference between the requested task and available resources. This approach attempts to minimize the unused resources as it considers the minimum difference between them.

A.3 Min Distance:

Min distance tries to minimize the distance between producers and consumers in order to reduce the communication cost between task and resources. The consumer is matched not only when enough resources are available but also tries to minimize the distance between the two-paired nodes. This is to minimize as much as possible the distance that has to be traveled between the two-paired nodes. As each node has an (x, y) co -ordinate, we compute the Euclidean distance between two nodes.

A.4 Matchmaking Function Illustration:

To illustrate the matchmaking approach, consider the following example given in table I, for first match and min difference function. Each of the four producers in the given population is given resources as function of four kinds of CPU, memory, bandwidth and hard-disk space and same is done for consumers.

When adopting the simplest matching function i.e. first match, consumer C_1 will be matched with P_1, C_3 with C_3, C_4 with P_4 but C_2 will not be matched with P_2 but it can be matched with P_1 or P_3 or P_4. When adopting min difference, C_3 will be matched with P_2; C_2 with P_3, C_1 with P_1 and C_4 will be matched with P_1. When adopting the min distance the euclidian distance between each pair (P_1, C_1), (P_1, C_2) ... (P_4, C_4) is calculated and C_i is matched to corresponding P_i where the distance between (P_i, C_i) is minimal. To calculate the distance between two nodes, their IP addresses are considered as measuring factor. We assume

nodes that belong to same network or near networks differs little in their IP address for example 192.35.67.22 is near to 192.35.67.30 than 194.35.67.22.

B. Evaluation Criteria:

To evaluate the centralized matchmaking mechanism, in terms of different matchmaking algorithms, various evaluation criteria such as maximum number of matches, task usage efficiency, resource usage efficiency and matching time are considered. For each of the matching functions following statistics are computed.

- **Resource usage Efficiency** : Resource usage efficiency can be defined as the percentage of available resources that are used by the allocated tasks. It can be calculated as the ratio of matches occurred to the total number of *producers* in the system.
- **Task Execution Efficiency** : Task execution efficiency is the percentage of tasks that could be allocated to available resources. It can be calculated as the ratio of matches occurred to the total number of *consumer* in the system.
- **Matching Time**: Matchmaking time is the time for a consumer to find a appropriate producer. The relative matching time is the time when matchmaker receives the requests submission till it finds appropriate match.

III. EXPERIMENTATION:

In this section of the paper, we discuss about the experimentation carried out for evaluating central matchmaking. At first, we perform each experiment: first match and min difference and min distance and then compare these functions based on the evaluation criteria defined above. The following part of the paper elaborate this experimentation in details.

At first, we evaluate the first match function in terms of matched number, matching time, task and resource utilization for different network conditions. We summarize the experimental data in table II. As it can be seen from table II, matching number is proportional with the number of agents - where there are less number of agents match found it also less. However, after certain population size, experiment couldn't be performed due to limitation in network bandwidth and limitation in hardware configuration for simulation. As far as resource and task execution are concerned, they are tightly related and any influence on task usage is immediately reflected to resource usage. The resource utilization and task execution are also affected by the timing constraints as producers/consumers start dying if they don't find matches and matchmaking constraints matchmaking can occur only $(C_{(CPU,Mem,BW,Disk)} \leqq P_{(CPU,Mem,BW,Disk)})$.

We also evaluate the min difference function in terms of matched number, matching time, task and resource utilization for different network conditions. This approach of matchmaking attempts to minimize the unused resources as it considers the minimum difference between them. We summarize the experimental data in table III. The first thing to notice about this matching function is the matchmaking constraints that it poses. As it can be seen from fig 2 the consumer matching time for min difference is larger than in first match. Also, for this function, for number of matches, task and resource usage, the similar observation can be made as in case of first match .

The last matching function that was used is the minimal distance between two agents. The minimal distance computes the euclidian distance between each consumer and a given set of producers which is more expensive in terms of computing cycles than min difference and first match. This function has also been evaluated based on the same criteria of matched number, matching time and task/resource usage. The experimental data are summarized in table the table IV. For resource utilization and task execution, the similar observation can be made as in cases of first match and min difference. Similar phenomenon also applies for matched number and matching time as well.

A. Comparing three different Matchmaking Functions:

In this part of the paper, we compare three different types of matchmaking function based on the same evaluation criteria: number of matches, matching time, task execution and resource utilization.

A.1 Number of Matches

The graph 1 shows the number of matches for three different matchmaking functions. The number of matches occur in case of first match doesn't significantly differ than other two match functions. However, this should hold theoretically, because of the structure of first match function. First match is simple and has less constraints while it finds matches. Min difference and min distance are little more complex. In min difference we consider the efficient resource utilization and min we also consider the minimal distance between the matching pair. The reason for this fluctuation in the graph 1 is the direct consequences of matchmaking constraints $(C_{(CPU,Mem,BW,Disk)} \leqq P_{(CPU,Mem,BW,Disk)})$ imposed by task and resources in our matchmaking and TTL limitation imposed by each producer and consumer.

A.2 Matching Time

As it can be seen from graph 2, the consumer matching time for first match is relatively low as compare to others

Producers	P1	P2	P3	P4
$P_{(CPU,Mem,BW,Disk)}$	200,40,400,50	10,40,40,50	200,40,400,50	400,40,400,50
Consumers	C1	C2	C3	C4
$C_{(CPU,Mem,BW,Disk)}$	100,20,400,30	200,40,400,50	10,40,0,50	200,40,400,50

TABLE I
MATCHMAKING ILLUSTRATION

	First Match: $(Producer \approx Consumer)$							
	$Number$			$Time$		$Usage$		
$Pop.size$	Con	Pro	$Match$	Pro	Con	$Task$	Res	$Match$
50	19	31	19	184	16	100%	61%	100%
50-1050	239	256	231	5354	1.83	99%	85%	98%
1050-2050	697	803	600	11462	0.03	85%	73%	86%
2050-3050	1132	1368	616	12895	0.36	55%	45%	54%
3050-4050	1589	1911	844	9705	1.13	53%	44%	53%
4050-5050	2002	2498	1240	12930	0.03	62%	50%	62%
5050-6050	2521	2979	2140	12510	0.21	88%	73%	85%
6050-7050	3176	3325	1359	12939	0.03	42%	41%	43%
7050-8050	3587	3913	2967	6910	1037	83%	75%	83%
8050-9050	3933	4567	1982	7240	366	50%	43%	50%
9050-6050	4526	4974	3163	7062	68	71%	64%	70%

TABLE II
MATCHING TIME, RESOURCE/TASK USAGE FOR TASK INTENSIVE NETWORK $(Producer \approx Consumer)$

	$MinDifference : (producer \approx consumer)$							
	$Number$			$Time$		$Usage$		
$Pop.size$	Con	Pro	$Match$	Con	Pro	$Task$	Res	$Match$
50-1050	233	267	233	4041	174	98%	87%	100%
1050-2050	699	802	408	5310	0.69	60%	52%	87%
2050-3050	1154	1346	541	6273	0.72	47%	41%	88%
3050-4050	1641	1860	789	5570	0.97	48%	42%	88%
4050-5050	2083	2417	1051	5019	15.4	51%	44%	86%
5050-6050	2504	2996	1389	6518	5.69	56%	47%	86%
6050-7050	2970	3530	1550	5035	34.5	53%	44%	83%
7050-8050	3483	4018	1856	4713	126	54%	46%	85%
8050-9050	4017	4484	1942	6576	1.11	49%	43%	85%
9050-6050	4250	5250	2369	2975	2761	56%	45%	80%

TABLE III
MIN DIFFERENCE: MATCHING TIME, TASK/RESOURCE USAGE IN BALANCED NETWORK

	$MinDistance : (producer \approx consumer)$							
	$Number$			$Time$		$Usage$		
$Pop.size$	Con	Pro	$Match$	Con	Pro	$Task$	Res	$Match$
50-1050	259	291	235	2692	19	94%	83%	91%
1050-2050	700	850	428	5394	1	63%	51%	61%
2050-3050	1168	1382	611	4324	1	53%	44%	52%
3050-4050	1648	1902	88	5717	10	54%	48%	5%
4050-5050	2082	2468	1119	4165	73	54%	46%	54%
5050-6050	2552	2998	1195	5388	3	47%	40%	47%
6050-7050	3046	3504	1439	5407	2	48%	41%	47%
7050-8050	3451	4099	1820	3739	38	54%	45%	54%
8050-9050	4114	4436	1987	5707	1	49%	45%	48%
9050-6050	4485	4916	2085	4604	36	47%	42%	46%

TABLE IV
MIN DISTANCE: MATCHING TIME, TASK/RESOURCE USAGE IN BALANCED NETWORK

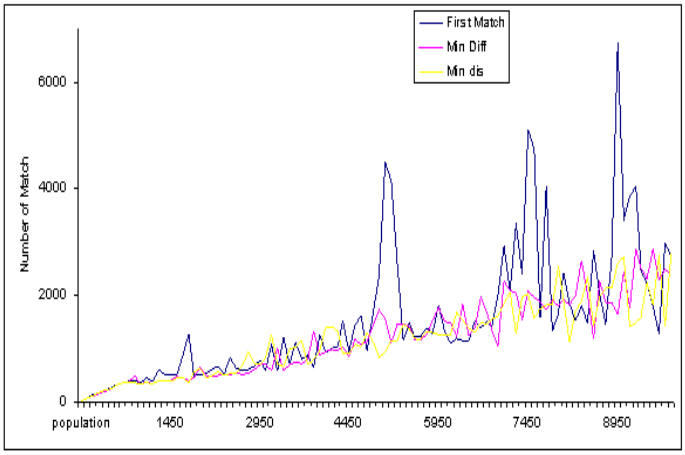

Fig. 1. Matched Number: First Match, Min Difference and Min Distance

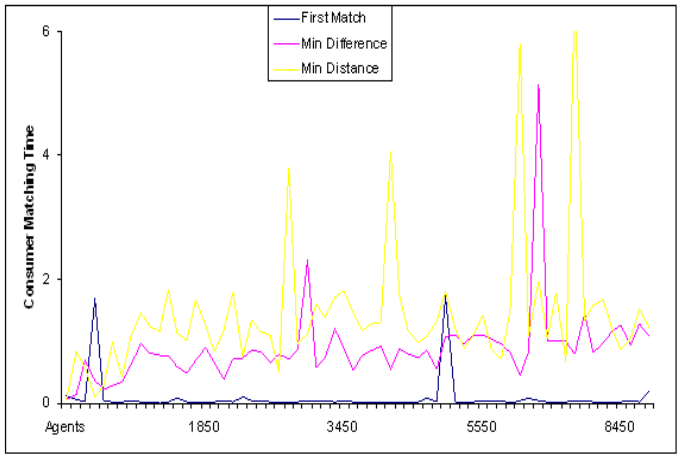

Fig. 2. Consumer Matching Time: First Match, Min Difference and Min Distance

two i.e. min difference and min distance. This is direct consequence of the structure of Min distance function. In the first match, matchmaker finds the first producer that can satisfy the resource requirement of consumer, while min difference tries to optimize the resource usage and min distance tries to reduce the communication overhead. So these algorithms take more time to find match than first match.

However, it can be seen from figure 3, the producer matching time in this case is quite unstable and fluctuate very much with population. This is the consequences of functional constraints that is imposed on the matchmaking functions i.e. matchmaking can only occur if $(C_{(CPU,Mem,BW,Disk)} \leqq P_{(CPU,Mem,BW,Disk)})$ and TTL limitation.

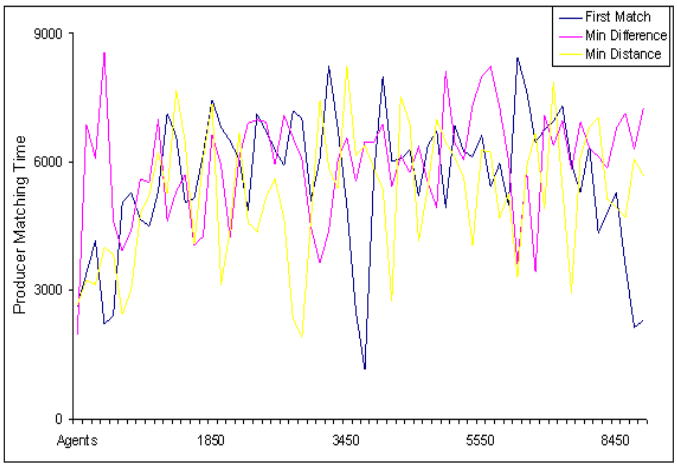

Fig. 3. Producer Matching Time: First Match, Min Difference and Min Distance

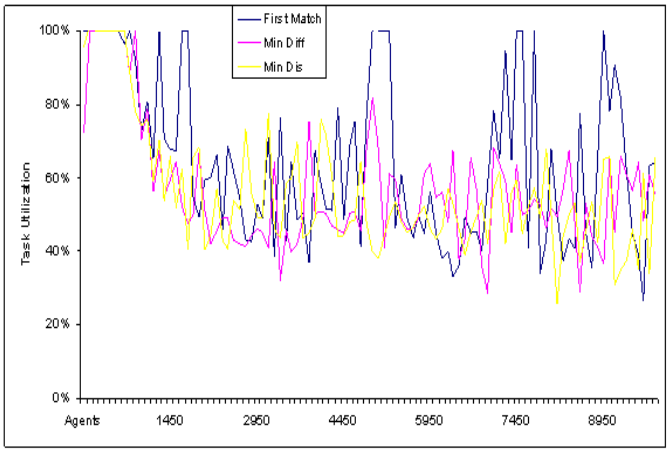

Fig. 4. Task Utilization: First Match, Min Difference and Min Distance

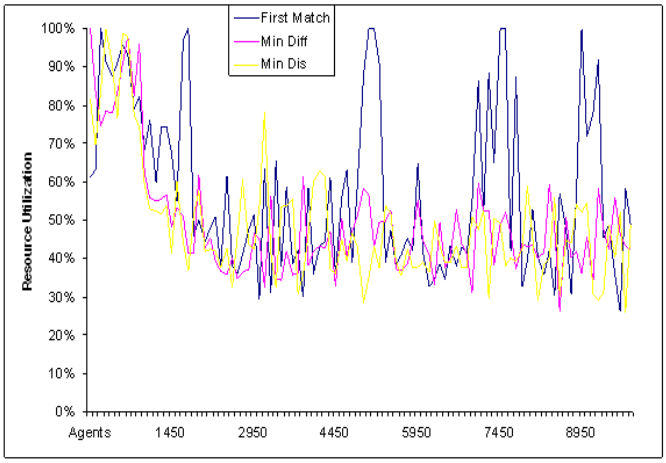

Fig. 5. Resource Usage: First Match, Min Difference and Min Distance

A.3 Task and Resource Utilization

Also, there is not much difference in resource utilization and task utilization efficiency in terms of first match as compare to min difference and min distance. Theoretically, resource utilization in min difference should be efficient compare to other two algorithms in terms of resource and task usage but since there are other constrains like TTL constraints and matchmaking constraints it is hard to claim anything obvious like that. The figures 4 and 5 compare the task execution and resource utilization for 3 different algorithms.

IV. Conclusion and Future Work:

In this paper, we studied the performance of centralized matchmaking mechanism in a single cluster of system using three matchmaking functions. Centralized matchmaking assumes a central matchmaker receiving both resource offers from producer and resource requests from consumer and matches them based on certain criteria. We simulated the centralized matchmaking experimentation for three different kinds of matchmaking algorithms: first match, min difference and min distance.

In a realistic setting, the matchmaking would occur in a asynchronous way where different offers and bids are treated as they are submitted. These offers and bids have a limited lifetime. In those cases where there are resources and timing constraints, simple matchmaking schemes like 'first match' perform more efficiently than other matchmaking functions as it can find the matches very quickly. With the increase in population size, the matching time also increases but the population size didn't significantly influence the matchmaking functions in a negative way. Future research will address how a more flexible and adaptable resource allocation can be implemented.

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] R. Buyya, D. Abramson, and J. Giddy. Economy driven resource management architecture for computational power grids, 2000.

[3] J. Cao, S. A. Jarvis, S. Saini, D. J. Kerbyson, and G. R. Nudd. Arms: an agent-based resource management system for grid computing. *Scientific Programming (Special Issue on Grid Computing)*, 10(2):135–48, 2002.

[4] K. Decker, K. Sycara, and M. Williamson. Middle-agents for the Internet. In *Proc IJCAI'97*, pages 578–583, 1997.

[5] 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.

[6] Shen W.and Li Y.and Ghenniwa H. and Wang C. Adaptive negotiation for agent-based grid computing. In *Proceedings of AAMAS2002 Workshop on Agentcities: Challenges in Open Agent Environments*, pages 32–36, Bologna, Italy, July 2002.

[7] Vasant Honavar. Tutorial on intelligent agents and multiagents systems.

[8] Somesh Jha, Prasad Chalasani, Onn Shehory, and Katia P. Sycara. A formal treatment of distributed matchmaking. In *Agents*, pages 457–458, 1998.

[9] K. Kant, R. Iyer, and V.Tewari. On the potential of peer-to-peer computing:classification and evaluation. In *Proceedings. of CCGrid, Berlin, Germany*, 2002.

[10] 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.

[11] Xiaotong Shen and Jianming Ye. Adaptive model selection. *Journal of the American Statistical Association*, 97(457):210–??, 2002.

[12] K Sigdel, K. Bertels, B Pourebrahimi, S. Vassiliadis, and L.S Shuai. A framework for adaptive matchmaking in distributed computing. In *proceeding of GRID Workshop Cracow-04*, January 2005.

[13] Katia Sycara. Multi agent system. *AI Magazine 19(2)*, 1998.