Tariq Abdullah

Mechanisms for Self-organizing Ad Hoc Grids

Mechanisms for Self-organizing Ad Hoc Grids

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.ir. K.C.A.M Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op woensdag 17 november 2010 om 12:30 uur
door

Tariq ABDULLAH
Master of Computer Science, IIUI Pakistan
geboren te Lawa, Pakistan

Dit proefschrift is goedgekeurd door de promotor:
Prof.dr.ir. H.J. Sips

Copromotor: Dr. K.L.M. Bertels

Samenstelling promotiecommissie:

Rector Magnificus	Voorzitter
Prof.dr.ir. H.J. Sips	Technische Universiteit Delft, The Netherlands, promotor
Dr. K.L.M. Bertels	Technische Universiteit Delft, The Netherlands, copromotor
Prof.dr. M. Boman	KTH Royal Institute of Technology
Prof.dr. J. Broeckhove	Universiteit Antwerpen
Prof.dr. F.M.T. Brazier	Technische Universiteit Delft, The Netherlands
Dr.ir. D.H.J. Epema	Technische Universiteit Delft, The Netherlands
Dr. L.O. Alima	LOA Dependable IT Systems S.A, Belgium
Prof.dr. C.M. Jonker	Technische Universiteit Delft, The Netherlands, reservelid

Tariq Abdullah

Mechanisms for Self-organizing Ad Hoc Grids
Delft: TU Delft, Faculty of Elektrotechniek, Wiskunde en Informatica - III
PhD Thesis, Technische Universiteit Delft.
Met samenvatting in het Nederlands.

ISBN: 978-90-72298-09-6
Subject headings: Ad hoc grids, self-organization, resource scavenging, social networks, infrastructure level trade-offs.

Printed in The Netherlands

This dissertation is dedicated to my late grandfather
Malik Fateh Khan
who, despite being illetrate and from a small remote village, dreamed the highest level of education for his kids.

Mechanisms for Self-organizing Ad Hoc Grids

Tariq Abdullah

Abstract

In this thesis, we present self-organizing mechanisms for resource management in ad hoc grids. By doing so, we enable the construction and deployment of ad hoc grids that can self-adapt to provide better resource allocation/utilization under dynamic conditions. The motivation for this study stems from the desire to harvest idle, unused computational resources in networked environments. Such networked environments could be from academic institutes, offices or the personal computers at our homes. The resources contributed in an ad hoc grid are volatile, unreliable and non-dedicated. Furthermore, the resource owners contribute these resources according to their own use/access policies. This phenomenon necessitates the study of the mechanisms that enable the ad hoc grid to modify its infrastructure in an autonomous way according to the varying resource availability/demand patterns. In the thesis, the infrastructure is defined as the mechanisms necessary to allocate the appropriate resources to the workers.

The infrastructure-level self-organizing mechanisms for resource management enable the ad hoc grid to adapt itself in order to provide the best resource allocation under varying circumstances. The infrastructure-level self-organization in an ad hoc grid emerges from self-organization at two levels; at the individual node level and the system level. An individual ad hoc grid node can be a consumer or a producer of resources. The node level self-organization enables an individual node to maximize its utility from the ad hoc grid in terms of task execution or resource consumption. The system level self-organization involves understanding under what conditions/circumstances a fully centralized or a fully decentralized infrastructure (or anything in between) is most appropriate.

We study and compare the mechanisms at two levels: at the resource allocation level and the overlay network level. Regarding the resource allocation, we propose a market-based Continuous Double Auction (CDA) approach and a modified Ant Colony Optimization (ACO) based approach. We also propose

an extension to the Pastry overlay network, which is required to (de)segment the ad hoc grid according to different resource availability and workloads.

In the market based CDA approach, the individual consumer/producer nodes use the price of a computing resource as an indicator of the resource availability/scarcity. The consumer/producer nodes self-organize themselves by increasing/decreasing their resource price according to their utility. The resource allocator (hereafter referred to as matchmaker) uses CDA as its matchmaking mechanism and applies segmentation and de-segmentation of the ad hoc grid for achieving system level self-organization. The segmentation and de-segmentation process results from the promotion and demotion of the matchmakers. The promotion and demotion is according to the overload and underload status of the matchmaker. An overloaded matchmaker shares its excess workload with a new matchmaker by promoting a normal node to a matchmaker. In this way, the ad hoc grid is segmented when new matchmakers are introduced, and the ad hoc grid segments are merged back when the matchmakers are demoted. We also studied the use of CDA for varying degree of the centralized control. One extension was the P2P case, where each node is its own matchmaker. We studied the impact of varying the degree of neighborhood.

Furthermore, a second infrastructure-level self-organizing mechanism is studied, namely the modified ACO approach. The modified ACO approach helped in achieving the system level self-organization in the presence of consumer/producer nodes of different resource types. The modified ACO approach also helped in dynamically forming the virtual resource segments of the ad hoc grid.

The main contribution of this work is that it presents mechanisms for self-organization in ad hoc grids. These mechanisms are implemented on a structured overlay network by using the algorithms proposed in this work. These mechanisms are studied in different network conditions. They helped in achieving a scalable, dynamic and a self-organizing ad hoc grid infrastructure. The study of these mechanisms identified the scenarios for determining the trade-offs for different ad hoc grid infrastructures

Table of Contents

List of Tables

List of Figures

List of Algorithms

List of Acronyms

AC Autonomic Computing

ACO Ant Colony Optimization

ACS Ant Colony System

AIS Artificial Immune System

ANN Artificial Neural Networks

BN Balanced Network

BOINC Berkeley Open Infrastructure for Network Computing

CAN Content Addressable Networks

CAS Complex Adaptive System

CDA Continuous Double Auction

DA Dutch Auction

DAS-3 Distributed ASCI Supercomputer 3

DKS Distributed K-ary System

DHT Distributed Hash Table

EA English Auction

EU DataGrid European Union DataGrid

FCFS First Come, First Served

FLOPS FLoating point Operations Per Second

FPA First Price Auction

FPGA Field-Programmable Gate Array

GRE Generic Route Encapsulation

GriPhyN Grid Physics Network

HTC High Throughput Computing

ICMP Internet Control Message Protocol

MAM Multiple Adaptive Matchmakers

MDTree Minimum-delay Dynamic Tree

P2P Peer-to-Peer

PI Principal Investigator

PL PlanetLab

PLC PlanetLab Consortium

PLE PlanetLab Europe

PPTP Point-to-Point Tunneling Protocol

PSP Proportional Share Protocol

RIN Resource Intensive Network

RSA key Rivest, Shamir and Adleman key

SSH Secure SHell

TCost Transaction Cost

TCP Transmission Control Protocol

TTL Time To Live

TIN Task Intensive Network

UDP User Datagram Protocol

VC Volunteer Computing

VM Virtual Machine

VMM Virtual Machine Monitor

ZRP Zone Routing Protocol

Chapter 1

Raising the Curtains-Introduction

THE term grid computing [65] is referred to as a distributed network infrastructure for promoting large-scale resource sharing. A computational grid is envisioned analogous to an electrical power grid. An electrical power grid interconnects different power plants and provides on-demand access through standardized, reliable and specialized interfaces to the users. In the same way, a computational grid is a closed network of computational resources (computational cycles, network bandwidth, database servers, specialized software and people), pooled together by several institutions, and are provided to the grid users through standardized, reliable and specialized interfaces on-demand. The computational grid has pre-defined access/use policies and provides non-trivial Quality of Service (QoS) assurance. Even though a computational grid provides the required QoS parameters, yet it forms a closed network due to its pre-defined access and use policies. For example, the EU DataGrid [82] and the GriPhyN [2] are only accessible to the employees of the organizations that build the grid infrastructure.

There is another class of computational applications that require transient, short lived and one-time collaboration among participants in most of the cases. One example of such computational applications is the scenario where the first group provides data analysis software; the second group contributes computational services; the third group pools the data visualization software, and the forth group provides the data storage repository for storing the experimental data. In this example, each participant wants to participate with his/her own use and access policies for a limited amount of time. The participating groups may find it infeasible or might be unable to build a for-

mal computational grid infrastructure for such an application. Furthermore, there are administrative overheads from such collaborations. The infeasibility and administrative overheads may make it impracticable to undergo a formal-computational grid establishment for one time collaboration, in most of the cases. In other words, the existing use and access models of the computational grid do not support the "ad hoc" collaboration among the participants.

Ad hoc grid[1] is an extension of the conventional grid and has geographically distributed, heterogeneous and ephemeral nodes, refer Section 2.2 for details of ad hoc grid. The development of an ad-hoc grid entails new challenges in comparison to a conventional grid. Resource management, security of an ad hoc grid from internal/external attacks, trust among the participants, QoS, etc are some of the challenges that need to be solved. The participating nodes in an ad hoc grid may have different ownership and varying use-policies. Resource management for nodes with heterogeneity, intermittent participation and varying use/access policies becomes a challenging undertaking. Different underlying network infrastructure and the variable participation of the nodes further increase the administrative complexity of the ad hoc grid, and hence resource management in an ad hoc grid becomes even more complex. Therefore, it is required to develop such a resource management system that will enable the ad hoc grid to self-organize and manage itself under varying workloads. Addressing these challenges is at the heart of this thesis.

A resource discovery mechanism defines how a node finds appropriate resources to perform its tasks and how it requests for additional tasks when its task queue is getting empty. The participating nodes in an ad hoc grid are geographically distributed, mostly heterogeneous, ephemeral and have their own use and access policies. In the literature, there exist different resource discovery approaches for the ad hoc grids varying from *fully centralized* [3, 27, 28, 46, 71, 103, 108] to *fully decentralized* ones [4, 24, 29, 48, 69, 74, 85, 93, 96, 98, 120, 123]. The fully centralized resource discovery mechanisms might be efficient for small-scale systems and may take less time in finding a required resource. However, these mechanisms are not scalable, and the centralized resource broker becomes a performance bottleneck. In contrast, the fully decentralized (peer-to-peer) resource discovery mechanisms do not have a single point of failure and are scalable. However, the drawback is that the fully decentralized approaches are computationally expensive and may take more time to find a resource in comparison with the fully centralized approach. Fully decentralized approaches are computation-

[1] also called Desktop Grid Computing [46], Global Grid Computing [43], Public Resource Computing [27] or Volunteer Computing [124].

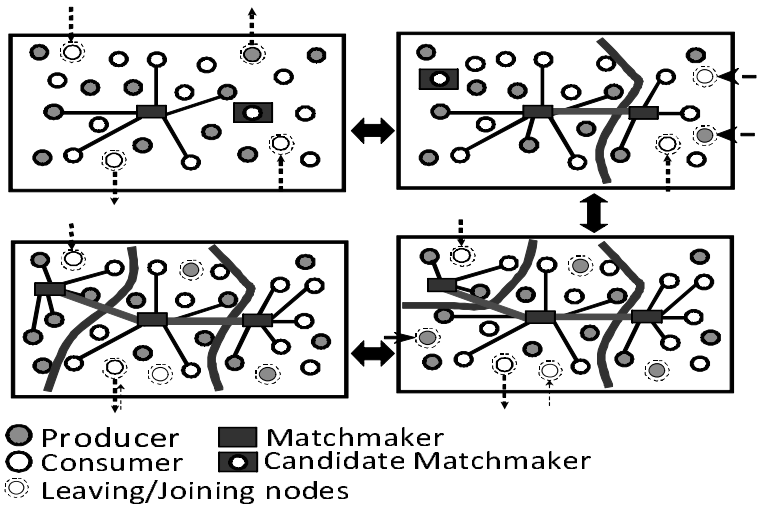


Figure 1.1: *System level self-organization and adaptation.*

ally expensive as each node is processing the message and more computation is done in comparison to a resource broker in a fully centralized approach. Furthermore, these approaches may take more time in finding a resource due to the increased message communication and message processing time at each node participating in the resource search process.

1.1 Research Context

Peer-to-Peer (P2P) and the fully centralized systems are often considered being mutually exclusive and residing on both ends of an infrastructural spectrum, however, 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. The question then boils down to how the system can determine what infrastructure, ranging from fully centralized to P2P, is most appropriate given its current status which is defined in terms of the available and the requested resources? The challenge is to find a way to generate system wide information on the basis of the individual states of the participating nodes. The context of the research aims to understand and develop appropriate mechanisms for self-organization and self management in an ad hoc grid at the infrastructure-level.

A generic representation of the system level self-organization and adap-

tation, proposed and discussed in this dissertation, is depicted in Figure 1.1. Consider an ad hoc grid, when there is a relatively low workload, a single centralized matchmaker suffices. However, in case the workload increases, the load on the matchmaker will increase too and in order to ensure a timely and appropriate match, the matchmaker may decide to look for support by promoting other nodes to become matchmaker(s). This could be done on the basis of, for instance, the price the matchmaker can charge for finding a match. In case of an overload of such requests, the price for finding a match will increase. The matchmaker may be constructed in such a way that once this price goes above a certain level, another matchmaker is initiated. This process may be repeated any number of times. In this way, the system organizes itself starting from a simple, centralized state and evolving successively into increasingly distributed ones. One extreme situation, the P2P case, would be where every single node in the ad hoc grid is now a matchmaker. Whenever, the price goes again below a certain level, the reverse could happen and the matchmaker nodes are demoted to become the ordinary nodes again in the system.

As explained above, a matchmaker is overloaded when it is unable to maintain its matchmaking capacity. The overloaded matchmaker promotes a normal node as a matchmaker. The newly promoted matchmaker starts performing matchmaking for some of the nodes that were under the responsibility of the overloaded matchmaker. The workload of the primary (overloaded) matchmaker is reduced. Thus, the overloaded matchmaker is able to maintain its matchmaking capacity and in this way the ad-hoc grid is segmented into two or more segments where each segment has its own matchmaker.

In order to understand the system level self-organization in an ad hoc grid, we looked into different self-organization approaches from the literature (Figure 1.2), which are detailed in Section 2.3. On the basis of the surveyed literature, we studied the system level self-organization in an ad hoc grid from micro-economic, social networking and the nature inspired Ant Colony Optimization (ACO) based approaches.

In this dissertation, we focus on the infrastructure-level self-organization in an ad hoc grid and aim to identify different scenarios where different ad hoc grid infrastructures can be opted. The main objectives of the dissertation are:

1. Propose scalable, system-level self-organization mechanisms for an ad hoc grid to introduce infrastructural-level self-organization, while maintaining the dynamism and the scalability of the ad hoc grid.

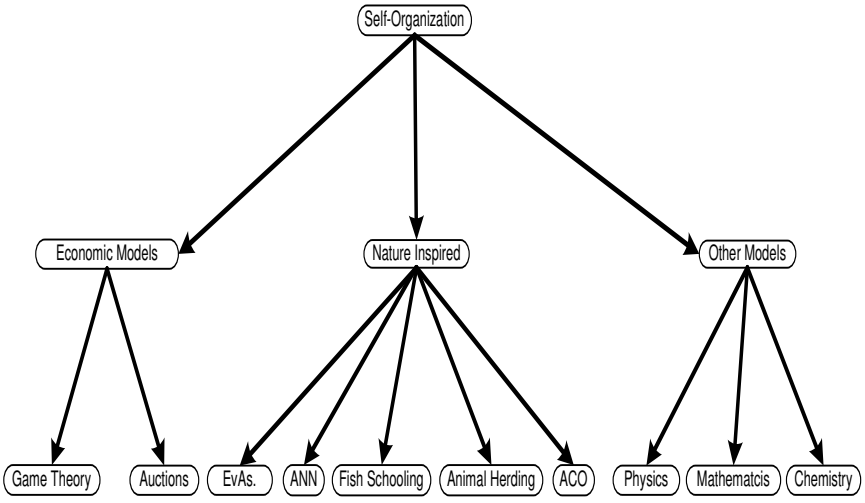


Figure 1.2: *A taxonomy of self-organizing approaches.*

2. Study the proposed self-organization from different self-organizing mechanisms such as micro economics, social networking and the nature inspired ACO mechanism.

3. To identify, based on the above mentioned studies, the trade-off options for a scalable self-organizing infrastructure for an ad hoc grid, while considering different level of centralization ranging from a fully centralized to a fully decentralized.

The dissertation is organized into three parts according to the above defined objectives. The first part comprises of Chapters 2, 3, 4 and 5 and describes the background knowledge, experimental platform and explains the proposed micro-economic based self-organizing mechanism. The second part of the dissertation comprises of Chapter 6 and explains the proposed mechanism from social networking domain. Whereas, the third part of the dissertation comprises of chapters 7 and 8. This part studies and explains the proposed nature inspired ACO based self-organizing approach, identifies the trade-offs and concludes the dissertation. A graphical representation of this organization is depicted in Figure 1.3. A summary of these chapters is provided in Section 1.4.

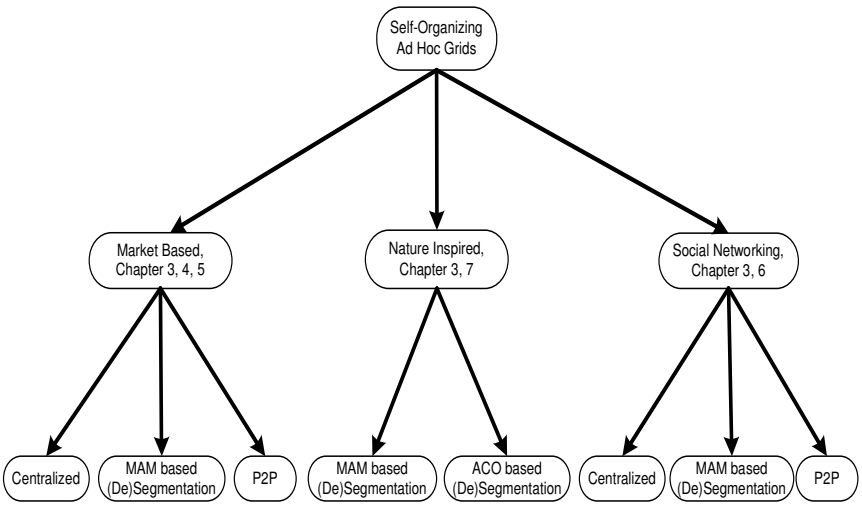


Figure 1.3: *Thesis organization.*

1.2 Research Challenges

A scalable, self-organizing mechanism for the ad hoc grid should address the following research challenges:

1.2.1 Scalability

In an ad-hoc grid, resource management mechanisms should be scalable and capable of dealing with the unpredictable availability of the resources. Fully centralized or fully decentralized resource management mechanisms are two ends of an architectural spectrum and each have their own efficiencies and deficiencies. Centralized methods with a central server have good throughput for certain population size, guarantee that a task finds the required resource (if it exists) and are able to find the best system-wide match. At the same time, they have low scalability because of the bottlenecks associated with the central server and are not robust as well. The whole system may fail when the central server leaves the network or goes down due to any reason. On the other hand, a fully decentralized and/or an unstructured mechanism is robust but lacks manageability and may cause a communication bottleneck. Moreover, it does not guarantee finding a match even if it exists in the network. Economic-based mechanisms provide scalable systems. These mechanisms limit the complexity of the resource allocation in large scale distributed sys-

tems by partitioning the complex global problem of resource allocation into a set of simple, independent sub-problems. This is provided by distributing decision-making across all the users and the resource owners.

1.2.2 Adaptation

In an ad-hoc grid, the participating nodes need to adapt themselves according to the varying resource availability of the non-dedicated resources. In such networks, a monitoring system or a centralized server to collect all the network information is not feasible. Therefore, embedding self adaptation mechanisms inside an ad-hoc grid is necessary. A way of providing adaptation is to learn the network condition through interaction with the environment. Learning from interaction is the founding idea underlying nearly all the theories of learning and the intelligence [130]. In a learning mechanism, agents in a network learn the condition using the reward they get from the environment and take a proper action to adapt to the new condition. The microeconomic mechanisms can be used to implement self-adaptation. In these mechanisms, individuals are able to make decisions based on their own preferences and according to the condition of the market. Price is the main concept in these mechanisms through which the decisions are made. A pricing mechanism that can learn from the changes in its environment is required to perform adaptation.

1.2.3 Emergent Behavior

Nodes are self-interested in an ad hoc grid. These self-interested nodes attempt to increase their profit without considering the global profit of the network. In such environments, it seems rational that each node can decide whether it needs additional resources or on the contrary wants to offer them, the main challenge is to make sure that they find the required resources or a user for their resources. One way to provide such a facility is to use the economic-based mechanisms. In the economic-based mechanisms, decision on resource allocation or task assignment can be resolved by providing the consumers/producers with a monetary system. A monetary system models the consumers/producers as buyers and sellers of the resources. Most economic models introduce money and pricing as the technique for coordinating the selfish behavior of the participating nodes. Each consumer is endowed with money that it uses to purchase its required resources, whereas, each producer owns a set of resources and charges consumers for the use of its resources.

1.3 Research Questions

The following open questions can be posed with respect to the self-organizing mechanisms for distributed systems in general and for the ad hoc grids in particular.

How do self interested participants react to the varying condition of an ad hoc grid? How do these reactions can affect the current state of an ad hoc grid?

How do self interested participants (consumer, producer and/or matchmaker) react to the varying load condition of an ad hoc grid? How does the emergent behavior affect (desired or undesired) the current state of the ad hoc grid? When and how the ad hoc grid can recover from these state changes? Chapters 4 and 5 address this research question.

What are different criteria for segmentation and de-segmentation of an ad hoc grid?

What can be the simple and an effective criterion for the matchmaker workload calculation? Chapters 4 and 7 address this research question.

Which system architecture suits best for self-organization in an ad hoc grid?

We are talking about the available choices for developing the architecture presented in this dissertation. Chapters 6 and 8 address this research question.

How does neighborhood of a node affect the resource discovery in an ad hoc grid?

What is the neighborhood of a node? How is it defined in an ad hoc grid? What will happen when neighborhood degree of a node changes on different points of the infrastructural spectrum in an ad hoc grid? Chapter 6 addresses this research question.

What would be the impact of adoption of a particular structure of the ad hoc grid from the infrastructural spectrum (ranging from fully centralized to fully decentralized extremes)?

How does the selection of a particular structure of the ad hoc grid affect the resource discovery in an ad hoc grid? What are different trade-offs for selecting a particular structure? Chapter 6 addresses this research question.

How can the self-organizing mechanisms found in the natural systems be applied for achieving self-organization in an ad hoc grid?

How do the self-organizing mechanisms found in naturally existing systems can be applied for resource discovery and self-organization in an ad hoc grid? How can these mechanisms be compared with the micro-economic based self-organizing mechanism? How can we achieve resource specialization in the distributed systems using these mechanisms? Chapter 7 addresses this research question.

1.4 Thesis Overview

A summary of the dissertation chapters is provided below:

Chapter 2 provides an overview of the background knowledge required to understand the rest of the dissertation, along with providing a summary of the literature related to the different resource discovery approaches in the ad hoc grids with a particular focus on self-organization. The resource discovery and self-organization mechanisms for the ad hoc grid on different points of the infrastructural spectrum; i.e. fully centralized, fully decentralized and any hybrid form; are described and analyzed. Advantages and disadvantages of the surveyed mechanisms, in comparison with the proposed mechanism, are also discussed. This chapter also provides an overview of the self-organization concept in different contexts and the mechanisms that are used to introduce self-organization into those contexts. It also provides an overview of the resource discovery and self-organization mechanisms based on ant colony optimization algorithm [54].

Chapter 3 explains the experimental platform and the experimental setup used in the experiments reported in this dissertation. This chapter starts by explaining the experimental platform, followed by the description of the experimental setup. It describes the reasons for selecting PlanetLab [1] over

other testbeds. This chapter also provides an overview of PlanetLab and an overview of the steps for executing an experiment on PlanetLab.

Chapter 4 proposes and explains the market-based self-organization mechanism for an ad hoc grid. An overview of the micro-economic based resource discovery approach used in the proposed model is provided. The mathematical formulas for the matchmaker workload calculation, hereafter referred to as Transaction Cost (TCost), are described. The experimental results for evaluation of the presented self-organization mechanism are presented. The focus of the presented results is on determining the upper and lower thresholds of a matchmaker's workload; which is used for promotion of a node to a matchmaker and demotion of a matchmaker back to a normal node.

Chapter 5 presents the algorithms that extend a structured overlay network, and are used for the self-organization mechanism presented in Chapter 4. This chapter starts with an overview of Pastry [123] structured overlay network before presenting the algorithms to extend Pastry structured overlay network. The algorithms for promoting a node as matchmaker, demoting a matchmaker to a node, node join/leave and the algorithm for discovering a responsible matchmaker by a node are presented and explained. Message complexity of the presented algorithms is also discussed. The experimental results showing that the proposed algorithms (joining ad hoc grid, finding a responsible matchmaker, promoting a matchmaker, demoting a matchmaker) work as expected are also presented and discussed.

Chapter 6 looks at the impact of adopting a particular infrastructure by exploring the following issues: First, it defines the degree of neighborhood of a node for resource discovery in a fully centralized, multiple adaptive matchmakers and a fully decentralized (P2P) environment for an ad hoc grid. Secondly, it analyzes the effect of varying the degree of neighborhood in a fully decentralized ad hoc grid. Thirdly, it compares the results of varying the degree of neighborhood in fully decentralized approach with fully centralized and multiple adaptive matchmakers approaches. Fourthly, message complexity analysis of the above mentioned resource discovery approaches is presented for understanding the communication cost. Finally, recommendations for the trade-offs in resource discovery on an infrastructural spectrum for an ad hoc grid are provided.

Chapter 7 presents the modified ACO algorithm for the micro-economic based resource discovery and self-organization in an ad hoc grid. Secondly, it presents the formulas for calculating the pheromone value. Thirdly, it presents a resource specialization mechanism based on the modi-

fied ACO algorithm. Finally, experimental results for resource discovery and resource specialization based on the mechanisms presented in this chapter are explained.

Chapters 8 summarize and conclude the dissertation. It presents a summary of the findings and the trade-offs for an ad hoc grid infrastructure in different network conditions. This chapter concludes the dissertation by summarizing our investigations, discussing the main contributions and proposing future research directions.

1.5 Thesis Contributions

This section provides an overview of the findings of this dissertation:

- We studied different schemes for infrastructure level self-organization in an ad hoc grid. These schemes are studied on an infrastructural spectrum that ranges from fully centralized to the fully decentralized extremes. As the ad hoc grids are characterized by intermittent, volatile participation of the nodes that show high fluctuations in resource availability/demands, therefore, a fixed infrastructure is not suitable for an ad hoc grid. Trade-offs for selecting a particular ad hoc grid infrastructure are discussed.

- The proposed infrastructure level self-organization mechanism is based on segmentation/de-segmentation of the ad hoc grid according to the workload of the resource allocator (matchmaker). This mechanism is studied in market-based and non market-based environments and is evaluated in different network conditions. We discovered that a market based mechanism takes care of individual participant's utility as well as the system level utility. Furthermore, a market-based mechanism is as efficient as a computationally less expensive, non market-based mechanism.

- We presented algorithms to extend a structured overlay network for the proposed segmentation/de-segmentation mechanism in an ad hoc grid. These algorithms include the algorithms for promoting a matchmaker, demoting a matchmaker, discovering a responsible matchmaker and node join/leave algorithm. The experimental results showed that these algorithms do not affect the working of the ad hoc grid and the ad hoc grid work as expected.

- We studied the impact of a node's social network on its ability for discovering the required resources by studying the effect of varying the degree of neighborhood of a node. The degree of neighborhood of a node is considered equivalent to the social contact(s) of a person in human society. We discovered that by increasing the degree of neighborhood of a node increases its ability to discover its required resources up to a certain degree. After that point, the increased degree of neighborhood does not help due to the increased communication cost with the social peers of a node.
- We also studied the infrastructure-level self-organization in an ad hoc grid by applying the nature inspired approaches. We applied a modified ACO algorithm in an ad hoc grid. This phenomenon was studied in a market based and a non-market based setting. We discovered that the Continuous Double Auction (CDA) does not help in discovering more resources than the FCFS setting. However, a CDA based modified ACO approach achieves node level and system level self-organization in an ad hoc grid.

1.6 How the Dissertation is Written?

This dissertation is based on the research articles that are the outcome of the research work carried out over the last couple of years, while working with Computer Engineering Laboratory, TUDelft. The order of the chapters in this dissertation does not coincide with the chronological order of the research articles.

Chapter 2

Background Knowledge and Related Research

A distributed system is a system in which components located at networked computers communicate and coordinate their actions only by message passing [50]. Distributed systems are typically divided into client-server model, peer-to-peer (P2P) model or any variation of these two models. A grid can be viewed as a distributed system that coordinates distributed resources using standard, open, general-purpose protocols and interfaces to deliver non-trivial qualities of service [65]. It is usually referred to as a high performing computing and data handling infrastructure and incorporates geographically and organizationally dispersed and heterogeneous resources. These resources may include computing systems, storage systems, instruments, real-time data sources, human collaborators and/or communication systems. The grid generally has a centralized architecture, fixed goals, pre-defined use policies, regulated control for membership and access privileges and availability of a stable, well-defined collaboration among the collaborating communities. P2P and grid computing both focus on the resource sharing within the distributed systems communities. However, different base assumptions lead to the different requirements and technical directions for these communities [64]. Grid computing focuses on infrastructure. It has smaller and better-connected groups of users with more diverse resources to share, whereas, P2P focuses on massive scalability, global fault-tolerance and not on the infrastructure. Despite these differences, a convergence of P2P and grid computing has been foreseen in literature [64, 134]. A grid and a P2P system can be compared in terms of their

Attribute	Grid	P2P
Ownership	Organizations	Individuals
Access policies	Pre-defined	No restriction
Use policies	Pre-defined	No restriction
Scale & Activity	Project dependent	Large
Applications	Dependent on interest & scale of the project	Vertically integrated solutions for specialized resource sharing
Resources	Powerful, diverse & well connected	Intermittent participation, loosely connected & highly variable
Content type	Well-defined contents	Any H/W & S/W resources

Table 2.1: *Comparing grid and P2P systems.*

ownership, access/use policies, scale, activity, supported applications, content type and the available resources. Table 2.1 gives a comparison of the grid and P2P.

2.1 Grid Categorization

A typical grid environment consists of grid users, a meta-scheduler and grid nodes as shown in Figure 2.1. The grid node runs the grid middleware client. A grid user submits its job to the meta-scheduler; the meta-scheduler schedules the job on the required number of grid nodes; the results are returned to the grid user after completing the job. Grid computing is used to solve the applications that cannot be completed on a single machine or will take too long to complete on a single machine. These applications may include, but are not limited to, video rendering, cryptography, parameter sweep, digital sky surveys and weather prediction etc. European Union DataGrid [82], National Fusion Grid [5], Grid Physics Network (GriPhyN) [2], AstroGrid [6], Particle Physics Data Grid (PPDG) [132] and Network of Earthquake Engineering Simulation (NEES) grid [7] are some example projects of grid computing. This section presents a survey of the grid according to its application types. The grid can be categorized into the following branches according to the application types:

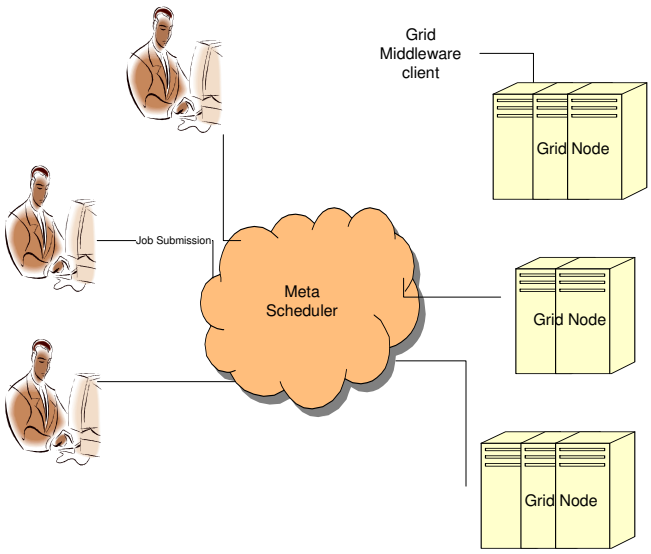


Figure 2.1: *A typical grid environment.*

Distributed Supercomputing

Applications that cannot be solved on a single machine and require lots of computing resources like computational power, memory and network bandwidth fall in this category. The required resources can vary from all supercomputers in a country to most of the desktop workstations in an organization and are dependent on the problem size. Examples of the distributed supercomputing include distributed interactive simulations [38], stellar dynamics and ab initio chemistry.

Data Intensive Computing

Vast amount of industrial or research data (several Peta bytes) is preprocessed and analyzed in data intensive computing. This data is stored in geographically distributed databases, digital libraries or data repositories. Examples of the data intensive computing include SETI@Home [8, 28], Folding@Home [9], EU DataGrid project [82], GriPhyN Project [2] and DataTAG [10].

Collaborative Computing

Participants collaborate with each other for a set of services that are distributed in the grid. These services are offered by the participating members. Examples of the collaborative computing include data exploration, collaborative design and education.

On-Demand Computing

Remote resources are integrated with local computation. This is often for a bounded amount of time. Examples of on-demand computing include medical instrumentation, network-based solvers and cloud detection.

High Throughput Computing

High Throughput Computing (HTC) environments deliver large amounts of processing capacity over long periods of time. Loosely coupled or independent tasks can be scheduled in HTC environments. Parameter sweep applications, video rendering applications, simulation of a complex system, cryptographic problems or other similar applications can be executed in HTC environments.

Volunteer Computing

Volunteer computing [124] aims at aggregating the pervasive desktop resources from individuals or from within an organization or from multiple organizations. It has been observed that desktop workstations or PCs are as much as 95% unused and idle at night and 85% during the day [94,133]. Volunteer computing exploits the availability of these idle and unused computing resources. Examples of the volunteer computing based projects/middlewares include BOINC [27], Entropia [46], SZTAKI [97], PC Grid [11, 66], SETI@Home [8,28] and Folding@Home [9]. As of November 1, 2010, BOINC has 309,704 volunteers with 515,159 computers and 3,738.14 TeraFLOPS are contributed in 24 hours on the average [12].

2.2 Ad Hoc Grid

In comparison to the applications discussed in Section 2.1, there are other classes of applications which are resource intensive but, at the same time, very

difficult to execute on the present day grid infrastructures, if not impossible at all. One such class of applications is the situation where the participants are offering different resources to collaborate on a common objective; a team of scientists may provide data analysis software, another team of scientists may pool visualization service and the third group may provide data storage repository for input of the analysis software. In this example, every participant wants to participate with his/her own usage policies and access rights to its resources for a certain limited amount of time, normally till the participant has some utility interest in the participation. Administrative overheads erupting from this type of experiments make it impractical for such transient communities to undergo a formal grid establishment process, probably one time collaboration in most of the cases.

Another example can be a grid market where grid resources are treated like a commodity. Individuals or organizations participate in this type of grid market for trading their resources with potential buyers/consumers. In this scenario, the participants want to optimize their respective objective function (utility). Every participant takes part in the grid market with its own objective function, pricing rules and usage policy. This type of grid market cannot be implemented and monitored by a single controlling authority. Moreover, the grid market has a metamorphic structure and it self-organizes itself over a period of time. The conventional grid infrastructures fail to support the grid market as they rely on some pre-defined network and structure-dependent services.

These classes of the present-day applications necessitate a new type of grid that does not require centralized control authority, no pre-defined underlying network infrastructure, no pre-defined usage rules and access policies, no fixed resource discovery mechanism and no initial commitments for participation in the grid from the participants. This new type of grid is known as *ad hoc grid*. It is difficult to have one comprehensive definition of the ad hoc grid. One way to define an ad hoc grid is; *"An ad hoc grid is a distributed computing architecture offering structure-, technology-, and control-independent grid solutions that support sporadic and ad hoc use modalities [25]*."

Ad hoc grid is sometimes referred to as Desktop grid computing [46], Public resource computing [27], Global grid computing [43] or Volunteer computing [124]. Ad hoc grid has geographically distributed, heterogeneous, and ephemeral nodes. The participating nodes may have different ownership with varying use-policies. Resource management for the nodes with the above characteristics becomes a challenging undertaking. Different underlying network infrastructure and the variable participation of nodes further increase the ad-

ministrative complexity of the ad hoc grid. Hence, resource management in an ad hoc grid becomes even more complex. Therefore, it is essential to study the resource management and self-organizing mechanisms in an ad hoc grid. Before discussing the self-organizing resource management in an ad hoc grid, we first provide an overview of the self-organization concept in different contexts in the next section.

2.3 Self-organization

A self-organizing system is intrinsically a dynamic structure that emerges from the interactions of the system's components. Such a system is nonlinear and is called *dissipative* (Section 2.3.5). The complex system theory also focuses on such nonlinear systems. An overview of the complex systems will help in better understanding the notion of *self-organization*.

2.3.1 Complexity and Complex Systems

Complexity itself is a complex concept. There is no general definition of complexity, since the concept achieves different meanings in different contexts like sociology, computation, biology, etc. Complexity stems from its Latin root "*complexus*", which means "entwined" or "embraced". In order to have complexity, there must be two or more distinct components and these components should be joined in such a way that it is difficult to separate them. It also implies that even separating these components apart will not help in understanding these components, as separation will destroy the connection(s) between the components.

Furthermore, these components are mutually entangled; a change in one component will propagate to the other components through connections of the component. This chained change process may affect the component(s) which originally initiated the change. Interactions among the individual components result in a global behavior of the system. The global behavior is entirely different from the behavior of the individual components. Human brain, living cell, the Internet, human society, sand dune ripples, the weather, bird flocking, ants food foraging, a city, an economy etc are all typical examples of the complex systems. All above mentioned systems depict a global behavior that cannot be reduced to the behavior of the participating components that produced the global behavior. There is no global definition of complexity and is explained in different meanings in different contexts [59].

Computing systems that became part of our lives by the introduction of user-friendly PCs in 1980's and the web in 1990's, now seem buried under complexity and confusion. It is estimated that 75% of the budget was consumed on acquiring new hardware and 20% for fixing the existing computing systems just 15-20 years ago. While now, $70\% - 80\%$ is consumed in fixing and maintaining the existing computing systems and the rest is consumed for developing the new computing systems. It is estimated that 66% of the IT projects fail or are delayed because of their complexity. It is very difficult now, if not impossible at all, to control or predict the interaction and behavior of the present day computing systems due to their dependence on many hardware/software modules, data sources, input and output peripherals, network devices and underlying network infrastructures. These systems are so complex that the human mind is unable to learn and remember all the procedures needed to use these systems. The number of possible interactions among different components of a complex system(s) increases exponentially with the addition of new components. According to Moor's Law, hardware components and their capacity increase exponentially, therefore, the complexity of these systems also increase exponentially. This complexity hinders the speed of further progress and results in systems that have security holes, corrupted data and other catastrophic side effects on the day to day human routine. In general, the present day computing systems can be characterized by indeterminacy, non-linearity and chaos.

The complex systems are expected to be adaptive and robust. These can be adaptive by compensating any perceived deviation from the desired course by using the positive/negative feedback loops. The feedback control loop can be highly effective if the reaction comes before the impact of action becomes large. However, a feedback control loop demands availability of broad repertoire of reactions. The complex systems can anticipate the appropriate reaction in response to a deviation; the complex systems learn anticipation from their experiences. The next section discusses adaptiveness and robustness of the self-organizing systems in detail.

2.3.2 Meaning of Self-organization

Our environment/universe is abundant with examples of natural organization found in the naturally existing Complex Adaptive Systems (CASs). The CASs are dynamic, decentralized networks and consist of many participating agents. Examples of CASs include bacteria colonies [118], sand dune ripples, mud ripples, a city, human brain, flock of birds, fish colonies and ant colonies [54, 55].

Property	Description
Absence of global control	There is no global control. Each sub-component is autonomous in its decisions.
Emerging structure	System behavior emerges from the local interactions of the sub-components.
Highly scalable	System performs as requested, regardless of the number of sub-components and performance degrades gracefully in the event of a failure.
Feedback loops	Feedback loops exist and can be positive or negative. Positive feedback amplifies & negative feedback suppresses the current operation.

Table 2.2: *Characteristics of self-organizing systems.*

These self-organized communities, waves, bodies and patterns are developed and later destroyed by some seemingly invisible hand. Different fields have explained the notion of self-organization from different contexts. These fields include cybernetics [139], thermodynamics, mathematics, autonomic computing [86], biology, computer science and other fields [81]. A comparison of different self-organizing mechanisms from nature and information technology is given in [99].

Although there is no unanimous definition of self-organization in the literature, however, the most commonly used definition of self-organization is: *"Self-organization is a process in which a pattern at the global level of a system emerges solely from numerous interactions among the lower-level components of the system. Moreover, the rules specifying interactions among the system's components are executed using only local information, without reference to the global pattern [42]."* Absence of global control, emerging structure, high scalability and feedback loops characterizes the self-organizing systems. These characteristics are summarized in Table 2.2 and explained below:

- **Absence of Global Control–** There is no global control in self-organizing systems. Each sub-component of a self-organizing system is autonomous and takes its decisions according to its state at any given instance of time. Hence, a self-organizing system can be viewed as a set of autonomous systems/components working collectively for a global objective.
- **Dynamic & Adaptive–** Self-organizing systems are dynamic, flexible

and adaptive. The components are changing their state relative to each other, and the system evolves over a period of time. The system depicts dynamism by continuously adapting to the changing environmental conditions. When the environmental changes are rapid and modifications in the system are out of the tolerance range of the system, temporary fluctuations/instability might be seen in the system. The self-organizing system attempts to achieve the best system state according to the present environmental conditions.

- **Emergent Behavior from Local Activity–** The complex system level behavior emerges from the local interactions among the participating sub-components. The local interactions usually follow some simple rules. The emergent global behavior is entirely different from the local interactions of the sub-components. Furthermore, the global behavior cannot be traced back to the local interactions of the sub-components and vice verse.

- **Robustness & Fault Tolerance–** Self-organizing systems are intrinsically robust and can withstand errors and perturbations. These systems have the ability to get back to their previous working state by repairing/correcting the damages or by appropriately responding to the perturbations. Self-organizing systems are fault tolerant as the work of a failed sub-component can be taken over by another component.

- **Non-Linearity and Feedback Loops–** There are feedback control loops in the self-organizing systems. The feedback control loops can be positive or negative. A positive feedback control loop amplifies the current operation in a self-organizing system, and a negative feedback control loop suppresses the current operation. The feedback control loop generated from one sub-component will propagate to all the connected sub-components due to the interconnection of the sub-components in a self-organizing system. This propagation may trigger a chained change process, consequently, affecting the components which initiated this feedback control loop initially.

- **Scalability–** Self-organizing systems are highly scalable due to decentralized control and continue working as requested with the addition of more and more sub-components. No performance degradation is observed/expected with the addition of new sub-components. Increased scalability results in decreased determinism. Scalability and determinism are inversely proportional for a self-organized system as depicted in

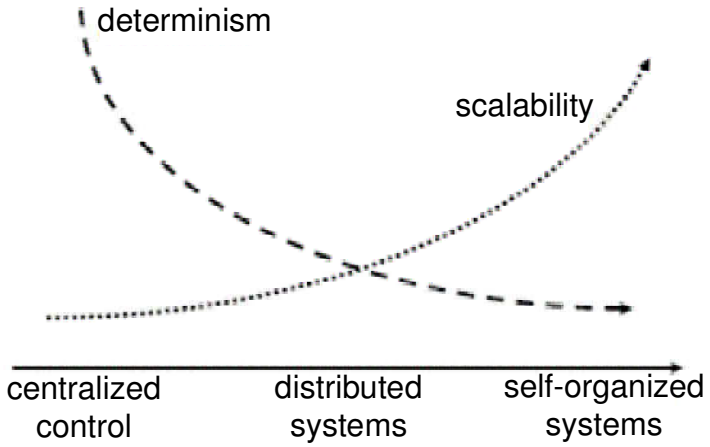


Figure 2.2: *Scalability vs determinism in a self-organized system [56].*

Figure 2.2. Systems with centralized control are not scalable and their global behavior is very deterministic. On the contrary, distributed and self-organizing systems are highly scalable and their global behavior is highly indeterministic.

In the remainder of this section, we discuss the self-organization phenomenon in different contexts such as autonomic computing, information systems, thermodynamics, mathematics, and natural systems.

2.3.3 Autonomic Computing and Self-organization

Autonomic Computing (AC) attempts to anticipate IT system requirements and resolve problems with the help of its self-managing autonomic capabilities in the same way as the human nervous system does for the human body, consequently, the IT system professionals can focus on the tasks of higher business value [86]. The self-managing autonomic capabilities of an AC system are limited to the extent delegated to it by its developers, whereas, the decisions made by the human autonomic nervous system are involuntary.

An autonomic computing system uses a control loop(s) to collect feedback information from the environment. An autonomic system takes an appropriate action according to the received feedback control loop information. The control loops become the attributes of a self-managing AC system. Self-organization is generally considered being a special property of AC systems

	Self Configuring	Self Healing	Self Optimizing	Self Protecting
Storage Tank		Y	Y	
Oceano			Y	Y
AutoAdmin	Y		Y	
SMART DB2	Y		Y	
OceanStore	Y	Y	Y	Y

Table 2.3: *Autonomic computing systems overview.*

[104]. Self-organizing systems are self-managing systems with an organization. The attributes of an autonomic computing system are explained below:

- **Self Configuring:** Enable an AC system to adapt to the dynamically changing environment by deploying new components or by removing the existing components of an AC system. AC systems with a focus on self-configuration include AutoAdmin, SMART DB2.
- **Self Healing:** Enable an AC system to detect system errors/malfunctioning and initiate a corrective action(s). An AC system becomes more resilient due to self-healing capability, as chances of failure of day-to-day operations of the system reduce. Storage Tank and OceanStore are example systems from literature with a focus on the self-healing capability.
- **Self Optimizing:** Enable an AC system to monitor and tune resources automatically. The AC system can divert its under-utilized resources to the low-priority tasks. This capability introduces high service standards to both *system end users* and *business customers*. Some example projects focusing on self-optimization include Storage Tank, SMART DB2, AutoAdmin, Oceano, GLAMOR and HAC.
- **Self Protecting:** Enable an AC system to detect hostile behavior– unauthorized access/use, virus infection and denial-of-service attacks– and take corrective measures to make the system less vulnerable. OceanStore is one example AC system that supports self-protecting capability. Furthermore, the work presented in discusses different methods to enable P2P systems self-protecting.

Different approaches that are used in developing these systems come from agent-based computing [88], control theory [80] and biologically inspired ap-

proaches such as neural networks and genetic algorithms. Different projects supporting more than one self-* capabilities of the AC systems are summarized in Table 2.3.

2.3.4 Information Systems and Self-organization

A computing system is usually developed for a well-defined purpose. In order to keep with the growth and demands of an organization, old components are removed or replaced with the new components. Consequently, the computing system expands organically and becomes complicated. In such a situation, there is no choice but to gradually switch from centralized to the decentralized control. As discussed previously in this section that decentralized systems are complex and indeterministic, hence the move from centralized to the decentralized control sows the seeds of complexity.

This creates the perfect conditions for a complex system behavior, in which many sub-components make autonomous/selfish decisions on the basis of the locally available information. It is important to learn to take advantage of the emergent properties resulting from the interactions among the system components and not try to counter these properties. Such a system can only be driven to a desirable state via self-organization. This process requires engineering the reasoning and decision-making engine running on the sub-components so as to promote the emergence of the desired collective behavior. Consequently, this implies adapting the predictive techniques of natural complexity science to meet the needs of the present day complex computing systems.

Fully decentralized information systems are also excellent candidates for self-organization. The examples of such systems include P2P systems for content distribution, knowledge management and scientific data and resource sharing applications. In these systems, individual participants take local decisions and perform local operations based on the localized information.

2.3.5 Thermodynamics and Self-organization

Self-organization is often characterized by the notion of increased/decreased order without an external agent. The negative of entropy represents the notion of order in thermodynamic systems. According to the second law of thermodynamics, the entropy of an isolated system only decreases and does not increase. Thus, the system evolves to the state of maximum entropy or thermodynamic equilibrium. Since, self-organizing systems require a constant feedback (in-

put of matter/energy) from their environment; therefore, these cannot be isolated. Self-organizing systems get rid of the internally generated entropy with the help of feedback. This phenomenon makes these systems as dissipative structures and maintains them far from the thermodynamic equilibrium [67]. Furthermore, description of a self-organizing system as a measure of entropy is dependent on the level of observer. A system can be self-organizing from a higher abstraction level and could be self-disorganizing at a lower detailed level.

2.3.6 Mathematical Systems and Self-organization

The fields of cellular automata and random graphs exhibit features of self-organization. Random graphs are used for self-organization in sensor networks and other complex systems [37]. A random process generates a random graph. Theory of random graphs lies at the intersection between the graph theory and the probability theory.

A cellular automaton is a discrete model consisting of an infinite, regular grid of cells. Each grid cell can be in one of a finite number of states, such as *on* and *off*. The set of cells at a distance of 2 or less from a given cell define the neighborhood for a cell. The neighborhood usually consists of the cell itself. An initial state (at time $t = 0$) is selected by assigning a state for each cell. A new generation (at time $t = 1$) is created according to a mathematical function, representing a general rule to be applied on all the cells in the grid. This rule determines the new state of each cell in terms of the current state of the cell and states of the cells in its neighborhood. An example rule might be that the cell is "off" in the next generation if exactly one of the cells in the neighborhood is "on" in current generation; otherwise, the cell is "on" in the next generation. A well known example of a cellular automaton is the *game of life [70].* It is also applied in self-learning systems and cryptography [140].

2.3.7 Self-organization in Natural Systems

There exist countless systems in nature that fit within the characteristics of self-organizing systems discussed in Section 2.3.2. Some examples of the naturally existing self-organizing systems include human brain, coordination of human movement, sand dune ripples, morphogenesis, human immune system, ants foraging for food and grouping behavior found in flocks of birds and fish schools (Figure 2.3), etc. Bio-inspired self-organizing techniques include arti-

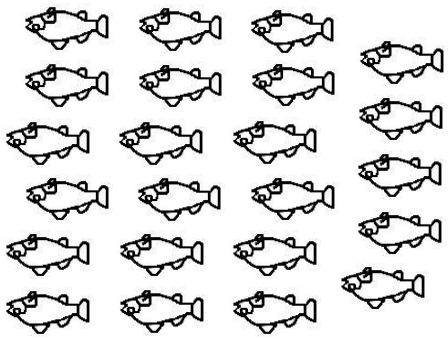


Figure 2.3: *Fish moving in a school to avoid predators.*

ficial intelligence, cellular automata, artificial neural networks, swarm intelligence [36], artificial immune system and evolutionary algorithms.

It has been proved experimentally that the pheromone trail following behavior of ants in an ant colony leads to the emergence of the shortest path from the nest to the food source [34, 52]. A colony of Argentine ants called *Iridomyrmex humilis* is studied for their foraging behavior under controlled conditions by using a double bridge [52]. Each bridge is of the same length and separates the nest of the colony of ants from the food source into two different paths. Each ant leaves a pheromone trail on the upper/lower branch of the bridge followed. It was observed that, after the initial fluctuations, one branch became more preferred than the other as each passing ant modified the following ant's probability of choosing that branch. Thus, the ants tend to converge on one of the two branches of the double bridge or, in general, on one of the many available paths from the food source to the nest over a period of time, as depicted in Figure 2.4.

2.4 Self-organizing Resource Management in Ad Hoc Grid

After explaining the notion of self-organization from different contexts in general and specifically in computing systems (Section 2.3), we now provide an overview of different approaches that are used for resource discovery and self-organization in the ad hoc grid. These approaches vary from fully centralized to fully decentralized ones. These approaches can be divided into following categories:

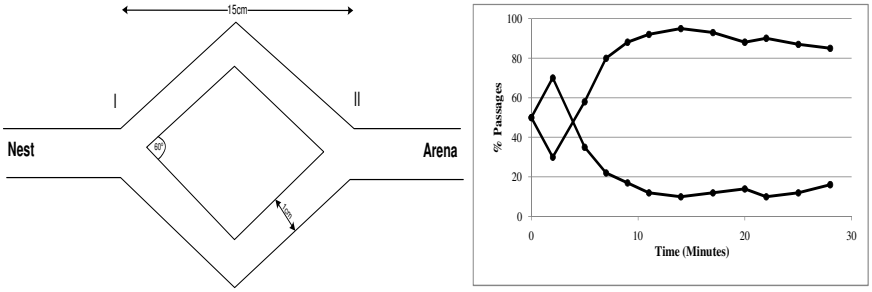


Figure 2.4: *Foraging behavior of ants in an ant colony.* ***(a)*** *Experimental setup with double bridge for ant foraging behavior analysis.* ***(b)*** *Percentage of ants per 3 minutes period passing on the two branches of the bridge in an ant colony of 1000 ants [52].*

- Centralized Approach
- Peer-to-Peer Approach
- Hybrid Approach

2.4.1 Centralized Approach

Traditional or fully centralized approaches maintain a centralized or a set of hierarchically organized resource allocators to index resource information in an ad hoc grid. The fully centralized approaches [3, 27, 28, 46, 71, 103, 108] for the ad hoc grids employ a client-server architecture. A typical client-server model works as follows: clients request jobs; a trusted server distributes jobs to the clients; the clients run the jobs; and the server collects the results. The centralized approaches have high throughput and guarantee finding a resource, if it exists in the ad hoc grid. The server is responsible for providing and maintaining robustness and reliability.

However, the following four reasons limit the efficiency of the traditional resource management approaches. First, these approaches may result in system bottlenecks in a highly dynamic environment where many resources join, leave and can change characteristics at any time. Second, systems with the centralized approach can suffer from a high computational overhead and may result in overall system performance degradation. Third, the centralized server or the root node of the hierarchical organization has the inherent drawback of being a single point of failure. Fourth, these approaches cannot scale

well to a large-scale and geographically distributed system across the Internet.

2.4.2 Peer-to-Peer Approach

In P2P approach, each node or a group of nodes negotiates for its required resources with other nodes. These approaches may apply a centralized indexing server [4, 29], query flooding [32, 49, 85, 96], decentralized indexing server [13, 74, 93] or Distributed Hash Table (DHT) based routing [24, 48, 69, 98, 120, 123] for resource discovery.

A centralized indexing server maintains information about all the resources contributed by participating peers in a centralized indexing based resource discovery approach for P2P systems. The centralized indexing server performs the resource discovery process. Matched peers communicate directly with each other for resource exchange. The centralized indexing approach is simple to implement and easy to administer. However, centralized indexing server can become a bottleneck and a single point of failure, thus resulting in system's failure. Therefore, centralized indexing approaches do not scale well and are only suitable for small grids.

In query flooding approaches, a resource request/query is broadcasted to all available peers in the P2P network by applying different query broadcast approaches [130, 135, 142, 144]. The matching peer responds to the requesting peer. These approaches are useful for small network environments with unpredictable infrastructures and in absence of the dedicated resources. However, flooding/multitasking based approaches suffer from huge bandwidth usage and do not scale well.

DHTs are used to map contents to network addresses (peers) in the structured peer-to-peer overlay networks. Contents are located by applying the hash functions on a resource request [131]. DHTs are very efficient in locating contents by one name or attribute and do not support multi-attribute range queries. MAAN [41] used customized hashing function for supporting the multi-attribute range queries in DHT based P2P networks. The problem with customized hashing functions is that they require prior knowledge of the attributes value distribution.

A locality aware, decentralized, flexible and co-allocation supporting approach for P2P supercomputing is proposed in [57]. This approach is not suitable for high throughput computing applications. A reactive, pull-based protocol for aggregate node selection in unstructured overlay networks is presented in [119]. Iamnitchi et al. [84] proposed a resource discovery approach

for completely decentralized grid environments and evaluated different request forwarding algorithms. Their approach employs time to live (TTL) for resource discovery. In their approach, TTL represents the maximum hop count for forwarding a request to the neighboring nodes. This approach is simple but may fail to find a resource, even that resource may exist somewhere in the grid.

2.4.3 Hybrid Approach

Hybrid approaches are a mix of fully centralized and fully decentralized (P2P) resource discovery approaches. These approaches use some trade-off and offer the best feature set of the centralized and decentralized approaches. Hybrid approaches attempt to achieve a balance between the efficiency of centralized approaches and the scalability, load balancing, fault-tolerance and node autonomy of the P2P approaches. Choi et al. [47] proposed a group-based scheduling mechanism in a peer-to-peer grid computing environment and called those groups as *volunteer groups*. The individual volunteer's properties define the volunteer groups. 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 clients, job allocation to different volunteer groups and collection of the results; therefore, it may become a performance bottleneck and can be a single point of failure. Padmanabhan et al. [107] 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. Main drawback of their work is that there is no load balancing mechanism among the groups formed. Zhou et al. [95, 148, 149] 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 a communication overhead.

Attribute encoding of static or dynamic computational resource information [44, 75] is another technique for resource discovery in a structured overlay network. The available resources are mapped to the nodes of a P2P structured overlay network in the attribute encoding approach. As 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.

Butt et al. [39] implemented a P2P based Condor flocking [60] to share resources in different Condor pools [133]. 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 [78] distributed the workload of one matchmaker among multiple matchmakers. It used one matchmaker for each resource type being traded in the ad hoc grid. A new matchmaker is introduced 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 [125] for grid resource discovery mainly focus on a best matchmaker selection from a pool of fixed number of matchmaker. In the super-peer model [14, 15, 68, 98, 117, 121, 141] some nodes maintain resource index and are called *super-peer nodes*. Super-peers form an overlay network, where each super-peer is responsible for a part of the overlay network. Mastroianni et al. [98] used unstructured P2P networks to construct a super-peer model for resource discovery in grids and applied experience based query forwarding.

A zone based hybrid resource/service discovery approach using Zone Routing Protocol (ZRP) [76] is presented in [102]. In ZRP, the network is divided into routing zones according to the hop distances between the nodes, and neither according to the network condition nor the workload conditions of the nodes. Thus, a ZRP based resource discovery mechanism is not suitable for infrastructure level self-organization in an ad hoc grid. Kim et al. [90] proposed parsimonious resource usage and job migration to lesser overloaded nodes, in order to balance the overall workload in a decentralized ad hoc grid. Mercury [35] used node leave/rejoin for load balancing. The overloaded node leaves and then joins the ring as neighbor of a lightly loaded node. Node leave/rejoin introduces message overhead.

Minimum-delay Dynamic Tree (MDTree) [143] organized the nodes in form of a tree, based on link delay of each of the joining node. MDTree splits a grouped set of nodes into subgroups when number of nodes exceeds a certain number of nodes in a group of MDTree. Fixed sized groups and the use of link delay as group formation parameter are the drawbacks of this approach.

Vadhiyar et al. [136] considered system load and job characteristics, before making a decision for job migration in order to achieve self adaptivity in the grid. The focus of their work is on load balancing and not the infrastructure level self-organization in the grid. Erciyes et al. [61] propose a dynamic load balancing middleware protocol for the gird, in which each cluster coordinator first attempts to balance the load in its own cluster. When a cluster coordi-

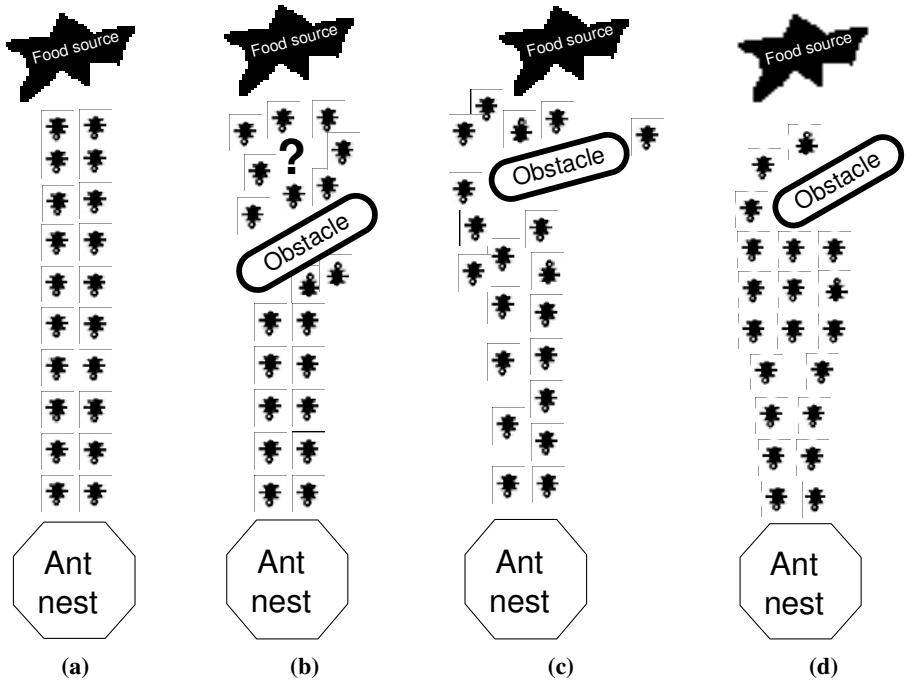


Figure 2.5: *Ant Colony System (ACS) illustrated.*

nator fails, it coordinates with the other cluster coordinators for transferring the workload. Padmanabhan et al. [107] proposed a self-organized grouping method that forms and maintains autonomous *resource groups*. These resource groups are formed according to some pre-specified resource characteristics, where each group contains a set of similar resources. The main drawback of their work is that there is no load balancing mechanism among the groups formed.

2.5 Ant Colony

List of naturally existing complex adaptive systems is very large. The CASs are dynamic, highly decentralized networks and consist of many participating agents. Human immune system, sand dune ripples and ant foraging are some examples of the natural CASs. Decentralized control, emergent behavior, robustness and self-organization characterize the CASs. The participating agents in these systems interact according to simple local rules which result in

self-organization and complex behavior at the system level.

Ant Colony System (ACS) [55], illustrated in Figure 2.5, is inspired by a colony of artificial ants co-operating in the foraging behavior. Ant colony optimization [54] is a heuristic algorithm that imitates the behavior of the real ant colonies in nature. In ACO algorithms, ants drop a chemical, called pheromone, on their way from nest to the food source and vice verse, while they search for a food source. The ants choose a path, from food source to their nest, with higher pheromone concentrations. The ant colony self-organizes by local interactions of the individual ants.

Ad hoc grids and similar computational distributed systems are inherently dynamic and complex systems. Resource availability fluctuates over time in ad hoc grids. These changes require adaptation of the system to a new system state by applying some self-organization mechanism. Current scientific problems, like protein folding, weather prediction, particle physics experiments are complex and require huge computing power and storage space. These scientific problems can be solved by using ad hoc grids. An overview of the related work from literature using the ACO algorithms for resource management in ad hoc grids is presented in the following paragraph.

Jaeger et al. [87] applied bloom filter for self-organizing broker topologies in publish/subscribe systems. The drawback of their work is that they only considered the similarity of notification messages in publish/subscribe system to reduce the total cost of forwarding and processing the notification messages. Ritchie et al. [122] proposed a hybrid ACO algorithm to select the appropriate scheduler in a heterogeneous computing environment. The proposed approach was only tested in solving a scheduling problem in a static environment for independent jobs. Deng et al. [53] proposed a resource discovery mechanism for P2P grids inspired by ACO algorithm. Fidanova et al. [63] attempted searching the best task scheduling for grid computing using an ACO based algorithm. Zeng et al. [146] proposed a dynamic load balancing mechanism based on the count of waiting jobs and the arrival rate of new jobs in distributed systems. Andrzejak et al. [30] compared different algorithms, including ACO algorithm, for self-organization and adaptive service placement in dynamic distributed environments. Messor [101] is implemented on top of the Anthill framework [106]. An ant can be in Search-Max or Search-Min states. The ant wanders randomly in the environment until it finds an overloaded node. The same ant, then, changes its state to Search-Min and wanders randomly again in the environment, while looking for an underloaded node. After these state changes, the ant balances the underloaded and the overloaded

node. However, considering the dynamism of the grid environments, this information may cause erroneous load balancing decision making.

2.6 Open Issues

The above discussed approaches are summarized as follows:

- Resource discovery approaches with centralized indexing [27,28,46,71, 108] employ a client-server architecture. These approaches have high throughput and guarantee finding a resource; however, they are not scalable, have a single point of failure, can suffer from a high computational overhead and may result in an overall system performance degradation.
- Different strategies/parameters are applied for forming multiple segments in an ad hoc grid. These include volunteer properties [47,107], attribute encoding [44,75], one matchmaker for each resource type [78], or finding the best matchmaker from a fixed pool of matchmakers [39,125]. These strategies do not consider the workload of resource allocators and may end up with an inappropriate infrastructure for the given state of ad hoc grid.
- Some approaches [61, 143] attempt to form the fixed-sized node groups or attempt to balance the workload of the fixed number of clusters; however, these approaches do not discuss the infrastructure level self-organization in an ad hoc grid.
- Node leave/join [35], customized hash functions [41] in structured P2P networks or super-peer model on top of unstructured networks [98]; all these approaches attempt to balance the workload of resource manager/matchmaker by sharing the workload; they may end up with an inappropriate infrastructure for the given state of ad hoc grid.
- Fully decentralized approaches apply indexing or query flooding for resource discovery. These approaches are useful for small networks; however, these approaches suffer from huge bandwidth usage and become less efficient with a large network.

In this dissertation, we aim to find trade-offs for infrastructural level self organization for a resource discovery mechanism in an ad hoc grid. Self organizing mechanisms for resource discovery will be presented and evaluated in the rest of this thesis.

2.7 Concluding Remarks

In this chapter, we presented the background knowledge and an overview of the related work for the work presented in this dissertation. This chapter aimed at understanding different approaches for self-organized resource management in ad hoc grids and the other distributed systems. Centralized indexing, client/server model, attribute encoding, a matchmaker for each resource type, using volunteer properties for forming groups, balancing workload among fixed size groups, decentralized indexing and query flooding are some of the approaches used for self-organized resource management in the distributed systems ranging from fully centralized to the fully decentralized extremes. Some open issues are identified after discussing the advantages and disadvantages of the studied approaches. In the subsequent chapters, we will propose our mechanisms for self-organized resource management.

Chapter 3

PlanetLab Based Experimental Platform

IN order to understand the proposed mechanisms for infrastructure-level self-organization in an ad hoc grid; we developed a PlanetLab based experimental platform. The platform helped in evaluating the effectiveness of the proposed mechanisms including micro-economic, social networking and nature inspired ACO approach. In this chapter, we provide an overview of the experimental platform, experimental testbed i.e. PlanetLab, and the experimental setup used in most of the experiments reported in this dissertation. The modifications, where applicable, to the experimental platform and/or the experimental setup will be mentioned in the respective chapter(s).

3.1 Experimental Platform

The experimental platform is developed in Java. It is implemented by extending a structured overlay network, Pastry [58]. A generalized architecture of the experimental platform is represented in Figure 3.1. It comprises 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*. The consumer agent has tasks and requires computational resources to execute its tasks, whereas, the producer agent offers its excess resources for executing tasks of the consumer agents. The matchmaker agent receives resource requests, resource offers and finds a

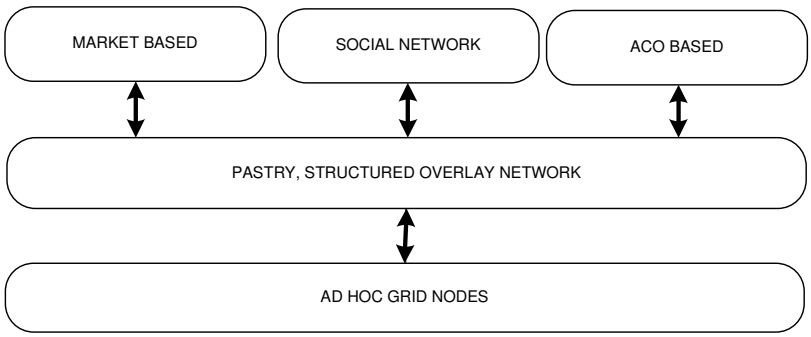


Figure 3.1: *Generalized system architecture.*

match between them. The matchmaker agent is also responsible for segmentation and de-segmentation of the ad hoc grid.

- Consumer Agent: The consumer agent estimates task execution time, required resource quantity, and the bid price in its *task manager* module. The consumer agent submits the resource request to the matchmaker or receives the matchmaking response through the *communication* module. Entire consumer-to-producer and consumer-to-matchmaker communication is done through the underlying structured overlay network. The consumer agent submits the matched job to the producer agent and receives the results in its *Job Manager* module.

- Producer Agent: The producer agent estimates its available resource quantity and calculates the ask price in its *resource manager* module. The producer agent submits the offer to the matchmaker or receives matchmaking response through the *communication* module. Entire producer-to-consumer and producer-to-matchmaker communication is done through the underlying structured overlay network. Receiving jobs from the consumer agents, execution of the consumer jobs and returning the results to the consumer agents are done in the *job Manager* module.

- Matchmaker Agent: The matchmaker agent receives the requests/offers from the consumer/producer agents. The *repository manager* module inserts the received requests/offers into the matchmaker's request/offer repositories. The *matchmake* module performs the matchmaking process. The matchmaker communicates with the consumer/producer agents through the *communication* module. The underlying structured

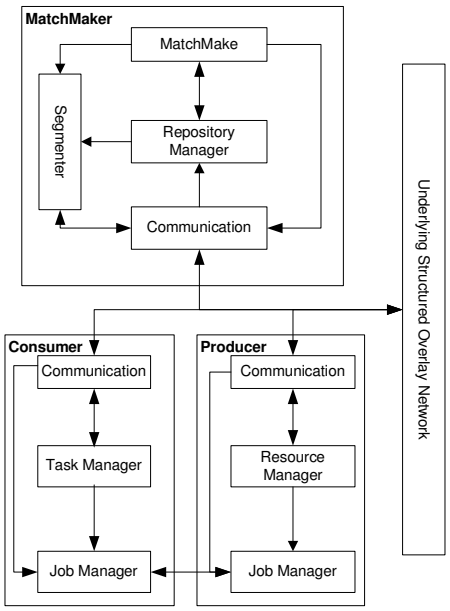


Figure 3.2: *Interaction among ad hoc grid node modules.*

overlay network does the entire matchmaker-to-consumer/producer and matchmaker-to-matchmaker communication.

- Segmenter: The *segmenter module* is part of the matchmaker module. This module contains the logic for calculating the workload of the matchmaker and for making the decision to promote a node as a matchmaker or demote a matchmaker as a normal node according to the workload of matchmaker. The workload of a matchmaker is referred to as Transaction Cost (TCost) and is explained in Section 4.3. The *segmenter* module is also responsible for sharing the excess workload, above the TCost *upper threshold* value, of a matchmaker with the other matchmaker(s). The segmenter module does so by forwarding the request/offer message to the other matchmaker(s). This module also manages the communication between different matchmakers through the underlying structured overlay network and uses the *communication module* of the matchmaker.

Interaction among different modules of the consumer, producer and matchmaker agents is graphically shown in Figure 3.2.

Name	Minimum Value	Maximum Value
Request Resource Quantity	500ms	4000ms
Offer Resource Quantity	500ms	6000ms
Resource Type	700MHz	4GHz
Consumer/producer ID	Unique IP Address	
Request/Offer TTL	10,000 milliseconds (ms)	
Request/ Offer ID	Unique Alphanumeric Value	
Transaction Price	Calculated by the matchmaker	
Transaction Cost (TCost)	Calculated by the matchmaker	
Ask Price	Calculated by producer during experiment	
Bid Price	Calculated by consumer during experiment	

Table 3.1: *Request/Offer message components specification.*

3.2 Experimental Setup

In this section, we describe the experimental setup used in our experiments. We first describe the experimental conditions, followed by the evaluation parameters and different network conditions.

In the reported experimental results, there are N nodes in each experiment. As each node can switch its role as a consumer/producer agent, according to its resource requirements/availability, so there can be $2 * N$ consumer/producer agents in an experiment. The value of N varies from $15 - 650$ PlanetLab nodes in our experiments. The number of matchmakers is varied from $1 - 5$ in these experiments. 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 back to the normal nodes.

The consumer/producer agent sends a request/offer message to the matchmaker and the matchmaker sends a reply message to the matching consumer/producer agents (refer to Section 4.1 for details). The components of a request/offer message and the consumer/producer reply message are as follows:

- **Request message:** ConsumerID, requestID, resource type, request resource quantity (aka task execution time), request TTL, bid price.
- **Offer message:** ProducerID, OfferID, resource type, offer resource quantity, offer TTL, ask price.
- **Consumer reply message:** ProducerID, requestID, offer TTL, Transac-

tion price, TCost.

- **Producer reply message:** ConsumerID, offerID, request TTL, Transaction price, TCost.

The request/offer message and the consumer/producer reply message components are explained in Table 3.2. TTL of a request/offer message is fixed to 10000 milliseconds and reflected the delays observed in PlanetLab. The task execution time and offered resource quantity are randomly generated according to the range specified in Table 3.1. The quantity of each request/offer resource quantity is varied for each request/offer message. The value ranges of other message components are specified in Table 3.1. Data represented in these results is extracted after $1/4^{th}$ of the experiment time has elapsed.

The proposed mechanisms are evaluated in terms of matchmaking efficiency, response time and TCost. Matchmaking efficiency is calculated in terms of the consumer/producer utilization. Consumer utilization and producer

Name	Description
Consumer / Producer ID	A unique identity for each node in the ad hoc grid. It is represented by a unique IP address assigned to each node.
Request/Offer ID	A unique number for identification of a request/offer.
Request/ Offer TTL	Represents the validity period of a request/offer message. A message is invalid after TTL expiry & is discarded by the matchmaker.
Request Resource Quantity	The amount of requested resource in terms of the task execution time. It is represented in milliseconds (ms).
Offer Resource Quantity	The amount of offered resource in terms of the available execution time. It is also represented in milliseconds (ms).
Ask Price	Minimum acceptable price of a producer for an offered unit of its available resources. Ask price formula is detailed in Section 4.1.3
Bid Price	Maximum price that a consumer is willing to pay for one unit of requested resource. Bid price formula is detailed in Section 4.1.3
Resource Type	Type of requested/offered resource like computational power (CPU), storage, network etc. CPU speed in represented in MHZ.
Transaction Price	Transaction price price is paid by consumer to the producer for using its resources & is calculated by using the formula in Section 4.1.3.
Transaction Cost (TCost)	TCost represents the workload of a matchmaker and is determined by the matchmaker using the formula given in Section 4.3

Table 3.2: *Request/Offer message components description.*

utilization (referred to as *cUtil* and *pUtil* respectively hereafter) is calculated according to Equations 3.1 & 3.2 respectively:

$$cUtil = \sum matchedRequest \ / \ \sum request * 100 \tag{3.1}$$

$$pUtil = \sum matchedOffer \ / \ \sum offer * 100 \tag{3.2}$$

where $\sum matchedRequest$ represents the total number of matched requests and $\sum matchedOffer$ represents the total number of matched offers in a unit time period. Similarly, $\sum request$ and $\sum offer$ represent the total number of requests and offers submitted by the consumer and producer nodes in a unit timer period respectively.

The 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 response time is calculated as:

$$RT = T_{match} - T_{receive} \tag{3.3}$$

where RT represents the response time. The time when the matchmaker agent found a matching offer/request for the received request/offer message is represented by T_{match}. Whereas, $T_{receive}$ is the receiving time of the received request/offer message. The formulas for calculating TCost are detailed in Section 4.3.

Experiments are executed in different network condition. These network conditions differ in the distribution of task-resource ratio for a network condition. These different network conditions are used for studying behavior of the participating agents (consumer, producer and matchmaker). Different network conditions used in the experiments are listed below:

- **Balanced Network (BN)** The task-resource ratio in a balanced network condition is $50\% - 50\%$. In this network condition, tasks and resources are generated with almost equal probability.
- **Resource Intensive Network (RIN):** The task-resource ratio in a resource intensive network condition is $20\% - 80\%$. In this network condition, resources are in abundance and tasks are scarce.
- **Task Intensive Network (TIN):** The task-resource ratio in a task intensive network condition is $80\% - 20\%$. In this network condition, tasks are in abundance and resources are scarce.

3.3 PlanetLab

There exist many testbeds for experimenting planetary scale distributed systems. We only discuss Distributed ASCI Supercomputer 3 (DAS-3) [16], OneLab [17] and PlanetLab [1].

DAS-3 represents the next generation grid infrastructure in The Netherlands. It is a five-cluster, wide-area distributed system for providing a common computational infrastructure to the researchers working, within The Netherlands, on various aspects of parallel, distributed, grid computing systems and on large-scale multimedia content analysis. As DAS-3 follows a clustering model instead of the volunteer computing, so it is not suitable for the resource requirement/availability model required for our experiments.

OneLab project represents PlanetLab Europe (PLE) [18] and other federated testbeds, which is the European arm of the global PlanetLab system [1]. PLE operates in cooperation with PlanetLab central, at Princeton University in the United States, and members of both PLE and PlanetLab central have full access to the combined global testbed. Migration of European sites from PlanetLab central to PLE is in process. The members of PlanetLab central also have access to PLE. Therefore, we selected PlanetLab central as our experimental testbed.

PlanetLab (PL) [1] was established as community testbed for planetary-scale networks services in March 2002. It is a global experimental test-bed that supports execution of the experiments at a larger scale than a LAN/WAN. PlanetLab is an open and geographically distributed computing environment/testbed. It is considered as a global research network that supports the development of the new network services. It has been used to develop new technologies for distributed storage, network mapping, peer-to-peer systems, distributed hash tables, and query processing. PlanetLab makes it possible to demonstrate the scalability and robustness of a research project with real network traffic, generated from real users while considering the inherent unpredictability of the Internet [109, 110]. As PlanetLab nodes are shared by multiple users, therefore, it is possible to get real workloads for a distributed systems research project. The member site in turn gets access to all the resources available on PlanetLab. Moreover, there is no centralized control over resources in PlanetLab.

Before going into the details of PlanetLab's principals (roles), architecture and how to work with PlanetLab, we would like to clarify some misconceptions about PlanetLab. It is not a distributed supercomputer or a simulation

	PlanetLab	**Grid**
Location Transparency	Small, long running services at specific locations.	Location independent large computations.
Application(s)	Low-level platform for testing distributed computing applications.	Standardized for large sclae computation of a particular application.
Motivation	Starts with a simple, well known interface and evolves for multiple, computing execution environments.	Starts from scratch and puts together an execution environment for a computation.

Table 3.3: *Comparing PlanetLab and the grid.*

platform or an Internet simulator or an arena for repeatable experiments or a grid. There are many differentiating points between the grid and PlanetLab. The grid aims at location transparency for large scale computations like protein-folding or a weather prediction job with a fixed deadline. Whereas, PlanetLab's purpose is to support small, long-running services in specific locations, for example, testing a new file sharing service for next 8 months on nodes located in x, y and z cities. Secondly, a grid starts from scratch and creates an execution environment, whereas, PlanetLab provides a simple, well-known interface for research community to execute multiple computing environments. Thirdly, a grid is constructed and standardized in one particular application domain for a large scale utility computing. On the other hand, PlanetLab provides a low-level platform where different distributed computing services can be tested. Table 3.3 summarizes the differences between the grid and PlanetLab.

PlanetLab Terminology

In order to understand a PlanetLab node architecture and the working of PlanetLab, it is essential to become familiar with the basic terminology of PlanetLab. This terminology includes site, slice, service, Principal Investigator (PI), PlanetLab user, PlanetLab Consortium (PLC) and node owner. These terms are explained as under:

A *site* represents a member institute/organization of PlanetLab and contributes a minimum of 2 computing nodes. For e.g., Delft University of Technology (TUDelft) is registered as a member institute on PlanetLab, so TU Delft is a *site* on PlanetLab. Each site contributes at least two nodes. These nodes

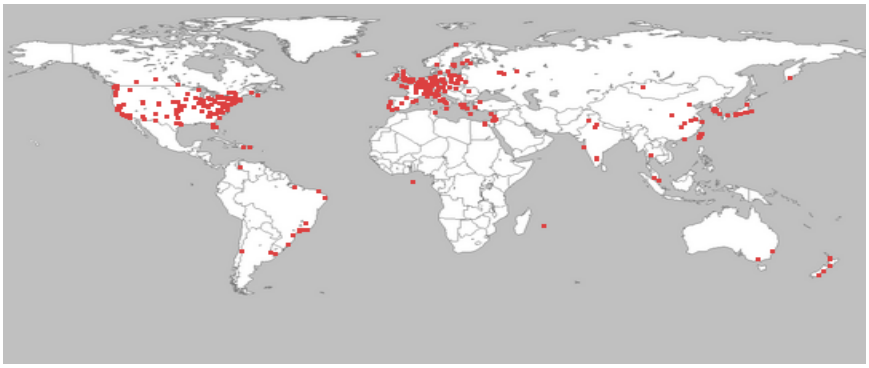


Figure 3.3: *PlanetLab nodes distribution [1].*

are x86-based, server class machines with minimum Fedora Core Linux installation. Table 3.4 summarizes the minimum hardware specifications as laid down by the PlanetLab consortium [19] for a PL node. It is evident from Table 3.4 that PlanetLab strives to give access to the latest available hardware to PlanetLab users. The contributing site is responsible for the security, working and upgradation of the contributed nodes so that 24/7 availability of the contributed resources is assured. As of November 1, 2010, Planetlab [1] has 1132 nodes, spanning 518 sites from 40 countries around the globe (Figure 3.3).

A *slice* runs as a Virtual Machine (VM) on a PlanetLab node. A slice isolates projects from each other and represents a set of allocated resources distributed across PlanetLab. A user adds nodes in a slice by itself. A slice has a finite lifetime and must be periodically renewed.

A research project in PlanetLab terminology is called a *service*. The global behavior of the nodes in a slice is defined and implemented by a service. Each service runs in its own network of virtual machines (slice). Over 600 services are currently running on PlanetLab.

	Until Dec'07	**Until Dec'08**	**Until Dec'09**	**Until Jun'11**
CPU	Intel Pentium 2.4 Ghz, AMD 2400+	Intel Pentium 3.2 Ghz, AMD 3200+	Intel Xeon 30X0 2.4 Ghz	Intel Quad core, 2.4 Ghz
RAM	1 GBytes	4 GBytes	4 GBytes	4 GBytes
Disk	180 GB	320 GB	500 GB	500 GB

Table 3.4: *Minimum hardware requirements for a PlanetLab node.*

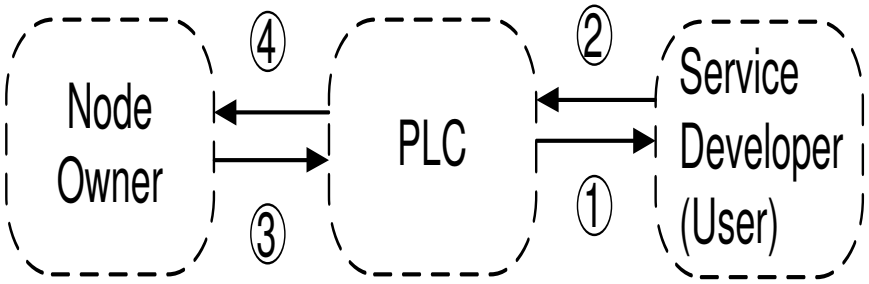


Figure 3.4: *Trust relationship among the PlanetLab node owners, PLC and the PlanetLab users [111].*

A *Principal Investigator (PI)* is responsible for account management, node management and overseeing all the slices that they create on behalf of the users at their site. A PI can create/delete slices, create/delete/enable/disable user accounts and assign users to the slices. A PI is also responsible for the physical maintenance of the nodes at its site. A PI is usually a faculty member from the participating member institute or a project manager from a commercial organization.

A *user* develops and deploys applications on PlanetLab. There can be more than one users in a slice and a user can be a member of multiple slices. Inspite of having access to multiple slices, a user cannot mix applications from different slices.

The *PlanetLab Consortium* is a trusted intermediary between the users and PlanetLab node owners. The PLC is responsible for managing PlanetLab nodes on behalf of the node owners, for creating slices and for a smooth and continuous working of PlanetLab.

A *node owner* is a site that owns its contributed PlanetLab nodes. A node owner site has ultimate control on its PlanetLab nodes; however, some of the control of the contributed PlanetLab nodes is delegated to the trusted PLC intermediary. There trust relationship [111] among node owners, users and the PLC intermediary is illustrated in Figure 3.4 and works as follows:

1. PLC trusts a user by accepting its credentials, creating a slice for the user and by granting access to the slice.

2. A user trusts PLC's role as its agent for checking its credentials and for creating a slice for the user.

3. A node owner trusts the PLC by delegating some of its responsibility of

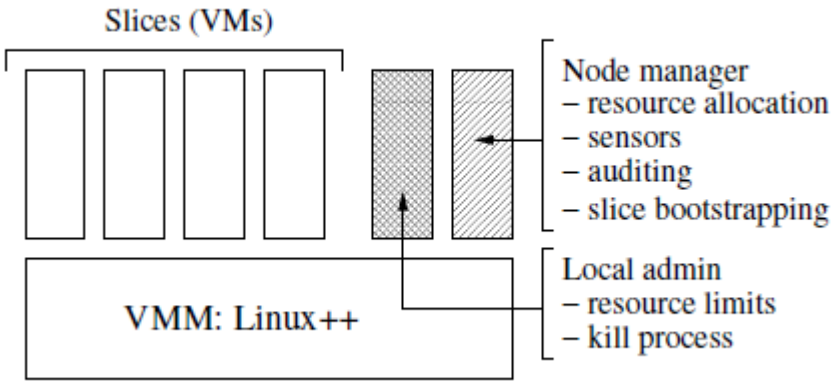


Figure 3.5: *A PlanetLab node architecture [33].*

node management to PLC.

4. PLC intermediary trusts the node owners for securing their contributed PlanetLab nodes.

3.3.1 PlanetLab Node Architecture

PlanetLab is designed around five organizing principles. These principles include distributed virtualization, unbundled management, chain of responsibility, decentralized control and efficient resource sharing. All these principles either represent a user requirement or a constraint imposed on PlanetLab from the environment in which PlanetLab operates. These principles are guidelines for a PlanetLab node architecture. Figure 3.5 shows the components and their interactions.

At the bottom level, each PlanetLab node runs a Virtual Machine Monitor (VMM). The VMM is combination of a Fedora Core Linux kernel and a set of kernel extensions including *vservers*, *schedulers* and *VNET*. A *vserver* is a patch to the Linux kernel 2.4 for providing multiple, independently managed virtual servers on a single node. A vserver is responsible for isolating multiple slices running on a PlanetLab node. A *scheduler* schedules CPU, memory, disk and link bandwidth fairly among all the slices executing on a PlanetLab node. A *VNET* enables a slice to send well-formed IP packets to non-blacklisted host and ensures that an IP packet is received by a slice destined for that IP packet.

VNET supports TCP, UDP, ICMP, GRE and PPTP network protocols.

A *node manager* is a privileged root virtual machine running on the VMM. The VMM is responsible for the management and monitoring of all the VMs running on that node. Resource allocation, auditing and slice bootstrapping are the main functions of a node manager. The *local admin* VM is for applying resource limits on slices and for killing a misbehaving slice/process. Apart from these two special-purpose VMs, all other VMs represent user slices [33].

3.3.2 How to Work with PlanetLab?

A PlanetLab node boots a small Linux OS from a CD or runs on a RAM disk. The node contacts a bootserver; the bootserver in return sends a signed startup script specifying a boot mode. The node can boot normally or can write new file system or can start sshd for remote PlanetLab admin login of the node. The boot up process can be remotely power-cycled. After this process, the node is ready for executing the user services (experiments) by dynamically instantiating slices on a PlanetLab node.

When a user logs in to PlanetLab, then the /bin/vsh (vserver) immediately switches to the account's associated vserver and chroot() command changes the root privileges and switches UID/GID to the account on vserver. This transition to vserver is transparent to the user and appears as if the user just logged into the PlanetLab node directly. There is a node-wide cap on outgoing network bandwidth to protect the world from PlanetLab services. Each node isolates multiple vservers. This isolation provides fairness by dividing the resources among N vservers as 1/N. Furthermore, the isolation also guarantees that each slice reserves a certain amount (e.g., 1Mbps bandwidth, 10Mcps CPU) of the resources on a PlanetLab node. The left-over resources are fairly distributed among the slices running on that node. Each of the N vservers gets 1/N of the resources during contention.

Connection to the PlanetLab nodes is only through *Secure Shell (ssh)* or any of its variants. PlanetLab uses public/private key combination for securing the communication between a user's machine and PlanetLab nodes. There are several console/GUI-based tools available to ease the experiment execution and management of the nodes in a slice. The console-based tools are simply Python or Perl scripts designed to run commands in parallel and include pShel, pSSH and AppManager. The GUI-based tools include PlMan, Plush and Neubla etc. Different tools available for executing and managing the ex-

Name	Institute	Format	Version	Description
Plush	University of California	GUI		Users describe experiments in XML using Nebula.
PlMan	University of Washington	GUI		Designed to simply the deployment, execution & experiment monitoring on PlanetLab.
Stork	University of Arizona	GUI		Software Installation utility akin to yum and apt.
pShell	McGill University	Shell	1.7	Provide a few basic commands to interact with a PlanetLab slice and with PlanetLab nodes
AppManager	Intel Research Berkeley	Shell	Client 9b, Server 13	Enables to centrally manage, install, upgrade, start, stop & monitor applications on a slice.
Enulab	University of Utah	GUI	4.188	Testbed for developing, debugging and evaluating systems
pssh	Intel Research Berkeley	Shell	1.4.3	Interface for few basic commands to interact with PL slices & nodes

Table 3.5: *PlanetLab tools for users.*

periments on PlanetLab are summarized in Table 3.5. These tools help users to execute commands in parallel on PlanetLab nodes in a slice. Following are the main steps for working with and executing an experiment on PlanetLab nodes:

Get a Slice

An authorized researcher of a PlanetLab member site submits a request for a slice and user account(s) to the respective PI by providing its credentials. The PI verifies the credentials and creates a slice and user account(s). As communication to PlanetLab nodes is only through SSH, therefore, a user needs to create a 1024-bit RSA public/private key for authentication and uploads the public key to PlanetLab using OpenSSH. In this way, the user can securely communicate with PlanetLab nodes in his/her slice.

Populate the Slice

Once a PlanetLab user has a user account and a slice, then the user identifies *good* PlanetLab nodes and populates its slice with those nodes. The good nodes can be identified by using the long running reporting services (like CoMon and CoDeeN) on PlanetLab.

Prepare the Nodes in a Slice

Considering the diversity of experiments running on PlanetLab nodes, a PlanetLab node does not have any user tools and is only equipped with minimum Fedora Core Linux installation. A user can install the required software on nodes in his/her slice according to the requirements of the experiments.

Transfer Experiment Executables

The user transfers project executables to the nodes in his/her slice after installing the required software. As the experimental platform (Section 5.1) for the experimental results presented in this dissertation is developed in JAVA, therefore, only JAR files are transferred to PlanetLab nodes.

Execute the Experiments

The experiment is executed from a Linux shell prompt. The status of an experiment is reported either success or failure on the console prompt.

Collect Experimental Data

The results are written in a text file on each PlanetLab node, which are later downloaded, stored in a database (MySQL database in our case) and analyzed.

The most attractive feature of PlanetLab is its *real world behavior*. Due to its geographical distribution, global scale, large number of nodes and dynamic nature of the Internet, it is very difficult if not impossible to execute experiments with the desired precision and accuracy. Furthermore, PlanetLab is not designed to be used for controlled experiments and is not suitable for reproducible results [112]. The nodes may fail to execute a command due to network problems, maintenance issue, or public/private keys mismatch. Results obtained from a few hours long tests only reflect the condition of the

network during that time period. Moreover, PlanetLab nodes are not backed up. Currently, we are able to identify around 650 stable nodes that we can connect to (most of the time) and run our experiments.

3.4 Concluding Remarks

In this chapter, we explained the experimental platform, described the experimental setup and provided an overview of steps needed for executing an experiment on PlanetLab. The experimental platform comprises of three main agents namely; Consumer, Producer and Matchmaker. The platform has enabled us in studying the proposed self-organization mechanisms. The evaluation criteria include consumer utilization, producer utilization and the response time. The experimental setup and the range of values of different evaluation parameters are discussed. In the subsequent chapters, the self-organizing mechanisms are presented and evaluated by using the experimental platform discussed in this chapter.

Chapter 4

Market-based Self-organizing Mechanism

THE mechanism to dynamically segment and de-segment the ad hoc grid is presented in this chapter[1]. We envision that the ad-hoc grid could have one or more matchmakers according to its size and consequently, the number of requests being sent by the individual nodes. For instance, as the grid grows, the number of nodes sending requests may become too big for one matchmaker to handle, thus requiring a second (or more) matchmaker to become active to assist the primary matchmaker in its task. As the grid shrinks, the system should be able to demote the matchmakers and return to a state with fewer matchmakers.

The question then boils down to defining these upper and lower bounds that will incite the system to, respectively, add or remove matchmakers to (from) the system. If the cost for making a transaction becomes excessively high, because of the incapability of the matchmaker to process all requests, then the grid itself becomes completely ineffective as it is no longer capable of using the available resources for the tasks to be executed by the grid. Where the use of markets and auctions is by no means original, its use in the context

[1] This chapter is based on the following research articles:

T. Abdullah, V. Sokolov, B Pourebrahimi, K.L.M. Bertels, "Self-Organizing Dynamic Ad Hoc Grids", In Proceedings of 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW 2008), Venice, October 2008.

T. Abdullah, L. Mhamdi, B Pourebrahimi, K.L.M. Bertels, "Resource Discovery with Dynamic Matchmakers in Ad Hoc Grid", The Fourth International Conference on Systems, March 2009

of such self-organization constitutes its main innovation. We will show that the simple basic mechanisms as described above, allow the system to scale its infrastructure automatically that better suited its current, and dynamically changing state.

To this purpose, we compute the cost of a single request made to the matchmaker, which is calculated for every request/offer message being submitted to the matchmaker [23]. The experiments described in this chapter define and calculate the upper and lower threshold of the matchmaker workload (TCost) in an ad hoc grid. The calculated upper and lower threshold values are used to increase or decrease the number of matchmakers dynamically. The results reported in this chapter are obtained from the experiments executed in PlanetLab. These results show that the mechanism enables the ad hoc grid to self-organize according to the workload conditions in an ad hoc grid.

4.1 Micro-economic based Resource Discovery Framework

An economy based resource management helps in building a large scale computational grid; distributes decision-making process; provides a fair basis for access to grid resources; enables both consumers and producers to maximize their utility; helps in regulating the demand and supply; and helps in developing user centric as well as system centric scheduling policies [40]. The use of market concepts for resource discovery is not new. Different research projects [26, 92, 105, 126, 138] used market concepts for resource management in computational clusters. Effectiveness of the market-based mechanism over the simple queuing algorithms like round-robin algorithm is proved in [73].

Macro- and micro-economic are the two branches of economics that study the behavior of economy at the aggregate level and of individual consumer/producer, respectively. Macro-economic focuses on the interaction among the individual components of a system, whereas, micro-economic focuses on the behavior of the individual consumer/producer by studying how much a buyer can pay for the needed quantity of a resource or how much a seller demands for the offered resources.

Supply, *demand*, *price* and *equilibrium* are some of important concepts in micro-economics. The *supply* represents the willingness of a producer to provide some quantity of a resource for a certain price. The *demand* represents the willingness of a consumer to acquire some quantity of a resource for a

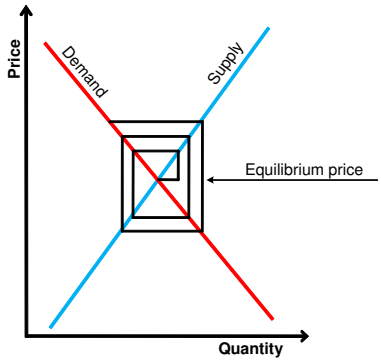


Figure 4.1: *Equilibrium point in micro-economics.*

certain price. The *price* indicates the value given to a resource by a user.

The price of a product increases with high demand and less supply, whereas, the price of a product decreases with less demand and high supply. The demand and supply curves gradually converge towards the point where the supplied resource quantity meets the demanded resource quantity. This point is known as the *equilibrium point* and the price at the equilibrium point is known as the *equilibrium price* (Figure 4.1).

As generally known from the literature, *law of supply* and *law of demand* describe the price variation with increase/decrease of supply and/or demand of a quantity respectively, as depicted in Figure 4.2a & 4.2b. According

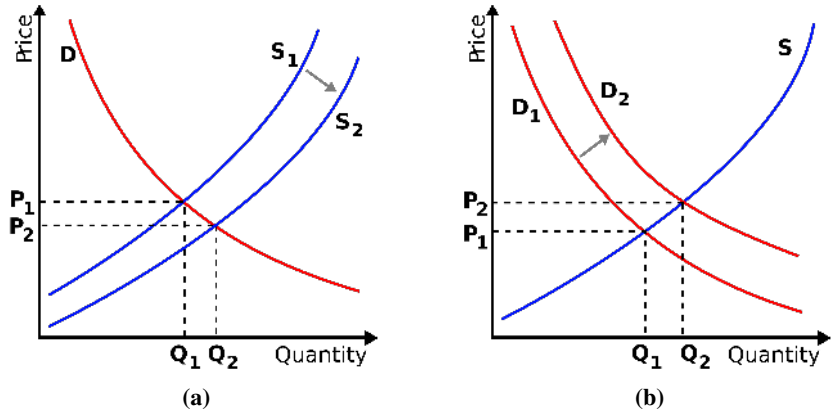


Figure 4.2: *Supply and demand laws from micro-economics theory.* ***(a)*** *Law of supply.* ***(b)*** *Law of demand.*

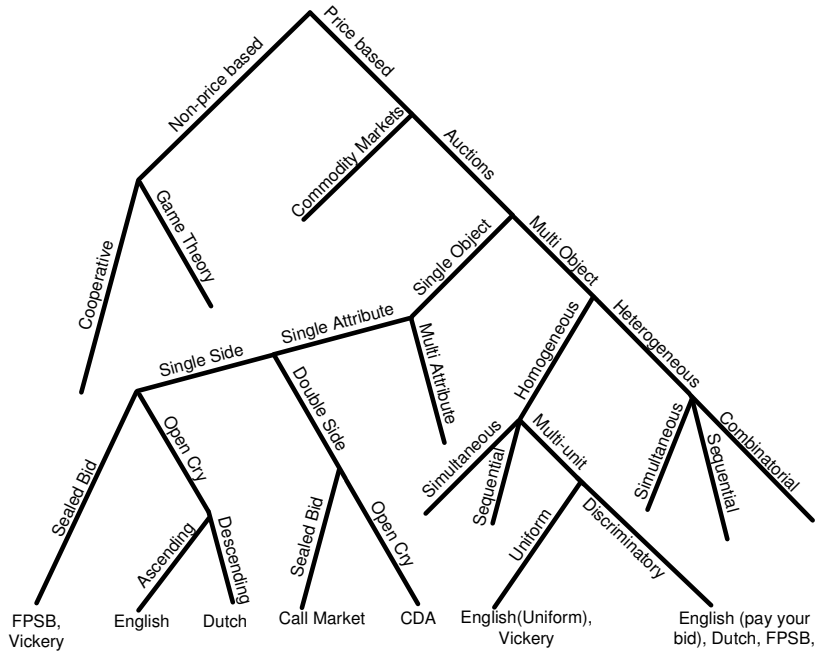


Figure 4.3: *Economic models topology.*

to law of supply, an increase in supply results a decrease in price, while a decrease in supply results an increase in price.

Micro-economic is classified into price-based and non-price based mechanisms [62,83]. A classification of economic mechanisms is summarized in Figure 4.3. In price-based mechanisms, the price represents supply/demand condition of resources in an economic market. Instead of price, there is a utility function in non-price based mechanisms. Non-price based mechanisms can be selfish or co-operative [79], depending on their focus for individual utility function of a participant [144,145] or global utility function of the system [91].

Price-based mechanisms are divided among auctions and commodity markets. Auctions support simultaneous participation of consumer/producer, observer resource deadlines, accommodate resource variations and avoid global information [113]. Different resources are treated as commodities in commodity market mechanisms.

Auctions do not require system-wide information for completing a transaction. Furthermore, auctions allow each consumer/producer participant to calculate its ask/bid price according to its internal evaluation of the re-

	Auction	**Commodity Market**
Price	Calculated by individual consumer/producer and is known as bid/ask price.	Calculated by market.
Matchmaking	Performed by a mediater called *auctioneer*.	Performed by a mediater called *market*.
Transaction price	Calculated by auctioneer.	Calculated by market.
Consumer/producer satisfaction	Attempt is made to satisfy multiple consumer/producer at a given price.	Usually one consumer/producer are satisfied at a given price.
Resource interchangeability	Interchangeable in many-to-many auctions. Not interchangeable in one-to-many auctions.	Resoruces are considered interchangeable.
Global information requirement	Not required. Price is calculated by the participants.	Required. Price is calculated by the market.
Implementation complexity	Low	High

Table 4.1: *Comparing auction and commodity market mechanisms.*

quested/offered resource quantity. The auctions are easy to implement and require fewer resources at execution over commodity markets. A comparison of auctions and the commodity market mechanisms is performed in Table 4.1.

In essence, avoiding requirement for global information, enabling consumer/producer to calculate their price for individual ask/bid, observing resource/offer deadlines, enabling decentralization, providing support for node as well as system level adaption to the changing environmental conditions and requiring less resources for execution makes auctions a suitable candidate for using them as a resource discovery mechanism in an ad hoc grid.

4.1.1 Why Continuous Double Auction?

Auctions are generally performed as *one-to-many* or *many-to-many* protocol. As the names imply, one consumer/producer agent initiates a one-to-many auction and many other consumer/producer agents can submit their bids/asks. *English Auctions* (EA), *Dutch Auction* (DA), *First-Price Auction* (FPA) and *Vickery Auction* (VA) are the examples of one-to-many auctions. On the other

hand, many consumer/producer agents can initiate a many-to-many auction and many other consumer/producer agents can submit their asks/bids. *Double Auction* (DA) is most widely used mechanism in many-to-many auctions. Different variants of double auction are *Continuous Double Auction* (CDA), *Periodic Double Auction* (PDA) and *Proportional Share Protocol* (PSP). Kant et al. [89] compared these variants from both system's perspective and user's perspective.

CDA is one of the many-to-many auctions. CDA supports simultaneous participation of the consumer/producer, observes resource/request deadlines and can accommodate the variations in resource availability. CDA adapts itself according to the variations in system states [45, 127]. CDA performs equally well in comparison to the non-market mechanisms like First Come, First Served (FCFS), in spite of the fact that there is no price constraint on the matchmaking process in FCFS as compared to CDA. Furthermore, CDA is compute intensive as compared to FCFS during matchmaking process. A comparative study of CDA with FCFS in terms of average task/resource utilization, resource/load balancing, average consumer/producer deadline satisfaction in balanced network, resource intensive network and task intensive network condition is done in [113], summarized in Table 4.2. CDA requires less communication bandwidth as compared to the other auction mechanisms like FPA, EA and DA [31]. A comparison of different auction protocols as a design choice for resource allocation in an ad hoc grid shows that CDA performs better or is interchangeable with Vickery auction and first price auction in terms of throughput, consumer/producer surplus, price volatility and consumer/producer deadline satisfaction in different network conditions [114].

		Task Util.	Res. Util.	RB^{+}	$LB^{\#}$	CDS^{*}	$PDS^{\wedge}$
BN	CDA	92%	92%	96%	98%	90%	81%
	FCFS	95%	95%	98%	98%	87%	65%
TIN	CDA	24%	95%	82%	94%	19%	94%
	FCFS	25%	100%	58%	100%	6%	99%
RIN	CDA	99%	26%	96%	81%	99%	18%
	FCFS	100%	26%	100%	62%	99%	20%

RB^{+} = Resource Balancing, $LB^{\#}$ = Load Balancing, CDS^{*} = Consumer Deadline Satisfaction, $PDS^{\wedge}$ = Producer Deadline Satisfaction

Table 4.2: *Comparing CDA and FCFS in different network conditions.*

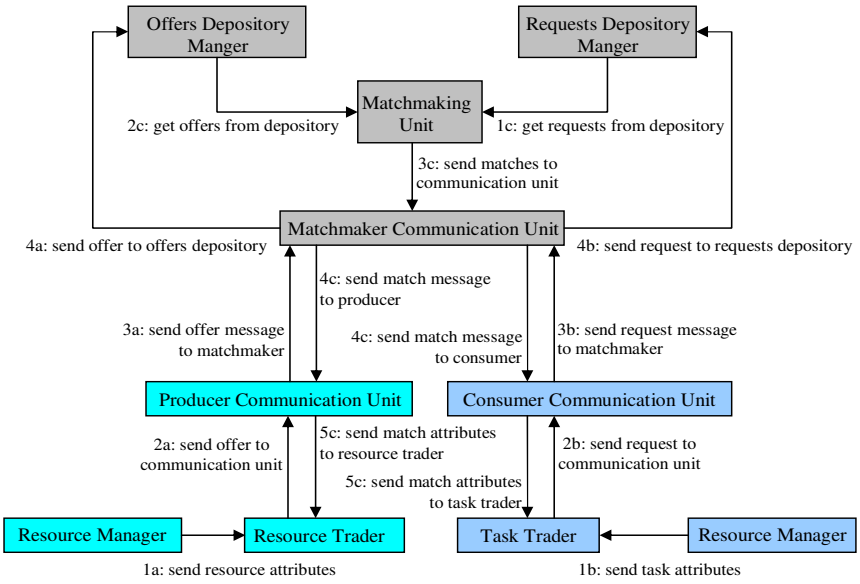


Figure 4.4: *Collaboration diagram of CDA based resource discovery mechanism [115].*

4.1.2 CDA based Resource Discovery Mechanism[2]

In this section, an overview of the economic framework is provided [113]. This framework serves as the starting point for the experimental results reported in this chapter. Several agents (*say consumers*) can initiate an auction and several other agents (*say producers*) can bid in the auction. The resource requests are called *bids* and the resource offers are called *asks*. The auctioneer (*matchmaker*) collects bids/asks and matches them immediately on detection of the compatible *bids*. A compatible bid is a pair of resource offer and resource request where task constraints (such as resource quantity, time deadline, ask/bid price) are satisfied. The auctioneer finds the matches between buyers and sellers by matching offers (starting with the lowest price and moving up) with requests (starting with the highest price and moving down). The matchmaker calculates the *transaction price* for the compatible bid. The transaction price is calculated as the average of ask price and bid price. It is the amount that a consumer pays to the producer for consuming the producer's resources. The matchmaker informs the matched consumer/producer along with the transaction price. The collaboration among different objects for the CDA based re-

[2]In [113], the author presented the economic framework.

source discovery mechanism is depicted in Figure 4.4.

Consumers/producers do not have global information about the supply and demand in the ad hoc grid and are not aware of the other's bids or asks. They submit asks/bids based on their local knowledge. The consumer/producer agents calculate their bid/ask price by applying a history based pricing strategy which is based on their previous experiences (Section 4.1.3). When a task query is received by the matchmaker, the matchmaker searches all the available resource offers and returns the best match. If no match is found, the bid/ask is stored in the matchmaker's request/offer repositories till the time-to-live expires or a match is found. The CDA based resource discovery mechanism used in the work presented in this dissertation is detailed in [115].

4.1.3 History-based Dynamic Pricing Strategy[3]

In an ad hoc grid, a manual price setting mechanism for each requested/offered resource quantity is not feasible. A manual price setting mechanism is also difficult to implement and is not expected to lead towards a stable and healthy economy. Furthermore, a dynamic pricing strategy in an ad hoc grid allows adapting to supply/demand changes in the ad hoc grid and helps consumer/producer agents to maximize their utility i.e., the producer can maximize its earnings and the consumer can acquire required resources with as little spending as possible.

Each consumer/producer agent uses a history based dynamic pricing strategy [116] to calculate its ask/bid price. This history-based dynamic pricing strategy is based on the interactions of an agent with its environments. A consumer/producer agent learns about its past resource/offer utilizations from its interactions with the environment and uses this information for its future pricing decisions. This process is graphically depicted in Figure 4.5 (adapted from [130]), where a consumer/producer agent receives feedback from its environment about its previous resource/offer utilization at time t as $r(t)$ and updates its price as $price(t+1)$ at time $t+1$ in form of its action/input to the environment.

In this approach, each consumer and producer agent joins the ad hoc grid with an initial pre-defined price and dynamically updates it by using the history-based dynamic pricing strategy. The calculated price is thus a reflection of the value of each resource unit which the consumer and producer agents are

[3] In [116], the author presented the history-based dynamic pricing strategy.

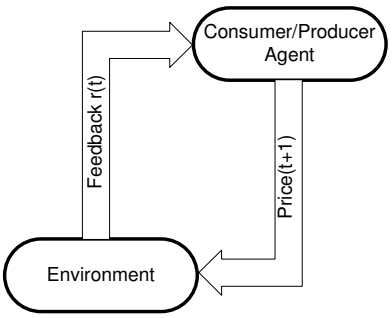


Figure 4.5: *Consumer/producer agent interaction with the environment.*

willing to buy or sell. An overview of the pricing function is given here.

The agents perceive the demand and supply of resources through their previous experiences and update their prices accordingly. Based on this strategy, ask/bid price at time interval t is defined as:

$$P(t) = P(t-1) + \triangle P \tag{4.1}$$

where $P(t)$ is the new price and $P(t-1)$ denotes the previous price. According to the previous history of resource/task utilization of the seller and buyer, $\triangle P$ for is defined. For the seller, $\triangle P$ is mathematically represented as:

$$\triangle P = \alpha(\mu(t) - \mu_{thR})p(t-1) \tag{4.2}$$

and for the buyer $\triangle P$ is defined as:

$$\triangle P = \beta(\mu_{thT} - \mu(t))p(t-1) \tag{4.3}$$

where α and β are the coefficients to control the price change rate. In this chapter, $\alpha = \beta = 0.8$ is used. The task and resource utilization thresholds are represented as μ_{thT} and μ_{thR}. The task/resource utilization of an individual node is represented by μ_t and defined as:

$$\mu_t = \sum_{i=t_0}^{t} x(i) / \sum_{i=t_0}^{t} N(i) \tag{4.4}$$

where $x(i)$ is a sold/purchased resource during time period $[t_o, t]$ and $\sum_{i=t_0}^{t} N(i)$ is the total number of offered/requested resources in time period $[t_o, t]$.

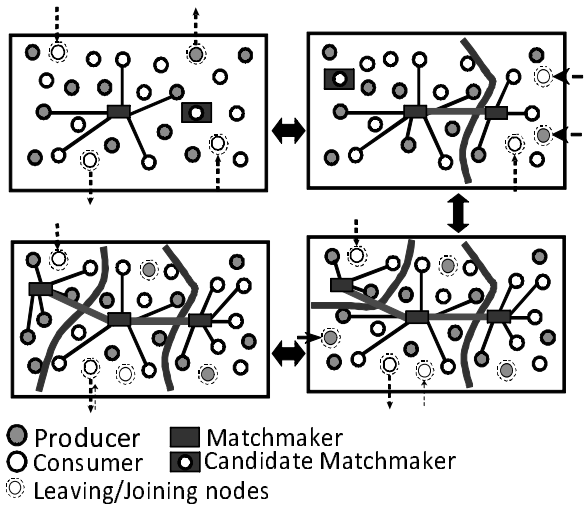


Figure 4.6: *System level adaptation.*

4.2 Ad Hoc Grid Segmentation & De-segmentation

We now present and explain the market-based self-organizing mechanism for segmentation and de-segmentation of the ad hoc grid. This proposed mechanism is an extension to the framework presented in [113].

The proposed mechanism introduces self-organization and scalability in an ad hoc grid, and is based on the workload of a matchmaker. The ad hoc grid divides itself into one or more segments when existing matchmaker is overloaded and is unable to maintain its matchmaking capacity. An ordinary node from the newly created segment is promoted as a matchmaker and the new matchmaker assists the existing matchmaker(s). On the other hand, when a matchmaker is underloaded and is not needed, then it is demoted back as a normal node and the ad hoc grid segments are merged back, as depicted in Figure 4.6.

We inspire ourselves on market-based, micro-economic mechanisms where all the information available at an individual node level condenses into a simple global metric, namely the *price*, thus reflecting the overall state of the system. The price incorporates all the available information which may reside at the level of the individual nodes and which is not necessarily shared among the other nodes. The basic institution in the economy is the market where producers and consumers can meet. A transaction is completed when both parties reach an agreement on the quantity and the price of goods (resource requests

or resource offers in our case). The way the basic market mechanism operates, is very simple: the price of a good goes up when there is more demand than supply and it goes down when there is more supply than demand. By analogy, we define the price of a resource to reflect the scarcity/abundance of that particular resource in the system.

Not only the price of a good is computed, but the cost related to find the appropriate good should also be taken into account. The transaction cost attached to submitting a request/offer by a node to the matchmaker will also be computed. The price is therefore an indicator on the scarcity of a particular good: if there is a shortage of resource, its price will go up and vice verse. The price can therefore be defined as a simple, single metric that summarizes the global state of a system.

Consider an ad hoc grid, when there is a relatively low workload, a single centralized matchmaker suffices. With the increasing workload in the ad hoc grid, the workload on the matchmaker will increase too and in order to ensure a timely and appropriate match, the matchmaker may decide to look for support by promoting other nodes to become matchmaker. This could be done on the basis of, for instance, the price the matchmaker can charge for finding a match. In case of an overload of such requests, and following the market logic, the price for finding a match will increase. The matchmaker may be constructed in such a way that once this price goes above a certain level, another matchmaker is initiated. This process may be repeated any number of times. In this way, the system organizes itself starting from a simple centralized state and evolving successively into increasingly distributed ones. One extreme situation, the P2P case, would be where every single node in the ad hoc grid is now a matchmaker. Whenever, the price goes again below a certain level, the reverse could happen and the matchmaker nodes are demoted to become ordinary nodes again in the system.

As explained above, a matchmaker is overloaded when it is unable to maintain its matchmaking capacity. The overloaded matchmaker promotes a normal node as a matchmaker. The newly promoted matchmaker starts performing matchmaking for some of the nodes that were under the responsibility of the overloaded matchmaker. The workload of the primary (overloaded) matchmaker is reduced. Thus, the overloaded matchmaker is able to maintain its matchmaking capacity and in this way the ad hoc grid is segmented into two or more segments, where, each segment has its own matchmaker. This system level self-organization and adaptation process in the ad hoc grid is summarized in Figure 4.6. The formulas for calculating a matchmaker's workload are

presented and explained in Section 4.3.

4.3 Matchmaker Workload Calculation

Ad hoc grid segmentation/de-segmentation is based on the workload of the current matchmaker(s) and is represented as the transaction cost. The TCost is attached to each request. The node submitting a request is supposed to pay this cost to the matchmaker. The transaction price is the amount that a consumer will pay to the producer for consuming the producer's resources and the TCost represents the matchmaker workload.

The matchmaker workload can be calculated in different ways. Even though, every choice of the cost function has some arbitrary aspect to it, we are looking for a simple and efficient one. We simply count the number of messages to be processed by the matchmaker before processing the newly received request/offer message[4]. The TCost is calculated for each request/offer message. The matchmaker agent maintains request and offer repositories in descending and ascending order of the bid and ask prices respectively. A new request/offer message is placed in a request/offer repository at its proper place by using insertion sort. If equations 4.5 and 4.6 represent the order of requests and offers before inserting a new request/offer message

$$requests = R_1 > R_2 > R_3 > ... > R_{N-1} > R_N \tag{4.5}$$

$$offers = O_1 < O_2 < O_3 < ... < O_{N-1} < O_N \tag{4.6}$$

and the newly received request/offer is placed at index L, where $0 < L < N$, then

$$requests = R_1 > R_2 > R_3 > ...R_{L-1} > R_L > R_{L+1}... > R_N > R_{N+1}$$

$$offers = O_1 < O_2 < O_3 < ...O_{L-1} < O_L < O_{L+1}... < O_N < O_{N+1}$$

TCost for a request/offer message:

[4] also used by [61] to calculate the workload.

$$TCost_{Request} = Count(R_1 ... R_{L-1}) \tag{4.7}$$

$$TCost_{Offer} = Count(O_1 ... O_{L-1}) \tag{4.8}$$

The TCost value for a matched request/offer pair is the average of their individual TCost values. TCost for a matched pair is calculated as:

$$MatchedPairTCost = (TCost_{Request} + TCost_{Offer})/2 \tag{4.9}$$

A matchmaker periodically calculates the average TCost which is calculated as follows:

$$AvgTCost = (\sum_{i=t_0}^{t} TCost(i)/(\sum_{i=t_0}^{t} N(i) \tag{4.10}$$

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

A matchmaker promotes a node as a matchmaker or demotes a matchmaker back to a normal node when its average TCost is above or below the matchmaker's *upper threshold* (see Section-4.5). In principle, the transaction cost could become infinitely high which would inhibit the grid to be operational as no matching could occur anymore. It is therefore, important to have mechanisms to counter such an event.

4.4 Assumptions

The following simplifying assumptions are adopted for the experimental results presented in this chapter.

1. We only consider the state of the ad hoc grid with one or more matchmakers and do not look at the P2P issues. Consequently, we do not consider the case where a single matchmaker(s) breaks down and the ad hoc grid goes into a P2P state;

2. We neither address the issue of routing of requests when more (or less) matchmakers are introduced nor the issue of how to create (or delete) new grid segments. We assume the necessary extensions to an overlay network that address this problem are available.

3. We assume that there is a known subset of nodes in the ad hoc grid that are the potential candidates for being promoted to matchmaker;

4. When a new matchmaker emerges, the initial matchmaker forwards requests to the new matchmaker in order to diminish its workload.

5. We do not address the issue of load balancing in this chapter even though arbitrage mechanisms between the matchmakers will be the key to this;

6. We assume that tasks are atomic and can not be subdivided. Similarly, one resource offer is allocated to one task only. Co-allocation of tasks and resources is not considered in this dissertation.

7. There is no budget constraint on a consumer/producer agent. In the presence of the budget constraint, the participation of a node is limited when it runs out of its allocated budget. As the dissertation discusses the segmentation/de-segmentation of the ad hoc grid based on the workload of the matchmaker and not according to the individual consumer/producer nodes, so this assumption is in place.

8. We assume that the participating agents (consumer, producer and matchmaker) are honest. Hence, we do not look into security, auditing and do not consider malicious behavior from the ad hoc grid nodes.

Most of these assumptions will be relaxed in the work presented in the subsequent chapters of this dissertation.

4.5 Experimental Setup and Results Discussion

The experimental results presented in this section evaluate the proposed self-organizing mechanism in terms of matchmaking efficiency, response time and

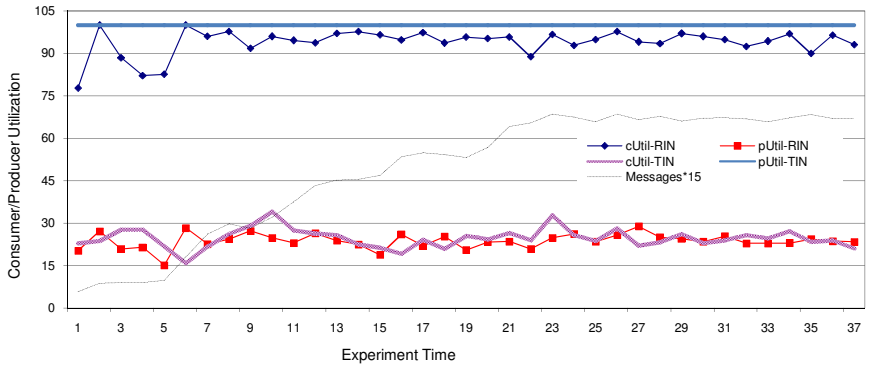


Figure 4.7: *Consumer/producer utilization of one matchmaker in RIN & TIN conditions.*

the TCost of a matchmaker. The matchmaker's matchmaking efficiency is represented in terms of consumer utilization and producer utilization and is calculated according to Equations 3.1 and 3.2 respectively. The formula for calculating the response time is given in Equation 3.3 and formulas for TCost are explained in Section 4.3. All the experiments are executed on PlanetLab. The number of nodes is varied from $15 - 650$, and the number of matchmakers varies from $1 - 5$. These experiments are executed in different network conditions.

In this section, we first present the experimental results with one matchmaker and discuss how we determine the lower and upper threshold levels for one matchmaker that will be used for promoting and demoting additional matchmakers. We then present the experimental results with multiple matchmakers to show the effect of dynamically introducing and removing multiple matchmakers. The threshold is applied to introduce multiple matchmakers in the ad hoc gird. The matchmaker attempts to adapt itself according to the varying workload.

4.5.1 One Matchmaker

The effect of request/offer utilization with increasing workload of the matchmaker in RIN and TIN conditions is depicted in Figure 4.7. Where, $cUtil-RIN$ and $cUtil-TIN$ represent the consumer utilization in RIN and TIN conditions respectively. Whereas, $pUtil-RIN$ and $pUtil-TIN$ represent the producer utilization in RIN and TIN conditions respectively. The workload of the matchmaker during the experiment is represented by $messages * 15$. As can be ob-

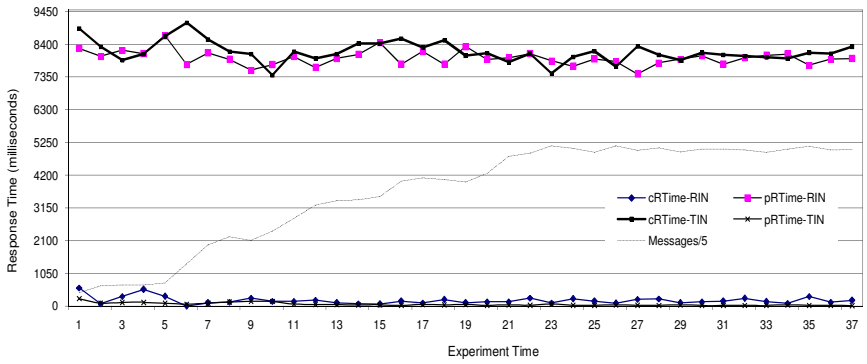


Figure 4.8: *Consumer/producer response time for one matchmaker in RIN & TIN conditions.*

served, the utilization of requests and offers, in both RIN and TIN conditions, fluctuates initially and becomes more or less stable over time.

This is explained by the initial low number of offers and requests that are submitted and can not be matched. As this number increases, the likelihood of finding a match increases, resulting in a higher utilization rate. As there are fewer resource offers than resource requests in TIN, therefore, resource offers get matched as soon as they are submitted to the matchmaker and hence the producer utilization ($pUtil - TIN$) is about 100% in TIN condition. Similarly, there are fewer resource requests than resource offers in RIN, therefore, resource requests get matched as soon as they are submitted to the matchmaker and hence the consumer utilization ($cUtil - RIN$) is about 100% in RIN condition. The consumer utilization in TIN ($cUtil - TIN$) and producer utilization in RIN ($pUtil - RIN$) is proportional to the ratio of the resource requests and resource offers in TIN and RIN conditions, which is 20%. The utilization becomes approximately constant for 350 message/minute or higher workloads.

The matchmaker response time for request/offer messages with the increasing workload of matchmaker is depicted in Figure 4.8. In this figure, $cRTime-RIN$ and $cRTime-TIN$ represent the matchmaker response time for consumer in RIN and TIN conditions respectively. Whereas, $pRTime - RIN$ and $pRTime - TIN$ represent the matchmaker response time for producer in RIN and TIN conditions respectively. The workload of the matchmaker, during the experiment, is represented by $messages/5$. The response time indicates a downward trend. This is because initially, the number of requests is low, and a request may have to wait for a longer period of time before being matched, since an increased number of requests induces a decrease in the response time.

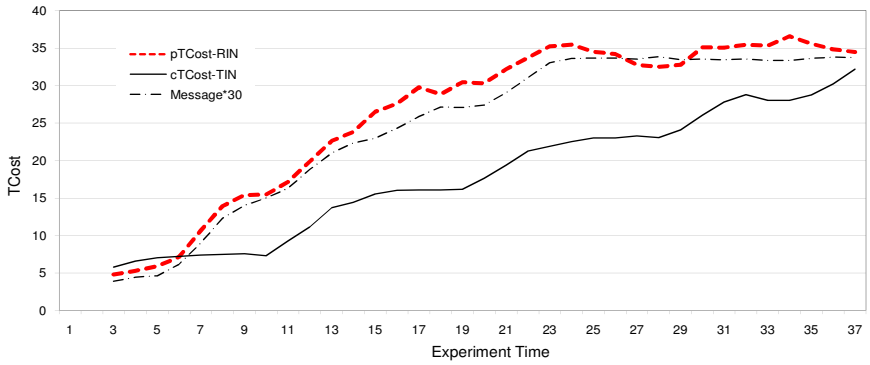
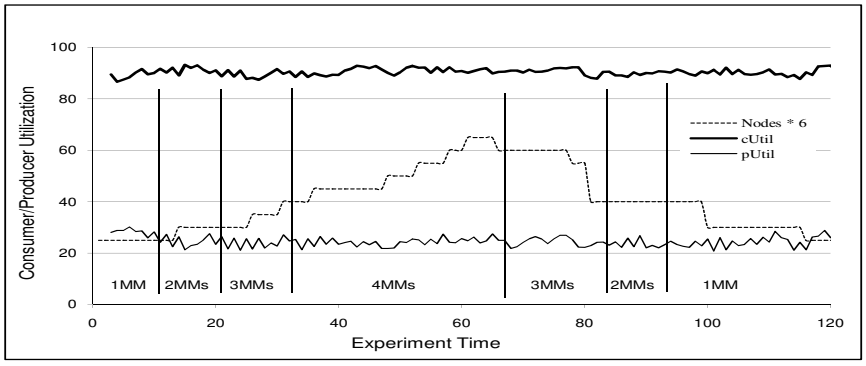


Figure 4.9: *Consumer/producer TCost for one matchmaker in RIN & TIN conditions.*

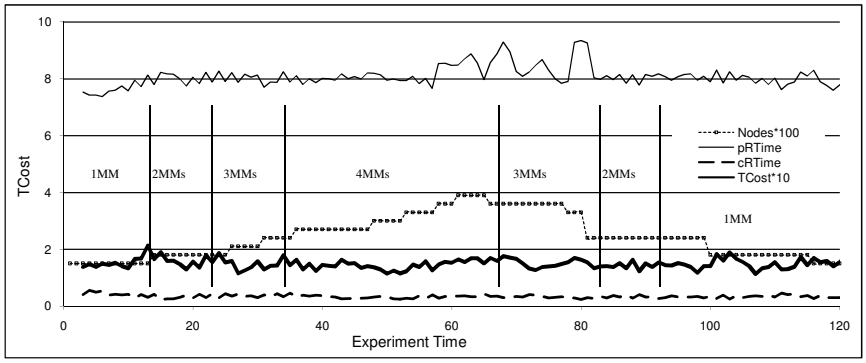
As more messages come in, the response time goes further down as the likelihood to be matched goes up. However, from our experiments, we can see that this downward trend is halted and tends to go up again as the number of messages that need to be matched grows, requiring more processing time.

A similar evolution can be seen in Figure 4.9, which depicts the effect of the consumer TCost variation in TIN condition ($cTCost - TIN$) and producer TCost variation in RIN condition ($pTCost - RIN$) with increasing workload of the matchmaker ($message * 30$). As can be expected, TCost is proportional to an increasing workload. However, the rate of the increasing trend is reduced for 350 message/minutes workload of the matchmaker in TIN, and 650 messages/minute in RIN. As resource requests in RIN and resource offers in TIN get matched as soon as they are submitted, therefore, the consumer TCost in RIN and producer TCost in TIN is observed almost equal to zero.

It can be concluded from Figure 4.7 & 4.8 that after 350 messages/minute in RIN condition and 650 messages/minute in TIN condition, the consumer/producer utilization does not show any change and the response time shows an increasing tendency, though at a very slow rate. However, TCost depicts an increasing trend proportional to the workload on the matchmaker (Figure 4.9). This status can be considered as the overload point of a matchmaker. At this point, the consumer/producer has to pay a higher TCost for availing the matchmaking service of the matchmaker. Furthermore, the matchmaker needs a second matchmaker at this point. TCost upper threshold, at the observed overload point, was 15 with the given experimental setup.



(a)



(b)

Figure 4.10: *Ad hoc grid with multiple adaptive matchmakers in RIN condition.* ***(a)*** *Matchmaking efficiency in RIN condition.* ***(b)*** *TCost & Response time (milliseconds) in RIN condition.*

4.5.2 Multiple Matchmakers in RIN

In the next set of experiments, we show that by adding new matchmakers dynamically have a positive impact on the matchmaking efficiency and an attenuating effect on the TCost and response time. The consumer/producer utilization (represented as *cUtil* and *pUtil*) with multiple adaptive matchmakers in a RIN (Resource Intensive Network) condition under varying workload of the ad hoc grid ($Nodes * 6$) is depicted in Figure 4.10a. The producer utilization in RIN is generally proportional to the percentage of the scarce commodity in RIN. The consumer utilization is about 90%. Some requests expire during the waiting period.

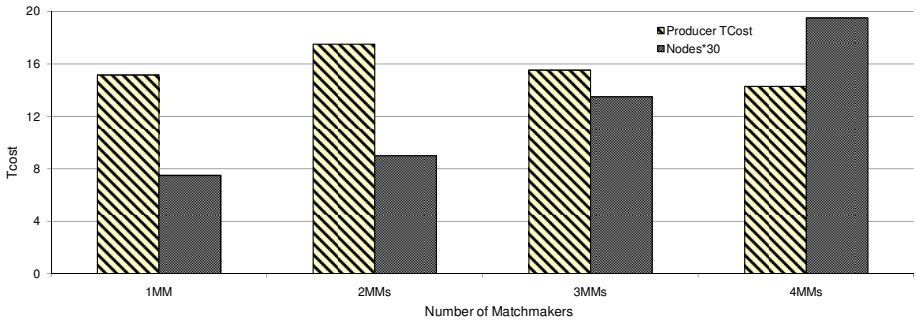
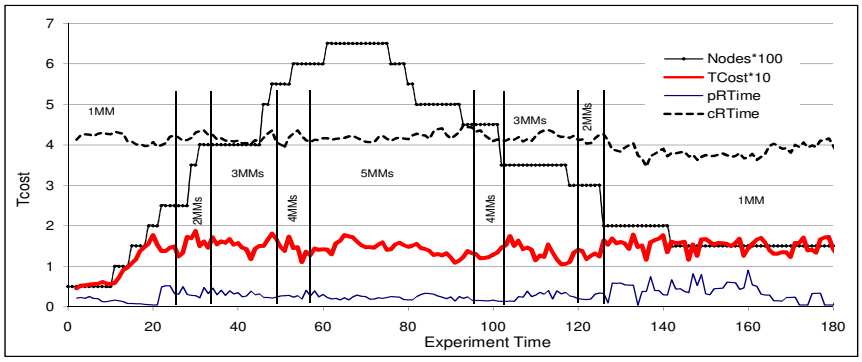


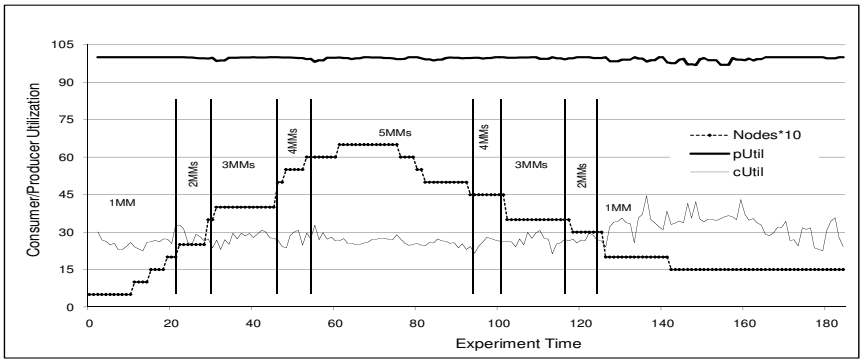
Figure 4.11: *Average TCost and the number of matchmakers in RIN condition.*

The evolution of the consumer/producer response time ($cRTime$ and $pRTime$) and producer TCost ($TCost * 10$) with multiple adaptive matchmakers under varying workload of the ad hoc grid ($Nodes * 100$) is graphically represented in Figure 4.10b. As tasks are scarce and resources are in abundance in RIN, therefore, a request is matched as soon as it is submitted to the matchmaker. Consequently, the consumer TCost is almost '0'. A new matchmaker is introduced when the first matchmaker reached its TCost threshold value. When new matchmakers are introduced, the TCost value decreases after the introduction of a new matchmaker. This phenomenon is reversed when a matchmaker is removed. Again, we can observe that the TCost remains stable irrespective of the number of messages the matchmaker receives from the ad hoc grid nodes. Temporary fluctuations in TCost value also occur, reflecting variations in the grid activity. The consumer response time is much lower than the producer response time due to the abundance of resources. It is easy for a task to find the appropriate resources, but the opposite is not the case. Figure 4.11 depicts the average TCost of the number of matchmakers with increasing workload. It becomes clearer from Figure 4.11 that the ad hoc grid can process more workload, and yet the average TCost of all the matchmakers at any given instance remains closer to the upper TCost threshold for one matchmaker.

When comparing consumer/producer utilization in the ad hoc grid with one matchmaker and multiple adaptive matchmakers in Figures 4.7 & 4.10a, we can see that the matchmaking efficiency remains stable in spite of an increase of the workload. This is because the workload is now spread over 2 or more matchmakers. When comparing Figures 4.9 & 4.10b, we can see that the upward pressure on the TCost has been neutralized again, thanks to the change in the number of matchmakers.



(a)



(b)

Figure 4.12: *Ad hoc grid with multiple adaptive matchmakers in TIN condition.* ***(a)*** *Response time and TCost in TIN condition.* ***(b)*** *Consumer/producer utilization in TIN condition.*

4.5.3 Multiple Matchmakers in TIN

The resources are scarce and the tasks are in abundance in TIN condition. When an offer is submitted to the matchmaker, it immediately gets matched. Consequently, the producer TCost is almost 0 in TIN setting. Figure 4.12a depicts the consumer TCost variation in presence of multiple adaptive matchmakers. The TCost value keeps on increasing with the 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

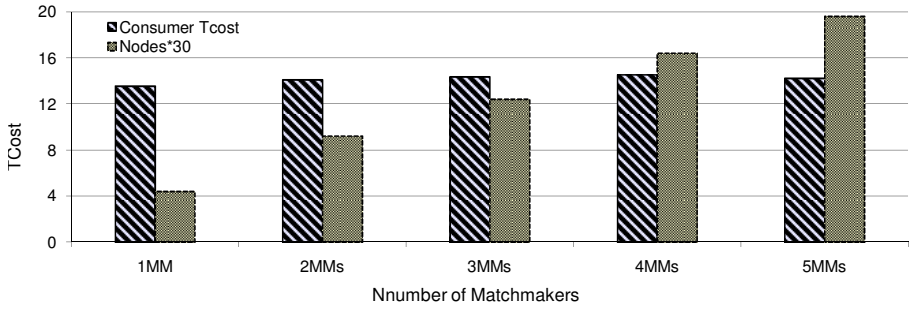


Figure 4.13: *Average TCost and the number of matchmakers in TIN condition.*

changes in the workload of the overall grid. Overall and compared with the TCost evolution of a single matchmaker with the increasing workload (Section 4.5.1), the TCost remains relatively stable and does not increase (Figure 4.13).

A similar observation can be made for the response time from Figure 4.12a. Rather than going up with an increasing workload, the increase of the number of matchmakers has a stabilizing effect on the response time. Figure 4.12b depicts the consumer/producer utilization in TIN with multiple adaptive matchmakers. The consumer utilization in TIN is generally proportional to the percentage of the scarce commodity in TIN, whereas, the producer utilization is about 100% in TIN condition.

Thus, the proposed self-organizing mechanism has a neutralizing affect on the TCost and response time of the ad hoc grid in TIN condition. Whereas, the consumer/producer utilization remains the same as that of one matchmaker (Section 4.5.1), in spite of the increased workload of the ad hoc grid.

4.6 Concluding Remarks

A self-organizing mechanism for a dynamic ad hoc grid infrastructure is proposed in this chapter. The proposed mechanism focuses on the workload of matchmaker to introduce self-organization in an ad hoc grid. The *upper* and *lower values* of the matchmaker workload threshold (TCost) are determined for different network conditions. The upper and lower threshold values are applied for dynamically introducing multiple matchmakers such that the grid can continue to execute all its tasks irrespective of the number of messages that are being sent and independent of its internal state (resource or task intensive). All

the experiments are executed on PlanetLab providing a realistic platform for testing the proposed mechanism.

The results show that the capability of the ad hoc grid to instantiate multiple matchmakers has a stabilizing effect on the TCost and response time without negatively affecting the consumer/producer utilization. This way, we guarantee that TCost and response time become invariant to the scale on which the ad hoc grid is operating. In the next chapter, we will relax assumptions 1, 2, 3 and 4 (described in Section 4.4) by extending a structured overlay network.

Chapter 5

Structured Overlay Network Extensions

A resource management mechanism that enabled the ad hoc grid to self-organize according to the workload of resource manager (*matchmaker*) was proposed in Chapter 4. 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 Chapter 4, the proposed micro-economic based self-organizing mechanism was studied under certain assumptions related to the underlying network (assumptions 1, 2, 3 and 4 in Section 4.4).

In this chapter[1], we define extensions of a structured overlay network [123] to relax the assumptions 1, 2, 3 and 4 (described in Section 4.4). These extensions are achieved by defining algorithms for node joining, finding a responsible matchmaker by a joining/existing node, ad hoc grid segmentation (by promoting nodes as matchmakers), and ad hoc grid de-segmentation (by demoting matchmakers as normal nodes) [20]. These algorithms are implemented on top of the Pastry structured overlay network. The experimental results presented in this chapter, verify that the proposed extensions enable the

[1]This chapter is based on the following research articles:

T. Abdullah, L.O. Alima, V. Sokolov, D Calomme, K.L.M. Bertels, "Hybrid Resource Discovery Mechanism in Ad Hoc Grid Using Structured Overlay", 22nd International Conference on Architecture of Computing Systems, March 2009.

T. Abdullah, V. Sokolov, B Pourebrahimi, K.L.M. Bertels, "Self-Organizing Dynamic Ad Hoc Grids", In Proceedings of 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW 2008), Venice, October 2008.

ad hoc grid to self-organize according to the workload of matchmaker. The proposed extensions make our hybrid resource discovery mechanism for ad hoc grid, dynamic and flexible.

5.1 P2P Structured Overlay Network

The proposed self-organization mechanism that extends a structured P2P overlay network, Pastry [123], is coded in JAVA and is tested on PlanetLab [1]. Before presenting the algorithms for extending a structured overlay network, we first present an overview of the existing distributed hash table based structured overlay networks.

An overlay network is a network built on top of an existing network, for example the Internet is built on top of the existing telephone network. Structured overlay networks are DHT based networks and are essentially built on top of the existing Internet. We selected structured overlay networks due to their flexibility, scalability and no adherence to any fixed network topology. Pastry [123], Chord [129], CAN (Content Addressable Networks) [120], DKS (Distributed K-ary System) [24], and Tapestry [147] are among the 2^{nd} generation of P2P routing and location schemes. DHT based networks route a message to a logical address, derived from a hash function, instead of an IP address. Message routing and content access are based on the name and a hashed value pair. Thus, any participant having a name of any particular node can retrieve the value maintained/stored by that node (Figure 5.1). In this way, the job of mapping names and values is distributed among the nodes so that addition or removal of a node is performed with minimal efforts.

Pastry is a completely decentralized, scalable and self-organizing structured overlay network with respect to node join/leave and overlay network maintenance [123]. Although, we used Pastry in this work, but it can be replaced with any other DHT based network like Chord, CAN, DKS or Tapestry. We used Pastry for node joining/leaving, for node-to-node and node-to-matchmaker communication.

Each node in Pastry is assigned a 128-bit unique identifier (nodeID) from a circular nodeID space that ranges from 0 to $2^{128} - 1$. This nodeID is randomly assigned while a node joins the pastry overlay network. The nodeID of a node indicates the position of a node on the circular nodeID space. The nodeID is generated by computing a cryptographic hash of node's IP address or public key or location or geographic distance, or a combination of any of these parameters. The randomly generated nodeID thus ensures, with high

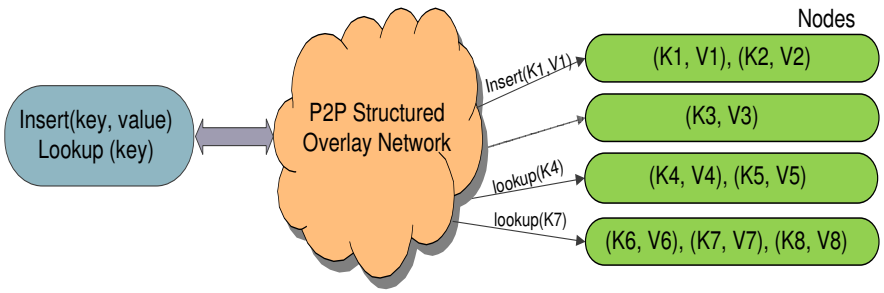


Figure 5.1: *Key lookup in a P2P structured overlay network.*

probability, that the nodes with adjacent nodeIDs are diverse in geography, ownership, etc.

Each pastry node maintains a *routing table* and a *leaf set*. The routing table is organized into $\log_{2^b} N$ rows with $2^b - 1$ entries in each row (N being the total number of nodes in Pastry overlay network and b is a Pastry configuration parameter with default value as $b = 4$). With $N = 10^6$ nodes and $b = 4$, a routing table will contain about 75 entries $(\log_{2^4} 10^6 \, x \, 2^4 - 1)$ and the maximum number of routing hops are $5\,(= \log_{2^4} 10^6)$; and with $N = 10^9$ nodes, a message is delivered in 7 hops.

A leaf set contains L nodes, with $L = 2^b$ as default value. The leaf set contains $|L|/2$ nodes of numerically closest higher nodeIDs and $|L|/2$ nodes of numerically closest smaller nodesIDs to any given node's nodeID. Pastry uses its routing table and leaf set in its message routing process as described below:

Pastry routes a message from the origin node to the destination node that exactly matches to the given key or is numerically closest to the given key. The message routing in Pastry works as follows: A node receives a message from some other node, and first checks whether the destination nodeID falls within its leaf set or not. If the destination nodeID exists in its leaf set, the message is directly forwarded to the destination node; otherwise, the node consults its routing table, and the message is forwarded to the node that shares a common prefix with the destination nodeID by at least one digit and is numerically closest to the destination key than the current node's nodeID.

For example, a message is to be routed from a node (with nodeID 34b22dc) with key b978b1 in Pastry structured overlay network. In first hop the first digit of the key and the nodeID of receiving node are common (b), in the next hop - 2 digits (b9), then - 3 (b97) and so on until the message finds the destination node or a node with nodeID closest to the desired one. Fig-

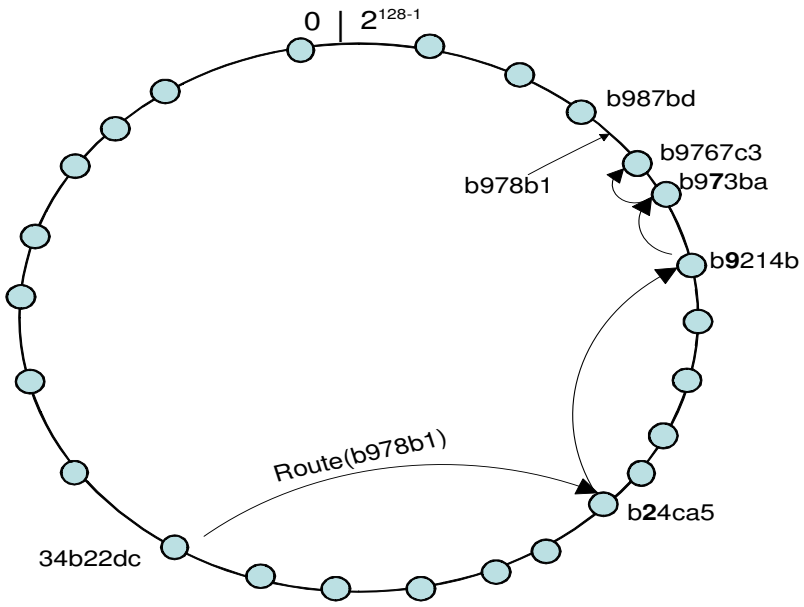


Figure 5.2: *Pastry message routing.*

ure 5.2 (adapted from [123]) summarizes the above explained message routing principle.

Routing performance of Pastry is scalable and robust. The maximum expected number of routing steps is $\log_{2^b} N$, where b is a Pastry configuration parameter with $b = 4$ as the default value and N represent the total number of nodes in Pastry [123] ring. Pastry routing performance degrades gradually in the event of node failures, unless $|L|/2|$ nodes with consecutive nodesIDs fail simultaneously; though the chances of such occurrences are least due to the expected diversity of the nodes with adjacent nodeIDs.

5.2 Pastry Extensions for Dynamic Segmentation and De-segmentation

In this section, we present the algorithms that extend a structured overlay network, Pastry, and focus on the dynamic segmentation and de-segmentation of

the ad hoc grid. These algorithms include; *promote a matchmaker*, *demote a matchmaker*, *discover a responsible matchmaker* and *node join/leave algorithm*.

There can be N nodes in the ad hoc grid. Each node can play the role of a *consumer*, *producer* or a *matchmaker*. The matchmaker receives offers and requests of resources from the nodes in ad hoc grid, which are then matched by the matchmaker. Each node is assigned a unique node identifier (nodeID). As the proposed algorithms are extensions of a structured P2P overlay network, principally any structured P2P overlay network can be used to implement the proposed algorithms. We used Pastry [123] to implement the proposed extensions. As 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 the N nodes ($M \leq N$).

The whole identifier space is divided into zones. Each zone has a responsible matchmaker. Each joining consumer/producer/matchmaker node knows the $ZoneNumber$ to which it belongs to and is provided with M and N. 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 the matchmaking process. The consumer/producer nodes under the responsibility of the 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 the matchmaker promotion (segmentation) is explained in Section 5.4.

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 their new matchmaker. The matchmaker demotion (de-segmentation) algorithm is explained in Section 5.5.

A consumer/producer node finds its responsible matchmaker node with the provided information (i.e. M, N and the node's $ZoneNumber$), after joining the ad hoc grid. If there is only one matchmaker in the ad hoc grid, then it becomes the responsible matchmaker for all the consumer/producer nodes. The consumer/producer node can submit resource requests/offers 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 matchmaker nodes. Sections 5.6 & 5.7 explain the algorithms for node joining and discovering a responsible matchmaker, respectively.

5.3 Matchmaker Underload and Overload

The algorithms for promoting and demoting a matchmaker are based on the overload and underload conditions of a matchmaker, respectively. A matchmaker is overloaded when its workload goes above the TCost threshold and is unable to maintain its matchmaking capacity. Similarly, an underloaded matchmaker is the one whose workload goes below the TCost threshold and is no longer needed to act as a matchmaker. The formulas for calculating TCost and its upper and lower threshold limits are detailed in Section 4.3 & 4.5, respectively.

5.4 Promote a Matchmaker

This algorithm is executed by an overloaded matchmaker 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}) for sharing its excess workload. In this way, the overloaded matchmaker (M_i) is able to maintain its matchmaking capacity.

The newly promoted matchmaker (M_{i-1}) changes its matchmaking status to active, as depicted in Figures 5.3a & 5.3b. The matchmaker M_{i-1} updates its predecessor matchmaker (M_{i-2}) about the change in its matchmaking status. The newly promoted 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 "cor-

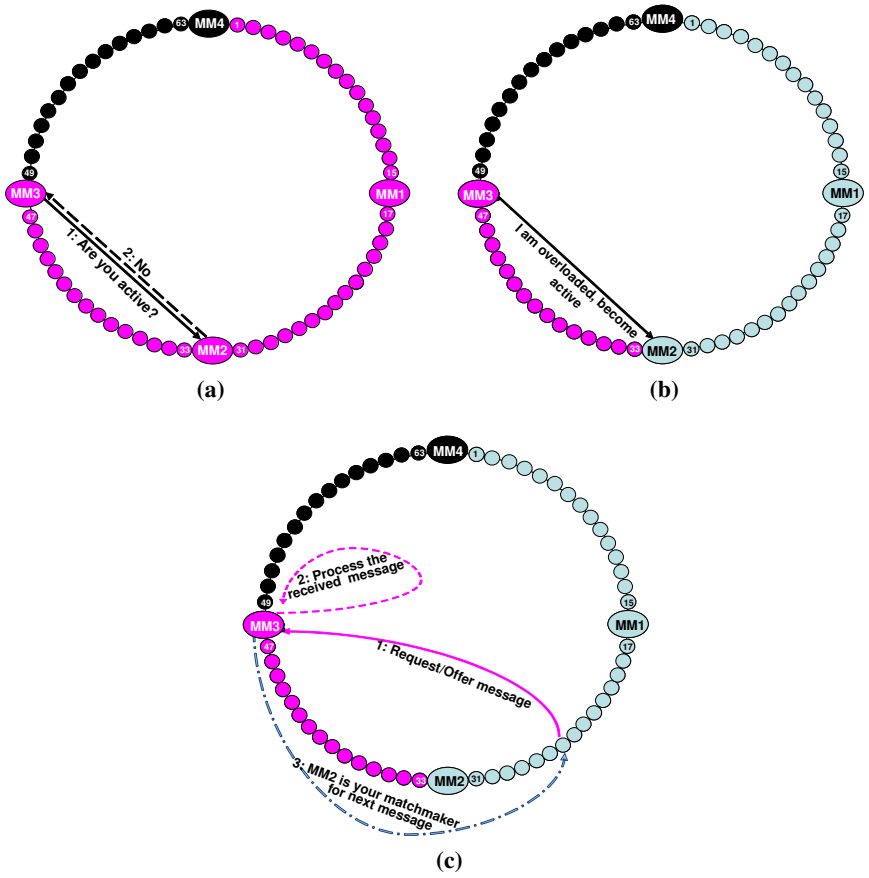


Figure 5.3: *Promote a matchmaker.*

rection 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 messages from the nodes belonging to zone $i-1$ and will update the respective nodes 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 of updating the consumer/producer nodes is graphically illustrated in Figure 5.3c.

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

Algorithm 5.1 Promote a 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 changes matchmaking status of M_{i-1} to TRUE
5: M_{i-1} updates its changed matchmaking status to M_{i-2}
6: M_i updates the matchmaker change to the nodes in zone $i-1$
7:ELSE
8: M_i Forwards excess workload to M_{i+1}
9:END IF
10:END IF

algorithm is listed in Algorithm 5.1.

As stated before, there can be N nodes (consumer/producer/matchmaker) in the ad hoc grid and out of the N nodes, there can be a maximum of M matchmakers ($M \leq N$). The ad hoc grid is thus divided into a 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 (Section 5.7). The number of effective zones increases with the promotion of matchmakers and decreases with the demotion of matchmakers (Section 5.5). In this way, the matchmaker promotion results in segmentation of the ad hoc grid and the matchmaker demotion results in de-segmentation of the ad hoc grid.

5.5 Demote a Matchmaker

This algorithm is executed by an underloaded matchmaker in the ad hoc grid. An underloaded matchmaker (say M_i, a matchmaker for zone i) demotes itself by changing its matchmaking status. The demoted matchmaker updates its predecessor matchmaker (M_{i-1}) and successor matchmaker (M_{i+1}) about the change in its matchmaking status, as depicted in Figures 5.4a & 5.4b.

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 M_i, after demoting itself, will forward the currently received request/offer messages, from nodes in zone i , to its successor match-

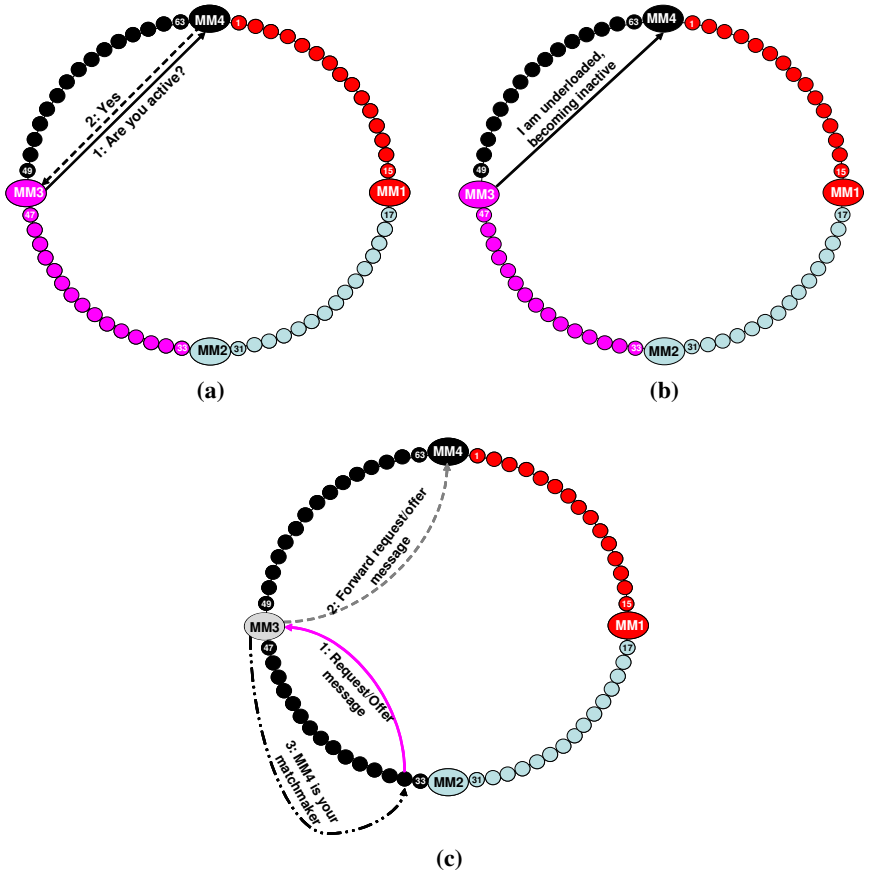


Figure 5.4: *Demote a matchmaker.*

maker (M_{i+1}). The matchmaker M_i also sends a message to the respective consumer/producer nodes, about the change of their responsible 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 5.4c. The matchmaker demotion algorithm is listed in Algorithm 5.2.

5.6 Node Join/Leave Algorithm

Each joining consumer/producer/matchmaker node knows the *ZoneNumber* to which it belongs to and is provided with M and N. We used Pastry node

Algorithm 5.2 Demote a 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

Algorithm 5.3 Node join algorithm.

1:Node join the ad hoc grid into zone i
2:IF (*Joining node is Consumer/Producer*) THEN
3: CALL $Discover - Responsible - Matchmaker$
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

join protocol [123] for node joining in our ad hoc grid. The consumer/producer node and the matchmaker node perform different set of actions after joining the ad hoc grid.

The consumer/producer node discovers its responsible matchmaker node with the provided information (Section 5.7). 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, it exchanges predecessor/successor matchmaking status information. The algorithm for joining the ad hoc grid is listed in Algorithm 5.3.

5.7 Discovering a Responsible Matchmaker

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

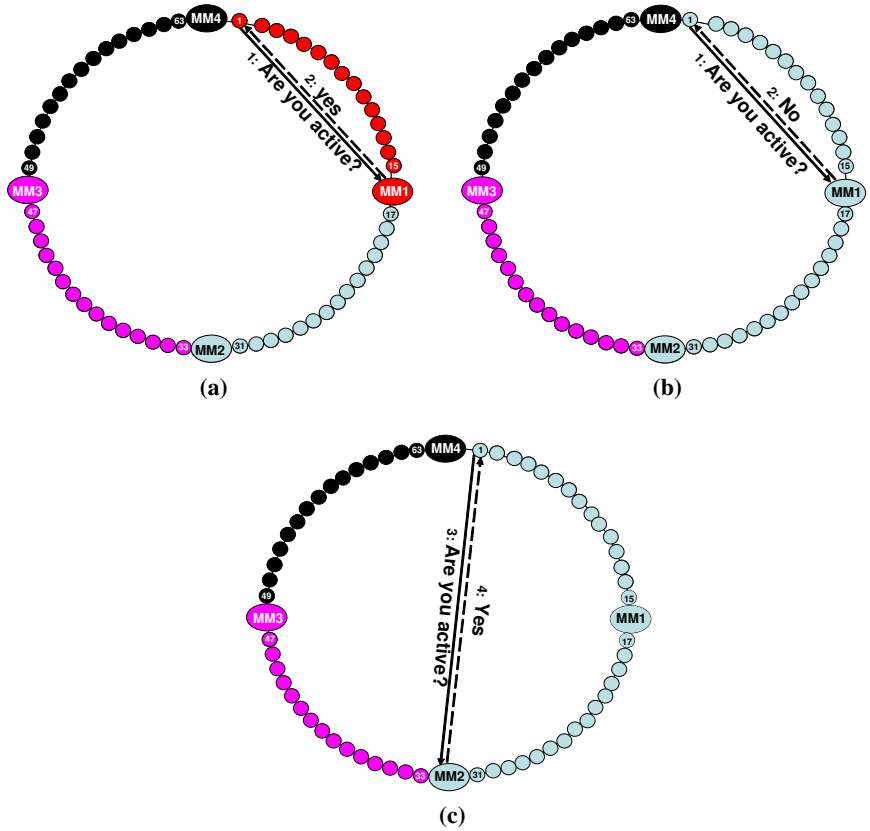


Figure 5.5: *Discovering a responsible matchmaker.*

is under the responsibility of a matchmaker.

This algorithm is executed by a 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 (Figure 5.5a). The consumer/producer node will send all its resource request/offer messages to this matchmaker.

If a matchmaker does not exist for a zone (say matchmaker M_i for zone i does not exist), then the consumer/producer node checks for the successor matchmaker (M_{i+1}) of the matchmaker M_i. The consumer/producer node continues searching for an active matchmaker node, until it finds one (Figures 5.5b & 5.5c). It is important to mention that the algorithm for finding a responsible matchmaker by a newly joining node does not cover the matchmaker node

Algorithm 5.4 Discovering a 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 l = i + 1
6:    set counter = 1
7:    WHILE (counter < M)
8:        Query Matchmaker_l
9:        IF (MatchMaker_l is active) THEN
10:           set MatchMaker_l as responsible matchmaker
11:           BREAK
12:       END IF
13:           l = l + 1
14:           counter = counter + 1
15:   END WHILE
16:END IF
```

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 5.4.

5.8 Message Complexity Analysis

Message complexity analysis shows the communication cost of the proposed algorithms. When a matchmaker node joins the ad hoc grid then it exchanges one message with its predecessor and successor matchmakers to get their matchmaking status information. The number of messages exchanged by a

Parameter	Messages in worst case
Consumer/producer/matchmaker node join	$O\ log_{2^b} N$
Predecessor/successor matchmaker update	2
Finding responsible matchmaker node	M
Matchmaker promotion /demotion	2
Updating each consumer/producer after matchmaker promotion / demotion	1

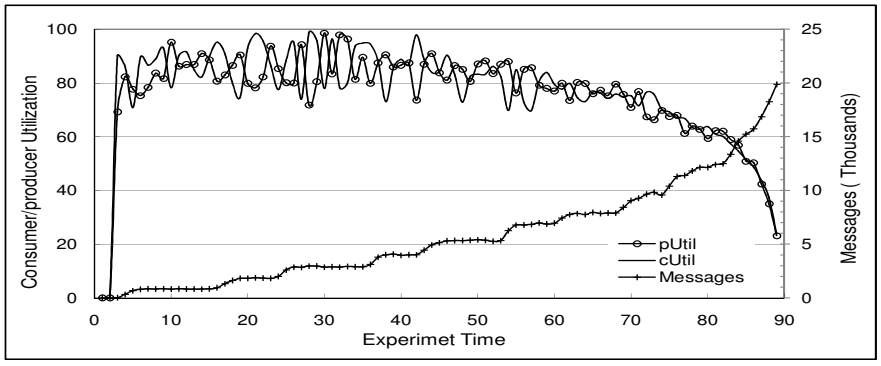
Table 5.1: *Message complexity analysis of the presented algorithms.*

consumer/producer node in finding its responsible matchmaker node depends on the number of active matchmaker in the ad hoc grid. The number of messages in finding a responsible matchmaker node will decrease with the increasing number of active matchmaker in the ad hoc grid. The number of messages to find a responsible matchmaker node varies between 1 and M (maximum number of matchmaker nodes in the ad hoc grid).

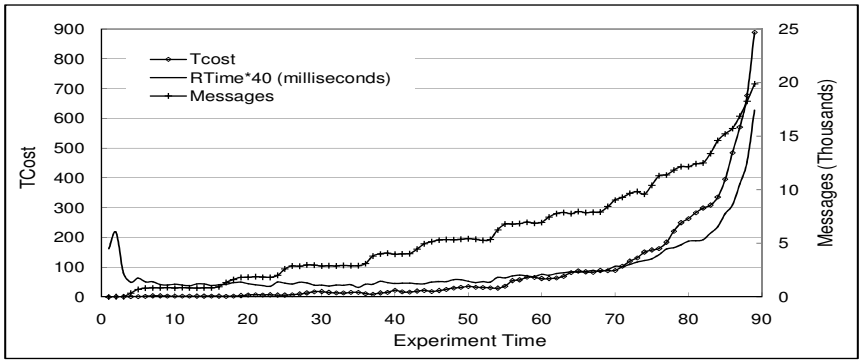
Only one message is sent from the overloaded matchmaker node (say M_i) to its predecessor matchmaker node (M_{i-1}), while promoting a node to a matchmaker node. The matchmaker (M_{i-1}) will send one message to update its predecessor matchmaker (M_{i-2}), about change of its matchmaking status. The overloaded matchmaker (M_i) will send one message, about the change of the responsible matchmaker, to each consumer/producer node belonging to zone $i-1$. The update message is sent only when the overloaded matchmaker (M_i) receives request/offer messages from the consumer/producer nodes belonging to zone $i-1$. The consumer/producer nodes change their responsible matchmaker node from M_i to M_{i-1}, after receiving this update message. Similarly, one message is sent from an underloaded matchmaker (say M_i) to its successor matchmaker (M_{i+1}) and the predecessor matchmaker (M_{i-1}) about the change of its status. The underloaded matchmaker (M_i) will send one message, about the change of the responsible matchmaker, to each consumer/producer node belonging to zone i. The update message is sent only when the underloaded matchmaker (M_i) receives request/offer messages from the consumer/producer nodes belonging to the zone i. The consumer/producer nodes change their responsible matchmaker node from M_i to M_{i+1}, after receiving this update message. Table 5.1 summarizes the message complexity analysis.

5.9 Experimental Setup and Results Discussion

The proposed algorithms are evaluated by consumer/producer utilization, matchmaker response time for a matched pair and the TCost. The consumer/producer utilization is calculated according to Equations 3.2 and 3.1. The formulas for calculating TCost are detailed in Section 4.3. The matchmaker response time for a matched request/offer is calculated as the time interval between a request/offer receiving time and the time when the matchmaker finds a matching offer/request. All the experiments are executed on PlanetLab in TIN, RIN and BN network conditions. The number of nodes is varied from $15-650$ and the number of matchmakers is varied from $1-5$.



(a)



(b)

Figure 5.6: *Ad hoc grid with one matchmaker in the balanced network condition.* ***(a)*** *Consumer/producer utilization of one matchmaker with increasing workload.* ***(b)*** *TCost & the response time of one matchmaker with increasing workload.*

In this section, we first present experimental results with one matchmaker. Later, we present the experimental results with multiple matchmakers to show the effect of dynamic promotion and demotion of matchmaker(s). In the second set of experiments, the effect of promotion (segmentation of ad hoc grid) and demotion (de-segmentation of ad hoc grid) of matchmaker(s) in an ad hoc grid on transaction cost, matchmaker response time and the consumer/producer utilization are compared with the ad hoc grid having only one matchmaker.

The effect of consumer/producer utilization with increasing workload of one matchmaker is depicted in Figure 5.6a. The consumer utilization is represented by *cUtil,* and producer utilization is represented by *pUtil* in Figure

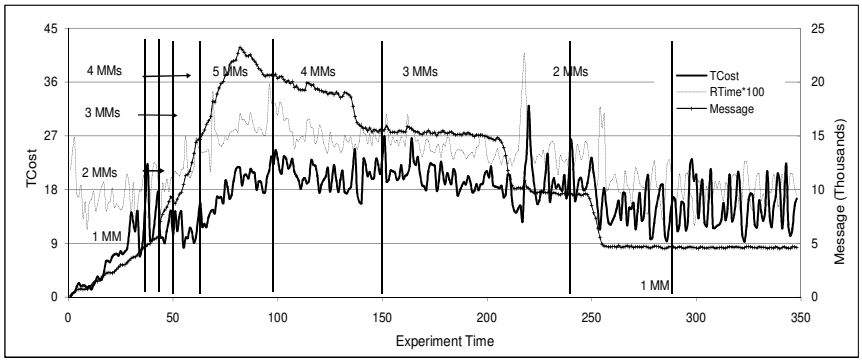
5.6a. Whereas, Figure 5.6b depicts the effect of the consumer/producer TCost and response time variation with increasing workload of one matchmaker. The horizontal axis represents the experiment time in both the figures. The primary vertical axis represents TCost and consumer/producer utilization in Figures 5.6a & 5.6b, respectively. Whereas, the secondary vertical axis represents the respective number of request/offer messages submitted to the matchmaker in both these figures.

TCost of the matchmaker and response time increase with the increasing workload and consumer/producer utilization decreases with the increasing workload of one matchmaker as depicted in Figures 5.6a & 5.6b, respectively. The increasing trend of TCost, response time and the 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 has to pay a higher TCost for availing the matchmaking service. The matchmaker needs additional matchmakers at the point, when its matchmaking efficiency starts decreasing. This is the point, where our mechanism becomes useful.

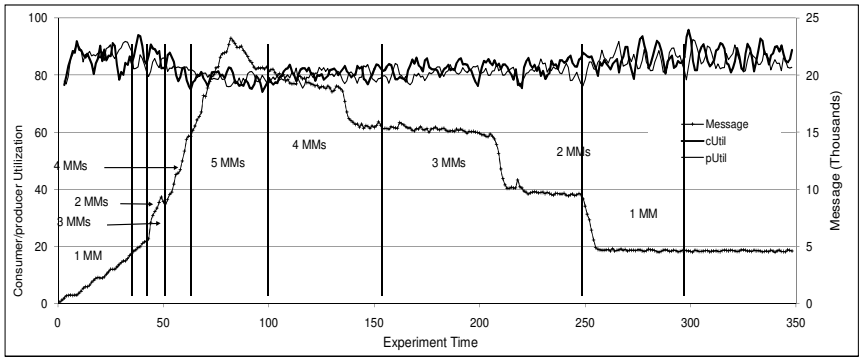
TCost and the response time variation in the presence of multiple adaptive matchmakers are shown in Figure 5.7a. The horizontal axis represents experiment time in both Figures 5.7a & 5.7b. The primary vertical axis represents the TCost and consumer/producer utilization in these figures. Whereas, the secondary vertical axis represents the respective number of request/offer messages submitted to the matchmaker(s) in both these figures.

TCost value keeps on increasing with the increasing matchmaker workload. A new matchmaker is introduced when the first matchmaker reached its TCost threshold value. TCost value decreases after the introduction of a new matchmaker. The opposite is observed when a matchmaker is removed. Evidently, the TCost also increases/decreases temporarily reflecting the changes in workload of the overall grid. Overall and compared with the TCost evolution with a single matchmaker and increasing workload (Figure 5.6b), the TCost remains relatively stable and does not increase (Figure 5.7a). Instead of going up with an increasing workload, the increase of number of matchmakers has a stabilizing effect on the response time. This is exactly what was expected.

The matchmaking efficiency with multiple adaptive matchmakers is depicted in Figure 5.7b. The matchmaking efficiency remains about 80% with the increasing workload of the matchmaker. Whereas, in case of one matchmaker, matchmaking efficiency showed a continuous decreasing trend with the increasing workload (Figure 5.6a). The experiments executed for RIN and TIN



(a)



(b)

Figure 5.7: *Ad hoc grid with multiple matchmakers.* ***(a)*** *TCost & the response time of ad hoc grid with multiple adaptive matchmakers in a balanced network condition.* ***(b)*** *Consumer/producer utilization with multiple adaptive matchmakers in a balanced network condition.*

conditions showed the similar behavior of TCost, response time and the consumer/producer utilization as observed and discussed in Section 4.5, therefore, the experimental results of the RIN and TIN conditions are not repeated here.

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 TCost and the response time without negatively affecting the consumer/producer utilization. This way, we guarantee that TCost and the response time become invariant to the scale on which the ad hoc grid is operating. These conclusions also confirm that the proposed algorithms for joining ad hoc grid, finding a responsible matchmaker, promoting a matchmaker, demoting a

matchmaker and message routing on a P2P overlay network work as expected.

5.10 Concluding Remarks

A dynamic, self-organizing mechanism for a dynamic ad hoc grid infrastructure was proposed in this chapter. The proposed mechanism focuses on the extensions of a structured overlay network to manage the (dis)appearance of matchmakers in an ad hoc grid and to route the messages to the appropriate matchmaker in an 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. In spite of the increasing workload, the matchmaking efficiency and capacity of the ad hoc grid was sustained by applying the proposed mechanism.

Chapter 6

Social Networks and Self-organization

In Chapter 4, the proposed infrastructure level self-organization mechanism was studied from the micro-economics perspective and the algorithms required for relaxing the simplifying assumptions, related to the underlying network, were explained in Chapter 5. This chapter focuses on the second part of the dissertation by studying the proposed infrastructure level self-organization mechanism from the social networks perspective.

In this chapter[1], we study the proposed infrastructure level self-organization mechanism from the social networks perspective and look at the impact of the adoption of a particular infrastructure, taken from the infrastructural continuum, by exploring the following issues. First, we define the term social network of a node in terms of the degree of neighborhood of a node in

[1]This chapter is based on the following research articles:
T. Abdullah, K.L.M. Bertels, L.O. Alima, Z. Nawaz, "Effect of Dgree of Neighborhood on resource discovery in Ad Hoc Grids", 23nd International Conference on Architecture of Computing Systems, February 2010

T. Abdullah, L.O. Alima, V. Sokolov, D Calomme, K.L.M. Bertels, "Hybrid Resource Discovery Mechanism in Ad Hoc Grid Using Structured Overlay", 22nd International Conference on Architecture of Computing Systems, March 2009.

T. Abdullah, V. Sokolov, B Pourebrahimi, K.L.M. Bertels, "Self-Organizing Dynamic Ad Hoc Grids", In Proceedings of 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW 2008), Venice, October 2008.

an ad hoc grid. The degree of the neighborhood of a node is defined in a fully centralized, multiple adaptive matchmakers and in fully decentralized (P2P) environments for an ad hoc grid. Secondly, we analyze the effect of varying the degree of neighborhood in a fully decentralized ad hoc grid. Thirdly, we compare the results of varying the degree of neighborhood in a fully decentralized approach to a fully centralized and with the multiple adaptive matchmakers approach. Fourthly, we perform the message complexity analysis of the above mentioned resource discovery approaches in order to understand the communication cost of a particular resource discovery approach. Finally, we give recommendations for trade-offs in resource discovery on an infrastructural spectrum ranging from fully centralize to fully decentralize extremes in an ad hoc grid.

Resource management is one of the important issues in the efficient use of grid computing in general, and poses specific challenges in the context of ad hoc grid due to the heterogeneity, dynamism and intermittent participation of the participating nodes in an ad hoc grid. Resource discovery approaches can be categorized into three broader categories; fully centralized, fully decentralized and the hybrid approach. These approaches and their advantages and disadvantages are discussed in Section 2.4. Fully centralized and fully decentralized approaches are often considered being mutually exclusive and residing on the two extremes of the infrastructural spectrum. A major issue concerning the effectiveness of these resource discovery approaches is the visibility and/or accessibility of resources to the tasks. We refer to this visibility as the *neighborhood* and study the effect of neighborhood on fully centralized, fully decentralized and on multiple adaptive matchmakers (hybrid) approach in an ad hoc grid. The notion of neighborhood and the degree of neighborhood at different points of an infrastructural spectrum in an ad hoc grid are discussed in the next section.

6.1 Degree of Neighborhood

In order to explore the difference in resource allocation efficiency between fully centralized and fully decentralized approaches, we introduce the notion of the neighborhood [22] in this chapter. The neighborhood of a node defines the visibility region of a node by defining the number of nodes accessible from that node. We explain the degree of neighborhood of node for resource discovery on the following points on an infrastructural spectrum that ranges from fully centralized to the fully decentralized extremes:

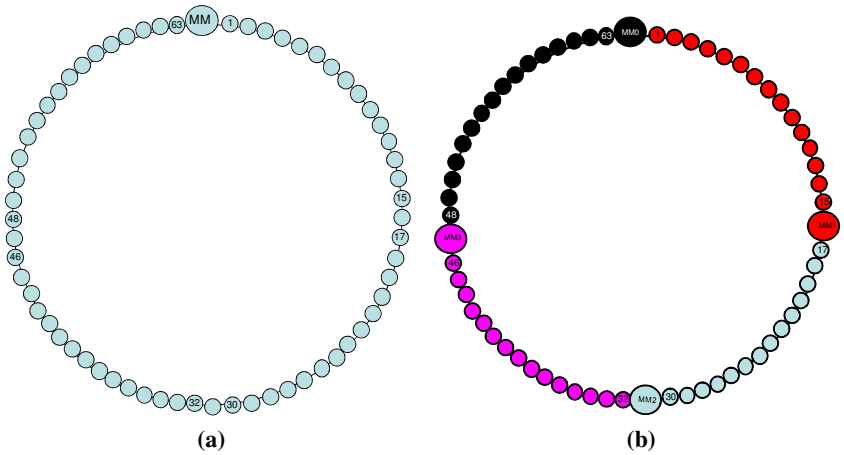


Figure 6.1: *Neighborhood on the infrastructural spectrum.* ***(a)*** *Fully centralized.* ***(b)*** *Multiple adaptive matchmakers.*

Fully Centralized Approach

In a fully centralized approach, there is only one matchmaker. All the consumer/producer nodes (Section 4.1) send their resource requests or resource offers to the matchmaker. The matchmaker finds matches for resource requests from the received resource offers and informs the matched consumer and producer nodes. As all the participating consumer/producer nodes send their request/offer messages to the matchmaker, therefore, the neighborhood of a consumer/ producer node is n (n being the total number of the nodes in the ad hoc grid). This is represented in Figure 6.1a, where there is only one matchmaker.

Fully Decentralized Approach

In a fully decentralized approach (P2P)**,** each node is its own matchmaker and looks for the appropriate resources from all the nodes in its neighborhood. The ad hoc grid is implemented on top of Pastry [123], a structured P2P overlay network. The degree of neighborhood of a node in our ad hoc grid is implemented and varied with the help of Pastry node's leaf set [123], and is explained below.

We consider a Pastry node with nodeID x for explaining the degree of neighborhood in the ad hoc grid. Each node in Pastry is assigned a 128 bits unique node identifier (referred to as *nodeID* hereafter). A Pastry node's leaf

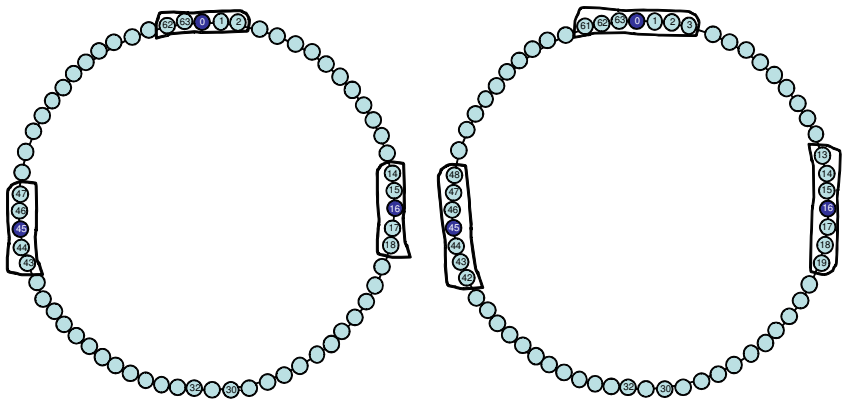


Figure 6.2: *Neighborhood on the infrastructural spectrum with varying the degree of neighborhood.* ***(a)*** *Fully decentralized degree=4.* ***(b)*** *Fully decentralized degree=6.*

set, L, contains closest nodeIDs to the nodeID x. The leaf set, L, comprises of $|L|/2$ numerically closest larger nodeIDs and $|L|/2$ numerically closest smaller nodeIDs, relative to any node's nodeID in a Pastry overlay network. Here, $|L|$ represents the cardinality of the leaf set L. The visibility of a node in the ad hoc grid increases with an increase in its degree of neighborhood. The neighborhood degree equal to 4 and 6 of three arbitrary nodes (with nodeIDs 0, 16 and 45) in a fully decentralized ad hoc grid is represented in Figures 6.2a and 6.2b, respectively.

Typically, a Pastry node can route a message to another Pastry node in less than $\log_{2^b} N$ steps, where N is the number of participating nodes and b is a configuration parameter with a typical value of 4. A Pastry node directly sends a message to its leaf set members. As the neighborhood is implemented as a leaf set, therefore, all the message exchange between our ad hoc grid nodes takes only *one* hop instead of $\log_{2^b} N$ hops. Pastry node leaf set and message routing details are in [123].

Multiple Adaptive Matchmakers Approach

In the multiple adaptive matchmakers approach, each consumer/producer node is under the responsibility of one matchmaker at any given point of time. The matchmakers are demoted or promoted according to the workload of a matchmaker(s) in the ad hoc grid. Then number of matchmaker(s) and the respon-

sible matchmaker of a consumer/producer node may also change by the promotion/demotion of matchmaker(s) [20]. As each consumer/producer node is under the responsibility of only one matchmaker at any given point in time, therefore, the neighborhood of the consumer/producer nodes is N/M (N being the total number of the participating nodes and M being the number of matchmakers). This is represented in Figure 6.1b, where different zones are under the responsibility of the matchmakers $MM1$, $MM2$, $MM3$ & $MM4$, whereas the consumer/producer nodes in each zone follow their responsible matchmaker.

6.2 Resource Discovery with Varying the Degree of Neighborhood

Each node generates a resource request/offer according to its resource requirements/availability. The generated resource request/offer is sent to all the nodes in its visibility region. The sender node waits for a matched response, after sending a resource request/offer to the neighboring nodes. The neighboring nodes receive and process the resource request/offer according to their resource status. A received resource request is discarded if the receiver node is also looking for resources. Otherwise, the receiver node (resource producer node) receives and processes the resource requests from the neighboring nodes. The producer's matchmaker agent attempts matching the node's resource offer with a resource request offering the highest price. The matchmaker agent also considers other resource request/offer constraints while finding a matched pair (refer Section 4.1). The receiver node notifies the matched sender node, after finding a match. Since, the sender node has sent its resource request to all the neighboring nodes, so, it may receive more than one matched responses. The sender node selects the first received response and discards the remaining ones. The matchmaker agent discards all the messages in its request/offer repositories, after finding a matched bid/ask pair. The sender node then contacts the producer node for its job execution. The producer node returns the results to the consumer node after executing the job.

6.3 Message Complexity Analysis

It is important to understand the cost is of a particular organization of the ad hoc grid. To this purpose, we analyze the number of messages exchanged for finding a matched pair in fully centralized, multiple adaptive matchmakers and

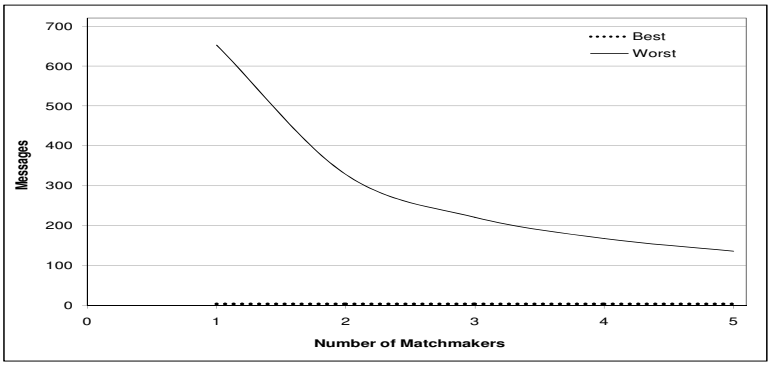


Figure 6.3: *Number of messages exchanged in the centralized and multiple adaptive matchmakers approach.*

fully decentralized resource discovery approaches.

First, we analyze the *fully centralized approach*. Let N be the total number of participating nodes. These nodes can play the role of a consumer or producer at any given time. There is only one matchmaker in the centralized resource discovery approach. In the best case, a consumer node sends a request to the matchmaker and a producer node sends a resource offer to the matchmaker. The matchmaker finds a match and a reply message is sent to both matching consumer and producer nodes. In the worst case, $N - 1$ nodes will send their offers to the matchmaker. Only then the matchmaker can find a suitable offer for the received resource request and a matched message is sent to both matching consumer and producer nodes. Hence, in the centralized resource discovery approach, only 4 messages are required to find a matched request/offer pair for the best case and $N + 2$ messages for the worst case.

In case of the *multiple adaptive matchmakers approach*, each matchmaker is responsible for a certain number of nodes out of all the participating nodes. An overloaded matchmaker forwards its excess workload to its neighboring matchmaker. The details of matchmaker(s) promotion/demotion and excess workload forwarding are discussed in Chapter 5. Let N be the total number of participating nodes, M be the number of matchmakers ($M < N$) and n_i be the number of nodes under the responsibility of matchmaker m_i, where $i = 1, 2, 3, ..., M$ in the ad hoc grid, such that:

$$N = \sum_{i=1}^{M} n_i$$

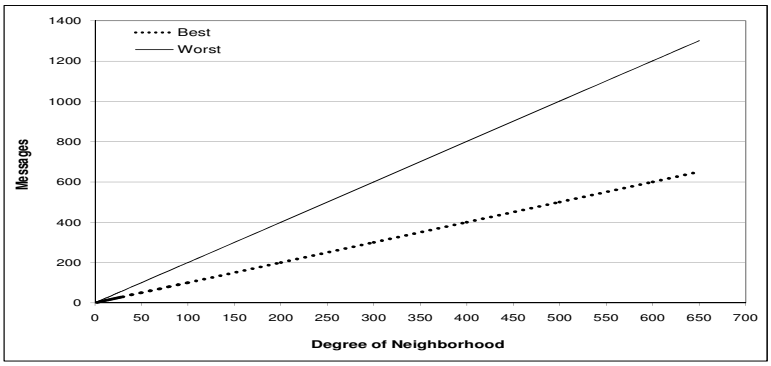


Figure 6.4: *Number of messages exchanged with varying the degree of neighborhood in a fully decentralized approach.*

The best case for a matchmaker in the multiple adaptive matchmakers approach is the same as that of the centralized approach. However, in the worst case, a request/offer message may be forwarded to at most $M-1$ matchmakers [20]. Therefore, the maximum number of messages to find a match will be $(M-1)+n_i+2$, where n_i is the number of nodes under the responsibility of $(M-1)^{th}$ matchmaker and 2 represents the matched message sent to both matched consumer and producer nodes. Figure 6.3 graphically depicts the number of messages required to find a match in the centralized and the multiple adaptive matchmakers approach in an ad hoc grid.

For a *fully decentralized approach (P2P)* with varying the degree of neighborhood**,** let N be the total number of nodes and D be the degree of neighborhood, such that $D = 2, 4, 6, 8, ..., N$ in the ad hoc grid. In the best case, all the neighboring nodes send offers to the current node, for its resource request, and one matched message is sent to the matching producer node. Hence, the number of messages will be $D+1$. The worst case scenario of this protocol, varying the degree of neighborhood, was explained in Section 6.2. In the worst case scenario, a node will send its resource request/offer to all neighboring nodes and all neighboring nodes will send a resource offer/request to the sender node. The sender node will send a confirmation message to the selected producer/consumer node. Total number of exchanged messages to find a matched pair will be $2D+1$.

The differentiating point in the analyzed resource discovery approaches is the matchmaker's ability to search for a required resource from the nodes under its responsibility or in its degree of neighborhood. The matchmaker agent can look at the submitted requests/offers of the nodes under its responsibility in

the fully centralized and multiple adaptive matchmakers approach. Whereas, the matchmaker agent, is limited by the degree of neighborhood (except when $D = N$) and cannot search the resources of all the participating nodes. Figure 6.4 graphically represents the number of messages required to find a match for varying the degree of neighborhood approach in an ad hoc grid.

The distribution of ad hoc grid nodes among two or more matchmakers may vary according to the workload of the matchmakers [20]. While comparing the message complexity for varying the degree of neighborhood in the fully decentralized approach with the other two approaches in Figures 6.3 and 6.4, we assume, $n_i = {}^{n}/_{m}$ for the case of multiple adaptive matchmakers approach. One matchmaker of the centralized approach is represented by $1MM$, whereas, $2MM, ..., 5MM$ represent two or multiple matchmakers of the multiple adaptive matchmakers approach in Figure 6.3. The number of messages exchanged in different resource discovery approaches is summarized in Table 6.1.

6.4 Experimental Setup and Results Discussion

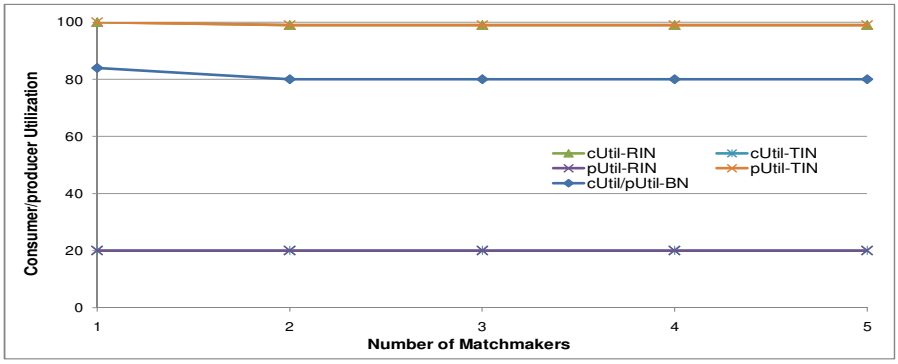
The number of participating nodes is varied from 15-650, and the number of matchmakers is varied from 1-5. The degree of neighborhood is varied from $2 - 250$. Matchmaking efficiency, response time and the number of messages exchanged are the analysis parameters. Message complexity analysis is explained in Section 6.3. The matchmaking efficiency of a matchmaker agent in the time interval $T = [T_{start}, T_{end}]$ is defined as: $(\sum_{T_{start}}^{T_{end}} Matched\ Message / \sum_{T_{start}}^{T_{end}} Total\ Message) * 100$, Where T_{start} and T_{end} represent the start and end time of the time interval $T = [T_{start}, T_{end}]$. The response time denotes the time interval, starting from the time a message is received, and ends at the moment when a match is found for the received message. The response time is calculated as: $RT = T_{match} - T_{receive}$, where RT repre-

Spectrum Point	Best Case	Worst Case
Fully centralized (One matchmaker)	4	$N + 2$
Multiple adaptive matchmakers	4	$M + n_i + 1$
Varying the degree of neighborhood in P2P	$D + 1$	$2D + 1$

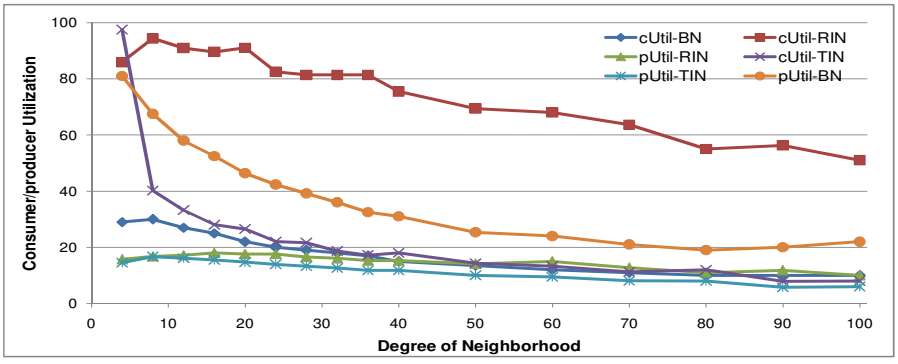
Table 6.1: *Message exchange for finding a match at different points of the infrastructural spectrum.*

sents the response time, T_{match} is the time when the matchmaker agent found a matching offer/request for the received request/offer message and $T_{receive}$ is the receiving time of the received request/offer message. All the experiments are executed in different network conditions, including *BN*, *RIN* and *TIN* conditions.

First, we look at the matchmaking efficiency in the fully centralized resource discovery approach (number of matchmaker $= 1$ in Figure 6.5a) and with multiple adaptive matchmakers resource discovery approach (number of matchmakers > 1 in Figure 6.5a). The *fully centralized approach* shows higher matchmaking efficiency for small workloads. However, one matchmaker can-



(a)



(b)

Figure 6.5: *Matchmaking efficiency.* ***(a)*** *Matchmaking efficiency of centralized and multiple adaptive matchmakers approach in different network conditions.* ***(b)*** *Matchmaking efficiency with varying the degree of neighborhood in fully decentralized approach in different network conditions.*

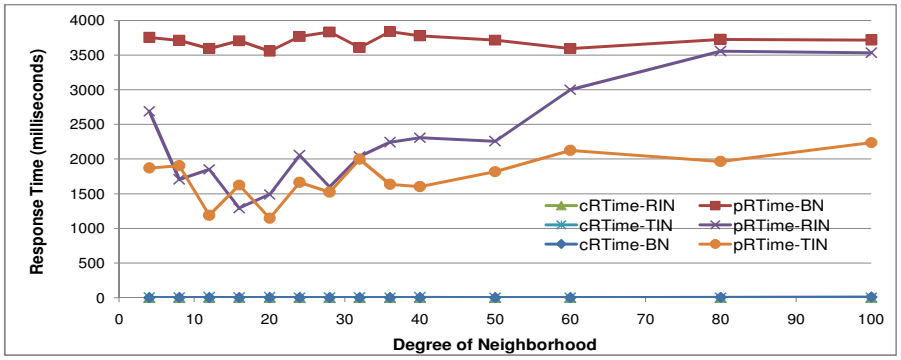
not maintain its matchmaking efficiency with the increasing work load. The matchmaking efficiency keeps on decreasing with the increasing workload of the matchmaker (Section 4.5). This phenomenon can be understood with the following explanation. With the increasing workload, the matchmaker has to process more messages, so it takes more time to find the matched pairs. The increased processing time results in an increased response time of the matchmaker. Since each request/offer message has a validity period (TTL), therefore, the TTL of the request/offer messages start expiring with the increased processing time of the matchmaker and, consequently, the matchmaking efficiency of the matchmaker decreases with the increasing workload. The workload threshold for one matchmaker and decreasing matchmaking efficiency with increasing workload of a matchmaker is explained in Section 4.5.

The matchmaking efficiency of *multiple adaptive matchmakers approach* is not affected by the increasing workload. The adaptive mechanism introduces more matchmaker(s) when needed by an overloaded matchmaker(s). Hence, the matchmaking efficiency remains the same with the increased number of matchmakers. The matchmaking efficiency of the fully centralized resource discovery approach is slightly higher than that of the multiple adaptive matchmakers approach (Figure 6.5a). The matchmakers in the multiple adaptive matchmakers approach communicate with the other matchmakers in order to promote/demote matchmakers and for sharing their access workload [20]. Some of the request/offer messages expire during this process. Since, there is no communication or workload sharing with the other matchmakers in the fully centralized approach, the maximum matchmaking efficiency of the fully centralized system is slightly higher than that of the multiple adaptive matchmakers system. However, the fully centralized approach is not scalable and can have a single point of failure [20, 23].

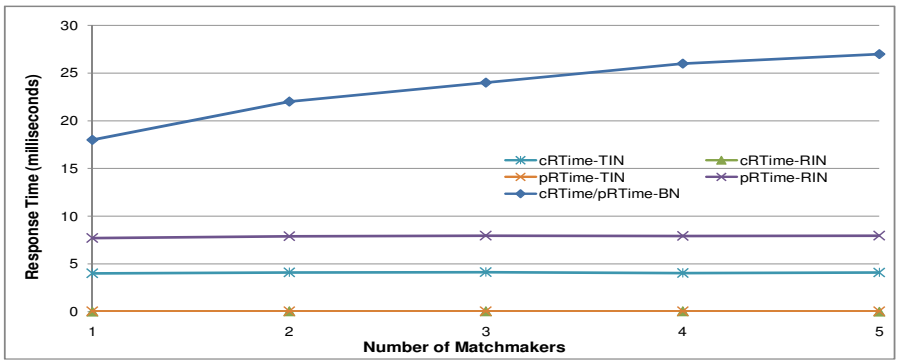
The matchmaking efficiency of the resource discovery approach with varying the degree of neighborhood in a *fully decentralized ad hoc grid (P2P)* is shown in Figure 6.5b. The matchmaking efficiency initially increases with an increased degree of neighborhood. This seems logical as with an increased degree of neighborhood, the chances for finding a required resource/offer also increase. However, this trend starts decreasing with further increase in the degree of neighborhood due to the increased number of request/offer messages (Figure 6.4). The matchmaker agent of each node has to process more messages. The increased processing time results in TTL expiry of the request/offer messages; consequently, a drop in the matchmaking efficiency. The experiments are repeated for $n = 100, 250$ and d is varied from $d = 2, 4, 6, 8, \ldots, n$. The same matchmaking pattern as in Figure 6.5b is

observed.

An alternative view of the above discussed phenomenon is to consider the average response time of the matchmaker. Figure 6.6a shows the average response time to find a match with varying the degree of neighborhood in a *fully decentralized ad hoc grid*. In our experiments, the response time stays stable up to 50 hops in the P2P case. The response time increases more than proportional once the number of hops goes beyond 60 (Figure 6.6a). The over proportional increase in the response time is due to the communication overhead incurred with the increased degree of neighborhood in a fully decentralized ad hoc gird.



(a)



(b)

Figure 6.6: *Response time for finding a match.* ***(a)*** *Response time with varying the degree of neighborhood in the fully decentralized approach in different network conditions.* ***(b)*** *Response time of centralized and multiple adaptive matchmakers approach in different network conditions.*

We observe an increase in response time of the *multiple adaptive matchmakers approach* with the increased number of matchmakers (Figure 6.6b). This increase is due to the ad hoc grid segmentation and due to the increased communication as explained in Section 6.3. The experiments were also executed under RIN and TIN conditions. We observed the same trend of the matchmaking efficiency and response time as discussed above.

It can be concluded from the above discussion that neither a fully centralized nor a fully decentralized is generally a suitable infrastructures for resource discovery in an ad hoc grid. A fully centralized infrastructure is not scalable and can have a single point of failure. On the other hand, a fully decentralized infrastructure incurs excessive communication overhead that results in an increased response time and the decreased matchmaking efficiency. An intermediate infrastructure having multiple adaptive matchmakers seems most efficient in terms of the response time and in finding matches. The intermediate infrastructure with multiple adaptive matchmakers should be preferred whenever possible in the ad hoc grid.

6.5 Concluding Remarks

In this chapter, we analyzed the effect of varying the degree of neighborhood on resource discovery in an ad hoc grid. For this purpose, we defined and implemented the degree of neighborhood for the participating nodes. The results were obtained for fully centralized, multiple adaptive matchmakers and fully decentralized resource discovery approaches in an ad hoc grid. The results show that the ad hoc grid becomes less efficient with the increased degree of neighborhood in a fully decentralized approach, due to the excessive messages being exchanged. The results also confirmed that an intermediate ad hoc grid infrastructure with multiple adaptive matchmakers is preferable in a local ad hoc grid.

Chapter 7

Nature Inspired Self-organization

THIS chapter focuses on the third part of the dissertation by studying the proposed infrastructure-level self-organization mechanism from the nature inspired ant colony optimization perspective. ACO and the other similar nature inspired mechanisms like artificial neural networks, swarm intelligence and evolutionary algorithms are based on the naturally existing complex adaptive systems. The CASs are dynamic, highly decentralized networks and consist of many participating agents[1]. Human immune system, sand dune ripples and ant foraging are some examples of the natural CASs. Decentralized control, emergent behavior, robustness and self-organization are the defining characteristics of the CASs. Participating agents in these systems interact according to simple local rules which result in complex behavior and self-organization at the system level.

Ad hoc grids and other similar computational distributed systems are inherently dynamic and complex systems. Resource availability fluctuates over time in ad hoc grids. These changes require adaptation of the system to the new system state by applying some self-organization mechanism. Current sci-

[1]This chapter is based on the following research articles:
T. Abdullah, K.L.M. Bertels, L.O. Alima, "Ant Colony Inspired Microeconomic based Resource Management in Ad Hoc Grids", 4th International Conference on Grid and Pervasive Computing, Geneva, May 2009.

T. Abdullah, V. Sokolov, B Pourebrahimi, K.L.M. Bertels, "Self-Organizing Dynamic Ad Hoc Grids", In Proceedings of 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW 2008), Venice, October 2008.

entific problems like protein folding, weather prediction and particle physics experiments require huge computing power and storage space and are complex. These scientific problems can be solved by using ad hoc grids.

Ant colony system [55] is inspired by the colony of artificial ants cooperating in foraging behavior. ACO [54] is a heuristic algorithm for ACS and imitates the behavior of the real ant colonies in nature. In ACO algorithms ants drop a chemical, called pheromone, on their way from nest to the food source and vice verse, while they search for a food source. The pheromone concentration of a path increases when more ants visit a path. The pheromone concentration of a path evaporates and that path may disappear, if the path is not visited by ants over time. Ants choose a path with higher pheromone concentration from the food source to their nest. The ant colony self-organizes by the local interactions of the individual ants.

We apply a modified ACO algorithm for micro-economic based resource management and self-organization in an ad hoc grid [21]. In this algorithm, a consumer/producer node generates request/offer ants. A matchmaker represents the food source. The proposed algorithm sends ants in search of resources/tasks to the matchmaker. The matchmaker sends a matched response to the requesting consumer/producer nodes. The proposed system is a hybrid model that modifies ACO algorithms by applying the concepts from micro-economics domain. The proposed mechanism from a user's perspective, can be viewed as a combination of centralized, such as Condor for submitting and running arbitrary jobs, and a system such as BOINC or SETI@home for distributing jobs from a server to a potentially very large collection of machines in a fully decentralized environment.

7.1 Micro-economic based Modified ACO Algorithm

In order to map ACS to an ad hoc grid, first we explain their relationship. Ad hoc grid node acts like a consumer/producer node in our modified ACO algorithm, and the matchmaker(s) are treated like the *food source(s)*. Each consumer/producer node is capable of generating and sending *request/offer ants* to the food source. The *pheromone value* indicates the weight of the matchmaker in the ant system. A matchmaker with higher pheromone value indicates that it has a higher probability of finding a compatible resource offer for a submitted resource request and vice verse.

Each joining node in our ad hoc grid is under the responsibility of a matchmaker and sends its resource request/offer ants to its responsible match-

maker. The joining node gets the pheromone value of the food source from its responsible matchmaker. The pheromone value of a matchmaker is updated for each resource request or resource offer received from a consumer/producer node. The pheromone value of a matchmaker is computed according to the equations given in Section 7.3. A matchmaker exchanges periodically its pheromone value with its immediate neighboring matchmakers (the successor and the predecessor matchmakers). The updated pheromone value is then sent to the consumer/producer nodes with a matched message. The consumer/producer node uses the pheromone value as an indicator of matchmaker's matchmaking performance. The consumer/producer node sends its next request/offer ant to a food source with the highest pheromone value.

7.2 ACO based Self-organizing Ad Hoc Grid Segments

This section explains the proposed ACO based self-organized segmentation and de-segmentation approach in an ad hoc grid. This self-organized, dynamic segmentation and de-segmentation process is based on the dynamic changing resource requirements and resource availability in an ad hoc grid. In the proposed mechanism, there can be as many segments of the ad hoc grid as many needed. These segments are created when needed and are removed, when not needed.

The proposed approach also considers the ad hoc grid segmentation for specialized resource requirements of the participating ad hoc grid nodes. These resources may include a pool of specialized hardware for a video rendering application or a pool of software resources for a data processing application or a pool of resources for remote collaboration on a scientific experiment.

A consumer/producer node knows about its resource category at the time of joining the ad hoc grid. The consumer/producer node joins the ad hoc grid by contacting one of the existing nodes in the ad hoc grid, referred to as a bootstrap node. A consumer/producer node discovers a responsible matchmaker for its resource category. The algorithms for node joining and finding a responsible matchmaker are detailed in [20]. All of the newly joining consumer/producer nodes discover their responsible matchmakers and the nodes of a similar resource category are under the responsibility of one matchmaker. An ad hoc grid node can change its resource category or its resource availability/demand status (being a consumer or producer of resources) at any time during its life span. A consumer/producer node generates and sends a request/offer ant after finding its responsible matchmaker.

Algorithm 7.1 Discovering the right food source.

1:IF (($Ant_{resCategory}$ equals $FS_{resCategory}$) AND ($FS_{phValue}$ is highest)) THEN
2: CALL Find matching request/offer algorithm
3: IF (no match found) THEN
4: IF ($Ant_{resCategory}$ equals $pFS/sFS_{resCategory}$) THEN
5: Ant visit pFS/sFS
6: GO TO Step 1
7: END IF
8: END IF
9: ELSE IF ($Ant_{resCategory}$ not equals $FS_{resCategory}$) THEN
10: Ant visit $successorFS$
11: GO TO Step 1
12: END IF
13:END IF

A matchmaker receives a request/offer ant, and first checks whether it is the right food source for processing the received request/offer ant or not. This process is listed in Algorithm 7.1 and is performed as follows: The matchmaker checks whether the resource category of the received request/offer ant ($Ant_{resCategory}$) is similar to its resource category ($FS_{resCategory}$). Secondly, the matchmaker also checks whether the predecessor's resource category ($pFS_{resCategory}$) and/or the successor's resource category ($sFS_{resCategory}$) is similar to its resource category. In case of a matched resource category, the matchmaker also checks that it has the highest pheromone value ($FS_{phValue}$). After determining from its local knowledge that it is the right matchmaker to process the received request/offer ant, the matchmaker attempts finding a matching offer/request for the received request/offer ant by following the steps listed in Algorithm 7.2. The matching consumer/producer nodes are directly notified about the match.

Algorithm 7.2 Find matching request/offer.

1:Store request/offer in request/offer repositories
2:Match request parameters with offer parameters
3:IF ($Offer_{parameters}$ match $Request_{parameters}$) THEN
4: Send matched message to consumer
5: Send matched message to producer
6: Update pheromone value
7:END IF

In case the matchmaker does not have the matching offer/request then the request/offer ant is forwarded to the predecessor/successor matchmaker with the second highest pheromone value. The predecessor/successor matchmaker node processes the received request/offer ant in the same way as explained above. When the resource category of the request/offer ant ($Ant_{resCategory}$) is different from the resource category of the matchmaker ($FS_{resCategory}$) and its immediate neighboring matchmaker nodes (successor and predecessor matchmakers), then the request/offer ant is forwarded to the successor matchmaker node. The successor matchmaker node again performs the steps in Algorithm 7.2.

After finding a successful match, the pheromone strength of that matchmaker increases. The pheromone value of a matchmaker is based on its matchmaking efficiency. The pheromone value of a matchmaker decreases, when it is unable to find a match. The formulas for calculating the increased/decreased pheromone value are explained in Section 7.3. The matchmaker periodically notifies its immediate neighboring matchmaker nodes (successor and predecessor matchmaker nodes) about the change in its pheromone value. The consumer/producer nodes communicate directly with each other for task execution after receiving a matched request/offer message from the matchmaker. The producer node returns the results back to the consumer node after executing the task.

The process of changing a node's segment is triggered when a node changes its resource category. This process is listed in Algorithm 7.3 and is performed as follows: Whenever a consumer/producer node changes its resource category, it receives a matched message reply from a new matchmaker ($matchedMessageSender_{FS-ID}$). A consumer/producer node changes its matchmaker node ($CPNode_{FS-ID}$) and sends its next request/offer ant to the new matchmaker. In this way, a consumer/producer node continues send-

Algorithm 7.3 Change segment.

1:IF ($CPNode_{FS-ID} != matchedMessageSender_{FS-ID}$) THEN
2: $CPNode_{FS-ID} = matchedMessageSender_{FS-ID}$
3: Join new virtual segment
4: Send request/offer ant to $CPNode_{FS-ID}$
5:ELSE
6: No change in virtual segment
7: Send request/offer to old $CPNode_{FS-ID}$
8:END IF

ing its subsequent request/offer ants to the matchmaker that sent a matched message reply. Whenever, an ad hoc grid node changes its resource category, it leaves its current virtual segment and becomes a member of another virtual segment. Therefore, the ad hoc grid keeps on changing its infrastructure and is divided into specialized resource segments. The consumer/producer nodes are not bound to a specific ad hoc grid segment. The nodes can dynamically leave one virtual segment and join another according to their changed resource category.

7.3 Pheromone Calculation

In this section, we describe the formulas for calculating the pheromone value in our modified ACO algorithm. As mentioned earlier in Section 7.1, the consumer/producer nodes generate and send the *request/offer ants*; the matchmakers represent the *food sources*; and the pheromone value indicates the weight of a matchmaker. The pheromone value of a matchmaker is calculated periodically according to the following formula.

$$\tau_{new} = \begin{cases} \alpha * \tau_{old} + (1-\alpha) * \triangle\tau & if\ \triangle\tau > 0 \\ (1-\alpha) * \tau_{old} + \alpha * \triangle\tau & if\ \triangle\tau < 0 \end{cases} \tag{7.1}$$

The parameter α represents the pheromone evaporation rate. The value of α varies between 0 and 1. τ_{old} represents the pheromone value during time interval $T1 = [t_{s_1},\ t_{e_1}]$. Whereas, $\triangle\tau$ is the change in the pheromone value between the time interval $T1 = [t_{s_1},\ t_{e_1}]$ & $T2 = [t_{s_2},\ t_{e_2}]$. The start time of both intervals is represented by t_{s_1} & t_{s_2} and t_{e_1} & t_{e_2} represent the end time of both the intervals, such that $T2 > T1$ & $t_{s_2} = t_{e_1}$. Value of $\triangle\tau$ is calculated as:

$$\triangle\tau = \sum_{i=1}^{n} \tau(i)/N \tag{7.2}$$

where N is the total number of messages received by the matchmaker and $\tau(i)$ is the pheromone value contributed by an individual ant. $\tau(i)$ for a consumer agent is calculated as:

$$\tau(i) = Perform(MM) * UPrice_{consumer} \tag{7.3}$$

$\tau(i)$ for a producer agent is calculated as:

$$\tau(i) = Perform(MM) * UPrice_{producer} \tag{7.4}$$

where $Perform(MM)$ represents the performance of a matchmaker and $UPrice$ represents the unit price of a requested or offered computational resource by an ant. $Perform(MM)$ is periodically calculated as:

$$Perform(MM) = {}^{Matched}/_{N} \tag{7.5}$$

where $Matched$ represents the number of matched pairs and N is the total number of request/offer ants processed by the matchmaker in the time interval $T = [t_{start}, t_{end}]$. Where t_{start} & t_{end} represent the start and end time of the time interval T respectively. The $UPrice$ for each requested or offered computational resource by an ant is calculated as:

$$UPrice_{request} = (\frac{RQuantity_{request}}{BPrice}) * SMachine \tag{7.6}$$

$$UPrice_{offer} = (\frac{RQuantity_{offer}}{APrice}) * SMachine \tag{7.7}$$

where $RQuantity_{request}$ and $RQuantity_{offer}$ represent the computational resource quantity of a request and offer respectively. The bid and ask price of a requested and offered computational resource are represented by $BPrice$ and $APrice$, respectively. The reference machine is represented by $SMachine$.

7.3.1 Assumptions

Following simplifying assumptions are in place for the experimental results reported in this chapter:

1. There exists at least one food source for a resource category.
2. Each consumer/producer node knows about or is under the responsibility of one food source at any given time.
3. The successor/predecessor food sources of a food source node (referred to as a matchmaker node) update the current matchmaker node after updating their pheromone value and vice verse.

7.4 Experimental Results and Discussion

Both the experimental setup used for evaluating the proposed mechanism and the experimental results are discussed in this section.

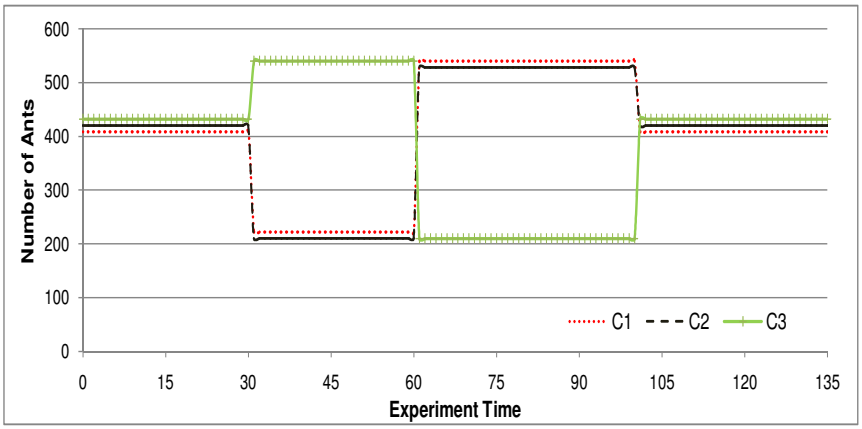


Figure 7.1: *Workload distribution of different resource categories.*

7.4.1 Experimental Setup

The modified ACO algorithm is implemented by extending a structured overlay network, Pastry [123]. Pastry forms an overlay network among the ad hoc grid nodes and performs the basic tasks required for maintaining an overlay network. The experimental results reported here are obtained by executing the experiments on PlanetLab [1].

These experiments are executed in a different network condition including balanced network condition, resource intensive network condition and task intensive network condition. The consumer-producer ratio is approximately $50-50$ in BN condition. Whereas, the consumer-producer ratio is $20-80$ and $80-20$ in RIN and TIN conditions, respectively. The number of participating nodes is varied from $15-650$. The number of different resource categories is 3 in the first set of experiments and number of matchmakers is 5 in the second set of experiments. Different parameters of resource request/offer like task execution time and resource quantity are randomly generated from a pre-specified range (refer to Table 3.1). The validity period (TTL) of request/offer message is set to 10000 milliseconds for accommodating delays observed in PlanetLab.

Each ant's pheromone is initialized by 1. In nature, each ant wanders randomly without any initial pheromone value as used in these experiments. However, if the initial pheromone value is set to 0, then the new pheromone value will always be zero. The value of α (rate of pheromone evaporation) is set to 0.8 in these experiments. Figure 7.1 shows the workload distribution. In this figure, the experiment time is represented on the horizontal axis, and the

number of ants of each resource category is shown on the vertical axis.

The analysis parameters are pheromone evaluation, consumer/producer utilization, response time, and the average ask/bid price of the participating producer/consumer nodes. The formulas for calculating the pheromone value are described in Section 7.3. The consumer utilization and producer utilization are calculated according to Equations 3.1 & 3.2, respectively. A general formula for the consumer/producer utilization is as follows:

$$(\frac{MatchedMessage}{N}) * 100$$

where $MatchedMessage$ represents the count of matched messages and N denotes the total number of request/offer ants processed by the matchmaker(s) in a unit time interval. The *response time* represents the time interval between receiving a request/offer message and finding a matching offer/request by the matchmaker. The response time is calculated as:

$$RT = T_{match} - T_{receive}$$

where RT represents the response time, T_{match} is the time when the matchmaker found a matching offer/request for the received request/offer and $T_{receive}$ is the receiving time of the received request/offer.

7.4.2 Experimental Results Discussion

The experimental results discussed below are divided into two sets. The first set of experimental results focuses on the self-organizing capability of the proposed mechanism for the ad hoc grid when nodes dynamically change their resource categories. The proposed ACO-based self-organizing mechanism is compared with different matchmaking schemes. These schemes include the simplest and less compute intensive first come, first served and a micro-economic based, compute intensive continuous double auction scheme (Section 7.4.2).

Whereas, the effect of an overload condition on a matchmaker in the proposed mechanism and a solution for avoiding the overload condition is discussed in the second set of experimental results (Section 7.4.3).

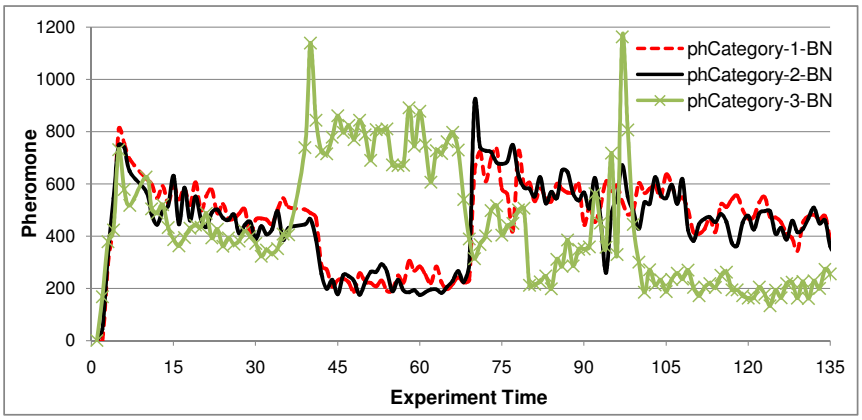
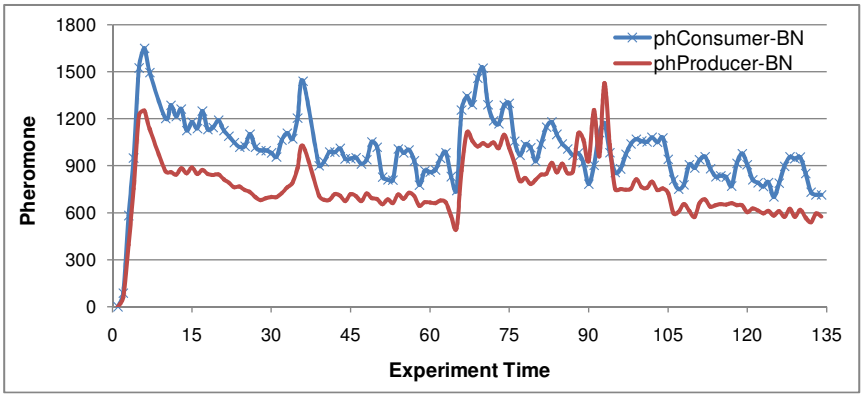


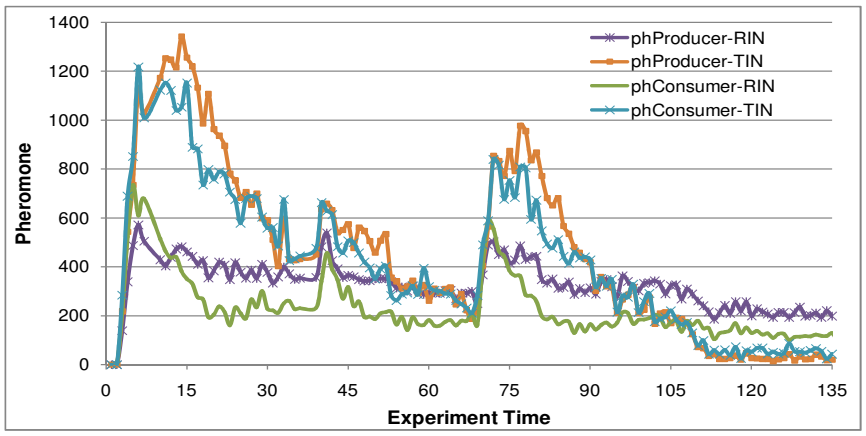
Figure 7.2: *Individual category pheromone evolution in an ad hoc grid.*

The overall pheromone evolution for each resource category (represented as $phCategory-1-BN$, $phCategory-2-BN$ and $phCategory-3-BN$) during the simulation in a BN condition is depicted in Figure 7.2. The experiment time is represented on the horizontal axis and the pheromone value is on the vertical axis. The pheromone of each category evolves according to the workload distribution of the respective resource category (Figure 7.1). Whenever the distribution of ants in different resource categories is changed, a change in the respective pheromone concentration is observed. The ad hoc grid self-organizes itself after each change. A stable pheromone value is observed before the next disturbance in the ant's distribution during the simulation.

The pheromone behavior of the individual consumer/producer nodes in BN condition in represented in Figure 7.3a, and TIN and RIN conditions are represented in Figure 7.3b. The horizontal axis of both these figures represent the experiment time and the vertical axis represents the pheromone value of the participating consumer/producer nodes in BN, RIN or TIN conditions of the ad hoc grid. The initial increase of the pheromone value for the consumer/producer nodes in BN, RIN and TIN conditions is followed by a decreasing trend that leads to a stable status of the ad hoc grid. Similar to the individual category pheromone value evolution, the overall consumer/producer pheromone also evolves similarly in different network conditions.



(a)



(b)

Figure 7.3: *Pheromone evolution for all resource categories of the ad hoc grid.* ***(a)*** *Consumer/producer pheromone evolution in BN condition of the ad hoc grid.* ***(b)*** *Consumer/producer pheromone evolution in RIN & TIN conditions of the ad hoc grid.*

The pheromone pattern is disturbed after a change of the ant's distribution in different resource categories. The proposed algorithm enables the ad hoc grid to re-structure itself into different virtual resource and in attaining a stable state.

As mentioned earlier, different matchmaking mechanisms are applied for evaluating the proposed nature inspired ACO-based self-organizing mechanism. The micro-economic based CDA mechanism is compared with the simplest and less compute intensive FCFS. The average consumer/producer

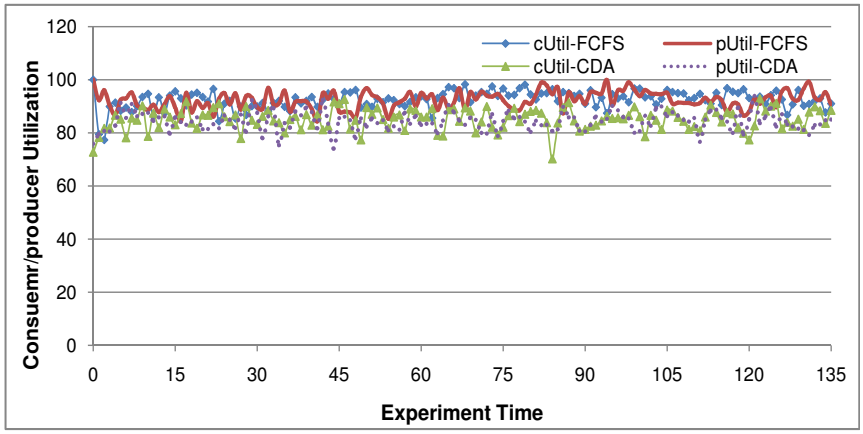
	Consumer Utilization			Producer Utilization		
	BN	RIN	TIN	BN	RIN	TIN
CDA	85%	92%	23%	85%	22%	87%
FCFS	92%	100%	25%	92%	25%	98%

Table 7.1: *Comparing consumer/producer utilization of different schemes in different network conditions.*

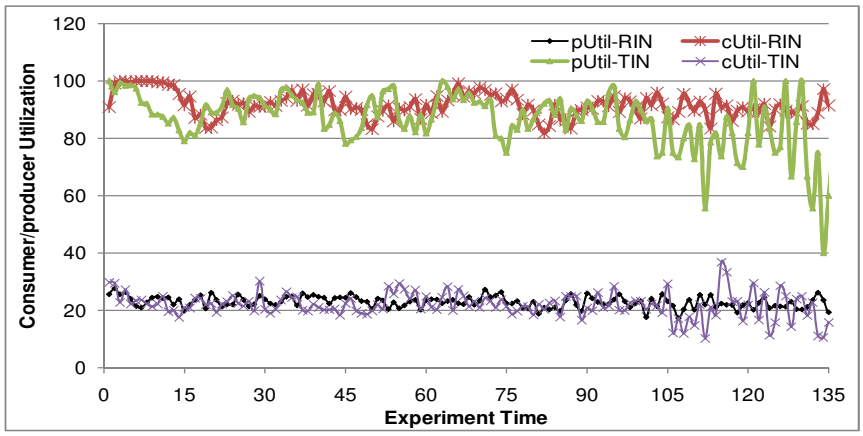
utilization of the participating consumer/producer nodes in an ad hoc grid with different matchmaking mechanisms is depicted in Figure 7.4a.

In-spite of changing workload of ants in different resource categories, the consumer utilization ($cUtil - CDA$, $cUtil - FCFS$) and the producer utilization ($pUtil - CDA$, $pUtil - FCFS$) in CDA and FCFS schemes under a BN condition remains above 80%. The fluctuations in the consumer/producer utilization refer to the activity in the ad hoc grid. This behavior implies that the compute intensive nature of CDA does not affect the matchmaking capacity of the ad hoc grid. Table 7.1 summarizes the average consumer/producer utilization in CDA and FCFS under different network conditions (BN, RIN & TIN). It can be concluded from Table 7.1 that, in spite of being compute intensive, the consumer/producer utilization in CDA is as good as in FCFS.

The consumer/producer utilization in RIN and TIN conditions is depicted Figure 7.4b. The task-resource ratio is $20\% - 80\%$ and $80\% - 20\%$ in RIN and TIN conditions respectively. In general, both the consumer utilization in TIN condition ($cUtil - TIN$) and the producer utilization in RIN condition ($pUtil - RIN$) are proportional to the ratio of scarce commodity in the respective network conditions. Whereas, the producer utilization in TIN condition ($pUtil - TIN$) and the consumer utilization in RIN condition ($cUtil - RIN$) vary between 80% and 100%. The higher consumer utilization in RIN and producer utilization in TIN conditions are due to the abundance of resource requests and resource offer of each category in the respective network condition during the simulation. Therefore, a request in RIN and an offer in TIN condition get matched as soon as it is received by its respective food source (matchmaker). The higher fluctuations in $cUtil - RIN$ and $pUtil - TIN$ refer to the points during the simulation when ants change their resource categories and the ad hoc grid becomes unstable. The consumer utilization in RIN ($cUtil - RIN$) and the producer utilization in TIN ($pUtil - TIN$) show a stable behavior and the variations in stable behavior are due to the change of consumer/producer resource category. In general, ad hoc grid shows a stable behavior in terms of consumer/producer utilization and accommodates the dynamic resource cate-



(a)

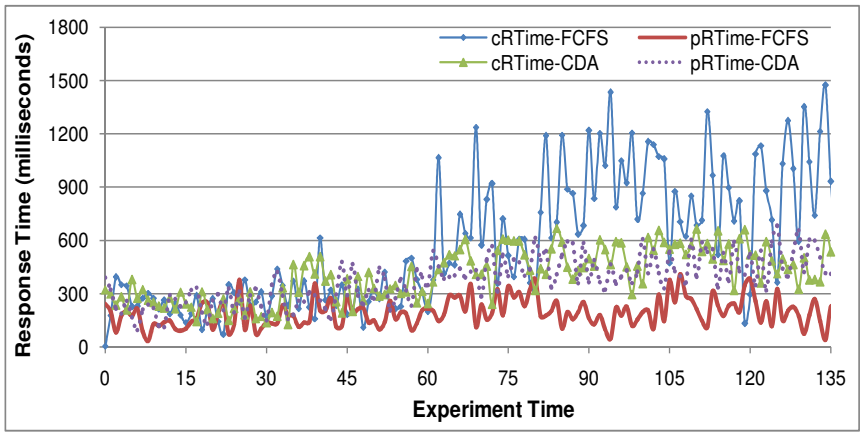


(b)

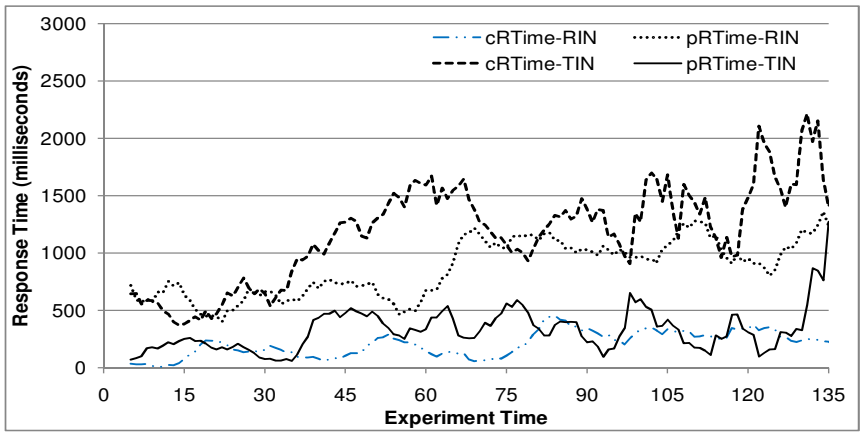
Figure 7.4: *Consumer/producer utilization.* ***(a)*** *Consumer/producer resource utilization in BN condition of the ad hoc grid.* ***(b)*** *Consumer/producer resource utilization in RIN & TIN conditions of the ad hoc grid.*

gory changes of the participating ad hoc grid nodes according to their changing resource requirements.

The proposed self-organizing mechanism can also be analyzed from the matchmaker response time for the consumer/producer nodes. The consumer response time and the producer response time for continuous double auction scheme and first come, first served scheme in a balanced network condition is represented in Figure 7.5a. Whereas, the consumer/producer response



(a)



(b)

Figure 7.5: *Response Time.* ***(a)*** *Response time in BN condition of the ad hoc grid.* ***(b)*** *Response time in RIN & TIN conditions of the ad hoc grid.*

time variation in RIN and TIN conditions is represented in Figure 7.5b. The response time shows a stable behavior and is not affected by the resource category change of the participating consumer/producer nodes. The consumer response time ($cRTime - CDA$, $cRTime - FCFS$) and the producer response time ($pRTime - CDA$, $pRTime - FCFS$) for both the schemes in BN condition are initially low due to the equal number of ants of different categories. When the consumer/producer nodes change their resource category, the ad hoc grid becomes unstable. The consumer/producer nodes have to wait longer for get-

ting a matched response. This longer wait time results in an increasing trend of the consumer/producer response time. The response time shows a stable behavior once the ad hoc grid attains back the stable behavior.

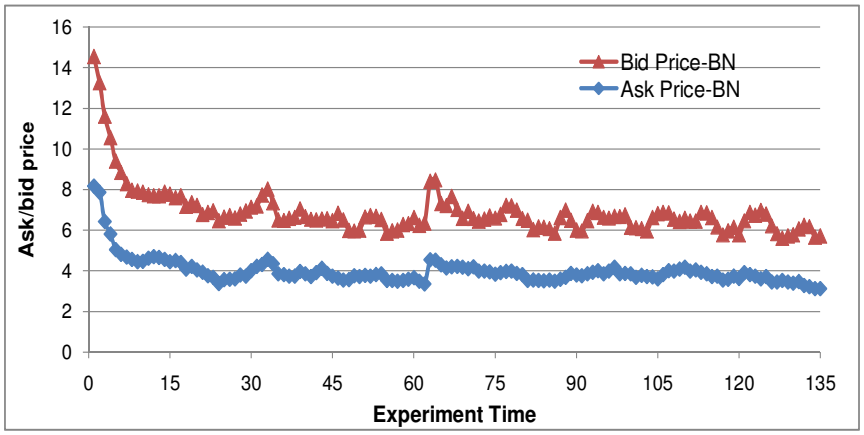
The consumer response time for FCFS ($cRTime - FCFS$) is higher than that of the consumer response time for CDA ($cRTime - CDA$) in a BN condition. The lower consumer response time in CDA can be understood by understanding the requests/offers handling process in the matchmaker request/offer repositories. The matchmaker stores requests in descending order and offers in ascending order of the price in its request/offer repositories. The consumer response time is less in CDA scheme as compared to the consumer response time in FCFS scheme, due to the sorted placement of requests/offers in the matchmaker repositories.

The consumer response time in RIN ($cRTime - RIN$) and the producer response time TIN network condition ($pRTime - TIN$) is low. Since, the consumers in RIN and producers in TIN are scarce; therefore, a request in RIN condition and an offer in TIN condition gets matched as soon as it is received by the matchmaker. Thus, resulting in low values of $cRTime - RIN$ and $pRTime - TIN$. The producers in RIN condition and the consumer in TIN condition are in abundance and may have to wait longer before getting matched. This longer wait time results in higher response time values for the consumers in TIN condition ($cRTime - TIN$) and producers in RIN condition ($pRTime - RIN$). The variations in response time refer to intervals when the consumer/producer nodes change their resource category and ad hoc grid tries to gain a stable condition once again.

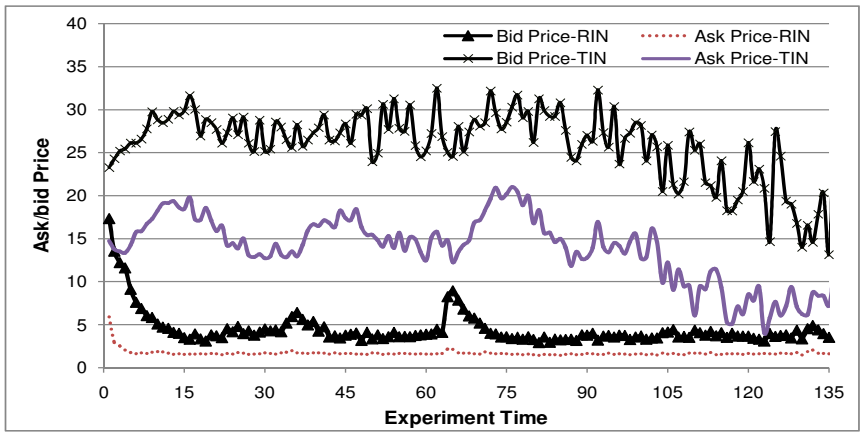
The average consumer/producer response time for CDA scheme and FCFS scheme in different network conditions is summarized in Table 7.2. The comparison leads to the conclusion that, in spite of being compute intensive, the consumer response time for CDA is less than that of the FCFS scheme.

	Consumer Response Time			Producer Response Time		
	BN	RIN	TIN	BN	RIN	TIN
CDA	406	203	1169	380	871	316
FCFS	584	21	3980	194	223	70

Table 7.2: *Comparing matchmaker response time for finding a match in CDA and FCFS under different network conditions.*



(a)



(b)

Figure 7.6: *Ask/bid price evolution in different network conditions.* ***(a)*** *Ask/bid price evolution in BN condition of the ad hoc grid.* ***(b)*** *Ask/bid price evolution in RIN & TIN conditions of the ad hoc grid.*

As the consumer/producer nodes applied a history based dynamic pricing mechanism (Section 4.1.3) for determining the value of the resources requested from the ad hoc grid or offered to the ad hoc grid in CDA based scheme, therefore, the affect of ask/bid price on the proposed mechanism is also analyzed. The consumer bid price ($BidPrice - BN$) and the producer ask price ($AskPrice - BN$) for the matched pairs in a BN condition is depicted in Figure 7.6a. A match is found when the bid price of a request is higher than

the ask price of an offer. As obvious from Figure 7.6a, the bid price is always higher than the ask price for a matched request/offer pair.

Furthermore, the fluctuations in ask/bid price refer to the intervals when the nodes change their resource category and the consumer/producer nodes increase their ask/bid prices for their requested/offered resources. The bid price of a consumer in RIN and TIN conditions ($BidPrice - RIN$, $BidPrice - TIN$) and the ask price of a producer in RIN and TIN conditions ($AskPrice - TIN$, $AskPrice - RIN$) depict a similar behavior as explained for BN condition (Figure 7.6b). The bid price in RIN condition is lower than that of the bid price in BN condition. As, there are more resources in RIN condition than in BN condition and hence, the competition among the consumer nodes is lesser in RIN condition than in BN condition. The less competition results in a lower bid price in RIN condition in comparison to BN condition.

It can be concluded from the above discussion that a nature inspired ACO-based, self-organizing mechanism with CDA scheme is preferred over an ACO-based mechanism with FCFS scheme. The compute intensive CDA based mechanism performs as good as a less compute intensive FCFS mechanism in terms of consumer/producer utilization, response time and the pheromone value evolution. The CDA based mechanism enables the individual consumer/producer nodes to value their resource requests and resource offers according to their previous experiences from the ad hoc grid. The consumer/producer nodes can increase/decrease their bid/ask prices according to the resource demand/availability in the ad hoc gird. Thus, a CDA based ACO mechanism enables the node level self-organization along with the system level self-organization.

7.4.3 Load Balancing Factor

In the second set of experiments, we applied a load balancing factor to balance the workload among the participating matchmakers. This factor is required for distributing the resource discovery load among all the participating matchmakers. The load balancing factor ensures a minimum level of matchmaking efficiency and response time of a matchmaker of the same resource category. As soon as a matchmaker reaches the threshold of matchmaking efficiency and response time, the overloaded matchmaker transfers its excess workload to the closest matchmaker. In these experiments, there is only one resource category and multiple instances of the matchmakers represent different food sources.

In this way, the overloaded matchmaker can maintain its minimum level

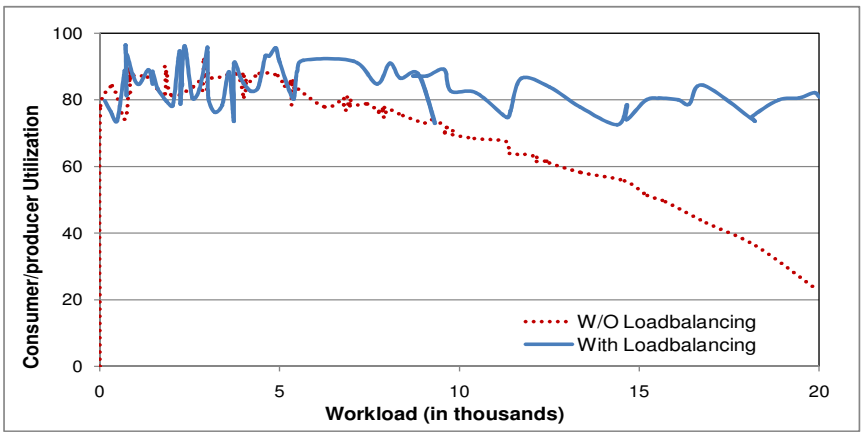


Figure 7.7: *Matchmaking efficiency of the ad hoc grid.*

of matchmaking. The details for calculating the overload threshold and sharing the excess workload are described in [23], whereas, algorithms for locating a closest matchmaker and for re-directing nodes to the other matchmaker(s) are described in [20]. We would like to point out that overload threshold calculation and matchmaker promotion and demotion mechanism proposed in [20, 23], are not used to promote/demote matchmakers in this work. We used the overload threshold calculation mechanism only for load balancing among the matchmakers. The analysis parameters and experimental setup are the same as explained in Section 7.4.

Figure 7.7 depicts matchmaking efficiency and Figure 7.8 represents the matchmaker response time. The horizontal axis represents the work load (messages processed by the ad hoc grid in a unit time interval) in both the figures and the vertical axis represents the matchmaking efficiency in Figure 7.7 and response time in Figure 7.8.

Initially, the experiments are executed without the load balancing factor. It is observed that the matchmaker with the highest pheromone value, out of all participating matchmakers, received and processed all request and offer messages from all the participating consumer/producer nodes. This phenomenon is expected for an ACO based resource management system. The high pheromone value of a matchmaker attracts more ants towards that matchmaker, which results in more workload for that matchmaker and a reduced workload for the other matchmakers. Ultimately, the matchmaker with the highest pheromone value receives and processes the complete workload of the ad hoc grid. At the same time, it is also observed that the matchmaking ef-

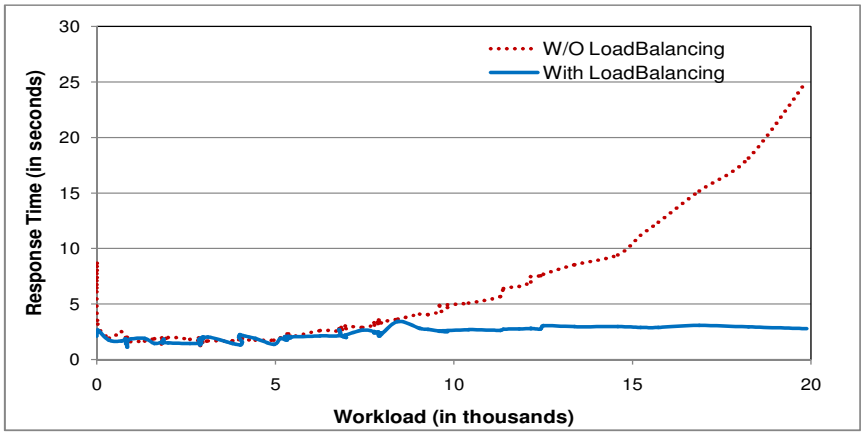


Figure 7.8: *Response time of the ad hoc grid.*

ficiency starts decreasing and response time starts increasing (Dotted line in Figures 7.7 & 7.8) with the increasing workload of the matchmaker having the highest pheromone value. However, the drop in matchmaking efficiency and increase in the response time to the consumer/producer nodes is not desired. A drop in the matchmaking efficiency means that the matchmaker is unable to find compatible request/offer pairs. A drop in the matchmaking efficiency also implies that the request/offer messages are being discarded by the matchmaker after their TTL expiry. The increased response time, with the increasing workload of the matchmaker, depicts the increased processing time of the matchmaker. It is observed that the response time becomes higher than the validity period (TTL) of the request/offer messages with the increased workload of a matchmaker.

The load balancing factor is applied when the workload of a matchmaker approaches the threshold value. When a matchmaker reaches its threshold values for matchmaking efficiency, the excess workload is processed by the matchmaker with the second highest pheromone value. We refer to [20,23] for the details of distributing excess workload among different matchmakers and threshold calculation, respectively. The load balancing factor results in increased matchmaking efficiency and a decreased response time (represented by the continuous line in Figures 7.7 & 7.8) for the same workload.

It can be concluded from the above discussions, that the proposed self-organizing mechanism enables the ad hoc grid to self-organize in different network conditions. Furthermore, the consumer/producer nodes can change their resource categories according to their resource demands. This resource

category change temporarily disturbs the stability of the ad hoc grid. However, the proposed ACO based self-organizing mechanism brings the ad hoc grid back to its stable state. We observed the same consumer/producer utilization and response time trend for CDA based and FCFS based approach. CDA based approach is flexible in comparison to FCFS based approach. Considering the dynamic nature of the ad hoc grid, an ACO algorithm with CDA is preferred over an ACO algorithm with FCFS towards self-organization. Furthermore, the CDA based modified ACO approach gives a stable behavior to the ad hoc grid and shows a better load balancing.

7.5 Concluding Remarks

An ACO inspired, micro economic based resource management approach for the ad hoc grids is presented in this chapter. We used matchmaking performance as the basic factor for calculating the pheromone value. The proposed ACO inspired CDA based approach enables node level as well as system level self-organization and supports resource specialization in an ad hoc grid. We also applied load balancing factor for distributing work load among all the participating matchmakers and for maintaining a minimum level of matchmaking efficiency. From the experimental results it can be concluded that the proposed mechanism gives a stable behavior of the system in resource management, and shows better load balancing.

Chapter 8

Lowering the Curtains-Conclusions

THIS dissertation dealt with the infrastructure level self-organization mechanisms in an ad hoc grid. These mechanisms use segmentation and desegmentation of the ad hoc grid according to the workload of the resource allocator (matchmaker). We study the proposed mechanisms for an infrastructural spectrum, ranging from fully centralized to fully decentralized extremes, from different perspectives in order to determine the trade-offs for an ad hoc grid infrastructure. The three main themes developed include: market based continuous double auction, social networking and the nature inspired modified ant colony optimization. There are, therefore, a number of conclusions from this work which are detailed in this chapter. It also summarizes the dissertation and highlights our future research directions.

8.1 Comparing Studied Approaches

In this section, we present a comparative summary of the findings from the studied approaches. The consumer/producer utilization of different schemes under different network conditions is summarized in Table 8.1. Whereas, Table 8.2 depicts the summary of the matchmaker's response time in different approaches under different network conditions. We performed message complexity analysis for finding a match in different schemes in order to understand the communication cost in each scheme. The message complexity analysis

		Market-based		Social	Nature Inspired	
		Centralized	MAMM	Networks	FCFS	CDA
Consumer Utilization	RIN	100%	100%	50%	100%	92%
	TIN	20%	20%	10%	25%	23%
	BN	85%	80%	10%	92%	85%
Producer Utilization	RIN	20%	20%	25%	25%	22%
	TIN	100%	100%	10%	98%	87%
	BN	85%	80%	10%	92%	85%

Table 8.1: *Comparing consumer/producer utilization in different schemes at different network conditions.*

representing the best and the worst case of each scheme is summarized in Table 8.3.

The proposed mechanism works equally well in different network conditions from micro-economics perspective. The consumer/producer utilization in balanced network condition becomes independent of the size of the ad hoc grid. Addition/removal of new segments does not affect the working of the ad hoc grid. In resource intensive network condition and task intensive network condition, consumer utilization and producer utilization is about 100% in the respective network conditions. Whereas, producer utilization in TIN and consumer utilization in RIN is almost equal to the ratio of the scarce commodity in the respective network conditions.

In social networking perspective, the behavior of the ad hoc grid varies is different network conditions. However, one conclusion is common for all the network conditions; the effectiveness of the social network decreases with an

		Market-based		Social	Nature Inspired	
		Centralized	MAMM	Networks	FCFS	CDA
Consumer Response Time	RIN	186	383	05	21	203
	TIN	8167	4079	05	3980	1169
	BN	2063	2197	05	584	406
Producer Response Time	RIN	7963	7819	05	223	871
	TIN	63	293	05	70	316
	BN	2158	2295	05	194	380

Table 8.2: *Comparing matchmaker response time in different schemes at different network conditions.*

	Market based		Social Networks
	Centralized	MAMM	
Best Case	4	4	$N+1$
Worst Case	$N+2$	$M+N_i+1$	$2N+1$

Table 8.3: *Comparing number of messages required in different schemes.*

increase in the degree of the social network of an ad hoc grid node. The communication cost and response time increases with an increasing degree of the social network of a node. The expected effectiveness of a large social network of a node is diminished by the increased overheads. BN and RIN conditions show better performance for small ad hoc grids in terms of consumer/producer utilization, response time and the communication costs. Whereas, TIN condition is not suitable due to the abundance of the task requests and scarcity of the resource offers. TIN condition in social networking perspective is not considered a choice under any circumstance.

In the nature-inspired modified ACO approach, we considered resource requests and resource offers of the ad hoc grid nodes as ants and the matchmakers as the food sources. The ad hoc grid self-organizes by dividing itself into different segments according to different resource categories. The ad hoc grid nodes looking for different resource types ultimately reside into their respective ad hoc grid segment. Furthermore, the addition of micro-economic based CDA mechanism helped the ad hoc grid nodes in making a wise evaluation of their resource requests and resource offers. The ACO based mechanism is not scalable with one resource category and requires the proposed segmentation/de-segmentation mechanism for scalability. The load balancing factor was needed for avoiding the overload condition of one matchmaker.

8.2 Selecting an Ad Hoc Grid Infrastructure

In this section, we describe the scenarios while selecting a particular ad hoc grid infrastructure on the infrastructural spectrum.

8.2.1 Fully Centralized

A centralized ad hoc grid infrastructure is preferred for a small network. However, there is no definition of the 'small', in our analysis small depends on the experimental setup for the ad hoc grid. Moreover, in our case, a fully centralized is not truly centralized. As, we distribute the node level decision making to the participating consumer/producer nodes. A centralized matchmaker performs the matchmaking process in the ad hoc grid. A centralized ad hoc grid infrastructure is not scalable. Furthermore, after a certain threshold ad hoc grid is unable to maintain its matchmaking capacity and the response time of the consumer/producer nodes showed an increasing trend. A matchmaker with CDA as the matchmaking mechanism works well for a centralized infrastructure. The CDA based mechanism also supports node level self-organization.

8.2.2 Fully Decentralized

A fully decentralized ad hoc grid infrastructure is not suitable according to our analysis. A fully decentralized infrastructure showed reduced matchmaking efficiency, increased message communication and an increased response time in comparison to the fully centralized or a hybrid infrastructure for the equal workload. Fully decentralized infrastructures are suitable for a small ad hoc grid where maintaining a separate matchmaker is not acceptable.

8.2.3 Hybrid

An ad hoc grid has a hybrid infrastructure when the ad hoc grid has more than one matchmakers and each matchmaker is responsible for a set of nodes in its segment. The dynamic promotion and demotion of matchmakers according to their overload status introduced a hybrid ad hoc grid infrastructure. The promotion of a node to a matchmaker divides the ad hoc grid into segments, whereas, the demotion of a matchmaker to an ordinary node merges back the ad hoc grid segments.

Hybrid infrastructure is scalable, dynamic and supports system level self-organization. In hybrid infrastructure segments are created and destroyed according to the workload in ad hoc grid. Dynamic promotion and demotion of the matchmakers helps in achieving the dynamic segmentation and desegmentation in the ad hoc grid.

8.3 Trade-offs for Ad Hoc Grid Infrastructure

Scalability, dynamism/adaptation and self-organization arising from the emergent behavior of the ad hoc grid nodes are identified as the main research challenges in Section 1.2 for this dissertation. The following trade-offs can be deduced for the above mentioned research challenges in view of the proposed mechanisms and the experimental results presented in this dissertation.

8.3.1 Scalability

A centralized infrastructure is not scalable. One matchmaker gets overloaded with the increased workload and is unable to maintain its matchmaking capacity. The proposed mechanism supports scalability in an ad hoc grid by segmenting the ad hoc grid and dynamically promoting ad hoc grid nodes as matchmaker(s). Each matchmaker is responsible for a set of nodes in the ad hoc grid. This process is repeated whenever a matchmaker is overloaded due to newly joining nodes in its segment or due to an increased frequency of the request/offer messages being submitted to the matchmaker. From the experimental results in Sections 4.5 and 5.9, it was observed that ad hoc grid became independent of the scale at which it operated in the proposed mechanism for all the considered network conditions (BN, RIN and TIN).

As each participating node in a fully decentralized ad hoc grid infrastructure (P2P) is responsible for finding its required resource or tasks, therefore, a fully decentralized ad hoc grid infrastructure is scalable by its nature. However, with increasing number of nodes, the P2P ad hoc grid is unable to maintain its matchmaking capacity and the response time for finding a match also increase due to the increased communication in a P2P ad hoc grid (Section 6.4).

8.3.2 Dynamism

Ad hoc grid nodes are dynamic in nature and show intermittent and volatile participation. Ad hoc grid needs a dynamic mechanism to handle the volatile and intermittent participation of the participating nodes. In the proposed mechanism, ad hoc grid is divided into more segments with the increased workload in one segment and the segments are merged back when the workload decreases. This segmentation and de-segmentation process is achieved by promoting and demoting the matchmakers according to the overload condition

of matchmakers in the ad hoc grid. Thus, the proposed mechanism is capable of handling the dynamic participation of ad hoc grid nodes and keeps on re-structuring the ad hoc grid according to the workload in ad hoc grid.

The dynamic nature of the proposed mechanism is observed in the market based CDA perspective, non market-based perspective and in nature inspired ACO perspective. The dynamic behavior of the proposed mechanism is observed in all studied network conditions including BN, RIN and TIN.

A fully decentralized ad hoc grid (P2P) infrastructure is inherently dynamic as the proposed mechanism is implemented by extending a structured overlay network, Pastry. However, a P2P ad hoc grid infrastructure does not perform well in terms of consumer/producer utilization and the response time for a matched message. This behavior is resulted from the increased processing time for finding a match due to the increased number of the messages and the increased communication overhead in a fully decentralized ad hoc grid.

8.3.3 Self-organization

The proposed mechanism enables the ad hoc grid to self-organize at an individual node level as well as at the system level. The matchmaker uses CDA as its matchmaking mechanism. Whereas, the ad hoc grid nodes apply a history based dynamic pricing mechanism for calculating the ask/bid price.

The history based dynamic pricing strategy enables an ad hoc grid node to calculate its bid/ask price for its subsequent request/offer messages according to its previous experiences from the ad hoc grid. A node increases/decreases its bid/ask price according to its consumer/producer utilization of the ad hoc grid. Each ad hoc grid node is decentralized and decides independently from its local knowledge only. In this way, the ad hoc grid node self-organizes for achieving its maximum utility form the ad hoc grid. The market based approach is applicable in BN, RIN and TIN conditions in centralized and hybrid infrastructures. This scheme may be applicable for a P2P ad hoc grid, if communication overhead could be neglected.

A system level self-organization is observed due to the local decisions of the ad hoc grid nodes. The ad hoc grid self-organizes and converges towards an equilibrium point at system level due to the informed decisions of the individual participating consumer/producer nodes. The node level and system level self-organizing behavior, in a market based perspective of the proposed mechanism, are observed in all studied network condition including BN, RIN and TIN. Moreover, at the infrastructure-level, ad hoc self-organizes itself with

the help of dynamic segmentation and de-segmentation mechanism whenever the workload condition changes.

System level self-organization is achieved in two different scenarios for the nature inspired modified ACO mechanism. In the first scenario, ad hoc gird nodes self-organize themselves into virtual segments according to their resource categories. These virtual ad hoc grid segments are re-structured whenever a set of participating ad hoc grid nodes changes their resource category. Section 7.4.2 proved that the proposed ACO based approach enabled the ad hoc grid to self-organize into virtual segments of different resource categories in BN, RIN and TIN conditions. The ad hoc grid maintained its matchmaking capacity and the response time while the ad hoc grid nodes re-structured themselves into different virtual resource segments.

In the second scenario, the ad hoc grid achieves self-organization by balancing its workload among different segments of the same resource category. The experimental results discussed in Section 7.4.3 confirm that the ad hoc grid is capable of maintaining its matchmaking capacity in the proposed self-organized approach while balancing the workload among different segments of the ad hoc grid. As the proposed nature inspired ACO based approach supports virtual specialized resource groups or load balanced resource groups of the same category, therefore, it is not applicable for a centralized ad hoc grid infrastructure by its definition. This approach works well in hybrid ad hoc grid infrastructures as proved in Sections 7.4.2 & 7.4.3. The applicability of the proposed ACO approach in a fully decentralized (P2P) ad hoc grid is left for future work.

8.4 Dissertation Summary

This section provides a summary of the findings from the dissertation chapters.

In Chapter 2, an overview of the necessary background knowledge regarding ad hoc grid, P2P systems and the concept of self-organization in different contexts was provided. This overview was needed to understand the work presented in this dissertation. An overview of the existing approaches to self-organizing resource management in an ad hoc grid was provided to highlight the research challenges addressed in this dissertation.

In Chapter 3, the developed experimental platform used for obtaining the experimental results reported in this dissertation was explained. Different components of the platform, their attributes and the interactions between these

components were explained. A summary of the experimental testbed, PlanetLab, and of the experimental setup was also provided.

In Chapter 4, the market based self-organizing mechanism was presented. A summary of the micro-economic framework in general and reasons for using CDA as the resource allocation mechanism was elaborated before explaining the proposed mechanism. The proposed mechanism was compared in different network conditions with non micro-economic based approach. The results helped us in understanding the emergent behavior of the self interested participants (consumer, producer and/or matchmaker) under varying work load condition in an ad hoc grid. Furthermore, we also understood the impact of the state changes of ad hoc grid due to the emergent behavior of participating nodes.

In Chapter 5, we provided extensions for Pastry, an overlay network. On top of this overlay network, we implemented new algorithms that prove effective, as they do not impact the correct behavior of the ad hoc grid while rendering the infrastructure more robust.

In Chapter 6, the proposed mechanism was studied by exploring the social network of a node. This study explained the behavior of the proposed mechanisms on an infrastructural spectrum, ranging from fully centralized to fully decentralized extremes. It also enabled us in determining the trade-offs for system-level adaption of the ad hoc grid and in a better understanding of the effect of an ad hoc grid infrastructure on resource discovery. Furthermore, it explained how the social network of a node affects resource discovery in an ad hoc grid.

In Chapter 7, the proposed mechanism was studied with the help of nature inspired modified ACO mechanism. This study was performed in a market-based and non market-based setting. The experimental results provided insight on the application of existing self-organizing mechanism, ACO specifically, for the resource discovery and self-organization in an ad hoc grid. These results further helped us to compare the ACO based self-organizing mechanism with the market-based self-organizing mechanism and on achieving resource specialization in an ad hoc grid.

In Chapter 8, a summary of findings was presented and trade-offs for an ad hoc grid infrastructure in different network conditions were identified.

8.5 Contributions

The main contributions of this works are as follows:

- We studied different schemes for infrastructure-level self-organization in an ad hoc grid. These schemes are studied on an infrastructural spectrum, ranging from fully centralized to fully decentralized extremes. A fixed infrastructure is not suitable for an ad hoc grid due to the high fluctuations in resource availability/demands patterns, which are caused by the intermittent and volatile participation of the ad hoc grid nodes. Trade-offs for selecting a particular ad hoc grid infrastructure are discussed.

- The proposed infrastructure-level self-organization mechanism is based on segmentation/de-segmentation of the ad hoc grid according to the workload of a resource allocator (matchmaker). This mechanism is studied in market-based and non market-based environments and is evaluated in different network conditions. We discovered that the market based mechanism takes care of individual participant's utility as well as of the system-level utility, and is as efficient as a non market-based mechanism, which is computationally less expensive.

- We presented algorithms to extend a structured overlay network for the proposed segmentation/de-segmentation mechanism in an ad hoc grid. These include the algorithms for promoting a matchmaker, demoting a matchmaker, discovering a responsible matchmaker and node join/leave algorithm. The proposed algorithms do not affect the functional aspect of the ad hoc grid. In addition, the resulting infrastructure of the ad hoc grid is more robust, as it leverages the self-organization of the underlying network.

- We studied the effect of a node's social network on its ability for discovering the required resources by studying the effect of degree of neighborhood of node. The degree of neighborhood of a node is considered being equivalent to the social contact of a person in society. We discovered that by increasing the degree of neighborhood of node, its ability to discover the required resources increases up to a certain degree. After that point, the increased degree of neighborhood does not help due to the increased communication cost with the social peers of a node.

- We investigated the infrastructure-level self-organization in an ad hoc

grid by applying the nature inspired self-organizing approaches. We applied a modified ACO algorithm in an ad hoc grid. This phenomenon was studied in the market-based (CDA) and non market-based (FCFS) settings. We discovered that CDA based ACO performs equally well in comparison with the simplest and less compute intensive FCFS approach. Moreover, the CDA based ACO approach helps in achieving node-level self-organization. The modified ACO algorithm also helped in forming specialized resource groups in an ad hoc grid.

8.6 Future Directions

This section will outline our future research directions beyond this work.

- Resource co-allocation: In this dissertation, one resource request (task) is allocated to only one resource offer (resource). After successful execution of a task, the producer node submits/announces the revised resource offer, a new task is matched with the revised resource offer and the producer node starts executing the new task. The cost and time will be reduced if a resource is co-allocated to more than one task. At the same time, a set of new challenges need to be studied with the co-allocation of resources. Resource co-allocation has been studied in the context of conventional grids [51, 100, 128] and not in the context of the ad hoc grids. This can be an interesting research direction for future.

- Self-organizing ad hoc grid for reconfigurable devices: Reconfigurable devices [72, 77, 137], like FPGAs, can be added for resource specialization in an ad hoc grid or a single segment of the ad hoc grid. The introduction of the reconfigurable devices will open up new challenges for the resource management in an ad hoc grid.

- Fault tolerant ad hoc grid: Failure handling of a consumer/producer node is managed by the failure handling mechanism of the Pastry overlay network. However, failure handling of a matchmaker is not discussed in this dissertation. Fault tolerance will enable an ad hoc grid segment to work even in the course of failure of a matchmaker. We will also look into a different QoS issues for the proposed mechanisms.

- Secure ad hoc grid: All the consumer/producer/matchmaker agents are assumed honest in this dissertation and no malicious activity is expected from the participating nodes. This assumption was in place to focus

on the study of the proposed mechanism and not on the issues due to the malicious behavior of the participating nodes. However, this assumption is difficult to guarantee. Different mechanisms to secure an ad hoc grid from micro-economic and from non micro-economic perspectives can be studied as another future research direction.

- Accounting service in ad hoc grid: Ad hoc grids are characterized by the volatile, intermittent participation of the participating nodes. These characteristics decrease the reliability of the resources. Mechanisms to encourage a reliable participation of the participating nodes need to be investigated in the context of ad hoc grid. Price is one mechanism in the micro-economic domain for attracting the resource providers for the reliable participation. Other mechanisms like reputation-based, bond-based, barter trade-based etc from micro-economic and non micro-economic domains could be investigated in order to have an accounting service in an ad hoc grid.

Bibliography

[1] PlanetLab Online, https://www.planet-lab.org/.

[2] GridPhYn Home, http://www.griphyn.org/.

[3] Globus Home, http://www.globus.org/.

[4] Open Source Napster Server, http://opennap.sourceforge.net/.

[5] FusionGrid Project, http://www.fusiongrid.org.

[6] ASTROGrid Project, http://www2.astrogrid.org.

[7] NEESgrid Home, http://access.ncsa.uiuc.edu/Stories/NEESgrid.

[8] SETI@home Project, http://setiathome.berkeley.edu/.

[9] Folding@home Project, http://folding.stanford.edu/.

[10] DataTag Project, http://datatag.web.cern.ch/datatag/.

[11] PC Grid, http://www.ud.com.

[12] BOINC Home, http://boinc.berkeley.edu/.

[13] Kazaa Home, http://www.kazaa.com.

[14] German National Grid, http://www.d-grid.de/.

[15] Gnutella Home, http://www.gnutella.com.

[16] DAS3, http://www.cs.vu.nl/das3/.

[17] OneLab, http://www.onelab.eu/.

[18] PlanetLab Europe, https://www.planet-lab.eu.

[19] PlanetLab Joining Guideline, https://www.planet-lab.org/joining.

[20] Tariq Abdullah, Luc Onana Alima, Vassiliy Sokolov, David Calomme, and Koen Bertels. Hybrid resource discovery mechanism in ad hoc grid using strucutred overlay. In *Proceedings of the 22nd International Conference on Architecture of Computing Systems*, March 2009.

[21] Tariq Abdullah, Koen Bertels, and Luc Onana Alima. Ant colony inspired microeconomic based resource management in ad hoc grids. In *Proceedings of the 4th International Conference on Grid and Pervasive Computing*, Geneva, May 2009.

[22] Tariq Abdullah, Koen Bertels, Luc Onana Alima, and Zubair Nawaz. Effect of degree of neighborhood on resource discovery in ad hoc grids. In *Proceedings of the 23rd International Conference on Architecture of Computing Systems*, February 2010.

[23] Tariq Abdullah, Vassiliy Sokolov, Behnaz Pourebrahimi, and Koen Bertels. Self-organizing dynamic ad hoc grids. In *Proceedings of the 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops*, October 2008.

[24] Luc Onana Alima, Sameh El-Ansary, Per Brand, and Seif Haridi. DKS(N, k, f): A family of low-communication, scalable and fault-tolerant infrastructures for P2P applications. In *Proceedings of the 3rd International workshop on Global and P2P Computing on Large Scale Distributed Systems (CCGRID'03)*, 2003.

[25] Kaizar Amin, Gregor von Laszewski, and Armin R. Mikler. Toward an architecture for ad hoc grids. In *Proceedings of the 12th International Conference on Advanced Computing and Communications, ADCOM*, Ahmedabad Gujarat, India, December 15-18 2004.

[26] Yair Amir, Baruch Awerbuch, and Ryan Borgstrom. The java market: Transforming the Internet into a metacomputer. Technical Report CNDS-98-1, Center for Networking and Distributed Systems, John Hopkins University, 1998.

[27] David P. Anderson. BOINC: A system for public-resource computing and storage. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, 2004.

[28] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. SETI@home: An experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61, 2002.

[29] Stephanos Androutsellis-theotokis. A survey of peer-to-peer file sharing technologies–white paper. Technical report, ELTRUN, Athens University of Economics and Business, Greece, 2002.

[30] Artur Andrzejak, Sven Graupner, Vadim Kotov, and Holger Trinks. Algorithms for self-organization and adaptive service placement in dynamic distributed systems. Technical Report HPL-2002-259, HP laboratories Palo Alto, 2002.

[31] Marcos Dias De Assuncao and Rajkumar Buyya. An evaluation of communication demand of auction protocols in grid environments. In *Proceedings of the 3rd International Workshop on Grid Economics & Business (GECON'06)*, May 2006.

[32] Sujoy Basu, Sujata Banerjee, Puneet Sharma, and Sung-Ju Lee. NodeWiz: Peer-to-peer resource discovery for grids. In *Proceedings of the 5th IEEE International Symposium on Cluster Computing and the Grid*, volume 1, pages 213–220, 2005.

[33] Andy Bavier, Mic Bowman, Brent Chun, David Culler, Scott Karlin, Steve Muir, Larry Peterson, Timothy Roscoe, Tammo Spalink, and Mike Wawrzoniak. Operating system support for planetary-scale network services. In *Proceedings of the First Symposium on Networked Systems Design and Implementation (NSDI'04)*, 2004.

[34] R. Beckers, J.L. Deneubourg, and S. Goss. Trail and U-turns in the selection of the shortest paths by the ant Lasius Niger. *Journel of Theoretical Biology*, 159:397–415, 1992.

[35] Ashwin R. Bharambe, Mukesh Agrawal, and Srinivasan Seshan. Mercury: Supporting scalable multi-attribute range queries. In *Proceedings of the ACM SIGCOMM*, 2004.

[36] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm intelligence–From natural to artificial systems*. Oxford University Press, 1999.

[37] Richard R. Brooks. Sensor network self-organization using random graphs. *International Journal of Distributed Sensor Networks*, 5(3):201–208, 2009.

[38] Sharon Brunett, Dan Davis, Thomas Gottschalk, Paul Messina, and Carl Kesselman. Implementing distributed synthetic forces simulations in metacomputing environments. In *Proceedings of the 7th Heterogeneous Computing Workshop (HCW 98)*, 1998.

[39] Ali Raza Butt, Rongmei Zhang, and Y. Charlie Hu. A self-organizing flock of Condors. *Journal of Parallel and Distributed Computing*, 66(1):145–161, 2006.

[40] Rajkumar Buyya, David Abramson, and Jonathan Giddy. An economy driven resource management architecture for global computational power grids. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'00)*, Las Vegas, USA, June 26-29 2000.

[41] Min Cai, Martin Frank, Jinbo Chen, and Pedro Szekely. MAAN: A Multi-Attribute Addressable Network for grid information services. *Journal of Grid Computing*, 2(1):3–14, 2004.

[42] Scott Camazine, Jean-Louis Deneubourg, Nigel R. Franks, James Sneyd, Guy Theraulaz, and Eric Bonabeau. *Self-Organization in Biological Systems*, chapter What Is Self-Organization? Princeton University Press, 2001.

[43] Franck Cappello, Samir Djilali, Gilles Fedak, Thomas Herault, Frederic Magniette, Vincent Néri, and Oleg Lodygensky. Computing on large-scale distributed systems: Xtrem web architecture, programming models, security, tests and convergence with grid. *Future Generation Computer Systems*, 21(3):417–437, 2005.

[44] Adeep S. Cheema, Moosa Muhammad, and Indranil Gupta. Peer-to-peer discovery of computational resources for grid applications. In *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, 2005.

[45] Shang-Wen Cheng, David Garlan, Bradley Schmerl, Peter Steenkiste, and Ningning Hu. Software architecture-based adaptation for grid computing. In *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing (HPDC'02)*, 2002.

[46] Andrew Chien, Brad Calder, Stephen Elbert, and Karan Bhatia. Entropia: Architecture and performance of an enterprise desktop grid system. *Journal of Parallel and Distributed Computing*, 63(5):597–610, May 2003.

[47] Sungjin Choi, Maengsoon Baik, Joonmin Gil, Soonyoung Jung, and Chongsun Hwang. Adaptive group scheduling mechanism using mobile agents in peer-to-peer grid computing environment. *Applied Intelligence*, 25(2):199–221, 2006.

[48] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of the ICSI Workshop on Design Issues in Anonymity andUnobservability*, pages 46–66, 2000.

[49] Brian F. Cooper and Hector Garcia-Molina. Ad hoc, self-supervising peer-to-peer search networks. *ACM Transactions on Information Systems*, 23(2):169–200, 2005.

[50] George Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed Systems: Concepts and Design (fourth edition)*. Addison Wesley, 2005.

[51] Karl Czajkowski, Ian Foster, and Carl Kesselman. Resource co-allocation in computational grids. In *Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing (HPDC '99)*, 1999.

[52] J.L. Deneubourg, S. Aron, S. Goss, and J.M. Pasteels. The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, 3:159–168, 1990.

[53] Yuhui Deng, Frank Weng, and Adrian Ciura. Ant colony optimization inspired resource discovery in P2P Grid systems. *The Journal of Supercomputing*, 49(1):4–21, 2008.

[54] Marco Dorigo and Christian Blum. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344:243–278, 2005.

[55] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics-Part B*, 26(1):29–41, 1996.

[56] Falko Dressler. Benefits of bio-inspired technologies for networked embedded systems: An overview. In *Proceedings of the Dagstuhl Seminar 06031 on Organic Computing - Controlled Emergence*, 2006.

[57] Niels Drost, Rob V. van Nieuwpoort, and Henri Bal. Simple locality-aware co-allocation in peer-to-peer supercomputing. In *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID '06)*, 2006.

[58] Peter Druschel and Antony Rowstron. Past: A large-scale, persistent peer-to-peer storage utility. In *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*, 2001.

[59] Bruce Edmonds. *In The Evolution of Complexity*, chapter What is complexity? -The philosophy of complexity per se with application to some examples in evolution, pages 1–18. Kluwer, Dordrecht, 1999.

[60] D.H.J. Epema, M. Livny, R. van Dantzig, X. Evers, and J. Pruyne. A worldwide flock of condors: Load sharing among workstation clusters. *Future Generation Computer Systems,*, 21:53–65, 1996.

[61] Kayhan Erciyes and Resat Umit Payli. A cluster-based dynamic load balancing middleware protocol for grids. In *Proceeding of the Advances in Grid Computing (EGC'05)*, volume 3470, pages 805–812, 2005.

[62] Donald F. Ferguson, Christos Nikolaou, Jakka Sairamesh, and Yechiam Yemini. *Economic models for allocating resources in computer systems*, pages 156–183. World Scientific Publishing, 1996.

[63] Stefka Fidanova and Mariya Durchova. Ant algorithm for grid scheduling problem. In *Proceedings of the 5th International Conference on Large-Scale Scientific Computations (LSCC '05)*, pages 405–412, 2006.

[64] Ian Foster and Adriana Iamnitchi. On death, taxes, and the convergence of peer-to-peer and grid computing. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, February 2003.

[65] Ian Foster and Carl Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[66] Munehiro Fukuda, Yuichiro Tanaka, Naoya Suzuki, Lubomir F. Bic, and Shinya Kobayashi. A mobile-agent-based PC grid. In *Proceedings of the Autonomic Computing Workshop*, pages 142–150, 2003.

[67] Nicolis G. and I. Prigogine. *Self-organization in nonequilibrium systems: From dissipative structures to order through fluctuations*. Wiley, 1977.

[68] Pawel Garbacki, Dick H.J. Epema, and Marteen van Steen. The design and evaluation of a self-organizing super-peer network. *IEEE Transactions on Computers*, 59:317–331, 2010.

[69] L. Garcés-Erice, E.W. Biersack, P. A. Felber, K. W. Ross, and G. Urvoy-Keller. Hierarchical peer-to-peer systems. In *Proceedings of the ACM/IFIP International Conference on Parallel and Distributed Computing (Euro-Par)*, pages 643–657, 2003.

[70] M. Gardner. The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*, 223:120–123, 1970.

[71] Cécile Germain, Vincent Néri, Gilles Fedak, and Franck Cappello. XtremWeb: Building an experimental platform for global computing. In *Proceedings of the 1st IEEE/ACM International Workshop on Grid Computing*, 2000.

[72] Maya B. Gokhale and Paul S. Graham. *Reconfigurable Computing: Accelerating Computation with Field-Programmable Gate Arrays*. Springer The Netherlands, 2005.

[73] Jacek Gomoluch and Michael Schroeder. Market-based resource allocation for grid computing: A model and simulation. In *Proceedings of the First International Workshop on Middleware for Grid Computing (MGC '03)*, pages 137–144, 2003.

[74] Nathaniel S. Good and Aaron Krekelberg. Usability and Privacy: A study of kazaa P2P file-sharing. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI'03)*, pages 137–144, 2003.

[75] Rohit Gupta, Varun Sekhri, and Arun K. Somani. CompuP2P: An architecture for internet computing using peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems*, 17(11):1306–1320, 2006.

[76] Zygmunt J. Haas and Marc R. Pearlman. The performance of query control schemes for the Zone Routing Protocol. *IEEE/ACM Transactions on Networking*, 9(4):427–438, 2001.

[77] Scott Hauck and Andre DeHon. *Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation (Systems on Silicon)*. Morgan Kaufmann, 2007.

[78] David Hausheer and Burkhard Stiller. Decentralized auction-based pricing with peermart. In *Proceedings of the 9th IFIP/IEEE International Symposium on Integrated Network Management*, 2005.

[79] Geoffrey Heal. Planning without prices. *Review of Economic Studies*, 36(107):347–62, 1969.

[80] Joseph L. Hellerstein. Self-managing systems: A control theory foundation. In *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, 2004.

[81] Francis Heylighen. *The Encyclopedia of Life Support Systems*, chapter The Science Of Self-Organization And Adaptivity, pages 253–280. EOLSS Publishers Co. Ltd., 2001.

[82] Wolfgang Hoscheck, Francisco J. Jaen-Martinez, Asad Samar, Heinz Stockinger, and Kurt Stockinger. Data management in an international data grid project. In *Proceedings of the First IEEE/ACM International Workshop on Grid Computing (Grid'00)*, pages 77–90, December 2000.

[83] Leonid Hurwicz. The design of mechanisms for resource allocation. *American Economic Review*, 63(2):1–30, 1973.

[84] Adriana Iamnitchi and Ian Foster. On fully decentralized resource discovery in grid environments. In *Proceedings of the 2nd International Workshop on Grid Computing*, pages 51–62, London, UK, 2001. Springer-Verlag.

[85] Adriana Iamnitchi, Ian Foster, and Daniel C. Nurmi. A peer-to-peer approach to resource discovery in grid environments. In *Proceedings of the 11th Symposium on High Performance Distributed Computing*, pages 419–434, August 2002.

[86] IBM. An architectural blueprint for autonomic computing. Technical report, IBM, June 2005.

[87] Michael A. Jaeger, Helge Parzyjegla, Gero Muhl, and Klaus Herrmann. Self-organizing broker topologies for publish/subscribe systems. In *Proceedings of the ACM symposium on Applied computing (SAC'07)*, pages 543–550, 2007.

[88] Nicholas R Jennings. On agent-based software engineering. *Artificial Intelligence*, 117:277–296, 2000.

[89] Umesh Kant and Daniel Grosu. Double auction protocols for resource allocation in grids. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05)*, pages 366–371, 2005.

[90] Jik-Soo Kim, Peter Keleher, Michael Marsh, Bobby Bhattacharjee, and Alan Sussman. Using content-addressable networks for load balancing in desktop grids. In *Proceedings of the 16th International Symposium on High Performance Distributed Computing (HPDC)*, pages 189–198, 2007.

[91] James F. Kurose and Rahul Simha. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Transactions on Computers*, 38(5):705–717, 1989.

[92] Spyros Lalis and Alexandros Karipidis. JaWS: An open market-based framework for distributed computing over the Internet. In *Proceedings of the First IEEE/ACM International Workshop on Grid Computing, (GRID '00)*, pages 36–46. Springer-Verlag, 2000.

[93] Nathaniel Leibowitz, Matei Ripeanu, and Adam Wierzbicki. Deconstructing the kazaa network. In *Proceedings of the 3rd IEEE Workshop on Internet Applications (WIAPP'03)*, 2003.

[94] Michael J. Litzkow, Miron Livny, and Matt W. Mutka. Condor: A hunter of idle workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems*, pages 104–111, June 1988.

[95] Virginia Lo, Daniel Zappala, Dayi Zhou, Yuhong Liu, and Shanyu Zhao. Cluster computing on the fly: P2P scheduling of idle cycles in the Internet. In *Proceedings of the 3rd International Workshop on Peer-to-Peer Systmes*, 2004.

[96] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th international conference on Supercomputing (ICS'02)*, pages 84–95. ACM Press, 2002.

[97] Attila Csaba Marosi, Gabor Gombas, Zoltan Balaton, Peter Kacsuk, and Tamas Kiss. SZTAKI Desktop Grid: Building a scalable, secure platform for desktop grid computing. Technical Report TR-0100, CoreGRID - Network of Excellence, August 2007.

[98] Carlo Mastroianni, Domenico Talia, and Oreste Verta. A super-peer model for building resource discovery services in grids: Design and simulation analysis. In *Proceedings of the European Grid Conference*, 2005.

[99] André Miede, Steffen Braun, Julian Eckert, Dieter Schuller, Nicolas Repp, and Ralf Steinmetz. A comparison of self-organization mechanisms in nature and information technology. In *Proceedings of the Americas Conference on Information Systems (AMCIS)*, 2009.

[100] H.H. Mohamed and D.H.J. Epema. Experiences with the KOALA co-allocating scheduler in multiclusters. In *Proceedings of the 5th IEEE/ACM Int'l Symposium on Cluster Computing and the GRID (CCGrid'05)*, pages 784–791, May 2005.

[101] Alberto Montresor, Hein Meling, and Ozalp Babaoglu. Messor: Load-balancing through a swarm of autonomous agents. Technical report, University of Bologna, May 2002.

[102] Rafael Moreno-Vozmediano. A hybrid mechanism for resource/service discovery in ad-hoc grids. *Future Generation Computer Systems*, 25(7):717–727, 2009.

[103] John P. Morrison, James J. Kennedy, and David A. Power. Webcom: A web based volunteer computer. *The Journal of Supercomputing*, 18(1):47–61, 2001.

[104] Gero Mühl, Matthias Werner, Michael A. Jaeger, Klaus Herrmann, and Helge Parzyjegla. On the definitions of self-managing and self-organizing systems. In *Proceedings of Kommunikation in Verteilten Systemen (KiVS 2007)*, 2007.

[105] Noam Nisan, Shmulik London, Ori Regev, and Noam Camiel. Globally distributed computation over the internet - The POPCORN Project. In *Proceedings of the 18th International Conference on Distributed Computing Systems (ICDCS)*, page 592. IEEE Computer Society, 1998.

[106] Özalp Babaoglu, Hein Meling, and Alberto Montresor. Anthill: A framework for the development of agent-based peer-to-peer systems. In *Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems (ICDCS '02)*, pages 15–22, 2002.

[107] Anand Padmanabhan, Shaowen Wang, Sukumar Ghosh, and Ransom Briggs. A Self-Organized Grouping (SOG) method for efficient grid resource discovery. In *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, pages 312–317, 2005.

[108] Adarsh Patil, David A. Power, and John P. Morrison. Economy-based computing with webcom. *Journal of Computer and Information Science and Engineering*, 1(2):82–89, 2007.

[109] Larry Peterson, Tom Anderson, David Culler, and Timothy Roscoe. A blueprint for introducing disruptive technology into the Internet. In *Proceedings of the First ACM Workshop on Hot Topics in Networks*, 2002.

[110] Larry Peterson, Andy Bavier, Marc E. Fiuczynski, and Steve Muir. Experiences building PlanetLab. In *Proceedings of the 7th USENIX Symposium on Operating System Design and Implementation (OSDI'06)*, November 2006.

[111] Larry Peterson, Steve Muir, Timpothy Roscoe, and Aaron Klingaman. Planetlab architecture: An overview. Technical Report PDN-06-031, Princeton University and Intel Research – Berkeley, May 2006.

[112] Larry Peterson, Vivek Pai, Neil Spring, and Andy Bavier. Using PlanetLab for network research: Myths, realities, and best practices. Technical Report PDN-05-028, PlanetLab, June 2005.

[113] Behnaz Pourebrahimi. *An economic framework for resource allocation in ad-hoc grids*. PhD thesis, Delft University of Technology, 2009.

[114] Behnaz Pourebrahimi and Koen Bertels. Auction protocols for resource allocations in ad-hoc grids. In *Proceedings of the 14th International Euro-Par Conference*, pages 520–533, August 2008.

[115] Behnaz Pourebrahimi, Koen Bertels, G Kandru, and Stamatis Vassiliadis. Market-based resource allocation in grids. In *Proceedings of the 2nd IEEE International Conference on e-Science & Grid Computing*, 2006.

[116] Behnaz Pourebrahimi, Koen Bertels, Stamatis Vassiliadis, and Luc Onana Alima. A dynamic pricing and bidding strategy for autonomous agents in grids. In *Proceedings of the 6th International Workshop on Agents and P2P Computing*, 2007.

[117] Diego Puppin, Stefano Moncelli, Ranieri Baraglia, Nicola Tonellotto, and Fabrizio Silvestri. A grid information service based on peer-to-peer. In *Proceedings of Euro-par*, pages 454–464, 2005.

[118] Nadav Raichman, Tamir Gabay, Yael Katsir, Yoash Shapira, and Eshel Ben-Jacob. Engineered self organization in natural and man-made systems. In *Proceedings of the 10th International Symposium on Continuum Models and Discrete Systems (CMDS)*, 2004.

[119] Imran Rao, Aaron Harwood, and Shanika Karunasekera. On problem for aggregate node selection for unstructured overlay networks. In *Proceedings of the 12th international Conference on High performance Computing and Communications (HPCC '10)*, pages 369–375, 2010.

[120] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *Proceedings of the ACM SIGCOMM '01*, pages 161–172. ACM Press, 2001.

[121] Matei Ripeanu, Ian Foster, and Adriana Iamnitchi. Mapping the Gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Ineternet Computing Journal*, 6(1):50–57, 2002.

[122] Graham Ritchie and John Levine. A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments. In *Proceedings of the 23rd workshop of the UK planning and scheduling special interest group*, 2004.

[123] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware '01)*, volume 2218, pages 329–350, 2001.

[124] Luis F. G. Sarmenta and Satoshi Hirano. Bayanihan: Building and studying Web-based volunteer computing systems using Java. *Future Generation Computer Systems*, 15(5-6):675–686, 1999.

[125] Mohammad Imran Shaik, S. Mary Saira Bhanu, and N. P. Gopalan. Distributed grid resource discovery with matchmakers. In *Proceedings of the 2nd International Conference on Semantics, Knowledge and Grid*, 2006.

[126] J. Sherwani, N. Ali, N. Lotia, Z. Hayat, and R. Buyya. Libra: A computational economy-based job scheduling system for clusters. *Software: Practice and Experience*, 34(6):573–590, May 2004.

[127] Ken'ichiro Shirose, Satoshi Matsuoka, Hidemoto Nakada, and Hirotaka Ogawa. Autonomous configuration of grid monitoring systems. In *Proceedings of the 2004 Symposium on Applications and the Internet Workshops*, 2004.

[128] O.O. Sonmez, H.H. Mohamed, and D.H.J. Epema. On the benefit of processor co-allocation in multicluster grid systems. *IEEE Transactions on Parallel and Distributed Systems,*, 21:778–789, 2010.

[129] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM '01*, pages 149–160, 2001.

[130] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[131] Ratnasamy Sylvia, Stoica Ion, and Shenker Scott. Routing algorithms for DHTs: Some open questions. In *Proceedings of the 1st International Workshop on Peer-To-Peer Systems (IPTPS'02)*, pages 45–52, London, UK, 2002. Springer-Verlag.

[132] PPDG team. The particle physics data grid (ppdg): From fabric to physics, final report, July 2006.

[133] Douglas Thain, Todd Tannenbaum, and Miron Livny. Distributed computing in practice: The Condor experience. *Concurrency and Computation: Practice & Experience*, 17(2-4):323–356, February 2005.

[134] P. Trunfio, D. Talia, H. Papadakis, P. Fragopoulou, M. Mordacchini, M. Pennanen, K. Popov, V. Vlassov, and S. Haridi. Peer-to-peer resource discovery in grids: Models and systems. *Future Generation Computer Systems*, 23(7):864–878, 2007.

[135] Dimitrios Tsoumakos and Dimitrios Tsoumakos. A comparison of peer-to-peer search methods. In *Proceedings of International Workshop on the Web and Databases (WebDB)*, pages 61–66, 2003.

[136] Sathish S. Vadhiyar and Jack J. Dongarra. Self adaptivity in grid computing. *Concurrency and Computation: Practice & Experience*, 17(2-4):235–257, February 2005.

[137] Stamatis Vassiliadis and Dimitrios Soudris, editors. *Fine- and Coarse-Grain Reconfigurable Computing*. Springer, 2007. ISBN: 978-1-4020-6504-0.

[138] Carl A. Waldspurger, Tad Hogg, Bernardo A. Huberman, Jeffrey O. Kephart, and W. Scott Stornetta. Spawn: A distributed computational economy. *IEEE Transactions on Software Engineering*, 18(12):103–117, 1992.

[139] Norbert Wiener. *Cybernetics Or Control and Communication In Animal And The Machine*. MIT Press, 1948.

[140] Stephen Wolfram. *A new kind of science*. Wolfram Media, 2002.

[141] Beverly Yang and Hector Garcia-Molina. Comparing hybrid peer-to-peer systems. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB '01)*, pages 561–570, 2001.

[142] Beverly Yang and Hector Garcia-Molina. Improving search in peer-to-peer networks. In *Proceedings of 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, 2002.

[143] Weishuai Yang, Nael Abu-Ghazaleh, and Michael J. Lewis. Automatic clustering for self organizing grids. In *Proceedings of the IEEE International Conference on Cluster Computing*, 2006.

[144] Yechiam Yemini. Selfish optimization in computer networks. In *Proceedings of the 20th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*, pages 374–379, December 1981.

[145] Yechiam Yemini and L. Kleinrock. On a general rule for access control or silence is golden. In *Proceedings of the International Symposium on Flow Control in Computer Networks*, pages 335–347, February 1979.

[146] Zeng Zeng and Bharadwaj Veeravalli. Design and performance evaluation of queue-and-rate-adjustment dynamic load balancing policies for distributed networks. *IEEE Transactions on Computer*, 55(11):1410–1422, 2006.

[147] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22:1, January 2004.

[148] Dayi Zhou and Virginia Lo. Wave Scheduler: Scheduling for faster turnaround time in peer-to-peer desktop grid systems. In *Proceedings of the 11th Workshop on Job Scheduling Strategies for Parallel Processing*, 2005.

[149] Dayi Zhou and Virginia Lo. WaveGrid: a Scalable fast-turnaround heterogeneous peer-based desktop grid system. In *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing (IPDPS'06)*, 2006.

List of Publications

Journal

- **T. Abdullah**, K.L.M. Bertels, ”Ant Colony Optimization based Self-organization in P2P Ad hoc Grids”, *to be submitted*.

- **T. Abdullah**, K.L.M. Bertels, L.O Alima ”Infrastructure-level Self-organization Mechanisms in Ad Hoc Grids”, *Manuscript in preparation*.

Proceedings

- **T. Abdullah,** K.L.M. Bertels, L.O. Alima, Z. Nawaz, “Effect of Dgree of Neighborhood on resource discovery in Ad Hoc Grids”, 23nd International Conference on Architecture of Computing Systems, February 2010.

- **T. Abdullah,** K.L.M. Bertels, L.O. Alima, “Ant Colony Inspired Microeconomic based Resource Management in Ad Hoc Grids”, 4th International Conference on Grid and Pervasive Computing, Geneva, May 2009.

- **T. Abdullah,** L.O. Alima, V. Sokolov, D Calomme, K.L.M. Bertels, “Hybrid Resource Discovery Mechanism in Ad Hoc Grid Using Structured Overlay”, 22nd International Conference on Architecture of Computing Systems, March 2009.

- **T. Abdullah**, L. Mhamdi, B Pourebrahimi, K.L.M. Bertels, "Resource Discovery with Dynamic Matchmakers in Ad Hoc Grid", The Fourth International Conference on Systems, March 2009.

- **T. Abdullah**, V. Sokolov, B Pourebrahimi, K.L.M. Bertels, "Self-Organizing Dynamic Ad Hoc Grids", In Proceedings of 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW 2008), Venice, October 2008.

ProceedingsLocal

- **T. Abdullah,** K.L.M. Bertels, "Agent based Local Ad Hoc Grids", proceedings of PRORISC, Veldhoven, The Netherlands, November 2007.

- **T. Abdullah,** K.L.M. Bertels, S. Vassiliadis, "Adaptive Agent-based resource management for GRID", 12th ASCI Conference, pp. 420-428, Lommel, Belgium, June 2006.

- **T. Abdullah,** K.L.M. Bertels, S. Vassiliadis, "Economy Based Self Organizing Grids", Architectures and Compilers for Embedded Systems (ACES), Edegem, Belgium, October 2006 (poster presentation).

Non-related Publications

- **T. Abdullah,** K.L.M. Bertels, "Agent Toolkits for Ad Hoc Grids", 32nd International Conference on Artificial Intelligence Workshops, Paderborn, September 2009.

Samenvatting

IN dit proefschrift presenteren we zelf-organisatie mechanismen voor het beheer van resources in ad-hoc grid-netwerken. Hiermee maken we de bouw en de inzet mogelijk van ad-hoc grid-netwerken die zichzelf kunnen aanpassen om een betere toewijzing/aanwending van resources te leveren onder dynamische omstandigheden. De motivatie voor deze studie vloeit voort uit de wens om beschikbare, maar ongebruikte computational resources in netwerkomgevingen aan te wenden. Dergelijke netwerkomgevingen kunnen we aantreffen in academische instituten, kantoren of de personal computers in onze huizen. De resources die toegevoegd zijn aan een ad hoc netwerk zijn vluchtig, onbetrouwbaar en niet-gereserveerd. Bovendien dragen de resource-eigenaars deze resources bij volgens hun eigen gebruiks-/ toegangsbeleid. Dit fenomeen creëert de noodzaak de mechanismen te bestuderen, waarmee ad hoc grid-netwerken hun infrastructuur op een autonome manier wijzigen in overeenstemming met de wisselende resource beschikbaarheids- en/of vraag-patronen. In het proefschrift wordt de infrastructuur gedefinieerd als de mechanismen die nodig zijn om de nodige resources toe te wijzen aan de werkers.

De zelf-organisatie mechanismen op infrastructuur-niveau voor het beheer van resources stellen het ad hoc grid-netwerk in staat om zich aan te passen ten einde de beste toewijzing van resources te bieden onder wisselende omstandigheden. De zelf-organisatie op infrastructuur-niveau in een ad hoc grid-netwerk ontstaat uit zelf-organisatie op twee niveaus; op het niveau van de individuele node en op systeem niveau. Een individuele node in een ad hoc grid-netwerk kan een consument of een producent van resources zijn. De zelf-organisatie op het niveau van de node stelt een individuele node in staat het nut van het ad hoc grid-netwerk in termen van taakuitvoering of resource-consumptie te maximaliseren. De zelf-organisatie op systeem niveau bestaat uit het begrijpen onder welke voorwaarden / omstandigheden een volledig gecentraliseerde of een volledig gedecentraliseerde infrastructuur (of iets daar tussenin) het meest geschikt is.

We bestuderen en vergelijken de mechanismen op twee niveaus: op het niveau van de toewijzing van resources en het niveau van het overlay-netwerk. Met betrekking tot de toewijzing van resources stellen we een op-de-markt-gebaseerde Continuous Double Auction (CDA)(doorlopende dubbele veiling) aanpak en een aangepaste Ant Colony Optimization (ACO) (mierenkolonie optimalisatie) aanpak voor. We stellen ook een uitbreiding voor van het Pastry overlay-netwerk, die nodig is om ad hoc grid-netwerken te (de)segmenteren op basis van verschillende beschikbaarheid van resources en takenpakketten.

In de op-de-markt-gebaseerde benadering van CDA maken de individuele consument- / producent-nodes gebruik van de prijs van een computational resource als een indicator van de beschikbaarheid/schaarste van resources. De consument- / producent-nodes organiseren zichzelf door hun resourceprijs te verhogen / verlagen op basis van hun nut. De brontoewijzer (hierna matchmaker (koppelaar) genoemd) gebruikt CDA als koppelingsmechanisme en past segmentatie en desegmentatie van het ad hoc grid-netwerk toe om zelforganisatie op systeem niveau te bereiken. Het segmentatie- en desegmentatieproces ontstaan door de promotie en degradatie van de matchmakers. De promotie en degradatie geschied op basis van de over- of onderbelasting van de matchmaker. Een overbelaste matchmaker deelt haar overtollige werklast met een nieuwe matchmaker door een normale node te bevorderen tot matchmaker. Op deze manier wordt het ad hoc grid-netwerk gesegmenteerd wanneer nieuwe matchmakers worden ingevoegd, en de segmenten van het ad-hoc grid-netwerk worden weer samengevoegd wanneer de matchmakers worden gedegradeerd. We hebben ook het gebruik van CDA onderzocht onder verschillende niveaus van gecentraliseerde besturing. Één uitbreiding was het geval van P2P, waar elke node zijn eigen matchmaker is. We bestudeerden het effect van het variëren van de reikwijdte van de neighborhood (buurt).

Bovendien is een tweede zelforganisatiemechanisme op infrastructuur niveau bestudeerd, namelijk de aangepaste ACO aanpak. De gewijzigde ACO aanpak hielp bij de verwezenlijking van de zelf-organisatie op systeem-niveau in de aanwezigheid van de consument- / producent-nodes van verschillende soorten resources. De gewijzigde ACO aanpak hielp tevens bij het dynamisch vormen van de virtuele resource-segmenten van het ad hoc grid-netwerk.

De belangrijkste bijdrage van dit werk is dat het mechanismen voor zelforganisatie in ad-hoc netwerken presenteert. Deze mechanismen zijn geimplementeerd op een gestructureerd overlay netwerk met behulp van de algoritmes voorgesteld in dit werk. Deze mechanismen zijn bestudeerd onder verschillende netwerk-omstandigheden. Deze mechanismen hielpen bij het bereiken van een schaalbare, dynamische en een zelf-organiserende ad hoc grid-netwerk infrastructuur. De studie van deze mechanismen identificeerde de scenario's voor het bepalen van de afwegingen omtrent verschillende ad hoc grid-netwerk infrastucturen.

Acknowledgments

I consider my PhD a big achievement in life, and would like to mention some people who shared the burden in making it happen through their help, guidance and encouragement during these years. First and foremost, I would like to thank my supervisor, Dr. Koen Bertels. This work would not have been possible without his advice and continuous encouragement. He has provided me with enormous freedom to pursue my own research interests, and at the same time guided my thinking on the basic research problems. His support has gone well beyond the academics over these years. Special thanks goes to Dr. Luc Onana Alima for his ideas and discussions. I would also like to thank Dr. Lotfi Mhamdi for his encouragement and positive thinking that helped me navigate the odds over the years. I sincerely thank my promoter Prof. dr.ir. H.J. Sips for his help at the final stages of my PhD. I would like to extend my thanks to my PhD committee members and especially to Dr. D.H.J. Epema and Dr. Luc Onana Alima for their detailed comments on my thesis.

I also found it very enjoyable to work with and talk to every member of the Computer Engineering (CE) Laboratory. I would like to thank Dr. Behnaz Pourebrahimi, my former colleague in this project, for all the help in the early period of my PhD and the long fruitful discussions. I express my sincere thanks to my officemate Sandra Irobi for always welcoming my interruptions, often non technical, but sometimes technical, during her work. I also remember my M.Sc student Vassiliy Sokolov who joined this project and did an excellent work. I also express my sincere thanks to Zubair Nawaz for providing an excellent company over the years and always being available to help. I am thankful to Roel Meeuws for translating synopsis and propositions into Dutch and Faisal Nadeem for proof-reading my thesis. My thanks also goes to all the staff of CE Laboratory namely, Bert, Erik and Eef for their technical support on network related issues, and to Lidwina and Monique for their administrative assistance throughtout these years. I am also indebted to the small Pakistani students community in The Netherlands and especially from Delft, that helped me overcome regular homesickness. Meeting and having so many special persons around me made me feel very lucky, and indebted to do much better.

Finally, I would like to thank my family for all their support. I am grateful to my parents for being my best friends, loving and trusting me unconditionally. Their devotion and sacrifices cannot be compensated by anything. I sincerely thank my brothers and sister for all their kindness, support and encouragement. A special thanks goes to my wife and kids

for their understanding and support over the years. My wife and especially my sons, Faran and Irfan, suffered a stressed, busy and sometimes grumpy husband and dad without complaints during my PhD studies. My sons, you should know that no matter how tired I had been, charming smiles; and loving looks from you cheered me up, made me forget the problems, and motivated me to go on. Without your love and support, this thesis would not have been possible. This is for you...!

Tariq Abdullah
Delft, The Netherlands, 2010

Curriculum Vitae

Tariq Abdullah was born on December 30, 1977 in Lawa, Pakistan. After graduating from University of Punjab, Lahore in 1997 he joined Department of Computer Science, IIUI for his M.Sc in Computer Science. From April 2000 to June 2001 he worked as a freelancer software developer. In July 2001 he joind the Department of Computer Science, IIUI as a System engineer. He was involved in software development projects and the supervision of undergraduate and graduate final term projects. In September 2005, he joined the Computer Engineering Laboratory, EEMCS faculty at Delft University of Technology for pursuing his Ph.D in the area of computer engineering.

He worked in Computer Engineering Laboratoy under the supervision of Dr. Koen Bertels. He worked on GRAPPA project, the results of which are described in this dissertation.