Resource Discovery with Dynamic Matchmakers in Ad Hoc Grid

Tariq Abdullah, Lotfi Mhamdi, Behnaz Pourebrahimi, Koen Bertels
CE laboratory, Delft University of Technology,
Mekelweg 4, 2628CD, Delft, The Netherlands
{tariq, lotfi, behnaz, koen}@ce.et.tudelft.nl

Abstract

Nodes in an ad hoc grid are characterized by heterogeneity, autonomy, and volatility. These characteristics result in varying workload of the resource manager in the ad hoc grid. Therefore it is required to develop a resource allocation mechanism that can balance the workload of the resource manager, hereafter referred to as matchmaker, and can enable the ad hoc grid to self-organize itself. In this paper, we define a mechanism that dynamically promotes and demotes nodes as matchmaker(s) and matchmakers back to the normal nodes in an ad hoc grid environment. The proposed mechanism uses the matchmaker workload as the basic criterion for promotion and demotion of the matchmaker(s). Simulation results show that our approach performs better than previously proposed solutions.

1. Introduction and Related Work

Ad hoc grid[1] nodes are control-, technology-, and structure-independent [1]. The participating nodes in an ad hoc grid may have their own use and access policies. These characteristics result in varying workload on the ad hoc grid resource manager. The ad hoc grids, with above characteristics, require such a resource management system that can enable them to self-organize under varying workloads. In this paper we present an economy based resource management mechanism that enables the ad hoc grid to self-organize under changing workload conditions. An overview of existing resource management mechanisms in different ad hoc grid project is presented, before going into the details of our mechanism.

There exists a variety of matchmaking approaches for ad hoc grids. These approaches can be categorized as: *Centralized*, *Decentralized* and *Hybrid*. Ad hoc grid projects with the centralized resource allocation approach [2, 5, 7, 12] employ one resource allocator/matchmaker. Centralized approaches guarantee finding a resource, if it exists, and provide high throughput. These approaches are however not scalable and can have a single point of failure. Systems with the centralized approach can suffer from a high computational overhead and may result in an overall system performance degradation.

In the decentralized or Peer-to-Peer (P2P) resource allocation approach, each node negotiates for its required resources with the other nodes. Adriana et al. [10] proposed an algorithm-based, fully decentralized resource discovery mechanism for grid environment. Kim et al. [11] proposed parsimonious resource usage and job migration to lesser overloaded nodes, in order to balance the overall workload in a decentralized ad hoc grid. Adeep et al. [4] applied attribute encoding for resource discovery grid. Attribute encoding is applied for mapping the resources to nodes in a P2P structured overlay network. As the majority of the encoded attributes may be mapped to a small set of nodes in an overlay network, therefore attribute encoding may result in a load imbalance condition. The decentralized approaches are scalable and do not have a single point of failure. However they may have low throughput and do not guarantee finding a required resource even if it exists in the ad hoc grid.

Hybrid approaches are a mixture of centralized and decentralized resource allocation approaches. Hybrid approaches use some trade off to get the best out of the centralized and decentralized approaches. Choi et al. [6] proposed a group-based scheduling mechanism in an peer-to-peer grid computing environment and called those groups *volunteer groups*. They are defined according to the individual volunteer properties. Each volunteer group has a coordinator that coordinates with its group members as well as with the volunteer server. Mobile agents distribute and schedule tasks to the members of a volunteer group. As the volunteer server manages volunteer registration, job submission by the clients, job allocation to different volunteer groups and the collection of results, therefore it can be a single point of failure and may become a performance bot-

[1] Ad hoc grid is also called Public Resource Computing [2], Desktop Grid Computing [5] or Global Grid Computing [7]

tleneck. Zhou et al. [16] exploited blocks of idle processing cycles and grouped them into geographic and night time aware overlay networks. Unfinished tasks are migrated to another night time zone when the current night time zone ends. The main drawback of this work is that the host availability model is not based on the resource requirements of the job. Furthermore job migration may result in communication overhead. CompuP2P [8] used the attribute encoding scheme in structured overlay network to create different markets for different amounts of a computing resource. Each market has a Market Owner (MO) that maintains information about buyers and sellers for matchmaking. As the majority of the encoded resources may be mapped to a small set of nodes in an overlay network, therefore the system may have a load imbalance. Butt et al. [3] implemented a P2P based Condor flocking to share resources in different Condor pools. Their work attempted to eliminate pre-configuration requirements for resource sharing in different Condor pools. They did not consider the overload condition of a Condor pool manager. Peermart [9] distributed the workload of one matchmaker among multiple matchmakers. It used one matchmaker for each type of resource being traded in the ad hoc grid. A new matchmaker is introduced only when a new resource type is introduced. It also did not consider the overload condition of a matchmaker for the introduction of new matchmakers. Multiple matchmakers [14] for grid resource discovery mainly focus on best matchmaker selection from a pool of fixed number of matchmakers.

The above discussed approaches use different parameters to distribute the workload of one matchmaker among multiple matchmakers. These parameters are volunteer properties [6], attribute encoding [4, 8], overlay networks based on night time zones [16], a matchmaker for each resource type [9] or attempt to find a best matchmaker [14, 3] from a fixed pool of matchmakers. As these approaches do not consider the workload of the matchmaker(s) so they may end up with overloaded matchmaker(s).

In this paper we propose an economy based, self-organizing mechanism for the segmentation and desegmentation of the ad hoc grid according to the workload of the matchmaker. This mechanism enables the ad hoc grid to dynamically change its infrastructure from a centralized to a hybrid form and/or back to the centralized form. The mechanism also strives to make resource allocation scalable, without noticeably affecting the performance or the administrative complexity of the ad hoc grid when the ad hoc grid dynamically grows or shrinks. The matchmaking is performed by using the Continuous Double Auction (CDA) based framework developed in our previous work [13]. We used the micro-economic based matchmaking mechanism because these mechanisms can express different system parameters into one representative value i.e. "price". The micro economic mechanism can be used to provide some kind of self management. In this work the "price" represents the matchmaker workload and is referred as the Transaction Cost (TCost). As the transaction cost for finding resources/tasks goes up too fast, additional matchmakers have a stabilising effect on the TCost value.

The results presented here are obtained from the ad hoc grid established in a local area network. All the experiments were executed in a balanced network condition, which means that approximately an equal number of consumers were competing against equal number of producers. We make the following simplifying assumptions which will be relaxed in our future work.

- We do not look into the P2P issues or any system state where the ad hoc grid may go into a P2P state. For example, when a matchmaker breaks down and the ad hoc grid segment goes into a P2P state.
- We assume that there is a known subset of candidate matchmakers nodes and that the consumer/producer node knows about the matchmaker to which it belongs to.
- We do not address the issue of routing of requests. We assume the availability of an overlay network that handles the routing of messages.

The contributions of this paper are as follows. This paper defines a mechanism to calculate the matchmaker workload (TCost) in the ad hoc grid. The paper compares the TCost variation in our approach (CDA) with the other ad hoc grid projects that use the Eager Scheduling (ES)[2] [2, 7] approach as their resource allocation mechanism. The paper defines an *upper threshold value* of TCost with one matchmaker in the ad hoc grid. The TCost upper threshold is then applied for dynamic segmentation and desegmentation in the ad hoc grid.

The paper is organized as follows. Section-2 provides a detailed description of the proposed model. Section-3 explains the experimental setup and discusses the results. Section-4 concludes the paper and briefs our future work.

2. Proposed Model

In this section we explain the proposed model that dynamically introduces/removes new matchmaker(s) according to the workload of a matchmaker in the ad hoc grid. The proposed model uses CDA as the matchmaking mechanism developed in our previous work [13]. The matchmaker considers different request/offer parameters, like resource quantity, job execution time, price, and consumer

[2]Eager scheduling is quite similar to FCFS (First Come First Served) or FIFO (First In First Out)[6]

budget, during the matchmaking process. A node in our ad hoc grid consists of three types of agents namely *Consumer*, *Producer* and the *Matchmaker*. A node can be a consumer or a producer or a matchmaker at any given instance. Consumers are the processing nodes that are looking for resources to execute their computing jobs. Producers are the nodes that want to share their available, idle resources. The matchmaker is the mediator agent that performs resource allocation/matchmaking. The consumer and the producer agents send their resource requests and resource offers to the matchmaker. The matchmaker searches for the match from the available offers while considering different request constraints such as requested resource size, resource availability, task deadline and the bid price. If no match is found for the request in the first attempt then this request is stored in the matchmaker's requests buffer. A request remains in the matchmaker's requests buffer until its TTL (Time to Live) expires or a match is found.

2.1. System Architecture

Each node is composed of three agents: *Consumer*, *Producer* and *Matchmaker*. The structure of these agents is depicted in Figure-1 (modified from [13]). The specification of these agents is summarized below:

Consumer/Producer Agent: Each ad hoc grid node has one consumer/producer agent. The consumer/producer estimates the task execution time or the resource availability duration in its *resource manager* module. It calculates the bid/ask price of the request/offer in the *job/resource trader* module. The agent submits the request/offer to the matchmaker through the *communication* module. It coordinates the job execution by the *job control* module.

Matchmaker Agent: The matchmaker agent performs the matchmaking in its *matchmake* module. The matchmaker communicates with the consumer/producer agents by *communication* module. It receives the requests/offers from the consumer/producer agents and inserts the received request/offer in its request/offer buffers by *consumer/producer repository manager* modules. The matchmaker agent uses Continuous Double Auction (CDA) protocol to perform matchmaking. We refer to [13] for the detailed specification of the three agents and for an overview of the CDA based matchmaking.

2.2. Segmenter Module

The *Segmenter module* is the core of the work presented in this paper. The segmenter module is developed as a part of the matchmaker agent, described in Section-2.1. The segmenter module is responsible for the following:

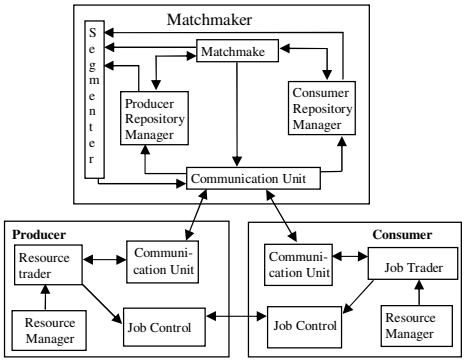


Figure 1: System Architecture

- Transaction Cost (TCost) Calculation : The TCost represents the workload of a matchmaker. The TCost is calculated by a matchmaker for every request/offer message, refer Section-2.3 for details.

- System and Node Level Self Organization: The segmenter module is responsible for system and node level self-organization by promoting a node as matchmaker or by demoting a matchmaker back to a normal node according to the workload (TCost) of the matchmaker, refer Section-2.4 for details.

- Balance the Matchmaker Workload: It is also responsible for balancing the workload, above the TCost *upper threshold* value, of one matchmaker with other matchmaker(s). The segmenter module does so by forwarding the request/offer messages to the other matchmaker(s).

- Matchmaker Communication: It is also responsible for communication between matchmakers. TCP/IP protocol is used for communication between matchmakers.

2.3. TCost Calculation

This paper presents a model to dynamically segment and merge back the ad hoc grid segments, by introducing/removing new matchmaker(s) based on the workload of the matchmaker. The matchmaker workload is represented as the Transaction Cost (TCost) . The TCost is calculated for each request/offer. The TCost value represents the number of request/offer messages to be processed by the matchmaker before processing the newly received request/offer message. The TCost value for a matched request/offer pair is the average of the their individual TCost values. Each matchmaker(s) periodically calculates its average TCost. A matchmaker promotes a node as a matchmaker or demotes a matchmaker back to normal node when its average TCost is above or below the matchmaker's *upper threshold* (see

Section-3). The TCost for individual request and offer message is represented by equation-1 and equation-2 respectively:

$$TCost_{request} = Count(requests) \quad (1)$$

$$TCost_{offer} = Count(offers) \quad (2)$$

$Count(requests)$ and $Count(offers)$ represent the number of request and offer messages to be processed by the matchmaker before processing the newly received request or offer message respectively. Average Periodic TCost is calculated as follows:

$$AvgTCost = (\sum_{i=t_0}^{t} TCost_{request}(i) + \sum_{i=t_0}^{t} TCost_{offer}(i))/N$$

Where $\sum_{i=t_0}^{t} TCost_{request}(i)$ and $\sum_{i=t_0}^{t} TCost_{offer}(i)$ is the sum of TCost of the messages processed in time interval $[t_0, t]$ *and N* is the total number of messages processed in this interval.

When the workload of the first matchmaker goes beyond the TCost *upper threshold value,* a new matchmaker is introduced to share the workload of the first matchmaker. The newly joining nodes send their request/offer messages to the new matchmaker. Similarly when both matchmakers are overloaded then a third matchmaker is introduced in the ad hoc grid and so on. When the matchmaker (s) workload decreases then the matchmaker (s) is demoted as a normal node in the same way as it was promoted and ad hoc grid segments are merged back.

2.4. System and Node Level Self-Organization

This model has two levels of self-organization: *system level self-organization* and *node level self-organization*. In system level self-organization, the ad hoc grid can accommodate increasing/decreasing workload of the matchmaker by introducing/removing new matchmaker(s). This process creates a segmented view of the ad hoc grid where each segment has its own matchmaker. When the matchmaker's workload decreases, the matchmaker is brought back to a normal node and segments are combined. In node level self-organization, nodes can be transformed from one form to another form. When the ad hoc grid needs more matchmakers, consumer or producer nodes can be promoted as matchmaker(s) with little modification. When the new matchmaker(s) are no more required in the ad hoc grid, they become the consumer or the producer nodes again. Hence, the node level self-organization enables the system level self-organization.

3. Experimental Setup and Results

The experiments were executed in a local ad hoc grid environment established in our local area network. The experiments were executed with varying numbers of participating nodes. The number of nodes was varied from 20-250 and the number of matchmakers was varied from 1-3.

Execution time, deadline and budget are randomly generated from a uniform distribution for each request. Quantity of requested/offered computational resource was varied for each request/offer message. The TTL of a request/offer message was fixed to 5000 milliseconds, in order to reflect the delays in our LAN. The consumer and the producer nodes start with an initial bid/ask price. They update their prices for subsequent requests/offers messages by using the pricing function [13]. The budget of each node increases or decreases according to buying/selling the resources. The experiments were executed in a balanced network condition, which means that approximately an equal umber of request and offer messages were generated by each participating node. The selection of any specific network condition is not important in the context of this paper. As the focus of experiments was to study the proposed segmentation/desegmentation mechanism by introducing/removing matchmakers, according to the workload of the matchmakers, therefore the workload of the matchmaker was considered as the main factor. Similar workload can be generated in unbalanced network conditions. Data is obtained after the system reaches a steady state, after fourth of the experiment duration has elapsed.

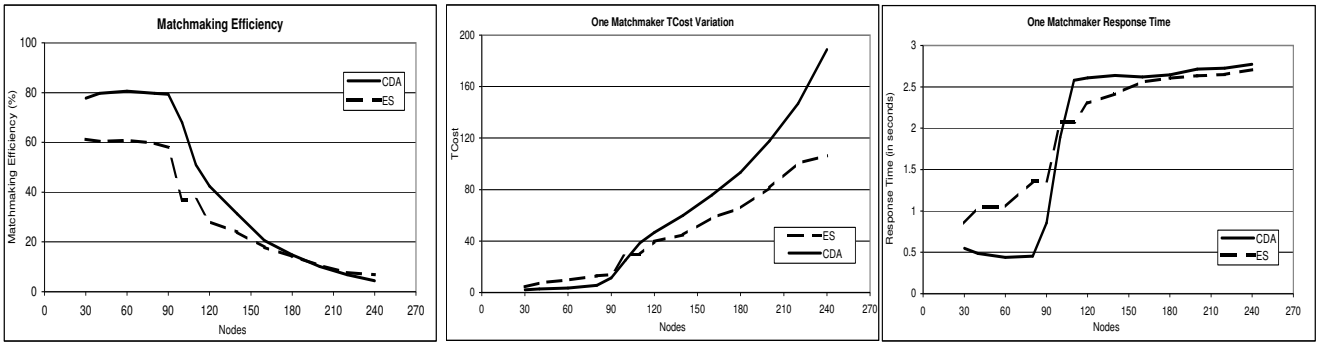
Experiments were performed to study the matchmaker(s) behavior in terms of matchmaking efficiency, matchmaker response time and the matchmaker workload (TCost) with varying numbers of participating nodes. The *matchmaking efficiency* is determined as $(\sum matched\ message/\sum message)*100$. The *response time* is calculated from the time a message is received by the matchmaker to the time the matchmaker makes a decision for the message. The *TCost* is explained in Section-2

3.1. Experimental Results

The first set of experiments was executed to determine the TCost *upper threshold* value of one matchmaker. The second set of experiments was executed with more than one matchmakers by applying the TCost *upper threshold* value. The effects on matchmaking efficiency, TCost and response time by dynamically introducing/removing the matchmaker(s) in both approaches were studied in an ad hoc grid environment. The experimental results in our approach (CDA) are compared with the Eager Scheduling (ES) [2, 7] approach.

3.2. One Matchmaker

Figure-2a depicts the matchmaking efficiency of one matchmaker. The matchmaking efficiency is 20% higher



(a) Matchmaking Efficiency (b) Matchmaker TCost (c) Matchmaker Response Time

Figure 2: One Matchmaker Throughput Compared in Continuous Double Auction (CDA) and in Eager Scheduling (ES)

in our approach as compared to ES for 20-90 nodes. As request/offer messages are sorted by the matchmaker before finding a match in our approach, therefore the matchmaking efficiency is higher in our approach than ES. We observe a fast decreasing trend in matchmaking efficiency of matchmaker for $N \geq 90$ nodes (Where N is the number of nodes) in both approaches. As more messages are received by the matchmaker than processed for $N \geq 90$ nodes, therefore the TTL of the majority of the messages expires before matchmaker can find a match for them. Consequently, the matchmaking efficiency of the matchmaker decreases with the same pattern in both approaches for $N \geq 90$ nodes.

Figure-2b depicts the TCost of the matched messages as a function of the number of nodes to one matchmaker for both approaches. TCost variation is less for 20-90 nodes for both approaches. As messages are unsorted in ES-based approach so a message may have to wait longer in ES approach. The longer waiting time results in a higher TCost value in the ES approach. As the ratio of incoming to the processed messages increases for $N \geq 90$ nodes, the TCost increases at higher rate for $N \geq 90$ nodes. Since the matchmaking efficiency is less for $N \geq 90$ nodes, we don't observe a steep increasing trend in TCost for both approaches. Moreover, as the messages are sorted by the matchmaker before finding a match in our approach, the sorting process results in higher TCost for $N \geq 90$ nodes in our approach as compared to the ES.

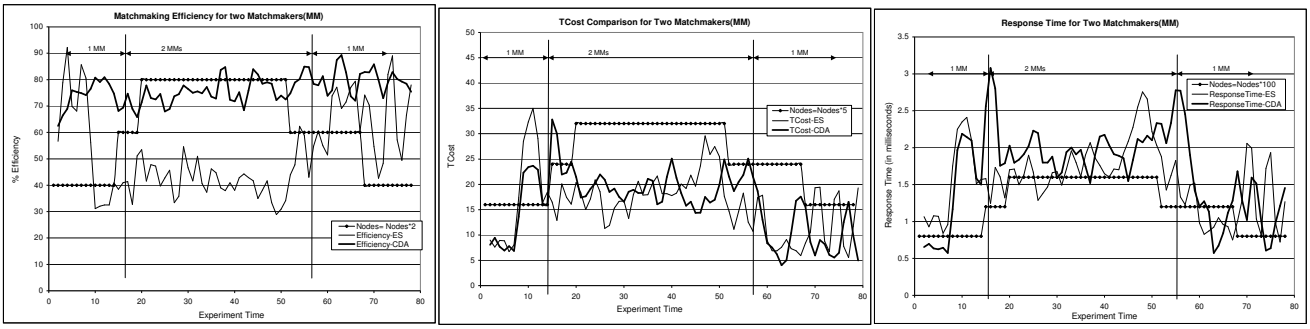
Figure-2c depicts the response time of matched messages as a function of the number of nodes to one matchmaker for both approaches. Initially response time is higher in CDA for less number of nodes. As the matchmaker has fewer messages in its request and offer buffers with less number of participating nodes ($N \leq 30$), therefore a message may have to wait longer for getting matched. Response time decreases with increasing number of nodes (from 30 to 70 nodes). This decrease is expected, as more messages are available in request and offer buffers of the matchmaker, messages get matched in less time. Since the matchmaker has to process more messages with the increasing number of participating nodes, therefore the response time increases for $N \geq 70$ nodes. The difference of response time, for 20-70 participating nodes, in both approaches, is due to the sorted and unsorted messages in CDA and in ES respectively.

The steep increase of response time in both approaches, for 80-100 participating nodes, can be linked with the maximum threshold of matchmaking efficiency (See Figure-2a) of the matchmaker. The matchmaking efficiency drops down for $N \geq 100$ nodes and the response time becomes approximately constant. As the matchmaking efficiency of one matchmaker drops down for $N \geq 90$ nodes, we don't observe the same steep increasing trend in response time or in TCost.

It can be concluded from the above discussion that for $N \geq 90$ nodes can be considered as the *TCost upper threshold* for one matchmaker with this particular experimental setup. The TCost upper threshold value for one matchmaker with this experimental setup was 25. This upper threshold indicates that a single matchmaker is not sufficient to entertain all the consumer/producer nodes in the ad hoc grid**.** The upper threshold of a matchmaker also implies that a new matchmaker is required to ensure the matchmaking capacity and efficiency of the matchmaking process. This trend is observed in both approaches.

3.3. Multiple Matchmakers

Comparison of both approaches (CDA and ES) with two matchmakers is presented in Section-3.4. Experimental results with three matchmakers based on CDA are discussed in Section-3.5. These experiments were executed with varying numbers of nodes in the ad hoc grid. The matchmaker(s) workload was varied in such a way that the complete proposed model can be analyzed. All these experiments were started with one matchmaker. The workload of the matchmaker was increased so that TCost *upper threshold* was



(a) Matchmaking Efficiency (b) Matchmaker TCost (c) Matchmaker Response Time

Figure 3: Comparison of Two Matchmakers Throughput in CDA and ES

reached and a second matchmaker was introduced. First matchmaker forwarded the incoming message, above its TCost *upper threshold*, to the new matchmaker. The same strategy was used to introduce the third matchmaker. When the TCost value dropped below its *upper threshold* for a matchmaker then the matchmaker(s) were demoted as a normal node in the ad hoc grid.

3.4. Two Matchmakers

Figure-3b depicts the TCost variation of two matchmakers in both approaches with varying number of nodes. The number of nodes is scaled down to represent properly in the graph (scaling factor on Figure-3b). When TCost upper threshold was approached then the second matchmaker was introduced in the ad hoc grid. First matchmaker forwarded all the messages to the new matchmaker that were above its TCost *upper threshold*. The workload of the first matchmaker and hence overall TCost was decreased below the upper threshold of one matchmaker by introducing the second matchmaker. The TCost varies temporarily, whenever there is change in workload of the matchmaker(s) or in the number of matchmakers. Rate of variation is dependent upon the variation of workload. TCost was 50 and 80 for $N \leq 160$ nodes with one matchmaker in CDA and in ES respectively (Figure-2b).

Figure-3a depicts the matchmaking efficiency of both approaches (CDA and ES) with two matchmakers. First we discuss matchmaking efficiency in our framework. Matchmaking efficiency reduced from 80% to 68% when number of nodes increased from 80 to 120 (Figure-3a). This phenomenon indicated the need of a new matchmaker. Matchmaking efficiency increased with the introduction of the second matchmaker. The matchmaker(s) became temporarily unstable with sudden variation (number of nodes increased from 120-160) of the workload on the matchmaker(s). However the matchmaker(s) achieve their normal matchmaking efficiency.

By introducing the second matchmaker, overall matchmaking efficiency remained between 72% -80% in our framework for $N \leq 160$ nodes. The variation in matchmaking efficiency between 72%-80% is due to the randomly generated values of the request/offer parameters in these experiments. The 3% decrease in matchmaking efficiency is due to the communication overhead among the two matchmakers.

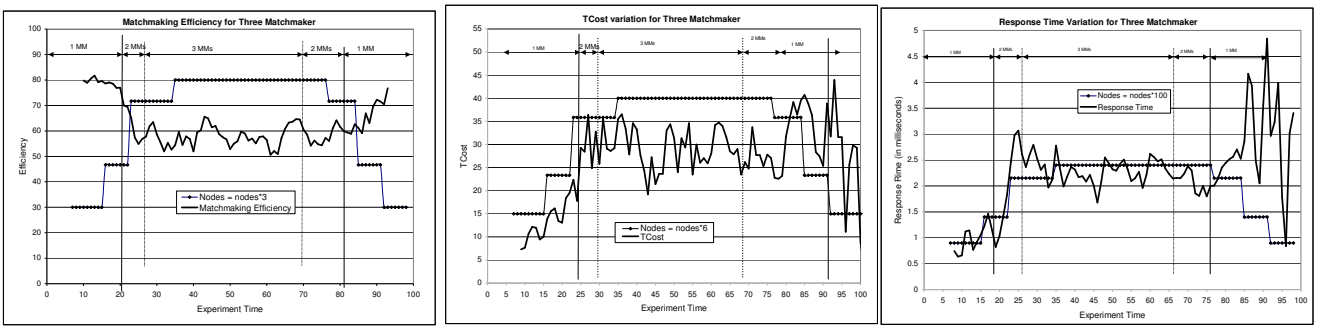
The matchmaking efficiency for ES approach varies between 40%-50% for $N \leq 160$ nodes. Matchmaking efficiency for ES approach was about 20% with one matchmaker for the same number of nodes (Figure-2a).

Figure-3c depicts the response time variation with two matchmakers in both approaches with varying numbers of nodes. The number of nodes is scaled up to represent properly in the graph (scaling factor on Figure-3c). Response time is less than 2 seconds for the higher matchmaking efficiency with two matchmakers (Figure-3a) as compared to approximately the same response time with decreased matchmaking efficiency for one matchmaker (Figure-2a) .

3.5. Three Matchmakers

The effects on matchmaking efficiency, TCost and matchmaker response time for 240 nodes with 3 matchmakers by applying our model are plotted in Figures-4a,4b,4c respectively. Same process was followed to introduce the third matchmaker that was used to introduce the second matchmaker. Second matchmaker forwarded all the incoming messages that were above its TCost upper threshold, to the third matchmaker.

The matchmaking efficiency variation (Figure-4a) follows the same pattern as was observed with two matchmakers. The matchmaking efficiency varies temporarily, whenever there is change in workload of the matchmaker(s) or in the number of matchmakers. Rate of variation is dependent upon the variation of workload. The decrease in matchmaking efficiency with three matchmaker as compared to two matchmakers is due to increased communication among



(a) Matchmaking Efficiency (b) Matchmaker TCost (c) Matchmaker Response Time

Figure 4: Throughput of Ad Hoc Grid with Three Matchmakers Using CDA

matchmakers.

The average TCost value for 240 nodes with 3 matchmakers is 26 (Figure-4b). Whereas the TCost value was 160 with one matchmaker for same workload (Figure-2b). The response time of the matched message is approximately equal to the response time of matched messages with one matchmaker (Figure-4c). These values of TCost and response time are with increased matchmaking efficiency as compared to the matchmaking efficiency of one matchmaker for the same workload.

It can be concluded from the above experimental results that the capability of the ad hoc grid to introduce new matchmakers, according to the workload of the ad hoc grids, has a stabilizing effect on the TCost and matchmaker response time without affecting negatively the matchmaking efficiency. In the proposed mechanism, we ensure, matchmaker work load and matchmaker response time become invariant of the scale of the ad hoc grid. Consequently, we can argue that the proposed framework can be used for ad hoc grid of any size.

4. Conclusion

A dynamic, self-organizing model to dynamically segment and desegment the ad hoc grid by introducing/removing new matchmaker(s) according to the workload (TCost) of the existing matchmaker(s) was presented in this paper. TCost upper threshold was calculated for one matchmaker in the ad hoc grid. The TCost upper threshold value was used to dynamically segment the ad hoc grid by introducing new matchmaker(s) and merging back the segments by demoting the matchmaker(s) as the normal nodes in the ad hoc grid. The TCost upper threshold was also used to share the workload of one matchmaker with the other matchmaker. A matchmaker forwarded all the new messages above TCost upper threshold to the other matchmaker. Experimental results indicate that the proposed model is scalable and does not increase the administrative complexity of the ad hoc grid. Future research will focus on relaxing the simplifying assumptions. We also plan to look for a dynamic TCost upper threshold calculation mechanism that will be independent of any specific experimental context.

References

[1] K. Amin, G. von Laszewski, and A. R. Mikler. Toward an architecture for ad hoc grids. In *ADCOM*, 2004.

[2] D. P. Anderson. BOINC: A system for public-resource computing and storage. In *Proceedings of Grid*, 2004.

[3] A. R. Butt, R. Zhang, and Y. C. Hu. A self-organizing flock of condors. *JPDC*, 66(1):145–161, 2006.

[4] A. S. Cheema, M. Muhammad, and I. Gupta. Peer-to-peer discovery of computational resources for grid applications. In *Proceedings of GRID*, 2005.

[5] A. Chien, B. Calder, S. Elbert, and K. Bhatia. Entropia: Architecture and performance of an enterprise desktop grid system. *JPDC*, 63(5):597–610, May 2003.

[6] S. Choi et al. Adaptive group scheduling mechanism using mobile agents in peer-to-peer grid computing environment. *Applied Intelligence*, 25(2):199–221, 2006.

[7] C. Germain, V. Néri, G. Fedak, and F. Cappello. Xtremweb: Building an experimental platform for global computing. In *The Proceedings of GRID*, 2000.

[8] R. Gupta et al. CompuP2P: An architecture for internet computing using peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems*, 17(11):1306–1320, 2006.

[9] D. Hausheer and B. Stiller. Decentralized auction-based pricing with peermart. In *Processdings of IM*, 2005.

[10] A. Iamnitchi and I. Foster. On fully decentralized resource discovery in grid environments. In *GRID*, 2001.

[11] J. Kim et al. Using content-addressable networks for load balancing in desktop grids. In *HPDC*, 2007.

[12] A. Patil, D. A. Power, and J. P. Morrison. Economy-based computing with webcom. *IJCISE*, 1(2):82–89, 2007.

[13] B. Pourebrahimi, K. Bertels, G. Kandru, and S. Vassiliadis. Market-based resource allocation in grid. In *eSicence*, 2006.

[14] M. I. Shaik, S. M. S. Bhanu, and N. P. Gopalan. Distributed grid resource discovery with matchmakers. In *SKG*, 2006.

[15] D. Zhou et al. Wavegrid: A scalable fast-turnaround heterogeneous peer-based desktop grid system. In *IPDPS*, 2006.