Hybrid Resource Discovery Mechanism in Ad Hoc Grid Using Structured Overlay

Tariq Abdullah[1], Luc Onana Alima[2], Vassiliy Sokolov[1],
David Calomme[2], Koen Bertels[1]

[1]Computer Engineering Laboratory, EEMCS, Delft University of Technology,
Mekelweg 4, 2624 CD, Delft, The Netherlands
[2]Distributed Systems Laboratory, University of Mons, Belgium
[1]{m.t.abdullah, v.sokolov, k.l.m.bertels}@tudelft.nl, [2]{luc.onana, david.calomme}@umh.ac.be

Abstract. Resource management has been an area of research in ad hoc grids for many years. Recently, different research projects have focused resource management in centralized, decentralized or in a hybrid manner. In this paper, we discuss a micro economic based, hybrid resource discovery mechanism. The proposed mechanism focuses on the extension of a structured overlay network to manage the (dis)appearance of matchmakers in the grid and to route the messages to the appropriate matchmaker in the ad hoc grid. The mechanism is based on the emergent behavior of the participating nodes and adapts with respect to changes in the ad hoc grid environment. Experiments are executed on PlanetLab to test the scalability and robustness of the proposed mechanism. Simulation results show that our mechanism performs better than previously proposed mechanisms.

1 Introduction

Ad hoc grid (also called Public Resource Computing [1], Desktop Grid Computing [2], or Volunteer Computing) has autonomous, geographically distributed, and heterogeneous nodes with intermittent participation. The intermittent participation of nodes results in varying workload of the resource manager. Furthermore, different underlying network infrastructures and varying access/use policies of the nodes increase the administrative complexity of the resource management system in the ad hoc grid.

There exist different resource discovery techniques for ad hoc grid. Centralized approaches are efficient and consume less time to find a resource. But these approaches have a single point of failure and are not scalable. In contrast to centralized approaches, peer to peer (P2P) approaches are scalable and depict low management complexity. Whereas, the P2P approaches are less efficient and can have time and network overhead while finding a resource. P2P and centralised systems are often considered to be mutually exclusive and reside on both ends of an infrastructural spectrum. We consider them to be part of a continuum where the system should be capable of restructuring itself in either of these states, or any intermediate state between those two extremes.

This paper is an extension of the work presented in [3], where, we proposed a resource management mechanism that enabled the ad hoc grid to self-organize according to the workload of the resource manager (referred to as *matchmaker* hereafter). The

mechanism is based on the emergent behavior of the participating nodes and adapts itself with respect to the changes in the ad hoc grid environment. In [3], we assumed the availability of a structured overlay network to handle node join/leave, de(segmentation) of ad hoc grid, message routing among matchmakers, node to matchmaker message exchange and matchmaker to node message exchange. Furthermore, a node belonged to only one matchmaker and there was no mechanism to find a new responsible matchmaker in the event of a matchmaker failure. We now address some of the open issues reported there.

This paper focuses on some of the open issues mentioned above. The main contributions of the paper are as follows. The proposed extension defines the algorithms for node joining, for finding a responsible matchmaker by a joining/existing node, for ad hoc grid segmentation (by promoting nodes as matchmakers), and for ad hoc grid desegmentation (by demoting matchmakers as normal nodes). Experiments are executed, on PlanetLab [4], to verify that the proposed extension enables ad hoc grid to self-organize according to the workload of the resource manager. The proposed extension makes our hybrid resource discovery mechanism, for ad hoc grid, dynamic and flexible.

The rest of the paper is organized as follows. An overview of the related work and necessary background knowledge to understand the proposed model is in section-2. Section-3 describes the proposed extension of the structured overlay network. Section-4 explains the experimental setup and discusses the results. Section-5 concludes the paper and describes our future work.

2 Related Work & Background Knowledge

In literature, we find different solutions for load balancing in distributed systems. Mercury [5] used node leave/rejoin for load balancing. The overloaded node leaves and then joins the ring as a neighbor of a lightly loaded node. Node leave/rejoin introduces message overhead. Cai et al. [6] used customized hashing function. The customized hashing functions required a prior knowledge of the attributes value distribution. Mastroianni et al. [7] used unstructured P2P networks to construct a super-peer model for resource discovery in grids and used experience based query forwarding. Attribute encoding of static or dynamic computational resource information for resource discovery in DHT based overlay networks is also studied in [8]. The majority of the encoded attributes may be mapped to a small set of nodes in the overlay network, therefore attribute encoding may result in a load imbalance condition. Padmanabhan et al. [9] proposed a self-organized grouping method that formed and maintained autonomous *resource groups*. These resource groups are formed according to some pre-specified resource characteristics and each group contained a set of similar resources. The main drawback of their work is that there is no load balancing mechanism among the groups formed. Butt et al. [10] implemented a P2P based Condor flocking to share resources in different Condor pools. They did not consider the dynamic introduction/removal of Condor pools or the workload condition of a Condor pool manager.

The above discussed approaches used different ways to distribute the workload of one matchmaker among multiple matchmakers. These include node leave/rejoin [5], customized hash functions [6], super peer model on top of unstructured networks [7],

or attribute encoding [8]. All these approaches attempt to balance the workload of the resource manager/ matchmaker by sharing the workload, they may end up with an inappropriate infrastructure for the given state of the grid.

2.1 Continuous Double Auction based Resource Allocation

Our ad hoc grid consists of autonomous, dynamic, volatile and loosely connected nodes that can join, leave or change their roles whenever needed. Each node is composed of three agents: *Consumer*, *Producer* and *Matchmaker*. Structure of these agents is depicted in Figure-1. The *Resource/Task Manager* module of the producer/consumer agent prepares *asks/bids*. The resource offers are called *asks* and the resource requests are called *bids*. An *ask* is defined by attributes like resource quantity, Time To Live (TTL) and resource price in this paper. A *bid* is represented by resource quantity, job execution time, bid price and Time To Live (TTL) in this paper.

The matchmaker uses Continuous Double Auction (CDA) to perform resource allocation in its *MatchMake* module. The matchmaker agent continuously receives *asks/bids* from producer/consumer agents and stores the received asks/bids in the offer/request repositories, maintained by the *Repository Manager* module. The matchmaker finds the matches between the producers and consumers by matching offers (starting with lowest ask price and moving up) with requests (starting with highest bid price and moving down). The matchmaker matches a compatible ask/bid pair immediately. A compatible ask/bid is a resource offer/request pair where resource request (*bid*) constraints are satisfied by the matching resource offer (*ask*). The received ask/bid remains in the offer/request repositories of the matchmaker till the expiry of its Time To Live (TTL). The *Segmenter Module* performs segmentation and desegmentation by promoting and demoting the matchmakers. All consumer/producer to matchmaker and vice verse communication is done through the *communication* module of the respective consumer/producer or matchmaker agent. The communication module uses underlying structured overlay network for communication between the consumer/producer or matchmaker agent.

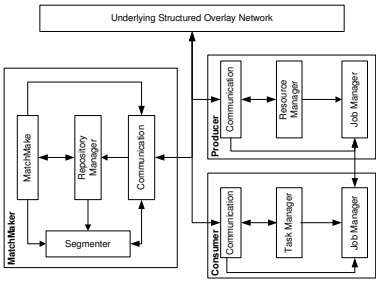


Fig. 1: System Architecture

The *Job Manager Module* of consumer/producer agents is responsible to execute the consumer jobs and return the computation results to the consumer, after receiving a

matchmaking notification from the matchmaker. The producer/consumer does not have a system wide (global) knowledge and are not aware of the other's ask/bid prices. The producer/consumer use a dynamic history based pricing function to calculate the ask/bid price for each offered/requested resource [3].

2.2 Transaction Cost (TCost) Threshold Determination

The matchmaker attaches a transaction cost (TCost) to each offer/request, which reflects the workload of the matchmaker. The node submitting a request/offer is supposed to pay this TCost to the matchmaker. Every node is given an initial budget which can be used to this purpose. Formula for calculating TCost, discussion of TCost upper and lower thresholds for promoting normal nodes as matchmaker and demoting underloaded matchmaker nodes as normal nodes are described in our previous paper [3].

3 Proposed Algorithm

There can be N nodes in our experiments. Each node can play the role of a consumer, producer or a matchmaker. A *producer node* offers its available resources (such as CPU, memory, disk space or bandwidth). A *consumer node* requests for the desired resources to execute its job. The node playing the role of a mediator between the consumer and the producer nodes are named as resource allocators or *matchmaker*s in this work. The matchmaker receives offers and bids of resources from the nodes in the ad hoc grid, which are then matched by the matchmaker. Each node is assigned a unique node identifier (nodeID). As the proposed algorithm is an extension of a structured P2P overlay network, principally any structured P2P overlay network, can be used to implement the proposed algorithm. We used Pastry [11] to implement the proposed extension. As the Pastry identifier space can have $2^{128} - 1$ unique identifiers, the maximum number of nodes (N) in our ad hoc grid can also be $2^{128} - 1$. There can be a maximum of M matchmakers out of N nodes.

The whole identifier space is divided into zones. Each zone has a responsible matchmaker. Each joining consumer/producer/matchmaker node is provided with M, N and the *zone number* to which the node belongs to. It is ensured that each consumer/producer node is under the responsibility of a matchmaker. When a matchmaker becomes overloaded then it promotes its predecessor matchmaker node to perform matchmaking. The consumer/producer nodes under the responsibility of overloaded matchmaker are now under the responsibility of the predecessor matchmaker. In case, the predecessor matchmaker is already performing matchmaking (i.e active) then the excess workload is forwarded to the successor matchmaker of the overloaded matchmaker. The algorithm for matchmaker promotion is explained in Section-3.1.

Conversely, when a matchmaker is underloaded then it demotes itself and informs its predecessor and successor matchmakers about the change in its matchmaking status. The successor matchmaker of the demoted matchmaker becomes the responsible matchmaker for consumer/producer nodes that were previously under the responsibility of the demoted matchmaker. After demoting itself, the demoted matchmaker will forward the request/offer messages to its successor matchmaker node. The demoted

matchmaker also informs the consumer/producer node, under its responsibility, about the change of its matchmaking status and about their new matchmaker. The matchmaker demotion (desegmentation) algorithm is explained in Section-3.2.

A consumer/producer node finds its responsible matchmaker node with the provided information, after joining the ad hoc grid. In case there is only one matchmaker in the ad hoc grid then it becomes responsible matchmaker for all the consumer/producer nodes. The consumer/producer node can submit request/offer to the matchmaker node after finding the responsible matchmaker node. Each matchmaker node maintains matchmaking status information (active/inactive) about its predecessor and successor matchmaker nodes. The matchmaker does so by exchanging matchmaking status information with its successor and predecessor nodes. The algorithms for node joining and matchmaker discovery are explained in Section-3.3 & 3.4 respectively.

3.1 Matchmaker Promotion: Segmentation

This algorithm is executed by an overloaded matchmakers in the ad hoc grid. An overloaded matchmaker (say M_i is matchmaker for zone i) promotes its predecessor matchmaker node (say M_{i-1}).

The newly promoted matchmaker (M_{i-1}) changes its matchmaking status to active, as depicted in Figure-2a & Figure-2b. The matchmaker (M_{i-1}) updates its predecessor matchmaker (M_{i-2}) about the change in its matchmaking status. The matchmaker (M_{i-1}) is now ready to perform matchmaking for the consumer/producer nodes belonging to zone $i-1$.

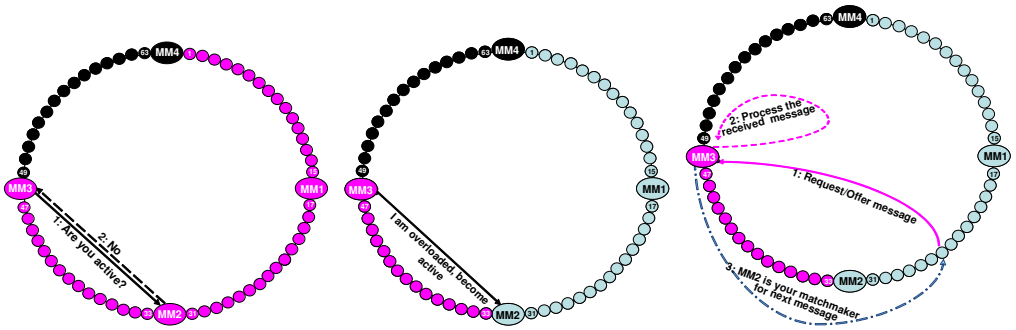
The consumer/producer nodes belonging to zone $i-1$ are still unaware of the change of their new matchmaker in their zone. We applied "correction on use" policy to update the consumer/producer nodes belonging to zone $i-1$. The matchmaker M_i , after promoting its predecessor (M_{i-1}) as active, will process the currently received request/offer message from nodes belonging to zone $i-1$ and will update the respective node, in zone $i-1$, about their new matchmaker (M_{i-1}). The consumer/producer nodes in zone $i-1$ change their responsible matchmaker to M_{i-1} from M_i and send the request/offer messages to their new matchmaker (M_{i-1}). This process is graphically illustrated in Figure-2c.

If the predecessor matchmaker (M_{i-1}), of matchmaker M_i, is already active then the overloaded matchmaker node (M_i) will forward its excess workload to its successor matchmaker (M_{i+1}). The matchmaker promotion algorithm is listed in Algorithm-1.

As stated before, there can be N nodes (consumer/producer/matchmaker) in the ad hoc grid and there can be a maximum of M matchmakers ($M < N$), out of N nodes. The ad hoc grid is thus divided into maximum of $(N/M) - 1$ zones. This zone information is only effective when there is an active matchmaker in that zone. If the matchmaker for a zone i does not exist or is inactive then the consumer/producer nodes, continue looking for an active matchmaker in the successor zones of zone i (refer Section-3.4 for details). The number of effective zones increases with the promotion of matchmakers and decreases with the demotion of matchmakers, explained in Section-3.2. In this way, the matchmaker promotion results in the segmentation of the ad hoc grid and the matchmaker demotion results in desegmentation of ad hoc grid.

Algorithm 1 Promote Matchmaker

1:IF(M_i is overloaded) THEN
2: M_i Query M_{i-1}matchmaking status
3:IF(M_{i-1} $matchmakingstatus\ is\ false$) THEN
4: M_i change M_{i-1} matchmaking status to TRUE
5: M_{i-1} update its changed matchmaking status to M_{i-2}
6: M_i update matchmaker change to consumer/producer nodes in zone $i-1$
7:ELSE
8: M_i Forwarded excess workload to M_{i+1}
9:END IF
10:END IF



(a) (b) (c)

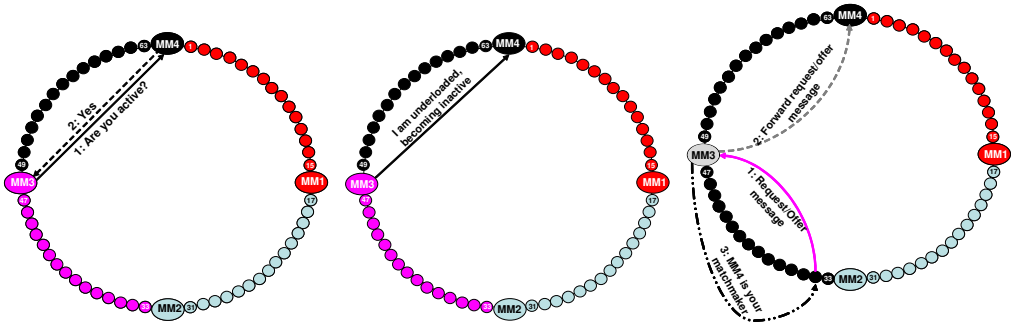
Fig. 2: Promote a Matchmaker

3.2 Matchmaker Demotion: Desegmentation

This algorithm is executed by an underloaded matchmaker in the ad hoc grid. An underloaded matchmaker (say M_i is matchmaker for zone i) demotes itself by changing its matchmaking status. The demoted matchmaker updates its predecessor matchmaker node (M_{i-1}) and successor matchmaker node (M_{i+1}) about the change in its matchmaking status, as depicted in Figure-3a & Figure-3b.

Algorithm 2 Demote Matchmaker

1:IF(M_i is underloaded) THEN
2: M_i change its matchmaking status to FALSE
3: M_i update its changed matchmaking status to M_{i-1}
4: M_i update its changed matchmaking status to M_{+1}
5: Notify consumer/producer nodes about change of matchmaker
6:END IF



(a) (b) (c)

Fig. 3: Demote a Matchmaker

The "correction on use" policy is also applied to update the consumer/producer nodes in zone i about the change of their responsible matchmaker. The matchmaker node M_i, after demoting itself, will forward the currently received request/offer messages, from nodes in zone i , to its successor matchmaker node (M_{i+1}). The matchmaker M_i also sends a message to the respective consumer/producer node, about the change of matchmaker from M_i to M_{i+1}. The consumer/producer nodes in zone i change their responsible matchmaker to M_{i+1} from M_i and send the request/offer messages to their new matchmaker (M_{+1}). This process is graphically illustrated in Figure-3c. The matchmaker demotion algorithm is listed in Algorithm-2.

3.3 Node Joining Algorithm

Each joining consumer/producer/matchmaker node is provided with M, N and the zone number to which the node belongs to. We used Pastry node join protocol [11] for node joining in our ad hoc grid. After joining the ad hoc grid, the consumer/producer and matchmaker node performs different set of actions.

The consumer/producer node discovers its responsible matchmaker node with the provided information (refer Section-3.4). It can send the resource request/offer messages after discovering the responsible matchmaker.

A matchmaker node maintains matchmaking status information (active/inactive) about its predecessor and successor matchmaker nodes. When a matchmaker node joins the ad hoc grid then it exchanges predecessor/successor matchmaking status information. The algorithm for joining the ad hoc grid is listed in Algorithm-3.

3.4 Matchmaker Discovery

As stated before, there can be N nodes (consumer/producer/matchmaker) in the ad hoc grid and there can be M matchmakers, out of N nodes. (N being the maximum number of nodes and M being the maximum number of matchmakers in ad hoc grid). The first node of each zone is considered the matchmaker for the previous zone. In this way each consumer/producer node is under the responsibility of a matchmaker.

Algorithm 3 Node Join Algorithm

1:Node join the ad hoc grid into zone i
2:IF ($Joining\,node\,is\,Consumer/Producer$) THEN
3: CALL $Find-Responsible-Matchmaker$
4:ELSE IF ($Joining\,node\,is\,Matchmaker$) THEN
5: Query predecessor node's matchmaking status
6: Query successor node's matchmaking status
7: END IF
8:END IF

Algorithm 4 Discovering Responsible Matchmaker

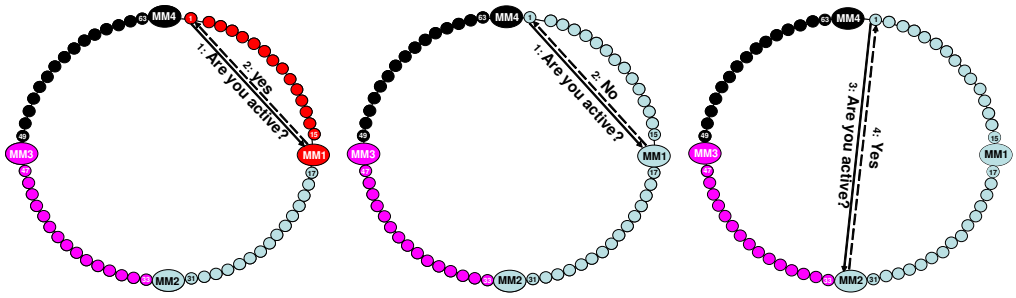
1:Consumer/producer node join the ad hoc grid into zone i
2:IF (M_i is active) THEN
3: set M_i as responsible matchmaker
4:ELSE
5: set $I = i + 1$
6: set $counter = 1$
7: WHILE ($counter < M$)
8: Query $Matchmaker_I$
9: IF ($MatchMaker_I$ is active) THEN
10: set $MatchMaker_I$ as responsible matchmaker
11: BREAK
12: END IF
13: $I = I + 1$
14: $counter = counter + 1$
15: END WHILE
16:END IF

The "Matchmaker Discovery" algorithm is executed by the joining consumer/producer node for discovering its responsible matchmaker node. If the matchmaker (say M_i) for zone i is active then the joining consumer/producer node sets this matchmaker (M_i) as its responsible matchmaker (refer Figure-4a). The consumer/producer node will send all its resource requests/offers messages to this matchmaker.

If the matchmaker for a zone does not exist (say matchmaker M_i for zone i does not exist), then the consumer/producer node checks for the successor matchmaker (M_{i+1}) of M_i. The consumer/producer node continues searching for an active matchmaker node, until it finds one (refer Figure-4b & Figure-4c). It is important to mention that the algorithm for finding a responsible matchmaker by a newly joining node does not cover matchmaker node failure. The matchmaker failure handling will be in our future work. The algorithm for finding the responsible matchmaker by a newly joining node is listed in Algorithm-4.

4 Experimental Setup & Results

The purpose of the experiments is to validate the proposed algorithm and to assess how it allows self organization. To this purpose, the proposed extension is implemented on



(a) (b) (c)

Fig. 4: Discovering a Responsible Matchmaker

Pastry [11] and is tested on PlanetLab [4]. Pastry[11] is a completely decentralized, scalable and self-organizing structured overlay network with respect to node join/leave or departure and overlay network maintenance. Although we used Pastry in this work, it can be replaced with any other structured P2P overlay network.

The message routing process in Pastry is briefly described as follows. When a node receives a message from another node, it first checks whether the destination nodeID falls within its leaf set or not. If the destination nodeID exists in its leaf set, then the message is directly forwarded to the destination node, otherwise the routing table of the node that received the current message is used and the message is forwarded to the node that shares a common prefix with the destination node ID by at least one digit. The routing performance of Pastry is scalable and the maximum expected number of routing steps is $\log_{2^b} N$ [11], where b is a Pastry configuration parameter with $b = 4$ as the default value and N is the total number of nodes in the Pastry ring. We refer to [11] for Pastry details.

PlanetLab is an open, geographically distributed computing environment/test-bed. PlanetLab makes it possible to demonstrate the scalability and robustness of a proposed mechanisms given real network traffic, generated from real users while considering the inherent unpredictability of the Internet. It [4] currently includes more than 893 machines, spanning 461 sites and 40 countries. Over 600 research projects, called *service*, are currently running on the PlanetLab. Each project runs in its own network of virtual machines, called *slice*. A slice isolates projects from each other. Moreover there is no centralized control over resources in PlanetLab. Data represented in the paper is extracted after $1/4^{th}$ of the experiment time has elapsed.

The matchmaker's throughput is determined in terms of its matchmaking efficiency, Transaction Cost (TCost) and the response time. The matchmaking efficiency is calculated in terms of request/offer utilization. The request utilization is calculated as

$$(\sum MatchedRequest/\sum Request) * 100$$

and the offer utilization is calculated as

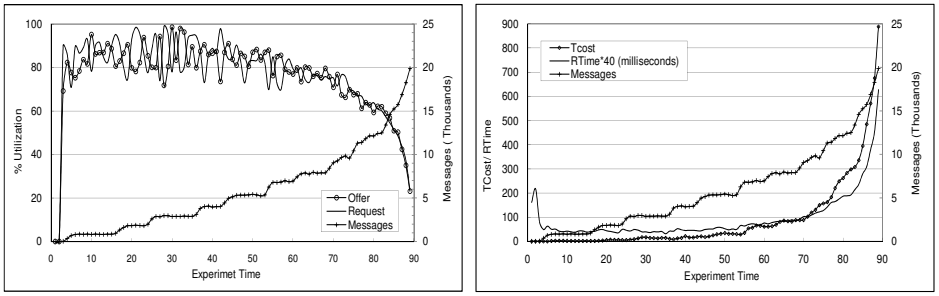
$$(\sum MatchedOffer/\sum Offer) * 100$$

The Transaction Cost (TCost) calculation and TCost upper and lower threshold determination is discussed in Section-2.2. The matchmaker response time is the time interval between receiving a request/offer message by the matchmaker and the time instance when matchmaker has processed a request/offer message.

The experiments are executed with varying numbers of the participating nodes. The number of the nodes is varied from 15-650 and the number of matchmakers is varied from 1-5. The workload is managed in such a way that the maximum number of matchmakers are needed and then gradually decreased to provoke the demotion of matchmakers to normal nodes again. The Job execution time, job deadline and required/offered resource amount are randomly generated from a predefined range. Quantity of requested/offered computational resource is varied for each request/offer message. The TTL of a request/offer message is fixed to 10000 milliseconds, reflecting the delays observed in PlanetLab.

4.1 Experimental results

In this section, we first present the experimental results with one matchmaker. Later, we present experimental results with multiple matchmakers to show the effect of dynamic promotion and demotion of matchmaker(s). The effect of promotion (segmentation of ad hoc grid) and demotion (desegmentation of ad hoc grid) of matchmaker(s), in ad hoc grid, on Transaction Cost (TCost), matchmaker response time and consumer/producer utilization are compared with the ad hoc grid having only one matchmaker.



(a) Request/Offer Utilization of One Matchmaker with increasing workload

(b) TCost & Response Time of One Matchmaker with increasing workload

Fig. 5: Ad Hoc Grid with One Matchmaker

Figure-5a depicts the effect of request/offer utilization with increasing workload of one matchmaker. *Request,* represents the consumer utilization and *offer* represents the producer utilization in Figure-5a. Whereas, Figure-5b depicts the effect of the consumer/producer TCost and response time variation with increasing workload of one matchmaker.

Figure-5a & 5b indicate that the TCost and the response time increase with increasing workload and consumer/producer utilization decreases with the increasing work-

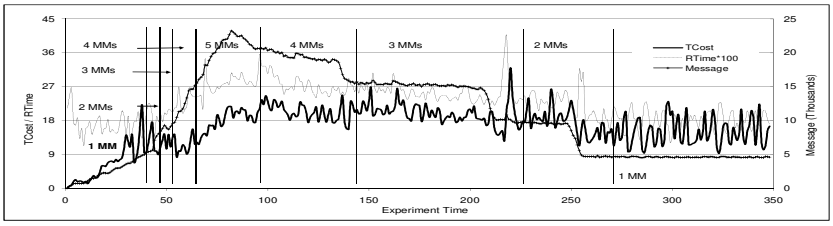


Fig. 6: TCost & Response Time with Multiple Adaptive Matchmakers

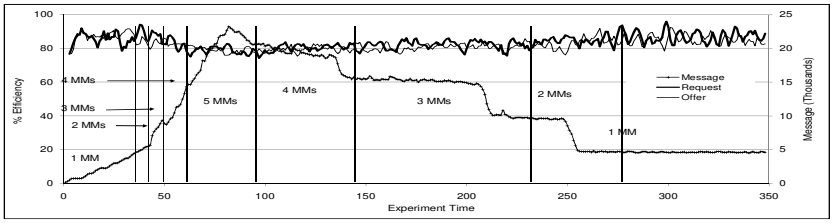


Fig. 7: Matchmaking Efficiency with Multiple Adaptive Matchmakers

load of one matchmaker. The increasing trend of TCost and response time, and decreasing trend of the consumer/producer utilization represent that the matchmaker is overloaded and is unable to maintain its matchmaking capacity. It also implies that the consumer/producer have to pay a higher TCost for availing the matchmaking service. The matchmaker needs additional matchmakers, at the point, when its matchmaking efficiency starts to decrease. This is the point, where our mechanism becomes useful.

Figure-6 depicts the TCost and Response Time variation in presence of multiple adaptive matchmakers. The TCost value keeps on increasing with increasing matchmaker workload. A new matchmaker is introduced when the first matchmaker reached its TCost threshold value. When a new matchmaker is introduced, the TCost value decreases. The opposite is observed when a matchmaker is removed. Evidently, the TCost also increases/decreases temporarily reflecting changes in the workload of the overall grid. Overall and compared with the TCost evolution with a single matchmaker and increasing workload (Refer Figure-5b), the TCost remains relatively stable and does not increase (Refer Figure-6). Instead of going up with an increasing workload, the increase of the number of matchmakers has a stabilizing effect on the response time. This is exactly what was expected.

Figure-7 depicts the matchmaking efficiency with multiple adaptive matchmakers. The matchmaking efficiency remains 80% with increasing workload of the matchmaker. Whereas, in case of one matchmaker, matchmaking efficiency showed a continuous decreasing trend with increasing workload (Refer Figure-5a).

In conclusion, we can observe from the above experiments that the capability of the ad hoc grid to instantiate multiple matchmakers has a stabilizing effect on the TCost and response time without affecting negatively the offers/request utilization. This way, we guarantee that the transaction cost and response time become invariant to the scale

on which the grid is operating. These conclusions also confirm that the proposed algorithms for joining ad hoc grid, finding responsible matchmaker, promoting a matchmaker, demoting a matchmaker and message routing on P2P overlay network work as expected.

5 Conclusions

A dynamic, self-organizing mechanism, for a dynamic ad hoc grid infrastructure was proposed in this paper. The proposed mechanism focuses on the extension of a structured overlay network to manage the (dis)appearance of matchmakers in the grid and to route the messages to the appropriate matchmaker in the ad hoc grid. The mechanism dynamically segmented the ad hoc grid into multiple segments and merged the ad hoc grid segments according to the workload in the ad hoc grid. The matchmaking efficiency and capacity of the ad hoc grid was sustained, in spite of the increasing workload, by applying the proposed mechanism. All experiments were executed on PlanetLab providing a realistic platform for testing the proposed mechanisms. Our future research will focus on the failure handling of the matchmaker node. We will also look into a different QoS issues for the proposed mechanism.

References

1. Anderson, D.P.: BOINC: A system for public-resource computing and storage. In: 5th IEEE/ACM International Workshop on Grid Computing. (2004)
2. Chien, A., Calder, B., Elbert, S., Bhatia, K.: Entropia: Architecture and performance of an enterprise desktop grid system. JPDC **63(5)** (May 2003) 597–610
3. Abdullah, T., Sokolov, V., Pourebrahimi, B., Bertels, K.: Self-organizing dynamic ad hoc grids. In: 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops. (October 2008)
4. PlanetLab Online, https://www.planet-lab.org/
5. Bharambe, A.R., Agrawal, M., Seshan, S.: Mercury: supporting scalable multi-attribute range queries. In: The ACM SIGCOMM Conference. (2004)
6. Cai, M., Frank, M., Chen, J., Szekely, P.: MAAN: A multi-attribute addressable network for grid information services. Journal of Grid Computing **2(1)** (2004) 3–14
7. Mastroianni, C., Talia, D., Verta, O.: A super-peer model for building resource discovery services in grids: Design and simulation analysis. In: European Grid Conference. (2005)
8. Gupta, R., Sekhri, V., Somani, A.K.: Compup2p: An architecture for internet computing using peer-to-peer networks. IEEE Transactions on Parallel and Distributed Systems **17**(11) (2006) 1306–1320
9. Padmanabhan, A., Wang, S., Ghosh, S., Briggs, R.: A self-organized grouping (SOG) method for efficient grid resource discovery. In: 6th IEEE/ACM International Workshop on Grid Computing. (2005) 312–317
10. Butt, A.R., Zhang, R., Hu, Y.C.: A self-organizing flock of condors. JPDC **66**(1) (2006) 145–161
11. Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: 18th IFIP/ACM International Conference on Distributed Systems Platforms. (2001)