Decentralised Economic Resource Allocation For Computational Grids

by

Simon Mark Davy

Submitted in accordance with the requirements for the degree of Doctor of Philosophy.

The University of Leeds
School of Computing

November 2008

The candidate confirms that the work submitted is his own and that the appropriate credit has been given where reference has been made to the work of others.
This copy has been supplied with the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

Abstract

Grid computing is the concept of harnessing the power of many computational resources in a transparent manner. It is currently an active research area, with significant challenges due to the scale and level of heterogeneity involved. One of the key challenges in implementing grid systems is resource allocation. Currently, centralised approaches are employed that have limited scalability and reliability, which is a key factor in achieving a usable grid system.

The field of economics is the study of allocating scarce resources using economic mechanisms. Such systems can be highly scalable, robust and adaptive and as such are a potential solution to the grid allocation problem. There is also a natural fit of the economic allocation metaphor to grid systems, given the diversity of autonomy of grid resources.

We propose that an economic system is a suitable mechanism for grid resource allocation. We propose a simple market mechanism to explore this idea. Our system is a fully decentralised economic allocation scheme, which aims to achieve a high degree of scalability and reliability, and easily allows resources to retain their autonomy.

We implement a simulation of a grid system to analyse this system, and explore its performance and scalability, with a comparison to existing systems. We use a network to facilitate communication between participating agents, and we pay particular attention to the topology of the network between participating agents, examining the effects of different topologies on the performance of the system.

Acknowledgements

I'd like to thank my supervisors Karim Djemame and Jason Noble for all their time, effort and knowledge. I'd like to thank the Biosystems Research Group at the School of Computing for much insightful discussion and enjoyable comradeship. For supporting me through the process of completing this thesis, I'd like to thank my wonderful wife.

Contents

Chapter 1

Introduction

1.1 Motivation

1.1.1 Computational Grids

The area of "grid computing", or more recently "cloud computing" is an ongoing research area looking at computing resource provision. It denotes a concept of computers as a utility, where computational resource can be provided as-and-when needed transparently to large and diverse numbers of users. It is similar in concept to other utilities, such as water or electricity, indeed the grid name owes in part to its similarity in concept to the National Grid. An ideal grid system would allow users to utilise massive computational resources as easily and as transparently as we plug something into a power socket at home and turn it on.

While allowing different people to share computing resources is nothing new, the grid takes the concept to a new level in several ways. Firstly, the sheer scale is much larger. Previously, a few hundred machines, or at the most a few thousand, have been harnessed together in some sort of grid-like system. The grid concept takes these numbers into the tens of thousand, or even millions. Secondly, and more importantly, it relies on the sharing of multiple resources owned and administered by different organisations or individuals. In previous resource sharing systems, the resources are all owned by a single entity. Grid computing, however, aims to allow many resource-owners to participate in sharing their resources together in a single unified manner. This is the ultimate aim of grid computing,

and presents some major research challenges. Arguably the most significant of these challenges, and the one with the fewest practical solutions, is resource allocation.

In a grid environment, a resource is some computational facility which users wish to utilise for their applications and programs. The challenges of deciding which users run which applications on whose resources are significant, especially in the face of huge numbers of resources and different administrative domains. Additionally, to achieve the full potential of grid computing, resource allocation needs to be done in a fast, automatic, robust and scalable manner.

The current dominant approach is to have a central point that maintains the state of all resources on the grid, and provides ways of selecting suitable resources to use. This is the historical approach to allocating computing resources, as most computational systems are centralised in nature. However, this method is not particularly scalable or robust, at least in current implementations, especially when you start to scale to thousands of resources. Current resource allocations systems do not deliver the necessary resource allocation capabilities necessary to realise the grid concept.

1.1.2 Economic Resource Allocation

The problem of allocating scarce resources is one that has been ever-present in human history. A resource is scarce if there is not enough supply of that resource to meet the corresponding demand for at a price of 0. The raw method used by nature for allocating such resources is through competition and survival. Humans have developed more sophisticated systems for facilitating this allocation, which is the subject of economics.

Economics provides a system for allocating scarce resources between different parties using understood mechanisms to swap (trade) one resource for another, usually money in exchange for some other resource. Many varying and complex economic systems have arisen over the centuries of human history to facilitate the exchange of scarce goods, and in recent centuries, these system have been much examined in their function and performance. Economic systems are recognised to be adaptive to change, highly scalable and robust to the failure of individual components. They have evolved to provide highly efficient methods of resource distribution at a large scale.

The property of economic systems that gives rise to these factors is that they are generally decentralised at the base level, except in extreme cases (such as governmental regulation). Whilst many existing economic systems utilise a central mechanism for efficiency, base economic systems developed from individuals acting independent of any central imperative. This decentralisation is present in two distinct ways. Firstly, individ-

ual participants in an economic system, that is, the traders themselves, are self-interested. The system simply provides them with an incentive to trade. There is no mandated co-operation, other than conforming to the common trading protocol, each individual agent is its own master, and utilises its own resources to participate in the trading. Secondly, no single trader has a complete picture of the whole system. Traders have knowledge of their own specific needs or resources, which are generally known only to themselves, although other traders may have some indication of the resources or needs of other traders in a more general sense. Additionally, traders usually only participate in a subset of all the trade interactions occurring in a given economic system, and thus have only a limited knowledge of others' trades. Yet despite these limitations, the interactions of many traders give rise to a global system behaviour that has the useful properties or adaptivity, scalability and robustness. These are also important properties of a grid environment.

1.1.3 An Economic Grid

There is a clear parallel between the needs of a resource allocation mechanism for grid systems and the economic resource allocation paradigm. The grid is envisioned as a large system, so any allocation mechanisms used must be highly scalable. Because of the large size, the number of individual components that fail in such a system will also be large, so a suitable mechanism must also be robust to these failures.

The economic paradigm is also well suited to the problem of multiple autonomous domains, as each resource can act as a self-interested agent in an intuitive manner. Each autonomous domain will have specific knowledge of its own needs and resources, and along with an inferred knowledge of other traders situations, can effectively reason on suitable trade prices.

A potential additional benefit of using an economic system for allocation is that it is also a revenue generating business model for the grid. This is an active research area in itself, but an economic system could combine the two. In theory, an economic system could be used for allocation and a different system for the business model. However, for an economic model to work, the currency used in trade prices would need to have real-world value or be limited in some other manner, or else the trader's resources would in effect be unlimited.

An economic allocation system would be very different from most of the current approaches to grid resource allocation, and represents an orthogonal approach compared to the current methods. If such a system could be implemented, it could provide the necessary transparency and scalability needed to implement a working grid system that

embodies the aims of the grid concept.

1.2 Objectives

This thesis explores the potential for a fully decentralised economic resource allocation scheme for a computational grid. Whilst economic allocation schemes for computational resources and grid systems have been developed previously, they are all centralised in nature, thus losing the potential scalability benefits of using such systems.

We also look to apply the work on simple trading agents in the literature to autonomous grid trading agents, something that has not been done in other economic grid allocation systems, where novel systems have been used.

We also apply network theory to our economic model. In most computational economic systems, communication between traders is assumed to be either all-to-all or random in nature. We look at different topologies for communication between traders.

1.3 Investigation Methodology

In order to investigate the range of situations desired, and to have full control over the environment, we use simulation to explore our economic approach for grid allocation. Real world grid systems are difficult to gain access to, as well as not being of sufficient size to investigate scalability, and would require elevated privileges to allow the implementation of a resource broker.

1.4 Thesis Contribution

The contributions of this thesis are as follows.

- We provide a detailed description of existing economic allocation mechanisms, and examine their suitability for use in a grid environment.
- We propose and investigate a novel decentralised economic allocation mechanism for grid allocation, and show that it is both feasible and has the desired scalability characteristics that economic systems can provide.
- We examine the effect of topological features on the network used by our decentralised mechanism, including a novel method of generating socially inspired topologies, and show that certain topologies have potential benefits.

- We compare our decentralised economic system with the current approach to grid allocation, and show that our system performs better at larger network sizes.

1.5 Thesis Overview

In Chapter 2 we review the literature from the field of grid computing, including the history and motivation of the grid concept, as well as a critique of the current solutions to resource allocation.

In Chapter 3, we review basic economic theory and examine previous work in modelling economic systems computationally, as well as existing work applying economics principles to computational resource management. We also take a brief look at the field of statistical network theory, summarising a variety of network types and their properties.

Chapter 4 outlines our proposed investigation, including a basic model of a grid system, and presents a simple economic model for market system and trader rationale.

In Chapter 5 we present the experimental details and initial findings based on the proposed solution in the previous chapter.

In Chapter 6 we compare our system with the current centralised approaches to allocation, and examine its ability to adapt to dynamic markets a variety of conditions.

The final chapter, Chapter 7, concludes the study, summarising the thesis contribution and suggests areas for future work.

Chapter 2

Resource Allocation on Grid Systems

2.1 The Grid Concept

Since the invention of the computer, there has been steadily increasing demand for more computational performance and resources. Initially this was met by the development of faster machines and more efficient software. The continued demand for more performance led to the use of many faster machines in parallel, and now we have massively parallel clusters of commodity hardware, with a recent move towards multiple processors on a single chip. And while the average desktop computer today has sufficient performance to handle the demand of a typical user, there are still many users that require more computational power. A current example of this is the Large Hadron Collider project [96] at CERN which has recently come on-line, and is expected to produce over 15 Peta-bytes of data per year [17]. The sheer quantity of data that will be generated is beyond both the storage and computational resources of any one institution.

To meet these needs, organisations often build large high performance computing (HPC) systems, which are capable of much higher computational performance. However, the cost of these HPC resources is significant. They are increasingly in demand, but the hardware is expensive to buy, maintain and run. Initial hardware costs can be millions of pounds, and the recurring costs, such as power, cooling, repairs, parts, and floor space can add up to a non-negligible monthly sum. And the larger the resource, the more skilled staff are needed to keep it running smoothing, adding significant salary costs. This means

that only those with sufficient resources (i.e. large companies or institutions) can feasibly make use of HPC resources.

These high costs lead to organisations looking to utilise their existing infrastructure as much as possible. Many make use of the spare computation available on idle resources, such as utilising staff desktops overnight. This also has an administration overhead, but can be more cost-effective than a purpose built HPC resource.

If computational resources were able to be utilised, and made available to users as a utility service, the demand for performance could be better met. For users of computational resource, it would allow them to pay a marginal cost for the use of those resources, and the owners of the resource would recoup the investment and operation costs, including deprecation costs.

On one hand, it would lower the barrier of entry for everyone, allowing modestly resourced organisations to access previously unavailable computing power. On the other, it would allow multiple HPC resources to be used simultaneously, which would allow for very large problems to be tackled, that otherwise would be computationally infeasible. It would also allow new kinds of services to be built upon previously unavailable on-demand access to HPC resources.

Also, it would help mitigate the costs of providing the resources, as they could potentially gain a better return on their investment and increased resource utilisation. Office machines could be made available overnight, providing a large pool of cheap computational power.

This is the idea behind the grid, providing high levels of computational resources as a utility for many. It has also been called Utility Computing or Autonomic Computing, but they are synonymous terms. Recently, the term Cloud Computing has come into popular usage, and indicates the use of some of these grid technologies to provide a specific type of general computational resource.

The idea of a grid is a relatively recent concept, even in the fast moving world of computer technology. It draws its inspiration from previous 'grids' that have developed, such as transport, telecommunication and power grids. These 'grids' supply affordable, scalable and powerful resources to the whole of modern day society. The resource 'the grid' will supply is mainly computational power, but also storage, network and specific application resources.

There will be other uses that are currently beyond our conception and vision. Governments, corporations, academic institutions and organisations will all be able to make productive use of grid technology. These capabilities elevate the grid to an important research area, and it has received much attention over the last decade.

2.2 History of Grid Research

The term "The Grid" was first introduced in a book by Foster and Kesselman [52], which defined a grid as "a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities". Foster and Kesselman's book was quickly followed by Foster's *Anatomy Of The Grid* [54], which further defined the grid concept in more detail.

However, there had been much work prior to this by various groups working on various utility computing projects, and the grid concept had emerged out of all these projects.

2.2.1 Early Work

An early effort towards grid systems was Condor [8], based at the University of Wisconsin. It was designed to make use of idle workstations to run independent jobs. If a workstation ceased to be idle, the job would be 'frozen' and migrated to another machine to continue. It was the earliest example of a fault-tolerant allocation algorithm (see Section 2.6), and made use of the ClassAd [98] system for matching jobs to resources. It was modernised to some degree with current grid concepts as Condor/G [55], but is not under active development anymore.

Legion [18] was started in 1993, and based on an Object Oriented view of the grid system. It presented the grid as a giant virtual machine to the user with which to execute jobs. Originally based at the University of Virginia, it was commercialised via Avaki Corporation as a complete grid infrastructure, and has had some success, being bought out by database providers Sybase.

The Unicore [101] project started in 1997 and was developed to provide access to HPC resources all across Germany. Like Legion, it is a complete grid implementation in JAVA, providing access through a single gateway, hiding much of the complexity from the user.

The leading grid implementation is the Globus [91] project, a collaboration of many organisations. It provides a toolkit to implement many of the low-level grid services, such as security, but leaves a lot of room for development of higher level services and applications. Originally released in 1998 [51], the Globus Toolkit [50] is now at Version 4 and counting, and has been adopted as the defacto standard grid implementation.

The key innovations demonstrated by these applications are that of use of multiple disparate HPC facilities, with different architectures and locations, connected using current networking technologies, as well as the automatic allocation of resources on these machines. These were the early attempts at implementing grid systems.

2.2.2 The Open Grid Forum

The multiple grid-orientated projects around the world prompted the need for standards and specifications for grid systems. The Global Grid Forum (GGF) [49] formed in 1999 to be a focal point for the emerging grid concepts and technologies, and a place for these different groups to work together on defining and building grid systems. The project has since re-branded itself as the Open Grid Forum (OGF).

One of the initial key proposals that came out of the OGF was the Open Grid Services Architecture (OGSA) [53], and its underlying infrastructure, the Open Grid Services Infrastructure (OGSI) [108]. This was an attempt to draw up an overall pattern for grid development. However, it revealed that key interoperability technologies were needed in order to implement grid systems.

In the web world the idea of Web Services (WS) [31] was gaining interest as an interoperability solution. Based on the Simple Object Access Protocol (SOAP [30]) protocol, WS (in theory) allowed remote procedure call between any systems supporting the WS standards.

The OGF decided to utilise the WS standards as the solution to their interoperability challengers and in 2002 refactored the OGSA to use WS [32], and released as the Web Service Resource Framework (WSRF) [33]. It defined the standard services a grid would need to function, and the interfaces between them. Additionally, the basic WS standards alone were not enough to implement the level of interaction needed for grid computing. Particularly the lack of *stateful* resource access was a critical limitation. The OGF continues to work in partnership with the W3C [29] to add stateful Web Services which would allow Web Services to be a standard way to implement the WSRF services need for a grid system. All of the major grid middleware applications are working towards providing the services described by OGSA via the WSRF standards.

2.3 Grid Challenges

The grid introduces a whole new level of diversity to the problem of HPC resource allocation. The level of heterogeneity in capacity, type, location, and ownership is much greater than in previous systems.

In order for this to be used on a large scale, any grid system would have to provide a service that was transparent, reliable, and affordable, like any other utility. A common analogy for the concept is the UK's National Power Grid. A user does not care which power station their electricity comes from, or the technology of the generator, or which

overhead lines it travelled down, nor that it needs to be transformed before being supplied to their house. They are simply concerned that it's there when they plug in and switch on. The generation and supply of the power is transparent to them. If they had to select and negotiate all the above factors, or if the supply was erratic and unreliable, or overly expensive, they may seek other sources of power.

An additional factor present in grid computing that our National Grid analogy does not capture is that of security. As information is now the resource being supplied, a grid system would need to ensure secure delivery and receipt of the workload and its results, whatever they may be. Individuals, both users and suppliers, would need to be authenticated and authorised, which in itself is no easy task.

The grid is effectively a new model of computing, and as such there are many challenges to overcome, in both understanding and implementation. Some of the challenges include:

- *Applications* : Currently applications are designed mainly around single systems. Many of the design methodologies, tools and environments are based on sequential systems. While there has been much work done on parallel software, it remains a specialist field. If grid enabled applications are going to be developed, then parallel programming models, tools and environments are going to need greater development.

- *Systems* : The level of distribution and communication on a grid would be much more than current architectures handle, and the routers and load balancing systems that control a grid will need to be able to handle this dramatically increased load. Networking protocols and infrastructure will also need much development if they are to be able to deliver the transparent access required.

- *Algorithms* : Problem solving algorithms that utilise the distributed nature of the grid will need to be developed in order to make the most of the grid environment.

- *Resources* : Given the multitude and variety of tasks that will be run on a grid system, resource handling is a key issue. The sort of resource management that will need to be done is different in scale to the low-level management technologies so far, and much work will need to be done in this area, particularly in large scale co-allocation and reservation, due to the autonomy and scale of the grid.

- *Security* : With the orders of magnitude increase in data transmission that a grid will bring, security is of even more importance than ever before, and is a significant

challenge. Given the diverse ownership of grid resources, this poses an even bigger problem.

- *End user systems* : Both the hardware and software design and implementation of end user systems will have to change in order to adapt them to be part of a grid. Current designs are based on being a single entity, and much work needs to be done in grid enabling these systems.

2.4 Grid Applications

A grid will enable many different types of application to run on top of it. It is these applications that provide the useful information and services that are the reason for the existence of the grid in the first place.

2.4.1 Types of Application

A brief synopsis of the various general types of applications that will be run on a grid system, is given in [52]. It is summarised and illustrated here.

- *Distributed super-computing* - also known as Grand Challenge applications, these solve very large or intensive problems that require either many processors (for speed) and/or vast amounts of memory (for large problem sizes). Many of the existing grid systems and their applications are based on solving these kinds of problems.

- *High throughput* - utilising idle systems to increase the aggregate computation or IO throughput of an existing application. For high computational throughput, an example would be trivially parallelisable job like micro-processor chip verification at idle engineers workstations. Other applications, like web servers and financial trading servers, require large IO throughput, and struggle to deal with spikes in data traffic.

- *On demand* - applications which dynamically allocate specific resources for a particular period upon request, for a specific purpose. Good examples are medical procedures that can utilise real-time image processing through on demand allocation of computational resources.

- *Data intensive* - processing vast amounts of data and synthesising new information. The processing of high energy physics experiment results requires petabytes of data

to be processed. Storing, transporting and managing these levels of data will require new applications.

- *Collaborative* - applications that allow collaborative working between multiple distributed parties. This includes the virtual laboratory concept.

Some applications may be of more than one type, but these are the various uses of grid systems that we can currently see. Again, others may emerge as the field matures. An example of a real grid application is the DAME project [110], a collaboration between Rolls Royce and the University of Leeds to build an online decision support system for aircraft maintenance. Engine data recorded during flight is downloaded and sent to the grid-based DAME application for analysis when the aircraft lands. This process uses available grid computing resources to complete the analysis and report the results back to the waiting engineers at the aircraft. The grid enables enough computing resource to be available on demand that an analysis can take place in a short enough time for the engineers to fix any potential problems before the aircraft is due to take off again. On conventional resources, the results of the analysis would take too long to perform and the aircraft will have taken off on its next flight before the results would be available.

2.4.2 Types of Grid Systems

The actual purpose and function of specific grid systems may vary widely depending on their creators' desires. One grid may not function the same way or for the same reason as another. Krauter *et al* [69] classify potential grid systems into three main categories, depending on function and purpose.

- *Computational Grids* - grid systems that have higher computational capacity for a single application than any component machine. This will replace the current stand alone super-computer idea, and allow extremely demanding applications to be run in practical time scales and with much larger problem sizes than previously possible.

- *Data Grid* - an infrastructure for storing, analysing and synthesising new information from data repositories. Similar in nature to a computational grid, but its focus is specifically on the storage and management of huge amounts of data.

- *Service Grid* - these are grid systems that provide computing services that are not available from a single machine. It provides an 'on demand' service to the user, such as interactive multimedia and collaborative conferencing.

All three categories of grid system will use the same underlying network and computing technologies, but the way in which resources are managed and allocated will be very different, and the intended usage and user groups are also diverse. For computational and data grids, it is unlikely that many end-user consumers will use their facilities directly. Rather, a service grid will allow for large scale use in the consumer market, especially the multimedia possibilities. It may be that other types of grid system emerge in the years ahead - it is still a new and largely undeveloped area.

Another distinction in grid types is less technical and more social in origin. The grid concept has largely been developed in academia, which is an open and public environment. The typical applications for HPC in academia are open and freely available anyway, so data security in not a big concern. In industry, however, this is often not the case. Companies can have large amounts of proprietary data that they wish to keep private, and as they would not own the resource that would execute their application, would need to be able to guarantee their data's integrity. Any disgruntled local system administrator could potentially spy on the applications execution. This leads to the possibility of "closed grids", or "virtual private grids" (similar to virtual private networks in networking), which are hired wholesale (or in isolated sections) from a trusted provider.

2.5 Architecture

The architecture of a grid is decomposed into distinct layers, similar to the current networking protocols on which they are based. Figure 2.1 is taken from [54] and describes a high level view of the current grid architecture.

We examine each of these layers in turn, starting from the bottom up. This grid 'layer' model bears much similarity to the original OSI reference model [89] for the internet.

2.5.1 Grid Fabric Layer

The elements at this layer comprise of the actual resources of the grid system, as well as the low-level interface to the resources themselves. Given their heterogeneous nature and differing administration policies, this is not a simple task.

The available resources can be broken down into three main types of resource.

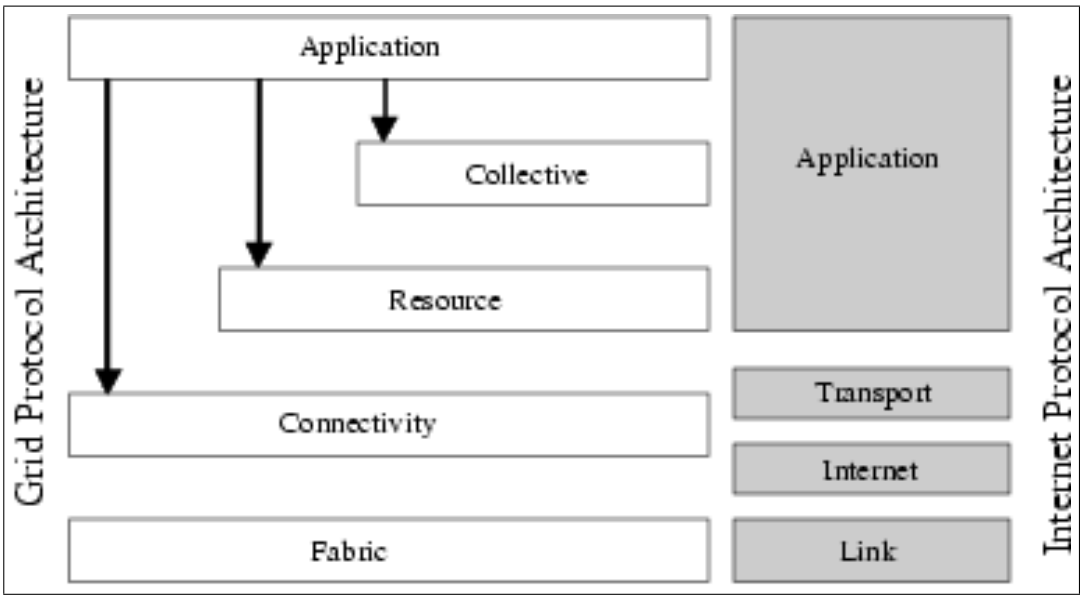

Figure 2.1: Grid architecture protocols with corresponding internet protocols (from Foster *et al* [54])

2.5.1.1 Computational Resources

These are the actual computers that do the work. In [52] four basic system types are suggested.

- *End user systems* : These are common computer devices, such as desktop machines and appliances. They are designed as a single functional entity with hardware and software working together in a homogeneous environment.
- *Clusters* : A cluster is a network of systems that is specifically designed to be used as a single high powered computational resource. Like an end user system, clusters are most often highly homogeneous and have a single controlling entity through which resource requests are made. They have arisen as a more affordable, scalable and possibly more powerful alternative to traditional integrated supercomputers.
- *Intranets* : These are large local networks belonging to a single organisation. They are generally heterogeneous in nature, with a large variety of resources available. Different parts of the system may be under different administration and there is less global knowledge about the system than in either a cluster or end user system.
- *Extranets* : These are large networks spanning multiple organisations. These are even more heterogeneous in nature than intranets, and have even less global knowl-

edge available. They often are much more geographically distributed also, increasing latency and other communication issues.

These are the systems that a grid is composed from, whether dedicated to grid usage or used when idle, and will provide the actual computational resource that the grid offers.

2.5.1.2 Storage Resources

These are dedicated storage machines for holding large amounts of data. This could be simply large file-systems, or complex databases for querying or updating. They make the data they manage available to other resources on the grid in a consistent manner. Given the vast quantities of data that a super-computing application can produce (in the order of GB/s), managing and sharing this data is a key objective of grid systems.

2.5.1.3 Network Resources

A network is made up of cabling and routers, but the resources it provides are measured in bandwidth and latency. Given different grid applications needs, grid network resources will need to be able to provide Quality of Service (QoS) as part of the grid networking infrastructure. It will be necessary to be able to reserve and allocate network resources such as bandwidth or latency on a whole variety of different network structures and capacities.

2.5.2 Grid Connectivity Layer

This layer is responsible for all aspects of actual connection between grid components. As well as the actual data and message transmission, it is responsible for the authentication of user and request, i.e. that they are allowed to use a particular resource. It is also responsible for security of transmission and the integration with diverse security policies and procedures on different resources. It must also track usage and keep account of resources used for each application, for determining costs incurred. The development of this layer is very much linked with the development of Internet technologies such as IPv6 and other QoS mechanisms.

2.5.3 Grid Resource Layer

This layer controls the reservation and allocation of a single resource, as well as taking responsibility for the execution of jobs on that resource. It is unconcerned with anything other than the resource management, and is unaware of the global state of the grid. It

provides information on the state, load, usage and free capacity of its own resources. It also allows direct management of applications executing on its resources, such as process creation, monitoring and reporting.

2.5.4 Grid Collective Layer

This is the layer that pulls all the resources together coherently. It allows access to and maintenance of a grid wide information service that can provide the state of any resource on the grid. This allows it to co-reserve and co-allocate resources across the whole grid. It is concerned with the concurrent management of all the resources available to the grid.

2.5.5 Grid Application Layer

This is similar to the internet application layer at the highest level. These are the applications written by users to run on grid systems. They use the other grid layer protocols to acquire and use the resources they need. An 'application' may actually be a grid 'portal' or entrance for another specific type of application.

2.5.6 Gird Architecture Summary

In this section we have detailed the high level architecture of grid systems. This work is focused on the Collective layer, where the details of the resource allocation are worked out. In the next section we look more closely at the resource allocation mechanisms that function at this layer.

2.6 Resource Discovery and Allocation

Resource allocation of multiple resources is an established work in the HPC field. HPC is not new, and many successful systems and technologies have been developed to manage the usage of resources. Typically, a HPC resource has been a single multi-processor parallel machine, or a cluster of many smaller machines. They are usually owned by a single organisation, although there may be many different groups of users within the larger body, and are usually based at a single location, rather than spread between several sites. However, grid systems break some of the underlying assumptions, so new techniques are needed.

Grid scheduling has seen much development recently, and we discuss current approaches below, but for a more in depth review see [82]. In current grid implementations,

resource allocation is broken down into two main steps, resource discovery, and resource selection or allocation. Resource discovery is the process of knowing what resources are available for use, as well as their current load and state, whereas resource allocation in this sense means the actual determination of which resource to execute an application on.

2.6.1 Resource Discovery

The dominant resource discovery provider is the Globus Toolkit's Monitoring and Discovery Service Directory (MDS) [34] which has been developed since the projects inception, and has undergone several major revisions as the OGF specifications have developed. It supplies a grid directory service and a protocol for querying the service for information about a resource or set of resources, in order to aid a user or application to select suitable resources to use. The directory maintains itself by both allowing registered resources to push information to the service and regularly pulling this information from the resources. Entries in a directory have a limited lifespan and must be renewed periodically or be removed from the directory. The design facilitates multiple directory servers that aggregate resource information, as a resource could be registered on multiple independent directories. This is aimed at increasing scalability and robustness, which has been a limitation and subject of investigation [64].

There is an inherent trade-off between the accuracy of information in a directory about a resource's state, and the number of resources registered. The more frequent the updating of state to the directories, the more accurate the grid information will be, but the greater the demand on the directory services. In a typical MDS deployment, of a few hundred resources, a directory's information is out of date by about 5 minutes. This is suited to grid environments where the demand is relatively stable and jobs are long running.

MDS provides the infrastructure on which grid meta schedulers could allocate grid resources, but does not implement any such algorithms itself. Often its left for specific applications or users to make final decisions. It is not a complete allocation mechanism, but provides an important service upon which to build such a mechanism.

2.6.2 Resource Selection/Allocation

Typical grid level allocation and scheduling query a directory service and perform some kind of algorithm on the query results to select the resources to be used.

Many schedulers are design to schedule jobs on local resources under a single domain. Condor-G [55] is a dynamic on-demand scheduler for dedicated and idle resources. Sun's GridEngine [11] uses a sophisticated priority queueing system and supports ad-

vance reservations. The Portable Batch System (PBS) [62] is one of the oldest schedulers around, but has been adapted to suit more modern environments. The most common commercial system is the Load Sharing Facilitator (LSF) [92], which is widely deployed. These system are often used to aggregate a group of heterogeneous resources to present a single resource interface to a larger grid system.

The EZ-Grid project [19] and the Grid Resource Broker [2] gather grid information using a centralised publish/subscribe (pub/sub) model. This is designed to help users make allocation decisions manually, but they do not actually schedule anything automatically.

There are actually very few functional grid-level schedulers. AppLeS (Application-Level Scheduling) [9] can schedule jobs in highly heterogeneous and dynamic environment. The Maui scheduler [93] has good provision for advance reservation and QoS. The Gridway project [95] provides a full scheduling system, with fault tolerance and job migration. It uses the Globus toolkit and focuses on interoperability between different local schedulers.

Of particular relevance to this study is the Nimrod/G [14] project, which is based on an economic paradigm. It is specifically designed for easily parallelisable parameter sweep applications, and is built upon the Globus Toolkit. It allows users to negotiate prices and completion deadlines, including advance reservations. See Section 3.3.2.1 for a more detailed discussion.

2.6.3 Limitations of Centralised Approaches

The current approaches to resource allocation have been developed out of the traditional HPC field, and have facilitated initial grid implementations. They are all centralised in nature, as this is the most straight forward implementation. However, they face some serious challenges as the size and use of grid systems continue to increase.

- Scalability. The chief challenge in building grid systems is size. Once a grid consists of more than a few hundred machines, current grid systems struggle to cope. If the full potential of grid systems is to be achieved, breakthroughs in improving scalability are essential.

- Dynamism. Different grid usage demand levels mean different policies and usage patterns. Current grid systems are not designed to adapt to high degrees of change in demand (and conversely, supply). If resource state is changing more rapidly than a centralised directory can keep up with, the accuracy of information is decreased, leading to incorrect allocation decisions being made. This can lead to decreased

performance of both the allocation system and the grid resources themselves, as more work is needed to successfully allocate jobs, and more load on the allocation system, which can spiral downhill.

- Single Point of Failure. Whilst this has been addressed to some degree by systems such as MDS, any centralised system, even if aggregated, is vulnerable if the central mechanism fails in some way. If it fails completely, the system may be totally unusable. If it's performance starts to degrade, the whole system's performance suffers.

Whilst there is much investment and knowledge in such centralised systems, and much work still to be done based on that paradigm, there are inherent limitations. Therefore it is important to explore decentralised approaches as potential solutions to this problem, as the limitations of centralised systems are the strengths of decentralised systems. And it is specifically these limitations which grid computing is trying to overcome.

2.7 Summary

The grid concept has been the focus of much development in the last 15 years, and has seen much progress. Much work has been done on existing technologies and their application to the grid. Work has also been done in identifying the issues involved with enabling a grid scale computing system.

However, while some of the challenges involved have been solved, others are still open challenges, particularly resource allocation. If the grid concept, as envisioned and stated by its proponents, is to become reality, large advances in the fields of networking transmission speeds, network management, and parallel software development will also be necessary.

However, the potential is clear, and recent "cloud" computing initiatives have drawn it into the main stream. The possibilities that a functioning grid system could provide are stimulating more investment in this area, and increase the need to solve some of these fundamental problems.

Chapter 3

Economics and Market Based Control

3.1 Economics

Economics is the study of the allocation of scarce resources between competing alternate uses. In a free economic market, allocation decision making is distributed amongst the many interested parties, whether individuals and organisations, with no ultimate central control (except in heavily centralised and regulated markets). It is driven by the availability of a resource (supply) and the need for that resource (demand). In particular, we are interested in *microeconomics*, the study of individual traders' behaviour and interactions, and their aggregated effects in a particular market.

We start with a summary of simple microeconomics principles. There is much published work within the field of economics, and we focus specifically on decentralised economic mechanisms. Most real-world and theoretical economic studies are based on some centralised regulation mechanism, and as such are not directly relevant to our study.

The term market is used to describe the interactions of traders trading a particular commodity or resource. The are many different types of markets, selling many different products, but the basic principles of a market are well defined. Here we introduce some key concepts about economic markets, and go on to look at three key components of a market; the traders who act in the market, the rules of the market, and a market place in which the interaction occurs. Much of this next section has been distilled from [86].

3.1.1 Supply, Demand, and Market Equilibrium

The basic model of equilibrating interactions supply and demand is well understood. A larger supply of a resource than demand for it will lead to the price falling as sellers of the resource compete with each other by offering lower prices to make a sale. Likewise, a high demand relative to supply would lead to the price rising as buyers compete to meet their demand. This is known as a competitive market equilibrium.

This key idea in microeconomics occurs when the price in a market for which the quantities supplied and demanded of a resource at that price are equal, and thus all supply of a resource is sold and all demand for that resource is met at that price. Any other price would result in a surplus or shortfall of resource. In an efficient market, the aggregate price of a transaction will tend to hover around this point over time. The buyer does not want to spend any more than necessary to meet their demand, and sellers do not want to waste their own resources to produce resource that is not needed. The further the aggregate trade price is from the equilibrium price, the more inefficient the market.

Note that there is not always an equilibrium point. Factors governing the nature of producing the resource, or other outside influences, can result in markets that have con-

stant shortfall or surplus of resource.

A particular market's supply and demand do not remain constant, and therefore neither does the equilibrium price. As they change, the market price will track the equilibrium price, as this is in the interests of all traders individually. Thus a market can adapt to its changing environment. This is one of the key properties which makes markets interesting for this work.

The rate of change of demand and supply varies in different types of markets. Some are slow and static with well understood supply and demand models, such as traditional resources like stone and timber. Others are highly dynamic and volatile (e.g. stock). This difference in dynamism leads to many different types of market. A less dynamic market can use a longer process to agree on a trade price, for instance, while a highly dynamic market needs to utilise a faster method. See Section 3.1.3 for more examples.

This concept of an "optimal" price for a particular supply/demand is key in evaluating economic systems, particularly when a market is concerned with global allocative efficiency, as is the case in Grid systems.

3.1.2 Market Traders

Traders are the individual agents that participate in a market, either buyer or seller. A trader has a private valuation of the current market price, and participates in a market process by offering a quote to another trader, or accepting a quote from another trader.

In real world markets, traders are people, who make decisions about trading according to some internal rationale. It is this rationality which has been the focus of much study and investigation. How does a trader decide what to bid? What information is available to the trader to make the decision? How does the trader adapt to changes in demand/supply?

Economic traders generally have the following basic characteristics as agents.

- *Self-interested*: A trader is fundamentally selfish and interested in its own utility or profit. While traders can cooperate, they do so only to their own advantage. No rational trader will knowingly trade at a loss.

- *Autonomous*: A trader can make its own trading decisions, without the need for a central authority. Often some sort of independent central authority is utilised, i.e. an auctioneer, but it is not essential.

- *Limited resources*: A trader has basic limitations. They have a finite amount of resource to sell, or money with which to purchase resource.

- *Limited knowledge*: The trader only has a limited view of the world, they are not omniscient. This limited view is formed from two main sources; public information that can be observed from the actions or communications of other traders, or is provided by a central arbiter, and private information, such as their own resources or valuation of a resource.

- *A rationale*

  Traders have some way of using the information that they have to estimate the market price and formulate a price to quote at, and, potentially, when to make a quote. For human traders, this rationale can be complex, and difficult to model. See section 3.2 for a discussion of this rationale.

It is these characteristics that lend the trader metaphor to the grid environment, as there are many similarities between traders and resources and jobs. The main similarity is in their self interest or autonomy, as this has been key challenge to traditional grid mechanism, but it is a base characteristic of economic systems.

3.1.3 Market Rules and Auctions

Any market has a set of well understood rules or conventions by which to communicate quotes between traders and agree on a price to trade at. These can be completely decentralised, such as simple "flea market" bilateral bargaining, or utilise a central arbiter, often computerised in modern markets.

Many forms have arisen over time that have varying characteristics as needed by a particular situation or market. The main recognised auction types, and the way in which they vary, are summarised below. Much of this review is based on [68].

In classical economics, the word "auction" is usually used to describe a specific set of market rules that use a central arbiter. In this work we use it in a more general sense, to mean any set of rules used to trade in market.

There are four 'classical' auction types recognised in the literature, each involving a single seller and multiple buyers, or vice-versa. These are usually single unit auctions (i.e. a seller selling a non-divisible amount of resource, or a buyer buying a non-divisible amount).

3.1.3.1 English Auction

An English auction (or ascending price auction) is one of the most commonly recognised auction mechanisms. It is a public auction with a single seller selling a single resource

and multiple competing buyers. The buyers start quoting low and raise the quote until no other buyer is willing to quote higher. The buyer with the highest quote wins the auction. Often a central independent auctioneer is used to control the quote rises, although the seller could act as the auctioneer.

It is commonly used for very scarce and unique resources such as works of art and large houses. In other words, high demand and low supply.

In general, the dominant strategy for a trader in an English auction is to bid up to your value, as the next to last bidder will hopefully drop out before then, leaving you with a gain in utility (in the case of a buyer, savings).

3.1.3.2 Dutch Auction

A Dutch auction is similar to the English auction in that there is a single seller and multiple buyers, but the quoting is reversed. The seller starts high and then repeatedly lowers the quote until the first buyer accepts the price. It is very fast auction and is utilised in fresh flower markets in Holland (from where it gets its name) where speed of trade is of the essence. The dominant strategy for a Dutch auction is for the seller to start higher than their price and for the buyer to wait until the price meets their price.

3.1.3.3 Sealed-bid Auction

In sealed-bid auctions the traders submit private quotes to a single opposing trader, (e.g. a buyer with many sellers). The best quote (lowest if buying, highest if selling) wins the auction. Government contracts commonly use this form. A variation of this is to make the bidding iterative, with the best bid of each round being communicated to all parties between quoting rounds. Strategically for the trader, this is the same as the Dutch auction, a trader should bid below their true value in the hope of maximising profit if they win.

3.1.3.4 Second Price Sealed Bid Auction

This is identical to the sealed bid auction, and the best quote still wins, but at the price of the next-best quote. It is strategically similar to an English auction, where the best price is not actually quoted, the winner wins when they offer just slightly better than the best price of the second best bidder. So the dominant trader strategy is to bid your true value. However, this kind of auction requires an independent auctioneer, to enforce the second price rule.

A second price sealed bid auction is a single unit specialisation of the general multi-unit Vickery [111] auction. In this kind of auction, with n units for sale, the top n bids

win, but at the price for the highest failed bid, i.e. the $n-1$th best bid.

3.1.3.5 Double Auctions

A double or 2-sided auction is one where both buyers and sellers quote. The previous auctions are 'single' auctions in this sense, as only the buyer or seller quotes, not both. A comprehensive review of the double auction is presented in [56].

Double auctions can be both very simple and highly complex. Simple bilateral bargaining, where a single buyer and seller alternately raise or lower their quote until they meet somewhere in the middle, has been around since civilisation began, as is still used today in many kinds of markets.

Probably one of the most recognised forms of this auction is the continuous double auction (CDA), where many buyers and sellers participate in quoting prices simultaneously.

The classic example of this is the open outcry auctions used by the stock exchange trading pits. Such auctions occur in a shared common space, where a trader will shout out his quote, while simultaneously listening to quotes from both other buyers and sellers. He then adjusts his quote according to what he hears and his trading rationale, until his price meets that of an opposing trader, or another trader's shouted quote is acceptable, and a trade is made. This has the advantage of being efficient in the sense of fast adaptation to the market equilibrium, and is used in highly dynamic markets such as stocks and commodities. A common stipulation, often called the New York Stock Exchange rule after its origin, requires that a quote persists after being given, and subsequent quotes must improve on that quote. The difference from the bilateral bargaining described above in that it is a public many-to-many auction, i.e. many buyers and many sellers (or traders who are both).

In the past, these types of auctions were often highly decentralised, with just aggregated information being provided to the traders centrally. With the advent of computers, many different kinds of CDA auctions have been implemented around the world. A comprehensive review of such systems can be found in [39].

One of the most common forms of computerised auction is a clearing house auction, also know as call auctions, which is centralised variation of a CDA. In a call auction, all traders submit their trade prices (asks and bids) to a central independent "order book". Periodically, the total perceived market supply and demand are then calculated, along with the equilibrium price, and all traders simultaneously trade at this price. Additionally, traders can review all current orders in the order book to aid decision making, as more information is available to them. This is a fast and efficient market form that is common in

finance markets, but is completely centralised in nature, as the central order system must be independent and trusted by all traders. Additionally, because it is used in specialised markets, it has not been scaled to large numbers.

3.1.3.6 Other Auction Mechanisms

Retail auctions, or posted offer auctions, are the most common form of market mechanism today, being used in most western consumer markets. The seller posts a price for particular good (e.g. bread) at a particular location (e.g. a supermarket). The buyer travels to the posted price location, and either accepts the fixed price or rejects it. The seller adjusts the posted price occasionally, in view with its perception of the market's supply and demand. This is used in high volume markets with low dynamics and is very scalable, although not particularly fast in terms of adapting to change in the market supply/demand. A local supermarket is a prime example of this type of "auction".

3.1.4 Computational Economics

The basic equilibrium model described in section 3.1.1 above is tried and tested, and used frequently in the study of market behaviour and dynamics. It lends itself to the mathematical modelling of markets, where supply and demand functions can be specified and the model's equations solved to provide results. This has been the dominant form of modelling markets in the field of economics. However, many of these mathematical models have had to make assumptions about the market to retain mathematical tractability. For instance, they may assume perfect rationality on the part of traders, or all to all communication between traders. It often requires fixed supply/demand curves, and thus it is difficult to analyse individual model agents' adaption to changing market conditions. The final state of the market can be examined, but not the dynamic state as a market 'finds' equilibrium. This is very much a 'top-down' model, attempting to capture the entire behaviour of a system in concrete manner.

The field of experimental economics represents a different approach to understanding markets, and was established by Vernon Smith in 1962 [105]. He created a simple experimental market, and used real human traders to explore market dynamics. This approach allows individual trader behaviour to be examined, which the mathematical models do not.

With the emergence of cheap and accessible computation now available to researchers, experimental computational modelling has become increasingly popular in economics during the last 20 years. Whereas the size and complexity of experiments in earlier work

was tied to human limitations, computational agents allow for many more options. Markets can actually be fully simulated and traced, and the behaviours of individual traders defined and examined. This allows us to look at the above assumptions in great detail, and study more complex questions that were not feasible with previous models. They can provide more detail about the rich dynamics that can be present in economic systems.

This approach is a 'bottom-up' model, as the individual agents give rise to system-level behaviour. It allows us to observe details like robustness and individual traders actions, and explore areas that were previously infeasible to examine.

3.1.5 Summary

Many different kinds of markets exist in the financial world. The dynamics of the supply and demand can be very different. In this section, we have reviewed a variety of common auction types, which vary in many ways:

- quotes can be private or public;
- quotes be communicated simultaneously or continuously;
- quotes can be given by the buyers, sellers or both;
- the number of quotes allowed can vary;
- the number of participants: one-to-one/one-to-many/many-to-many;
- the need for a central arbiter or not;

For our work, we will need auction mechanisms that can be decentralised, and are suited to the Grid environment, as well as the usual desirable properties such as enabling market efficiency due to accurate predication of the equilibrium price.

The individual rationales of the traders involved can be very different (e.g. the consumer market for clothes versus the market for company stock). In the next section, we look at understanding and modelling human trader rationality, and the work being done to automate this.

3.1.6 Terminology

For the rest of this thesis, the following terms are defined as described below.

- *quote*: a price offered from one trader to another for an amount of resource.

- *bid*: a price offered by a buyer to a seller.
- *ask* or *offer*: a price offered by a seller to a buyer.
- *transaction*: a trade between two traders at an agreed on price.
- *market price*: the current aggregate price of a transaction.
- *equilibrium price*: the theoretical price where supply and demand meet and the market is most efficient.

3.2 Computational Economic Traders

In the academic world, much effort has gone into modelling and artificially reproducing individual trading behaviours and rationales, with some success. The industrial world has also had increasing interest in automating trading, as results have shown that fast automated traders can both increase individual utility as well as improve overall market efficiency by small amounts. In this section we review this work, in particular the recent work inspired on simple generalised traders.

3.2.1 Modelling Human Traders: The Trading Agent Competition

The main focus of academic effort in the area of artificially intelligent traders has been around the Trading Agent Competition (TAC) [114]. Every year since 1999, an international competition has been held to allow different artificial agents to compete in complex market situations.

The original TAC market defines a multi-commodity travel market. Each agent must buy flights, hotel accommodation and different event tickets as entertainment. Each agent in the competition represents 8 customers, who all have different preferences for the various commodities. Each commodity is sold as a separate type of centralised auction, with different characteristics and market rules. Some auctions (for hotels) closed at specific times during the game. In the entertainment market, the traders could both buy and sell event tickets rather than just buy, as in other markets. A successful agent must maximise its clients' utility in terms of the clients preferences regarding travel dates, hotel quality and entertainment value.

A successful agent in this market is SouthamptonTAC [60]. This agent utilised fuzzy logic whose rules were based on whether related hotel markets were still open or had closed recently, or a while ago. It also included an environment sensor which used historical data from previous games to detect the dynamics of the different markets, and classify them as either competitive, semi-competitive, or non-competitive. This was used to decide on which of three 3 different basic strategies based around risk-aversion, risk-neutrality and risk-seeking to employ in bidding. A risk-seeking strategy was to buy flights first, and fit everything else around that. A risk adverse strategy involved delaying hotel purchase until flights and entertainments were closer to being acquired.

Another successful agent model was Walverine [20]. This agent takes a more analytical approach, and commits to buying flights early. It then uses a linear integer optimiser to decide what price for a good to bid on to maximise a trip's utility. The optimiser uses predicted prices for the different markets based on the Walrasian competitive equilibrium

(the set of prices at which all markets would clear).

A new TAC market was introduced in 2003, based on supply chain management, and called TAC-SCM [4]. It simulates a computer production supply chain, where a computer is made up of a CPU, a motherboard, some Memory and a Disk. These different goods are supplied by different suppliers with their own quantities and prices, and the agents must manage the whole chain of components, orders, assembly in a factory, and shipping.

A successful agent by the same people in this market was SouthhamptonSCM [61], which split the trading into three agents, a component agent to talk to the suppliers, a customer agent for orders, and a factory agent to assemble and ship. Each agent had a hand-coded custom trading strategy, and communicated via the factory agent, which balanced the supply and demand coming from the other agents.

The paradigm for these agents is a 'top-down' approach to artificial intelligence. That is, they attempt to capture human thought patterns at a high level, and develop methods of implementing similar strategies computationally. This means that the architecture of these agents is often sophisticated and multi-layered, as human strategies are often complex and intricate.

Additionally, these agents are designed for a specific market that is well defined and very specific to a particular situation. They are also very complex markets to simulate and implement in the first place.

Whilst it would be potentially feasible to implement similar algorithms in our exploratory work, the agents are complex to implement and would require refactoring to be more general. Additionally, it would be difficult to extract the generalised trading rationales from the market specific strategies. We therefore look to a different approach for trader rationale than the TAC-orientated algorithms.

3.2.2 Simple Computational Traders

A simpler model of trader rationale would be more useful in this investigation. Gode and Sunder [58] developed such an agent and investigated the effect of rationality in market performance. They formulated a Zero-Intelligence (ZI) trader model, where agents bid randomly in a double auction market. They added the constraint that the traders' random bids must be within their limit prices (ZI Constrained, or ZIC). That is, a ZIC buyer's bid is uniformly random between its limit price and the market minimum, whilst a seller's is between its limit price and the market maximum. They found that even with no bidding strategy, just these simple constraints, markets with ZIC traders still reached equilibrium. They proposed that market structure was a partial substitute for trader rationality. This

model has been much investigated, and been used as a baseline market predictor [47] amongst further investigations into areas such as general market efficiency [59], and information transmission [87].

3.2.2.1 ZIP Traders

In [26], Cliff analysed the ZI traders performance and demonstrated that in certain market situations ZIC traders failed to find the equilibrium price. These were markets which flat supply curves, which means the ZIC agents have no variance in their constraints, and thus are unable to move towards an equilibrium. Cliff shows that the accurate equilibrium's reached by the original ZIC traders were artifacts of the original simulation supply and demand curves. While these curves are not unrealistic, in real world markets they can vary significantly.

Following on from this analysis, Cliff proposed and investigated an improved trading algorithm, called Zero Intelligence Plus, or ZIP traders. A ZIP trader's rationale is split into two parts. Firstly when to adjust price (and in what direction), and secondly, how much to adjust the price by.

The first part is formulated as follows. When a ZIP trader hears a quote, it will adjust its price in the following three situations.

- When the quote was successful and the quote price is better than the traders current price (i.e.. lower for a buyer, higher for a seller), it raises its profit margin (i.e. lowers price for a buyer, raises it for a seller). The quote shows the possibility of successfully trading at a better price.

- When the quote is successful with a worse price and is an opposite quote (i.e. ask for buyer, bid for seller), the trader lowers its margins (i.e. raises price for buyer, lowers price for seller). The quote indicates a successful trade that the trader would not have won at its current price.

- When the quote is unsuccessful with a worse price, and is a competing quote (i.e. bid for buyer, ask for seller), it lowers its margins. The quote indicates that a competitor failed with a worse price, so we should adjust ours accordingly.

The second part, the question of how much, is provided by using a Widrow-Hoff delta rule to adjust the price. The delta rule, commonly used in back-propagation in neural networks, allows the price to descend the error curve in proportion to the degree of error.

$$\Delta u = (1-\gamma)\beta(q-t) + \gamma\Delta u_{last} \tag{3.1}$$

Each ZIP agent has a limit price λ and profit margin u. The margin is adjusted as in equation 3.1, where β is the learning rate, γ is the momentum, q is the quote price, t is the target price, and Δu_{last} is the last calculated change to the margin. The target price is calculated with a stochastic function $t = (1 \pm R)q \pm A)$ where R is a small relative perturbation, A is a small absolute perturbation, q is the observed quote price, and both R and A are randomly drawn from a uniform distribution. When the intention is to lower the price, R and A are subtracted, whereas when the intention is to raise the price, R and A are added. The price is then generated as $p = \lambda(1+u)$ for sellers and $p = \lambda(1-u)$ for buyers, then constrained to the the agents' limits if greater (buyer) or lesser (seller).

The purpose of this rule is to approach the perceived equilibrium price. If quote prices are constant, it would asymptotically approach the constant price at the learning rate. The momentum parameter is included to dampen oscillation as the perceived price changes from quote to quote. The stochastic function provides a small amount of noise to improve the adaption.

Using a small centralised auction Cliff found that ZIP traders out performed ZIC in finding market equilibrium, and were also able to successfully find equilibrium in the market environments in which ZIC traders were unsuccessful.

Cliff went on to further optimise this original ZIP algorithm using evolutionary optimisation [25], including developing a novel partially-double auction mechanism. Additional investigation was done on whether the results were depending on the synchronised "trading days" used in the simulation [24], but ZIP continued to perform in a continuous trading situation.

Further extensive work on optimising ZIP can be found in [48]. Here, the original 8 parameters were extended to 60 parameters for further evolutionary optimisation. The base 8 parameters are the lower and upper bounds for γ, β and initial profit margin, and A and R. Different sets of these 8 parameters were used for each of the 3 adjustment conditions described above, separately for buyers and for sellers, giving 6 sets of 8 parameters. Also, rather have A and R set system wide, they were drawn randomly for each set of parameters from a bounded uniform distribution, providing six sets of 10 parameters. On the whole, the elite ZIP60 agents out performed the elite ZIP8 agents from [25], although not by significant amounts, and with more variance, and cast doubt on the earlier partial double auction result. The initial ZIP60 optimisation work presented suggests that the distinction of different sets or parameters for buying/seller has less impact than using different sets for raising/lowering market price, and a more sophisticated genetic encoding to explore this is suggested for future work.

ZIP traders are an established trader rationale and have been used in variety of market

studies [72] [73] [35], as well as apparently being used to some degree in real markets, although this work is understandably not published.

3.2.2.2 Decentralised ZIP Traders

All the experiments using ZIP have utilised a global centralised auction mechanism. Its use in a decentralised auction is problematic, as there is no default method to know the success or failure of a quote, which the ZIP algorithm depends upon. Particularly relevant to this work, an exploratory method of using ZIP in a decentralised P2P setting is presented in [37], which attempts to guess the likelihood of the success or failure of an observed quote. It uses ZIP's delta learning rule to track the observed ask prices and buys prices separately. When a quote is observed, it is assumed that it will eventually trade at a price halfway between its current price and the traders delta-observed price for the opposite quote price. For a bid, this is half way between the bid's price and the estimated current ask price, and for an offer, this is half way between ask's price and the estimated current buy price. If the quote price is better than the current estimated price of the opposite quotes, then a successful quote is observed at that price by a normal ZIP algorithm. For a bid, if its price is greater than the currently estimated ask price, it is observed as a success, else is is observed as having failed. Whereas for an ask, if its price is less than the current estimated bid price, it is observed as succeeded, or else as having failed.

The simulation details presented are minimal, but indicate a single unit decentralised CDA auction. The results showing only market efficiency losses, which overall were worse than that of the centralised ZIP auction, which is to be expected given the decentralisation. Efficiency was shown to improve with more traders, which implies an all to all topology between traders, although this detail is not discussed. The efficiency was most improved by increasing the number of quotes an agent sent, presumably by specifying a minimum number before being allowed to accept. Results were also presented for an asymmetric markets, with many more buyers than sellers. Here the efficiency dropped considerably when there were more than twice the number of buyers than sellers. While the simulation details are minimal and analysis of this result is lacking, the ideas are an interesting development on the ZIP paradigm, and relevant to this work.

3.2.2.3 GD Traders

The Gjerstad-Dickhaut (GD) trader model was introduced in [57]. It defined a method of using a recent history of market activity to calculate a bidding price. This history includes the price, whether it was an ask or a bid, and whether it was successful or not.

In the GD model, a "belief" function $f(p)$ is used to calculate the likelihood of a price p being acceptable to any trader, based on this recent market history. For the seller it is formulated as:

$$f(p) = \frac{AAG(p) + BG(p)}{AGG(p) + BG(p) + UAL(p)} \tag{3.2}$$

where for the order in the history, AAG is the number of accepted asks above p, BG is the number of bids above p, and UAL is the number of unaccepted asks below p. Conversely, for the buyer:

$$f(p) = \frac{ABL(p) + AL(p)}{ABL(p) + AL(p) + UBG(p)} \tag{3.3}$$

where ABL is the number of accepted bids below p, AL is the number of asks below p, and UBG is the number of unaccepted bids above p.

For each price in the order history, this belief function is used to calculate a hard value point, which indicates the likelihood of this price being acceptable in the current market (as defined by the history). Other price's belief values are then interpolated using a cubic-spline based on the nearest hard values for prices above and below it, and clamped to [0,1]. The price that maximises the likely utility, defined as $(p)(p-l)$ for sellers and $(p)(l-p)$ for buyers, where l is the agent's limit price, is chosen as the next ask or bid price, respectively.

The GD model was modified in [107] (MGD) with three small modifications. Firstly, the MGD agent remembers the highest and lowest successful trades in the previous trading period, and uses them to limit the belief function above and below those values. That is, the belief function values for prices below the minimum successful price are set to zero, and those above the maximum to 1. Secondly, the values provided by the belief function are clamped to $[0,1]$ before the interpolation, rather than afterwards, as in the original. Thirdly, for agents with multiple units, agents are allowed to bid using the least valued unit.

These modification are designed to reduce the volatility of the GD algorithm in certain situations (such as no rejected bids in the history) and to improve the agent's surplus.

Tesauro further extended the GD model, along different lines to MGD, in [106], by using dynamic programming techniques to optimise for longer term prediction (GDX), which was show to slightly outperform both MGD and the original ZIP8 algorithm.

3.2.2.4 Simple Agent Comparison

A competition between different agents was presented in [107]. Agents were compared against competing homogeneous populations of agents in a multi-unit CDA. The agents included ZIC, ZIP, GD, MGD and Kaplan "sniper" agents [65]. Kaplan agents wait until the last possible moment before submitting a bid, thus hoping to win the trade by slightly under cutting the current best price. They do not adapt to market price, just undercut the current quotes, even if they are far from equilibrium. The results presented showed in general that GD, MGD and ZIP out-perform ZIC and Kaplan agents, and MGD was slightly better than ZIP, and both were slightly better than GD.

A further investigation was carried out on heterogeneous populations of agents, including ZIP, MGD and Kaplan agents. Interestingly, this experiment used an order book CDA auction which required minor modifications to MGD and ZIP. It allowed agents to switch tactics by observing another random agent, and switching to that agent's strategy if it would improve their surplus. The most likely strategy equilibrium in this initial CDA auction was a mix of approximately 45% ZIP and 55% Kaplan agents. Although other equilibria gave a higher market payoff (e.g. MGD 55%, ZIP 35%, Kaplan 10%), they were less stable equilibria. They added an artificial improvement to the MGD agents, which resulted in a swing in most likely equilibria being all MGD agents, if the MGD could achieve 5% more utility.

In [35], a comparison between artificial agents (ZIP/GD) and human traders in a simulated computerised market is presented. The ZIP algorithm was slightly modified, but is essence the same. It was however the unoptimised version for the original study, rather than the optimised one released later. Both the ZIP and GD out-performed the human agents in this simple market, by a similar amount, suggesting the simple automated traders such as ZIP and GD could be used advantageously in the real world.

3.3 Market Based Control

Market Based Control is the field of Artificial Intelligence that uses economic principles to design intelligent systems, of for the purposes of resource allocation. Early efforts in this approach were collected in a work entitled "The Ecology of Computation", edited by Bernado Huberman [63] in 1988. The collection included early efforts at applying market ideas and principles to distributed computer system design. Since then, the ideas have continued to be developed, with a lot of recent effort focusing on allocating resources on Grid-like systems.

3.3.1 Early Pre-Grid Work

3.3.1.1 Agoric Open Systems

Under the moniker "Agoric Open Systems" [79], Drexler and Miller presented a simulated MBC system [40] that used a rental model to share a single machine's processor and storage resources. It utilised a centralised, sealed-bid, second price auction mechanism on a single machine. It included a bidding method of escalating bid prices over time to ensure a degree of fairness, and to provide opportunities to change a process's current bid. The design was specific to a single machine, dealing with garbage collection and pointer management, a much lower level model of computation than the grid. It was an exploratory study on the potential for economics systems,

3.3.1.2 Enterprise

The Enterprise mechanism [77] was an early job allocation system that utilised a decentralised, first-price, seller quotes, single-round, sealed-bid auction. The price metric was user-estimated job completion time, and it included a cancellation/reneging protocol. It was optimised toward system level mean job flow rate, or throughput, as opposed to fairness, and there was no actual currency involved

It utilised an all-to-all communication network over a local network, which scaled well up to 90 machines. However, such a system would eventually hit a limit of communication overheads for its all-to-all connections, especially if some machines were located non-local networks with increased communication costs, particularly for broadcasting communication. This limited the potential scalability of the approach.

Additionally, using a user supplied completion time as a price metric encouraged users to be dishonest in their estimates in order to get quicker scheduling, and while they address

this somewhat with a penalty mechanism, it would be a problem in applying their solution to the grid.

3.3.1.3 Spawn

Similar to to the Enterprise project, Waldspurger *et al* developed Spawn, a distributed computational economic allocation system [112]. The resources are a network of heterogeneous machines, and the Jobs are modelled as easily parallelisable Monte-Carlo simulations.

It utilised a sophisticated sponsorship hierarchy for allocating complex jobs across multiple resources. A root agent, controlled by a human, instantiates bidding agents on a variety of resources with different rates of funding supplied. A static global knowledge of resources is assumed. A single-round sealed-bid second-price auction was used on each resource to allocate a 60 second slice of execution time, and the winner could extend their time by continuing to pay the auction price if needed. The root agent could then manually adjust the funding rate to its spawned agents to used the cheapest resources, if so desired. Various topologies of connected resources were used, including small all-to-all topologies, up to 64 machines in a regular ring network, although they discuss the possibility of scaling using a random network.

Tycoon (see section 3.3.2.3) is the effective successor to this system and is examined in more detail below.

3.3.1.4 Other Pre-Grid Systems

A collection of pre-grid MBC systems was presented in [22], that built upon the work reviewed above a variety of early systems. Examples include a system for trading hot and cold air in a building's air-conditioning system [23], scheduling of jobs in a factory production line [6] and allocation of bandwidth in ATM networks [80]. A critical review of these systems is presented in [27], which points out that all the system presented either used a centralised auction mechanism that required a global view of the market, and in some cases have no real auction mechanism at all.

Additionally for this work, the markets are all specific to particular problems, which are not grid-like. In every case the auction was under the control of a single administrative domain, unlike a potential grid market.

More modern, grid-orientated systems have developed in recent years, and we examine these in the next section.

3.3.2 Grid MBC Systems

The idea of an economic model for grid computing is not new. A review of distributed economic systems is presented in [12], outlines the various potential auction mechanisms for a grid market place, similar to those outlined here, and provides an overview of some distributed economic allocation systems. Most of these systems are designed for a specific purpose, and require administrative control over the resources, and as such are not applicable to a grid system. The Grid Economics and Business Models conference [3] is the focus of much of the activity in the Grid community in this area. However, most of the work is directed at the business model for the grid, as opposed to utilising economic principles for actual allocation, although some systems have been developed along these lines, and are discussed below.

In the more general MBC literature, a recent review [66] of current grid MBC systems shows much progress, and a collaborative effort is underway in the UK to build an economic grid system [41] on top of existing Grid systems.

3.3.2.1 Nimrod/G

One of the earliest Grid focus MBC systems is Nimrod/G [14] [1]. It allows users to define a maximum cost and execution deadline, and resources define a base cost. The scheduler uses the centralised Globus MDS to discover potential resources and their associated costs, and uses a cost-deadline optimisation algorithm [13] to select suitable resources. The cost and estimated completion time for these resources is passed back to the user, who can then adjust their cost or deadline amounts if desired, or accept the allocation and dispatch the job. The resource prices are set manually by resource owners based upon their own perception of market demand. It does support advanced reservation in this way.

Nimrod/G thus acts as an economic facilitator with humans as the traders, and effectively a posted price/retail auction mechanism. This has the potential to scale well, as retail auctions do, but will be less effective in more dynamic markets, where faster adaptation to changing supply/demand is required. Additionally, the trader strategy relies on human interaction, and therefore is not automatic, and is not as performant or as scalable as utilising computational agents to automate the bidding process. It is also based around a centralised resource discovery mechanism (Globus' MDS), which further limits its scalability.

3.3.2.2 Bellagio

The Bellagio [5] system uses a distributed peer-to-peer search mechanism to locate a set of resources matching a user's criteria. This mechanism is the SWORD [88] service deployed on the PlanetLab Grid system. Bellagio uses SWORD to populate a list of suitable resources on demand.

The bidding mechanism uses a centralised combinatorial auction mechanism called SHARE [21]. This uses threshold rule, similar to a Vickery auction, that provides an incentive for honest bidding, and calculates a reasonable estimate for allocation, given the optional allocation is NP complete.

Human users construct bids with a bidding language called XOR [10]. It is capable of handling bids for multiple resources, and short-term advance allocations (less than 1 week). The timescale for allocation is quite long, with the SHARE auction clearing every hour, and taking several minutes to perform the combinatorial calculations.

Bellagio is particularly focused on multiple resources. Buyers can use the XOR language to bid for slices of CPU time, memory, disk space and bandwidth.

It is based a round a slow, centralised combinatorial auction mechanism, which limits both scalability and dynamic adaption to market conditions, and such would not be useful in an on-demand market. Additionally, it requires human participation to manually construct complex bids, further limiting is ability to adapt compared to automatic trading agents.

3.3.2.3 Tycoon

A real world economic allocation system called Tycoon has been implemented by HP Labs [75] [67] [74] [102]. Tycoon separates the bidding into two layers, a top-level strategic bidding level which expresses the high-level preferences of the buyer, and a lower level per-resource auction mechanism which takes the supplied preferences and allocates actual resources. In many ways it is similar to the seminal *Spawn* system (see section 3.3.1.3).

The strategic top layer decides on which resources to utilise, using a Parent Agent. This Parent agent is customised per-application, including a budget, deadline and number of resources required. It also includes a description of the performance requirements of the application (e.g. a low latency web server, or a high-throughput simulation).

It uses a simple centralised service discovery service that is updated every 30 seconds with resource utilisation information. The update also includes the key needed to communicate with that resource directly.

The Parent Agent queries the central discovery service with its requirements, and

receives a list of potential resources. It then weights these potential resources according to its supplied preferences for performance, and selects the number of resources it requires that have the best weights. A Child Agent is instantiated on each selected resource, to participate in the lower level mechanism. It is supplied with an initial amount of funding, plus an indication of the expected interval e_i at which more funding will be granted by the Parent Agent.

The lower level mechanism consists of a modified proportional share auction on each single resource, call Auction Share. Each Child Agent specifies a bid amount b_i at their budget and CPU time amount q_i. Agents of high-throughput jobs would set $q_i = e_i$, as their priority is to get as much CPU times possible,w whenever. Agents of a job needing low-latency would set $q_i < e_i$, holding some in reserve to win more CPU time when needed. Every 10ms, the auctioneer calculates the winning bid as the best of $\frac{b_i}{q_i}$, and that Agent's application runs until it has run for q_i or a better bid is submitted. The Child Agent is then charged in proportion to the amount of time they actually executed for.

This allows the Parent Agent to fund Child Agents differently over the life-time of the application. If a resource becomes cheaper, the Agent can fund that Child Agent more (or less if the resource is expensive), thus increasing the Parent Agents utility. A particularly expensive resource can be dropped in favour of finding a new one.

The Tycoon system is in production, and performs reasonably well with moderate numbers of machines. The authors estimate scaling potential of up to 75K machines and 2500 Parent Agents, which seem over-optimistic, given that the central service would have to handle over 1.5 million connections per second.

Also, the bidding strategy is still very manual, there is little automatic adaption to market price, or any computational agent rationality. The onus is on the user to produce a customised Parent Agent implementation for their application, and is not generic or automatic.

Additionally, the system still relies on a centralised service to some degree. While the action allocation is decentralised, it still has the weaknesses of a single point of failure and lack of scalability.

3.3.2.4 The SE3D Competition

In 2005 HP Labs ran an experimental computational market for selling compute time for rendering 3D animations called SE3D [71]. Specialised rendering resources were provided to a variety of client production companies via an MBC based allocation system. Given that all the resources were owned by a single entity (HP Labs), a centralised system was used. Results from a simulation based on SE3D are reported in [15].

There are two different auction mechanisms used. The first was a simple proportional share auction, where agents would be awarded resources based on their bid's share of all the bids. As resource was consumed, it was deducted from the agents budget. An agent's bid for this auction was calculated from a polynomial equation based on the current time t and the deadline d, parametrised with 6 constant parameters $c_1,\ldots,c_6$, as shown in equation 3.4, where B^a is agent a's remaining budget.

$$b^a/B^a = (c_1^a/d + c_2^a + c_3^a d + c_4^a d^2) \times (1 + c_5^a t + c_6^a t^2) \tag{3.4}$$

The second auction was a generalised Vickery auction (see section 3.1.3.4) where users submit a multi-unit bid of a price for each desired quantity of resources. The resource allocation is then calculated as the allocation that would maximise the sum of maximum prices, thus providing the highest global gained utility. The agent strategy used equation 3.4 to control 3 parameters, the total bid b_{max}, the maximum quantity of units N_{max} and a risk factor r.

The above strategies were optimised via a genetic algorithm, and the generalised Vickery mechanism in general performed worse than the proportional share mechanism.

While an interesting real world experiment, this approach has limited application for grid-like systems. Each agent is identical, under one domain's control, and they do not adapt to market conditions, simply using a function based on only their own internal information. Additionally, both auction mechanisms used are centralised, and the actual auction mechanisms used do not scale well in terms of computational performance.

3.3.2.5 Hewlett Packard's Utility Data Centre

HP Labs developed systems based on grid ideas under the name Utility Data Centres (UDC). These were essentially large sheds which housed many high-density, multi-processor machines, with the goal being 50,000 processors in a single centre. HP Labs explored a number of market-based approaches to allocation resources, including a centralised combinatorial optimisation auction [16].

Of particular relevance to this work is the simulation of decentralised ZIP agent approach in 2002 [100]. This work repeated the previous work on evolving ZIP trading agents (see Section 3.2.2.1), and then used ZIP agents to allocate resources on a simulated UDC. The UDC was modelled as a ring of processing nodes with a certain task capacity, and both a ZIP buyer and ZIP seller. Each node is connected to its nearest n neighbours, typically 5 (a regular network, see Section 3.4.3). A task arrives at a node along with a ZIP buyer, and the node gives an auction "call" to its neighbouring nodes.

If any of the neighbours wish to participate, they indicate this to the original node. Once all neighbours have responded, an identical auction to the original ZIP work is then held at the original node, using the pool of sellers who responded positively at the task's ZIP buyer.

In order to spread the load across the network, nodes can "task-shift" a currently running task away to another node. In this case, when a node is full and receives a new auction call, it can initiate its own auction process (with its own ZIP buyer) to unload one of its tasks to free up resource for the new task. This new auction process must complete before the node can send back its participation response to the original node's auction. This means that these task-shifting auctions can cascade across the entire network, spreading the load out, but also delaying the completion of the original auction considerably.

Three different task-shifting policies where examined; no task-shifting, always task-shift, and randomly task-shift on one node when all neighbours for an auction are full. When tested again drip-feeding tasks into single node, a no task-shifting policy limited the exposure to that node's neighbours, as expected. An always task-shift policy spread the load more evenly, but led to longer allocation times due to the cascading auctions, which decreased overall utility. The random task-shift policy spread the load, but not as well as the always task-shift policy in this case. However, when applied to a more general case of tasks arriving at random nodes as opposed to drip feeding into a single node, the random task-shift policy was shown to out-perform the always task-shift policy in both the "spread" of tasks and in overall utility, due to its lesser disruption of running tasks.

While an interesting experiment, the simulation had some simplifications and limitations. Given that the target model was based on the singly-owned UDC, some of the mechanisms used would only be feasible in a singly administered non-grid environment. For example, the most successful task-shifting policy of random task-shifting only works when the original node has the authority and ability to choose and enforce the node to be task shifted. It is difficult to see how this would work in a grid environment, where each node may be separately owned and self-interested. A real economic grid nodes would need to base the decision to task-shift on a more complex cost/benefit analysis of the cost of task-shifting against the benefit of performing this new task. Given the more competitive environment, a node may not have the time to complete any sub-auctions in time to compete in the original auction, and thus may have to risk bidding on the original task (add possibly winning) without having yet secured new resources for its current task.

The simulation was based on a continuous time-step model, similar to the original ZIP work. This is easier to implement and enables unaltered use of the original ZIP market simulation model, but has many simplifying assumptions. For example, in a real world

implementation, the timing of arrival of quote messages may have an affect on the auction process outcome. A model with a more realistic communication model would be needed for further experimentation.

The model also relies wholly on task-switching for distribution of jobs, and given the challenges associated with a method of self-interested grid-based task-switching (as outlined above), this makes task-shifting alone a limited method of distributing tasks.

3.3.2.6 CATNETS

The CATNETS system [43] [45] [44] [46] explores a decentralised economic mechanism with similarities to this work.

It uses the BRITE [109] topology generator to generate an overlay network of nodes, although they fail to specify what topologies were chosen or why.

It uses two separate markets, one for Applications buying Services, and one for Services buying Resources. A Complex Service Agent (CSA) is a buyer for Services and a Basic Service Agent (BSA) is a seller for Services and a buyer for Resources. The purpose of splitting the system into two distinct markets is a design simplification, justified by the fact that resources and services (composed of resources) are separate commodities. They expect Services to vary widely, but for Resources to be fairly standardised. This allows for the possibility of using separate market/trading mechanism for each market, which may prove more efficient overall.

The CATNETS system uses a hybrid auction approach. It combines the resource discovery phase with initial bidding using a private, single round, best price, seller quotes auction, as follows. A CSA at a given node advertises need for a Basic Service (BS) to all other nodes within 2 network hops. BSAs at these nodes respond with an initial quote. After a 500ms timeout the CSA selects the best quote from the BSA, and begins further bilateral negotiation between the CSA and the BSA until an agreement is reached, or not. If not, then the next best quoting BSA from the initial auction is selected, etc.

This second negotiation protocol is a bargaining protocol, with buyer and seller alternating offers until accepted or abandoned. It is based on the AVALANCHE [90] protocol. Each agent has a limit price, with initial estimations of current market price set at a random point within their valid range.

The decision of whether to accept, reject or counter offer is controlled by 5 parameters.

- priceNext : the target price for this negotiation, based on an improvement of the last negotiation price.

- priceStep : the fraction of the initial starting price difference that concession prices are set at, defined at the beginning of each negotiation. For example, if the sellers starting price was 100, and the buyers 50, and the priceStep was 0.2, buyer concession prices would be 90 initially, then 80, 70, etc. The sellers would be 60, then 70, 80, etc.
- weightMemory : a weight ratio of current to historical pricing information.
- satisfaction : the chance that an agent will counter offer if the last offer's price is above its estimated market value, or abandon the negotiation.
- acquisitiveness : the probability an agent will not make a counter offer. For example, a agent with acquisitiveness 1 will never adjust its price, but with 0 will always counter offer at the next concession price.

These parameters are tuned at run time using the STDEA [104] evolutionary learning algorithm to facilitate adaption to market conditions.

In the simulation results presented in [46], a Complex Service only needed a single instance of a Basic Service, which only need a single instance of a Resource, and each Service had a constant execution time of one second. A fixed number (1000) of complex jobs were allocated per simulation run.

The results showed poor convergence on equilibrium for large numbers of traders, as the demand was not kept in proportion with the number of traders. This left traders starved of information with which to estimate market price.

The homogeneous nature of the Service and Resource quantities meant that the Resource market was essentially identical to the Service market, somewhat removing the need for having separate markets in the first place.

The trader rationale is also limited, as it relies on a wholly stochastic approach to decision making. Market adaption was provided via evolutionary algorithm, which suggests it would not adapt quickly to highly dynamic markets. Overall, while the only other work to aim for a fully decentralised approach, it falls short in its design, in both experiment methodology (failing to increase trades for larger markets) and design (arbitrarily split markets for no real gain).

3.3.3 Summary

A common factor to most of the systems discussed in this section is that they are centralised in nature, requiring a central auctioneer process. This centralisation loses some of

the potential benefits of an economic system and imposes severe limitations on scalability, which is key for any grid-sized market. Many operate within a single administrative domain, unlike the grid environment, which has multiple competing domains.

Additionally, the type of auction mechanism used is more often than not some variation of a call auction which requires a global market view and thus cannot be decentralised, as well as potentially being very computationally intensive. Many are also specifically designed for particular applications and specific markets, and would be difficult to abstract to a general grid type market.

This tendency towards centralised markets is understandable, as they generally provide a global market view and are thus more efficient, as well as being easier to implement.

In terms of trading rationale, all the above systems use simple static models customised to their specific market situations and implementations. There is no use of modern trading agent rationality.

Of the four systems that implement decentralised auctions, two (Enterprise and Spawn) rely on an all-to-all communication, which limits scalability, as well as both not being actively investigated any longer. The third (Tycoon), uses a central service, with decentralised auctions. This approach does potentially provide a greater degree of scalability and adaption, but still has centralised dependencies and constraint. The fourth, CAT-NETS, is the most promising in terms of decentralisation. It uses a overly network for decentralisation, and a distributed hybrid auction, consisting of an initial sealed bid auction followed by a bilateral bargaining auction. However, the simple custom model of trader rationality is limited and would not adapt well in dynamic markets. The results reported so far are unconvincing as to its performance, especially the scalability of the system.

There are no current systems that provide a fully decentralised economic system, or that utilise work from the adaptive trading agent area. Additionally, none of the above systems consider any topologies other than all-to-all or random. This is not the case in real world markets, and in the next section we review some of recent work in statistical network analysis of different topologies, to aid in understanding the various properties of different topologies, and how they might apply in an economic setting.

3.4 Network Theory

Recent years have seen much development in statistical network theory. This field provides tools to categorise, create and analyse large networks and their topologies and behaviour. We provide a brief overview of some types of network topologies, and the techniques that can be used to analyse them.

3.4.1 Motivation: Modelling the Marketplace

A marketplace is traditionally a location at which traders meet to trade. Many factors have contributed over time to the formation of marketplaces. For example, social factors like language and culture, or geographical factors such as presence or absence of natural resources and ease of travel and transportation, can affect the trading relationships in a market. Technology has played a pivotal role in this area, and the 20th century saw sweeping changes in the way in which marketplaces develop. The advent of the internet has impacted markets hugely, as now there are few geographical limits on potential trading partners and opportunities.

Whereas before markets needed to have central physical locations to facilitate communication between traders, modern communication technology allows for potentially anybody to trade with anyone else. This has the effect of decentralising marketplaces in terms of geography to some degree. However, the same technologies have also resulted in a centralisation of many markets also, as they allow many more traders to efficiently participate. While physical location has become less important, the higher efficiencies provided by centralised mechanisms mean many markets are centralised to around specific market areas, such as commodities or stock, where the number of traders is relatively small.

One aspect of marketplaces that has not changed is the fact that trading is still a human activity (for the moment). This implies trading relationships that are based on underlying social interactions between people. The network of interactions between real world human traders is therefore a social network.

In this section, we examine various networks including social networks in order to develop a suitable virtual marketplace for allocation on grid systems. We look at recent development in network theory, particularly statistical analysis of networks. We also look at different types of network, particularly social networks, which form the basis of real world market places. We are particularly interested in whether or not the properties of social networks make for more efficient marketplaces. For reference see [84] for a full review of network theory, from which much of this section has been distilled.

3.4.2 Network Characteristics

A network is simply a set of nodes (vertices) connected by some number of links (edges). The London tube map is an excellent example of a network, with each station being a node and each section of tube line being an link. The possible connection topologies for even a small number of nodes is very large. Note that we are only interested in undirected graphs (where a link goes both ways), so we exclude discussion of directed graphs.

3.4.2.1 Statistical Analysis

Large networks are complex to analyse. Beyond a few hundred nodes, simple visualisation techniques are generally unhelpful for analysis. Network theory uses statistical analysis of large networks in order to properly understand their structure and behaviour. Fist we define some key statistical properties that can be used to describe a topologies characteristics. We then look at some typical real world topologies and describe them using these statistics.

- Mean degree: The degree of a node is the number of edges it has, and the mean node degree gives a measure of a network's density (proportion of actual edges over possible edges).

- Degree distribution: the distribution of node degrees can vary. For example, a small number of nodes could have a very high degree, with the majority of nodes having very low degrees. A simple measure of this can be to calculate the skewness of degree distribution. A positive skew would indicate the majority of nodes have low degrees, with a fewer having higher degrees, while a negative skew indicates a larger number of nodes with a higher degree than lower.

- Degree Correlation: also called the assortivity coefficient [83]. This is a measure of correlation between linked node's degrees. That is, a high correlation means that nodes usually have links to other nodes that have a similar degree (the network is assortatively mixed). A low correlation means that nodes usually link to nodes of a much different degree (the network is disassortively mixed).

- Average Path Length: this is mean of all the shortest paths between every pair of nodes in the network.

- Transitivity: also called the clustering coefficient, transitivity represents the probability a node's linked nodes are also linked, forming a triangle of links between

three nodes. Its is measured as the achieved proportion of the maximum possible number of triangles in the network.

3.4.3 Types of Network

Many types of network have been categorised. We review a selection of real-world topologies that occur in large real-world networks

3.4.3.1 Random Networks

The classic basic random network is Erdös and Rényi's random graph [42] . This is defined by the probability p that any two nodes are connected. These networks typically have a small average path length, low transitivity and a degree correlation of 0. These types of networks are generated by iterating through all possibly edges in the graph, and using a uniform random variate compared with p to deciding if those two nodes are linked. These types of networks have been much studied, and are used on P2P systems today. They are useful as a base with which to compare other network topologies.

3.4.3.2 Regular Networks

A regular network has a uniform structure of links for each node. For example, a two dimensional lattice where each node has a link to its north, south, east, and west neighbours. Another common regular network structure is a ring, where the nodes are placed in a ring and each node is connected to a number of nodes along either side of the ring. They are simple to visualise and understand. They often have high average path lengths, transitivity and degree correlation. A key concept in regular network is that of locality - where you are in relation to other nodes.

Generation of regular network requires an underlying model of n-dimensional space in which the nodes are located. For a ring structure, this is a one dimensional position, whereas a lattice is a 2D space, and many other models are possible. Once a measure of locality is defined, edges are created between local nodes, depending on the regular structure the network is a based on.

3.4.3.3 Small World Networks

First formalised in [113], small world networks are observed in many different real world situations. Its presence in social networks gives rise to its name, the "small world" effect,

the idea any two people can be linked one to the other via a short number of other people (edges).

Typical generation of such topologies is based on a regularly connected network, such as a grid or ring, and randomly rewiring some proportion of links across the whole ring. This maintains the high transitivity of the original regular network, while significantly lowering the average path length, with only a small proportion of rewired global links.

3.4.3.4 Scale Free Networks

Scale free networks were defined by Barabási and Albert [7] as a network whose degree distribution follows a power law (which implies a positive skew). In the regular lattice network described above, for example, each node has a degree of 4, and the degree distribution is therefore uniform. However, in many real world networks, the degree distribution is not uniform, and often follows a power law. Examples include inter website links on the world wide web, links between internet routers and protein interaction networks.

Scale free networks exhibit similar characteristics as small world networks, such as high transitivity and low average path length, with the additional property of negative degree correlation. There are few nodes with high degrees, with a larger number of nodes with lower degrees.

Such topologies can be created using a preferential attachment algorithm, in which a node is more likely to link to a node with a higher degree than a low degree. In a sense, scale-free networks are in part hierarchical, with high degree nodes at the top of the hierarchy, and low degree nodes at the bottom.

3.4.3.5 Social Networks

Social networks are the depiction of interaction between humans, and have traditionally been hard to quantify. In recent years, examples of networks such as high school student friendships and scientific publication collaborations have been available to study and have allowed interesting insights. Newman and Park [85] show that social networks have a particularly unique combination of properties, that is, a high transitivity (your friends are also friends with each other) and high degree correlation (popular people know other popular people, loners know other loners). This type of network is particularly relevant to this study as economic networks are based on social networks. However, a social network may not be the best topology for an online market - other topologies may be more efficient, but previously have not been feasible to implement.

There are no standard techniques for generating social topologies, and we propose a

novel mechanism for creating them in section 5.4.1.

3.4.4 A Grid Marketplace

A decentralised grid market place requires an underlying network of trader relationships to function. Random, small world, scale-free and social network topologies all have properties that maybe useful in implementing a market. BitTorrent [28] utilises random networks to great effect in achieving scalable bandwidth distribution. In a previous study [87] we investigated the effects of topology on a social information transmission and a simple economic market, using a small world and scale free networks. It was found that small world and scale free networks had regions in the space of their construction parameters that provided a better transmission of information and economic utility, which is a starting point for our investigations. However, we did not model communication costs, so the more realistic model of this study will test our earlier findings.

3.5 Summary

In this chapter we have reviewed current economic approaches to grid allocation, as well as work on economic artificial agent rationale and basic network theory. In the next chapter, we build a decentralised economic mechanism for allocating grid resources using some of the work discussed here.

Chapter 4

Methodology

4.1 Overview

In this chapter we present the basic model of grid resource allocation upon which the experiments detailed in Chapter 5 are based.

Ideally, a real world implementation of a resource allocation system would provide the best evaluation of its potential, but is not feasible for several reasons. Firstly, it requires a lot more investment in the implementation, as operating system process management and network communication would be necessary, and it would also need to be fault tolerant to a greater degree as it is running on real hardware. Secondly, and more significantly, it requires an actual grid system on which to deploy. While access to a small grid system may be feasible, it would limit the size of experiments run to the size of the actual grid. There are some larger grid systems available, such as the National Grid Service [103], TeraGrid [97] and the EGEE [94]. However, access to these for this work is not feasible, particularly as it would require modifying or running alternate allocation middleware from what is currently employed. Also, while the number of individual processors in these Grids is rather large, the number or participating resource owners is relatively small (10-20). As a particular focus of this work is investigating allocation with large numbers of independent parties on larger systems, a real implementation would be both impractical and restrict this focus of the investigation. Therefore, a simulation of a grid system has been the experimental vehicle for this study. This allows us to make many useful

abstractions and greatly simplify the grid model used, as well as look at arbitrarily large grids.

This chapter details this simulation framework and the model of the grid hardware, users, jobs, resources and infrastructure. It also explains the basic economic model, including the overlay network model. Finally it looks at how best to evaluate the performance of such a grid, and reports on an initial implementation.

The base components that make up our grid model are;

- Resource: a computational resource.
- Job: a unit of work that can be executed on a Resource
- Site: the location of a resource and administrative domain
- Buyer: an agent responsible for negotiating and executing a particular Job on a Resource
- Seller: an agent responsible for negotiating and executing multiple Jobs on a particular Resource.

4.2 The Grid Model

4.2.1 Resources and Jobs

A Resource on our model is a representation of a computational resource that can be scheduled to run Jobs. For the purposes of simplicity in this investigation, we ignore other types of resource, such as network requirements or storage.

A Resource is modelled by a unit capacity, which represents an amount of work it is capable of carrying out simultaneously. Given that parallelisation is the dominant factor for achieving greater performance, this is meant to loosely represent a number of processors. We make several abstractions here.

1. Hardware comes in many different types and with different performance characteristics. Given the growing virtualistion of computing resources, as well as for simplicity, we abstract these details. Thus all processors are modelled as being equivalent in throughput, with the number of parallel units being the primary performance metric. This trend can be seen in the hardware world, as CPU clock speeds have hit a wall, and the manufacturers are focusing on multi-CPU chips.

2. Software also varies greatly. Again, virtualistion technologies are quickly making this a non-issue as users can bundle their application load in the operating system of their choice. So we assume a Resource can provide whatever software platform a particular application requires.

Closely linked with a Resource, a Job represents an amount of work to be carried out on a grid. This is modelled with a fixed unit size or work and a fixed duration. The size is meant to represent a Job's minimum resource requirements and applies directly to a Resource's capacity. A Job consumes its size of a Resource's capacity when executing on that Resource for a length equal to its duration. A Job that has a larger size than a Resource's free capacity cannot be executed on that Resource. This abstraction is made for simplicity, but overlooks two key challenges in this area of grid systems.

1. It is technically challenging to correctly estimate the actual execution time of a given grid Job, and this is an active research area [99] [38]. The user can provide an estimation of a Job's requirements, but it is not guaranteed to be accurate. However, this problem is not our focus in this work so we assume that a job describes itself accurately and will run at that size for that duration. It is relatively simple to add a degree of run-time variance to the Job model at a later date along with an appropriate penalty model for providing a bad estimate or overrunning. This is discussed further in Section 7.3.

2. In reality, the "size" of a Job is variable. A large job can still be run on a small resource, but will take a very long time to complete. Likewise a small job can potentially be scaled up and be completed faster on a large resource, although this is not as easy as the former, as it may be not able to be parallelised easily. This introduces a completion time into the equation when considering which Resource to run on. While this is a feasible option for the model, it has been omitted from the basic model for simplicity. Also, our primary focus is for "on demand" allocation (see Section 2.4). This means that it is reasonable to assume that the Job description exactly represents the amount of Resource needed, no more or less, at that particular time. Again, further ideas for modelling this aspect of grid Jobs are discussed in section 7.3.

For both Jobs and Resources, we do not consider the storage or network requirements (e.g. bandwidth) and availability in our model. We assume that there is enough of each to allow the Job to execute successfully. This is a reasonable assumption given that our titled focus is on computational grids, not storage grids. Possible ways to include these type of resources are discussed in Section 7.3.

4.2.2 Grid Sites

Our model uses the notion of a grid Site to capture the idea of diverse domains of ownership of resources as well as physical location. It represents potentially many individual resources under one autonomous administrative domain. These resources are not necessarily physically co-located but are owned by the same group (or virtual organisation, in Grid terms). It is assumed that multiple resources at a site would have a faster intra-connection speed between themselves compared to that between separate Sites and thus can be represented as a single aggregated Resource consisting of the sum of the capacity of all resources at that Site.

The Site is the basic location of our simulation and the basic node in our network (see Section 4.4). A Site contains a Resource and a Seller for handling the negotiations about the resource.

A Site also has a Server object to model the performance and communication costs between it and other Sites. The Server is a software system assumed to be running on dedicated hardware (i.e. not the Resource itself). This allows us to model the execution of grid Jobs (the actual performance of which we are less interested in) separately from the execution of the allocation system (which we are more interested in). The Server's performance is modelled by a sampling execution time to process a message from a stochastic distribution (see Section 4.7.1), processing a single message at a time, with a wait queue of messages to be processed.

This is intended as a general measure of the Server's ability to process requests from other Servers, not necessarily the performance of a specific machine. It may be run on a cluster or larger machine, but this is simply modelled by increasing the overall performance of the Server, rather than introducing more complex hardware models, with a similar rationale to the section above on our Resource model.

A Site also acts as an entry point to the grid network for Buyers, allowing them to communicate with other traders at other Sites. So any particular Site will have a number of currently active Buyers utilising that Site's Server to find an allocation.

The management of a Job by a Seller in our simplified model consists of starting the Job when successfully allocated and finishing it when completed (i.e. it has run for its specified duration). In reality, the Seller has to do much more, including setting up the Job, monitoring its status (and reporting it to the Job's user) during execution, and cleaning up and returning the results when completed. We assume this to be automatic in our model, as it is not part of the problem of resource allocation that we are looking at.

The actual local scheduling algorithm used by our Resource is very simple. If a resource has enough free capacity for a Job at the current time, it reports being capable of

scheduling it, and if is told to do so, simply executes it straight away, and removes it when completed. Given that we are focusing on "on demand" grids, and not doing any advance reservation or completion time modelling, this is as complex as it needs to be.

4.2.3 Other Abstractions and Assumptions

Our model assumes perfect execution of Jobs and no Resource or trader failure. Fault-tolerance is an important part of any realistic grid allocation solution, but is not necessary for the first exploration of a decentralised economic system. Real economic systems suggest many possible methods to handle a breach of contract, but this is outside the scope of this initial investigation. A discussion of extending this model to include it can be found in section 7.3.1.

4.3 The Market Model

4.3.1 Commodities - What Are We Trading?

We are interested in trading a particular amount of the single commodity of computational resource. If we were simply modelling as a number of units sold, that would be straightforward. However, we are technically hiring the resources for a fixed period of time, not a one-off transfer of ownership. This changes the unit of resource quantity that traders are quoting for. Our unit of commodity is Job size multiplied by Job duration, or processor-seconds. The quoted price in our markets is in this measure, and traders consider the price of a quote relative to this value, rather than size or duration alone.

Our opposite commodity is a simple integer model of monetary value, possibly a real currency, more likely a "Grid currency". An integer value gives us simple, rounding error free arithmetic. We denote the currency using a $ sign, for which approximate real-world might be a 10000th of a penny. So, a Job with a size of 20 and a duration of 100s, which traded at a price of $250 (or 0.025 pence per processor second) would in total cost the buyer $500,000 (about $50).

4.3.2 Trader Rationale

Our model of trader rationale is purposely simplistic, following after Gode and Sunder's Zero Intelligence (ZI) trader model [58], as opposed to the Trading Agent Competition (TAC) [114] trader models (see section 3.1.4 for details). This desire for simplicity derives from our interest in the performance of the system as a whole as opposed to the

relative competitive performance of individual agents. Additionally, the TAC agents are designed for a specific market and commodity, with particular dynamics, whereas Gode and Sunder's ZI traders are general to any market, due to their complete lack of strategy.

The main attribute of a trader in our model, after the ZI model, is simply a limit price. This is the price a seller won't go below, and a buyer won't go above. For the baseline agent, we implement the Gode and Sunder's ZIC method (see Section 3.2.2). This means the agents bid randomly between their limit price and a system maximum (for Sellers) or minimum (for Buyers) price. This is designed to act both as a baseline comparison for different systems, and also as an exercise in how simple a working system could be.

4.3.3 Buyers and Sellers

A Seller object in our model represents a Site and its Resource. A Buyer object represents a Job. Both are located at a particular site, with a single Seller trading the Site's Resource, and multiple Buyers using the Site as their grid entry point. We could model the Buyers as separate 'Buyer only' nodes on the grid network, that came and went as buyers finished their work. However, this would require a dynamic network, and our network model (see 4.4) is static, so we link the Buyers to a Site to avoid the complexity of implementing a dynamic network. Additionally, this allows to maintain the network parameters and metrics as constant for a whole experiment, rather than needing to dynamically recalculate them as the network changes. This is also a likely real world scenario - that Buyers will join the grid at some point in its existing infrastructure (e.g. a portal or service). Both Buyers and Sellers utilise the Site's Server to model their execution.

There is a major difference between Buyers and Sellers; Sellers are long running processes linked to a specific resource, whereas a new Buyer is instantiated for each new Job in the system. In theory, this puts the Buyers at a disadvantage when they start to trade, as they have not been participating in recent trade and are thus unaware of the current market state. Therefore, when a buyer is created for a new Job, we allow a delay for the buyer to observe the market before commencing trading.

4.3.4 Auction Rules

The defining element to our design of auction rules is the need for a decentralised auction system. This places some large constraints on the practicality of many normal auctions. Some auction mechanisms rely on a central arbiter, which we are looking to avoid to achieve scalability, or assume all interested parties can hear all quotes in the market (i.e. they are in the same place). The nature of our network (our market place) means we

cannot guarantee that all traders will hear all the quotes for an auction, as it is not an all-to-all system.

An English auction assumes that all buyers are able to hear all other buyers quotes, usually facilitated by being in the same place. In theory, a seller could act as this auctioneer if all buyers are still able to hear the quotes. In our model, the Seller could take on the responsibility of doing this, and broadcast the quotes to all interested buyers. However, we couldn't guarantee the Seller would not attempt to alter the quotes in their favour somehow, possibly reporting different quotes to different buyers. This would effectively be a kind of double-auction, but with dishonest trading. While this may work if Buyer's are still willing to accept the falsified quotes, it would undermine the auction mechanism and could have unknown dynamics. As such, we do not use this mechanism in this study.

As discussed in 3.1.3, a Vickery auction is semantically the same as an English auction in terms of bidding incentive. It will not work in a decentralised setting however, as there is no incentive for the trader holding the auction not to accept the best quote at the best quote's price, rather than at the second best quote's price - it requires an arbiter.

In theory, a posted price auction could be used. However, given no central location to post static prices, the Sellers would have to actually quote individually to each trader, or the buyers query each Seller, thus losing the potentially useful scalable aspect of such auctions.

Of the auction mechanisms discussed in 3.1.3, the CDA and the sealed bid auctions can most easily be used in a decentralised setting. A sealed bid mechanism is simpler to design, but the CDAs ability to quickly adapt to market changes is an obvious attraction. A sealed bid auction allows for private bids, which the Dutch and CDA do not. This suits a diversely owned grid environment where traders naturally may prefer to bid privately. Its implementation is also simple.

In this study, we implement both a decentralised sealed bid auction and a explore a decentralised CDA mechanism. We discuss the implementation of a Sealed Bid Auction method as well as a CDA method in more detail sections 5.2 and 5.3.

4.4 Economic Communication Model

In this section we define a model of a decentralised grid "marketplace" in which traders can conduct their business.

4.4.1 Trader Relationships

In the real world, trading usually occurs around a specific physical location, such as a city street, a specific event somewhere like a farmer's market, or special purpose venue like the London Stock Exchange. With the advent of the Web, these places can be virtual such as eBay auction pages. These locations provide a meeting place for traders to conduct business with other traders. They may not trade with every trader present at such a location, but will form relationships of some form with other traders that they can try and sell to or buy from.

With a decentralised grid system, we do not have a centralised location. All traders are connected to the Internet, and could in theory trade with any and all other traders. However, all-to-all communication for any significant number of traders would quickly become impractical. It may be possible to have a number of central "trading floors" that could provide a central point of contact, and this has some real world examples, such as the various stock exchanges around the world. But we are focused on exploring a fully decentralised system.

We need to define which traders a particular trader knows about and can trade with. Ideally this needs to be as many traders as is it is feasible to communicate with simultaneously over the Internet. We model this with an overlay network between traders, commonly called a peer-to-peer (P2P) network. A node in the network is a Site with both Buyers and Sellers, and a link between any two sites indicates that the Sites (and their respective traders) know each other, and have a potential trading relationship. This attempts to capture the relationships between traders in a scalable manner, allowing trade to occur between traders that are linked to each other.

As a simple optimisation to avoid unnecessary transmission of messages, each message that is being broadcast keeps a copy of which nodes in the network it has already been to. While duplication can still occur with this method, as message copies may follow different routes, it is much reduced, and Servers automatically drop any message they have seen before, so the same message is not processed twice.

4.4.2 Network Topology

The topology of this network is a key area of study. The topology could potentially affect the efficiency of the market, and its degree of scalability and penetration.

As an initial baseline topology, we use a modified version of Erdös and Rényi's [42] classic random graph. We modify it because we have some real world constraints on our system, and we are not interested in random graphs per se, but in utilising them for a basic

economic system.

This basic network is generated in two stages. A desired mean degree is specified. We first iterate through all nodes and link them to one other uniformly chosen random node. This ensures that each node has at least one connection, and significantly reduces the chance of generating a graph with disconnected components. It also means the mean degree cannot be less than 2, but as such a sparse graph would be of little interest to study, this is not a problem, although it does affect the degree distribution to some level. We then choose a node at random and connect it to a new node via the same method. This is repeated until we have acquired a specified amount of links.

This modified random network is our base network for testing the performance of the system, and is very similar to some of the peer-to-peer generated networks in use today [28]. However, we are interested in other topological features, such as the shortening of average path length that small world graphs provide. A variation of the above method for generating graphs based on other topologies is discussed and results are presented in Section 5.4.1, where we look at generating and examining the effect some of the other topologies discussed in section 3.4.3, such as small-world, scale-free and social networks.

We also base our network on a physical torus-like model of spatial distribution. The grid concept is globe spanning, and so we model the location of our nodes over a wrap-around two dimensional space. Thus, nodes are a certain 2D distance apart. This distance is used to group nodes into $n \times m$ regions, as well as an indication for communication costs. Nodes within the same region can communicate faster than in those in different regions. This concept of regions could be seen as analogous to the different stock exchanges and other economic centres around the world. We use a 3 by 3 grid layout of 9 regions over a 2D coordinate space as the base network. Note that the regions simply serve as a grouping of nodes within which communication costs are lower than when communication with nodes in other regions.

In order to spread the jobs across the grid, we utilise a network broadcast and a relocation mechanism. Traders are able to use the network as a broadcast channel to pass on other traders' quotes to increase the "shout radius" of the quoting trader. This is similar to general P2P search techniques, and the topology of the network is a key part of the performance of such systems. This allows a job exposure to a much larger set of resources and increases the likelihood of a successful allocation.

If a buyer fails to find a suitable allocation, the Buyer can relocate to another random node to try again, which facilitates the spread of jobs across the grid. The addition of the notion of regions as above allows Buyers to try other region completely in the event of being unable to successfully trade in one region. For ease of implementation we assume

that each region will have a approximately up-to-date registry of machines from which to choose a random new node.

An alternative method would be to initiate a local search for a new node using the overlay network. As well as being more complex, this would likely lead to the new location having a large intersection of neighbours with the old location. This would make the success of the second auction less likely, as the same sellers are unlikely to respond differently than previously.

It would be useful to also allow task-shifting after the UDC work [100]. However, developing a mechanism for doing this in a self-interested grid environment is a significant research challenge (see Section 3.3.2.5). It would require each node's seller to also control a buyer, and perform a run-time risk/benefit analysis for each possibility. Whilst this would be a useful addition, it is a significant area of work, and we do not implement it here, instead relying on network broadcasts and relocation to spread load across the grid.

4.5 Model Evaluation

4.5.1 Metrics

In order to assess the performance of the allocation system, we need to record various results they produce. We have two main performance aspects to consider. While most other computational economic systems are primarily interested in optimising market performance, our primary interest is the performance of the grid allocation scheme itself compared to current approaches. Of course the characteristics of the market and interdependent to some degree with the performance of the allocation system

To measure the grid's performance, we use the following metrics.

1. Mean Resource utilisation - the proportion of usage of resources across the whole grid.
2. Mean Server utilisation - the proportion of Server load, as a measure of the cost of all the communication costs
3. Mean Server queue delay - the mean time for message spent waiting in a queue
4. Mean allocation time - length of time taken to successfully allocate a Job
5. The different properties of Jobs that failed or succeeded, such as size, it's buyers limit price and the degree of the node it traded on.

For evaluating the market we can use many of the standard measures used to analyse market performance in computational economics.

1. Trader utility/profit - mean traders utility (for buyers and sellers)
2. Market Efficiency - a measure of allocative efficiency as the proportion of surplus achieved of maximum surplus.
3. Smith's α value - a measure of convergence to equilibrium price. This is the root mean square of the difference of trade prices from the theoretical market equilibrium price.

The system mean of these values are measured for each experiment, in addition to being traced at regular intervals within a single simulation run, which allows us to examine the internal dynamics of the system. In general, for any parameter values, the simulation was run 100 times, and the mean and standard error (defined as the standard deviation of the distance from the mean) values were calculated. For statistical analysis, we use the WilcoxonMannWhitney U test [78] [115] where appropriate to verify the significance of the results. More specifically, we use the Kruskal-Wallis test [70], which is a generalisation of WilcoxonMannWhitney to three or more groups.

4.5.2 Comparison

In order to correctly assess the performance of an economic grid allocation system, we need to compare its performance with other allocation systems. We use two such comparison systems that do not use an economic allocation mechanism, but a centralised one.

We implement a simulated comparison system that is a version of the current state of grid allocation schemes. This follows the idea that each Site publishes its own resource state regularly to a central Broker. This Broker is queried by a new Job to find the best location for itself. The Broker attempts to optimise system throughput by returning an allocation that fits the Job into the smallest free resource capacity available. Additional Brokers can be added to scale the system, with each Broker periodically exchanging its current grid view with the others. This is similar to the way that the Globus Monitoring and Discovery System (MDS) [34] works, and is aimed at a realistic current day grid implementation.

Both these models use the same underlying network and Site model used by our economic system.

4.6 Preliminary Investigation

A early version of the above model was reported by the author in [36]. This used a similar basic grid model as outlined above, but a different trader rationale and network.

The trader rationale was not designed as a self interested profit-making algorithm, but rather as a static system utilisation optimisation function. A Seller bid higher for a Job the more the Job's size filled its available free capacity. Thus the Seller whose Resource would have maximum utilisation would win the auction. A simple sealed bid algorithm was used, with a Buyer advertising to all connected Sellers, and accepting the lowest of the first three offers that it heard.

As no actual per-trader variability was used in the trading process, this was not really a market-based economy, but more of an experiment in using a decentralised overlay network based system for grid allocation. It showed it was a feasible approach, even without an actual market economy, proving the validity of an overlay network model approach to communicating on a grid system and achieving acceptable performance.

4.7 Implementation Details

4.7.1 Approximating Reality

There is little real world data from grid systems with which to chose sensible representations of grid usage and provision. However, many similar scheduling and allocation problems have been observed and can be utilised here. We rely on [76] for much of this section. In particular, their discussion of distributions in machine shop scheduling are applicable here.

We have several main numerical system aspects to model as accurately as possible. Table 4.1 shows the details of major random variate distribution and default mean values.

4.7.2 System Load

A key parameter to define is the mean interarrival time of Jobs into the system, which is drawn from an exponential distribution. We derive this mean value from several of the above values that are inter-dependent, along with a higher-level parameter, *load*. This variable represents the potential maximum resource utilisation. A load of 1.0 indicates a situation in which, hypothetically speaking, 100% of the grid's Resources would be in use. In market terms, it means there are equal supply and demand quantities.

Model	Description	Distribution	Notes
Grid	Resource capacity	Gamma ($\alpha = 5, \beta = 20$)	$\mu = 100$ (1)
	Job size	Gamma ($\alpha = 5, \beta = 4$)	$\mu = 20$ (1)
	Job duration	Gamma ($\alpha = 5, \beta = 20$)	$\mu = 100$
	Server service mean	Normal ($\mu = 0.05, \sigma = 0.012$	(2)
Communication	Trader on same Node	0	
	Overlay latency mean	Normal ($\mu = 0.1, \sigma = 0.025$)	(2)(3)
	Regional latency mean	Normal ($\mu = 0.2, \sigma = 0.05$)	(2)(4)
	Global latency mean	Normal ($\mu = 0.4, \sigma = 0.1$)	(2)(5)

Table 4.1: Note: Normally distributed variates are re-sampled if negative, introducing a slight positive skew. (1) These are integer values. (2) The actual values used are drawn from am appropriate gamma distribution with this mean. (3) For node linked in a trading relationship in the network. (4) For unlinked nodes in the same regions. (5) For unlinked nodes in different regions.

A load value is supplied as a simulation parameter, and using Resource capacity and Job size/duration we calculate the appropriate inter arrival mean to generate that level of load on the system (see equation 4.1, where μ_x indicates the mean of the labelled value x, and $n_{resource}$ is the number of resources). We fix the values of Resource capacity and Job size/duration to sensible defaults, thus allowing us to alter the Job inter-arrival time as a key parameter driven by a simple load value. We can then easily compare the system on a variety of load values. Note that the above method means our load is spread out over one average job duration period of time. As such, on an empty grid, it would take one full mean duration period before the grid achieves the specified load level. For this reason, we commence measurement of our system-wide metrics after this time (typically after 100s).

$$\frac{\mu_{job\_size} \times \mu_{job\_duration}}{n_{resource} \times \mu_{resource\_size} \times load} \tag{4.1}$$

Note that with a load of 1.0 even an optimal system will be unable to achieve 100% resource utilisation, as our Jobs are modelled as discrete sizes, so there will always be some unused space, as Jobs are unlikely to fit perfectly into any remaining free space. As such we would expect an optimal load in the mid 90% range.

4.7.3 Supply and Demand

The values given to traders for their limit prices are a key parameter in our simulation. They define the supply and demand curves and thus the equilibrium price of the market. We use uniformly distributed values between two points to populate these with our traders in a variety of market situations, as shown in table 4.2.

Market	Buyer Limits	Seller Limits	Eq. Price
A	Uniform(100,400)	Uniform(100,400)	$250
B	Uniform(100,400)	Constant(250)	$250
C	Constant(250)	Uniform(100, 400)	$250
D	Constant(300)	Constant(200)	$250

Table 4.2: Standard limit price values for traders. The market minimum is $1 and the market maximum is $500

- A: The "normal" situation of supply and demand of equal and opposite magnitudes. This is the usual situation in our experiments unless stated otherwise.
- B: Flat supply prices and variable demand prices as above.
- C: Flat demand prices, with variable supply prices.
- D: Flat supply and demand prices. This situation has the characteristic that the trading is limited mainly by available quantity, as all traders should be able to trade with any other trader on price.

However, unlike most other computational economic experiments, our supply and demand curves are not precise, and thus also the equilibrium price is not precisely known. Our prices are fixed, but the quantities available for both Buyers and Sellers vary, thus altering the slope of the curve.

For Sellers, the supply curve depends on the size of the network, which is usually constant for a particular series of experiments. However, given that resource sizes are randomly sampled, the curve will only be approximately similar, with some variance.

For Buyers, the effect of varying the load value as described above affects the length of the Buyer's demand curve. At levels below 1.0 it produces a over-supply situation, and higher values produce and under-supply situation. And given the random sampling, even a load of 1.0 can produce situations of slight over/under supply.

Figure 4.1 demonstrates this "stretching" of demand quantities for each market type for a variety of load values. Note that in markets B, C and D, ZI-C traders have been shown to be unable to find the optimal equilibrium. See Section 5.2.3 for more detail.

4.7.4 Decentralised Trading

Another big difference between conventional computational economic experiments and this work is the distributed nature of the trading. Usually, there are a small number of agents (< 30) and a central auctioneer that employs synchronised bidding. This means

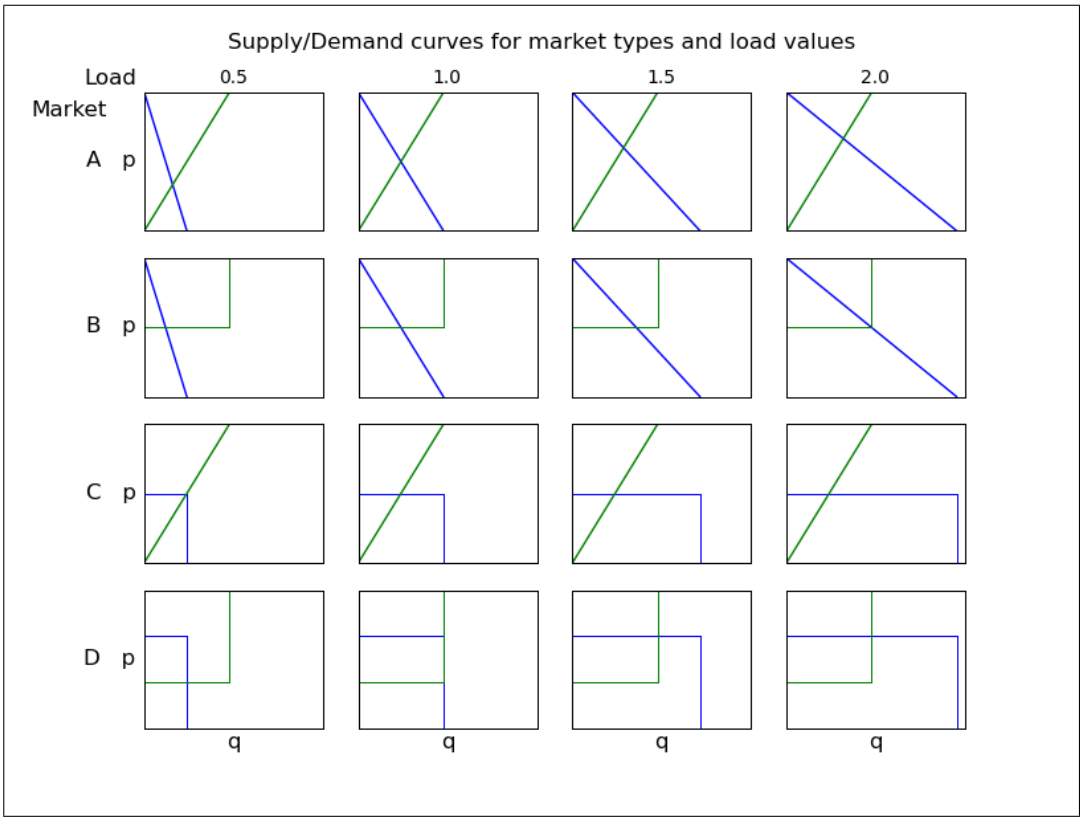

Figure 4.1: The theoretical market supply and demand curves for markets A, B, C and D for loads above and below 1.0. Note that the values have been chosen specifically to provide a theoretical market price of $250, except for market A, which will vary depending on demand, and market C at loads below 0.5, which will be less than $250.

that all traders get a synchronised authoritative response to a quote, and in the case of a public auction, all other traders also know that response. This poses two problems in our systems given the time-separation and uncertainties inherent in distributed systems, and the lack of a central arbiter.

4.7.4.1 Acceptance Protocol

Firstly, in a distributed system there is an inherent communication delay between receiving a quote and then responding to it. In that time, the situation may have changed. Additionally, it is possible that a server might be overloaded, and thus not respond to a specific request for some time or at all, leaving the sender hanging.

Our system addresses these issues with a simple generalised confirm/reject and timeout protocol. If trader A wishes to accept a quote it has heard from trader B, it sends trader B an acceptance message and sets a timeout. If no response is received from that trader B within the specified timeout, the trader A sends a cancellation message (in case trader B has responded positively, but the message is in transit), and moves on to other quotes from other traders.

If a confirmation response is received from trader B, the transaction is completed. If a rejection response is received, it means trader B can no longer fulfil the original quote (e.g. a Seller who has since filled up its Resource) and trader A moves on to other quotes.

When Trader A times out and sends a cancel message to Trader B, Trader B can be in one of three states, depending on what messages it has seen.

- Trader B received the accept message, but rejected it for some reason (e.g. a Seller who is now too busy). However, the reject message was not received by Trader A before its accept timeout occurred. In this case, the cancel message is ignored.

- Trader B received the accept message, and sent a confirmation. If trader B is a Seller, this involves starting the job. However, the confirmation did not arrive in time and the Buyer timed out and cancelled. In this case the Seller cancels the running Job. The confirmation, when it arrives at the buyer is ignored. This done for simplicity, but would potentially need better handling in a real system.

- Trader B did not receive the original accept message, so never responded, and ignores the cancellation message, and the accept message when it arrives.

4.7.4.2 Observing Success/Failure

The second problem is the uncertainty about whether a quote has been rejected or accepted. For ZIC agents this information is unimportant. However, for traders with any intelligence (such as ZIP), this knowledge is necessary.

A quote than has been accepted can be observed by the traders who send/receive and accept messages for that quote, but there is no explicit "reject quote" message in our system, as this would add extra messaging burden when it has to be shouted to all traders.

We investigate two possible solutions for this problem that provide success of failure information to the ZIP trader algorithm.

Firstly we use an observation timeout protocol. If a traders hears a quote (either from other traders or its own quote), an observation timeout is set. If the trader receives a corresponding acceptance message from another trader, it observes a successful trade at that price and cancels the timeout. If the timeout fires before an acceptance message is received the trader observes a failure for that price. If the observation timeout value is relatively long however, it could mean that in general there is a small amount of latency before the price might start to adapt to a general price fall compared to a price rise. Additionally, an acceptance message could be received after the timeout has fired, introducing further error into the estimation of market price.

As an alternative, we implement the experimental solution in [37], which is proposed for such decentralised systems as ours. In this approach, a traders uses the Widrow-Hoff rule from the ZIP algorithm to observe bid (B) and ask (A) prices separately. It then uses this information to guess the success or failure of a quote, and then uses the normal ZIP algorithm based on this decision, as follows.

If a bid of price p is heard, the current estimated ask price A is compared against an estimate trade price of $(p + A) / 2$. If greater, a successful trade at the calculated trade price is observed by the normal ZIP algorithm. Otherwise, a failure at that trade price is observed. For offers, the process is the same, except the estimated trade price is calculated with and compared to the currently observed buy price B. This allows traders to guess the success or failure without needed explicit notification of success.

4.7.5 Technical Details

Earlier investigations used a custom C++ simulation tool, which performed well but was difficult to modify for different experiments. In order to allow for easier development and experimentation the simulation was re-implemented using SimPy [81], a discrete event simulation tool written in the python programming language. This allowed rapid

development of different ideas and simplified making changes to the model and execution, at the cost of increased execution times and memory usage.

4.8 Summary

In this chapter we have outlined the basic simulation model and defined various key system parameters. In the next chapter, we explain the market process and parameters in more detail and present our findings using the metrics defined in this chapter.

Chapter 5

Performance Results

5.1 Overview

In this chapter we present the results and analysis of the proposed allocation scheme as implemented in our simulation model.

We present a decentralised sealed-bid auction model in Section 5.2, using random networks and ZIC traders. This shows both the feasibility and basic performance level of a simple fully decentralised economic system. We use this to analyse the performance of the grid infrastructure, and set suitable values for experiment parameters for further investigations. We investigate the performance of this system under a variety of different conditions.

We then introduce the implementation of a CDA auction in Section 5.3, and investigate using ZIP traders as an option for trader rationale. We compare this with the baseline sealed bid/ZIC model.

In Section 5.4.1 we develop a network model that allows exploration of varying network topologies, and investigate its effect on the performance of the system.

5.2 Sealed-Bid Auction

5.2.1 A Decentralised Sealed Bid Auction

This Section details the implementation of a single round, sealed bid, first price auction used to allocate resources. The sealed bid auction lends itself well to implementation in our model for a number of reasons.

- The private quoting is attractive in a decentralised grid system, as public quoting presents some challenges due to the lack of a central arbiter. Without a central independent information point, public quotes must be broadcast and forwarded by other traders, and thus depend on other interested parties being honest. For example, a trader could refuse to forward quotes better than their own current quote. The heard quotes therefore lack the authority provided by the independent arbiter. Private quoting avoids this issue, as quotes are point to point between the two primary parties.

- In a synchronised bidding system, the traders know the success or failure of quotes at the same time. This is certainly true in the simulated CDA markets used in most experiments. However, the asynchronicity introduced by network transmission and server load means that traders receive the quote information at different times in our system. Potentially, trades could be agreed on by nearby traders that have not seen the same quotes. Whilst this is to be expected to some degree given our decentralised approach, it could further decrease the efficiency of the market.

A sealed bid auction avoids these difficulties by having one way quoting (sellers only in our model) and private bids. As baseline system, we use ZIC traders (see Section 4.3.2), and random topologies (see Section 4.4.2). ZIP traders are not used in this sealed bid auction, as quotes are private, and therefore the information the ZIP algorithm needs is not available.

5.2.1.1 The Sealed Bid Auction Protocol

The normal flow of a trade is as follows. When a Buyer arrives into the system, it sends an "advert" to all its connected Sellers, including the one at its own Site. This advert includes the job details (size and duration), but no price information - it is simply a statement of interest. The Buyer sets a trade timeout to occur later, in order to trigger making a decision. In effect, the Buyer is advertising its own sealed bid auction.

The advert has a "shout radius" which indicates the number of hops the advert will be broadcast along the network, thus increasing the penetration of the advert into the grid system. Any Seller who hears the advert can then make an offer on the job directly back to the Buyer.

Upon hearing an advert, Sellers check they have the free capacity available at the moment to perform the Job. If they do, they make an offer using ZIC (i.e. a random value between their market price and the market ceiling price). If they cannot perform the job, they simply ignore the advert. This offer is private, communicated directly to the Buyer. Thus, only Sellers are using a strategy of any kind in this model.

A Buyer records the offers it hears from Sellers, resetting its timeout each time a quote is heard. If the offer is below its limit price, it keeps a record, else it discards the offer. When no offers have been received for the timeout period the Buyer selects the lowest of all the offers it has heard, and sends an accept message to the offering Seller, setting an acceptance timeout.

The acceptance model is as described in Section 4.7.4.1. Upon receiving an accept message from a Buyer, the Seller checks if it is still able to honour the offer (execute the Job), since it may have had other offers accepted in the meantime, and now be too busy. If it has enough free capacity, it sends a confirmation to the Buyer and begins executing the Job. When the Buyer receives the confirmation, it records the successful trade and is removed from the system. If the Seller can not now honour the original quote, it sends a cancel message to the Buyer. Upon receiving this, the Buyer removes that quote from its list, and repeats the acceptance process with the next best offer. If the Buyer's acceptance timeout occurs before receiving anything from the Seller, the Buyer sends a cancellation message and proceeds with the next best quote.

If the Buyer reaches the point when it has no valid offers (either never having received any, or receiving later rejections for the ones it did receive) then the Buyer can migrate to another region and start the process again. This is analogous to a trader moving to another market when failing to trade successfully in one market, and is included to allow a greater chance that the Job will be allocated by exposing it to more of the system. A Job can only migrate a limited number of times before reporting failure and leaving the system.

Table 5.1 shows the various parameters used in implementing this sealed bid auction mechanism.

Parameter	Value
Max. Buyer Migrations	3
Buyer Quote Timeout	5s
Buyer Accept Timeout	10s

Table 5.1: Parameter values for the Sealed Bid Auction mechanism

5.2.2 Basic Performance

The performance of the network of Sites and Servers, or "infrastructure" is investigated first. This allows us to set the key network parameters of mean degree and shout radius for later experiments.

Degree	Radius	Shout Load	Shout Load (ld 1.0)	msgs/node (ld 1.0)	msgs/node (ld 2.0)
2	2	7	12.4 (±2.98)	2.6 (±1.12)	5.7 (±2.33)
4	2	21	18.5 (±0.37)	3.7 (±0.17)	7.9 (±0.27)
6	2	43	34.9 (±0.53)	6.5 (±0.31)	14.3 (±0.59)
8	2	73	52.1 (±0.58)	9.4 (±0.45)	19.1 (±0.47)
10	2	111	67.2 (±0.58)	12.1 (±0.70)	20.1 (±0.57)
2	3	15	12.8 (±1.57)	5.8 (±3.21)	12.3 (±6.61)
3	3	40	27.3 (±1.08)	5.3 (±0.30)	11.3 (±0.61)
4	3	85	48.7 (±1.24)	8.9 (±0.45)	17.7 (±0.52)
5	3	156	67.2 (±1.25)	12.2 (±0.67)	19.5 (±0.54)

Table 5.2: The base network parameters investigated, and the measured message load on the system for load values of 1.0 and 2.0. The shout load is the mean number of Servers that process a broadcast/shouted message. Msgs/Node is the mean number of messages per server per second. Values in parentheses indicate the standard deviation for 100 runs. Note that for a load value of 2.0 and higher network values, the mean number of messages processed per node is near our set mean maximum of 20 messages per second, indicating that the infrastructure system is near full capacity. Also note that our simple shout load reducing optimisation (see Section 4.4.1) does successfully reduce the shout cost on the system. The larger standard deviations for mean degree's of two are due to the low network density at these levels, which means a Buyer must migrate more often, staying active in the system longer and increasing message load.

There is a trade-off between the network mean degree and shout radius, and the network performance overall. The more connections a site has, the more information it will have available, and the more traders on that node will be able to trade successfully. However, the more connections a node has the more time it spends processing messages, which leads to increasing message delays, which can affect the performance of the allocation system.

Additionally, with the concept of a broadcast message with a shout radius, additional message processing load is added to the system polynomially. For example, a network with a mean degree of 4 and a shout radius of 3 can actually cause a theoretical maximum

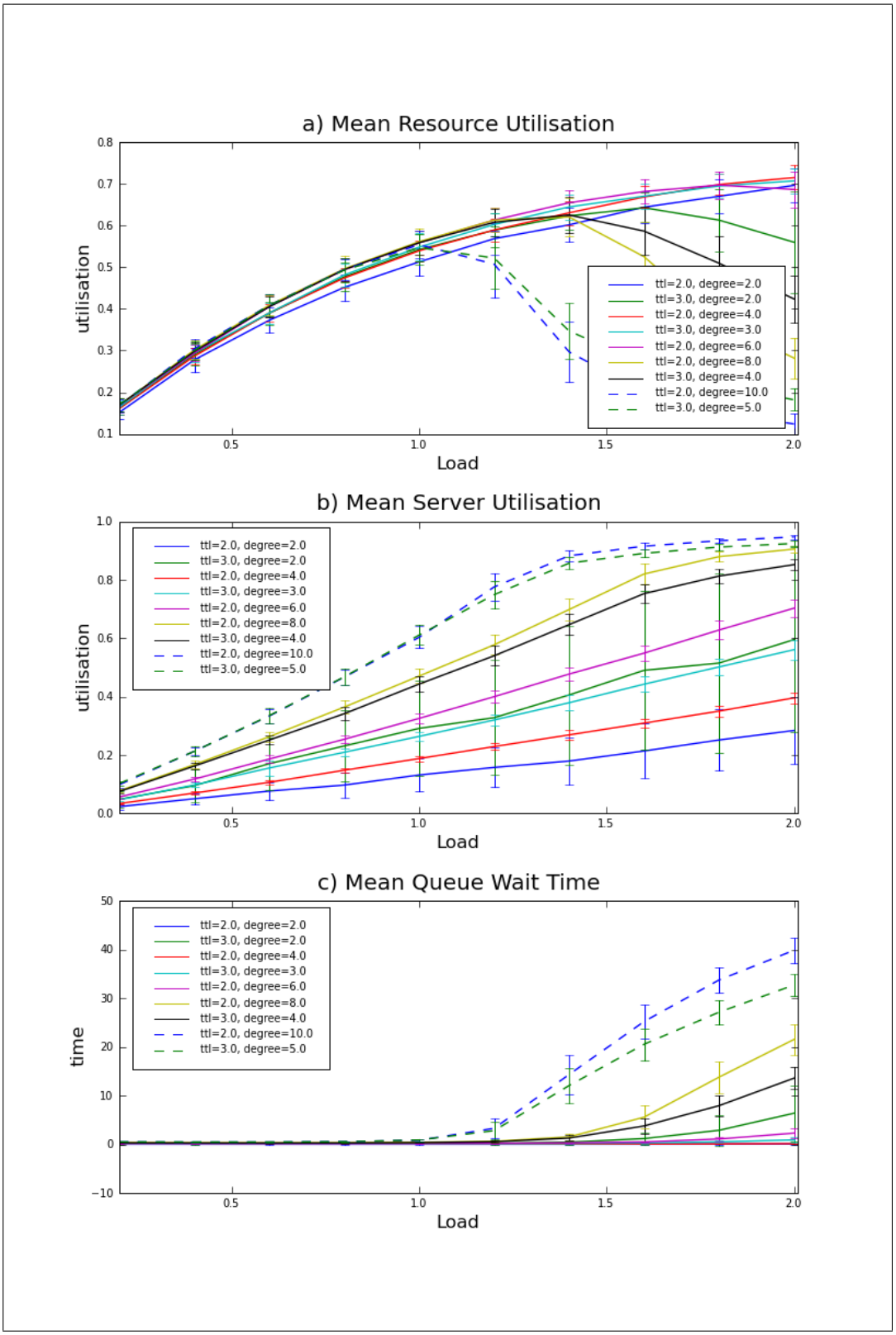

Figure 5.1: The mean server utilisation for the system against load and by mean degree and shout radius. The overall utilisation clearly tends towards 1 as the load increases for higher degree/shout radius values.

of $4^0 + 4^1 + 4^2 + 4^3 = 85$ Sites to process that message (given a trader will send any broadcast message to its own node as well as its neighbours). The actual message load was less in most cases, because of the duplication avoidance mechanism described in Section 4.4.1

A further impact on the message load of the infrastructure are market supply and demand conditions. Market A prices will mean more chance of not finding a successful trading partner, which could mean more migrations and thus more load on the system, whereas market D prices would provide a much greater chance of finding a suitable trade. As our goal in this instance is to investigate the infrastructure load, we use market A for these experiments.

We investigate our grid system under a variety of different load values, from 0.25 to 2.0. The key parameters that affect the grid's infrastructure performance are mean degree, shout radius and the Server's mean Server service time. We want to set these at levels that allow the grid infrastructure to cope with the messaging level for high load situations. We set the mean service time as in table 4.1. This value allows for a mean throughput of 20 requests per second. This is very modest compared to many systems, but we wish to allow for a wide variety of Sites to be represented, and thus keep the mean performance relatively low.

We vary the mean degree and shout radius as shown in table 5.2. Given the desire to investigate the effects of topology, we do not experiment with a shout radius of 1 (i.e a single hop), as this effectively negates the effects of any network topology characteristics on the results. Shout radius values higher than two quickly overload the network, so we experiment with shout radius values of 1 and 2. A sample network size of 100 was used to provide statistically viable results without overly impacting execution times. Table 5.2 also shows some results for the actual message load on a system under load values of 1.0 and 2.0.

Figure 5.1 shows the basic grid performance for the mean degree/shout radius values specified above. More values were tested but a selected subset have been shown to improve the clarity of the graphs. The other values tested followed in the same patterns as the ones shown, that is, lower values were virtually identical to the lower values shown, and higher ones continued to degrade the system performance further. For now, we do not discuss the actual allocation performance, as we are looking at the infrastructure. See Section 5.2.3 for a discussion on the allocation.

The main factor emerging here is that our simulated Servers can get overloaded easily for high mean degree, shout radius and load values. Particularly, the large increase in wait time as the Servers get too busy to handle messages, which adversely affects the resource

utilisation. This is especially the case for higher load values.

While higher values do offer marginal improvements in resource utilisation for load values around 1.0, the severe degradation at higher load values is not worth the gain. Based on these results, a standard mean degree of 4 and shout radius value of 2 seem the most appropriate to use for further exploration. These values essentially give the grid infrastructure the capacity to investigate what we are interested in (the resource allocation problem) rather than the performance of the infrastructure itself, which is not the aim of this work.

At a higher level, by setting relatively low mean degree/shout radius values, we continue to direct our work in its pursuit of investigating how simple systems, with limited agent knowledge, can solve the resource allocation problem.

It should also be noted here that there are a variety of changes that could be made to the model to allow for higher mean degree/shout radius values. For one, currently there is only one Server processing all the messages for a Site, which is an implementation abstraction. In reality, every trader will probably be running on its own computing resources to some degree or other. However, the central server will still be the "gateway" to the traders at that node, and will have to route and store all broadcast messages, if not the point-to-point accept/reject messages. Additionally, the facility to dynamically adapt the number of connections or change the shout radius value depending on market conditions (i.e. current load) would be important for a real system. This is discussed further in Section 7.3.2, but is not investigated, as the study of dynamic networks is both complex and not the stated focus of this work.

5.2.3 Grid Allocation Efficiency

We next investigate our prime performance interest, that of the allocation of resources. Using a mean degree of 4 and shout radius of 2, we investigate the allocative performance of the system for the various different market conditions described in Section 4.7.3. For these experiments, we fix our network size at 256 (see Section 5.2.6 for a discussion of network sizes).

Figure 5.2 shows the resource utilisation for a network size of 256 for each of market types A, B, C and D, against a range of load values. A theoretical optimum for each market is also indicated to aid comparison. For each market, when demand is lower (low loads), utilisation is close to optimum. As load approaches 1.0, we see that all markets except D fall below the optimal level - this is an indication of the loss of efficiency from lack of centralisation. As the system goes into over-demand (loads above 1.0), utilisation

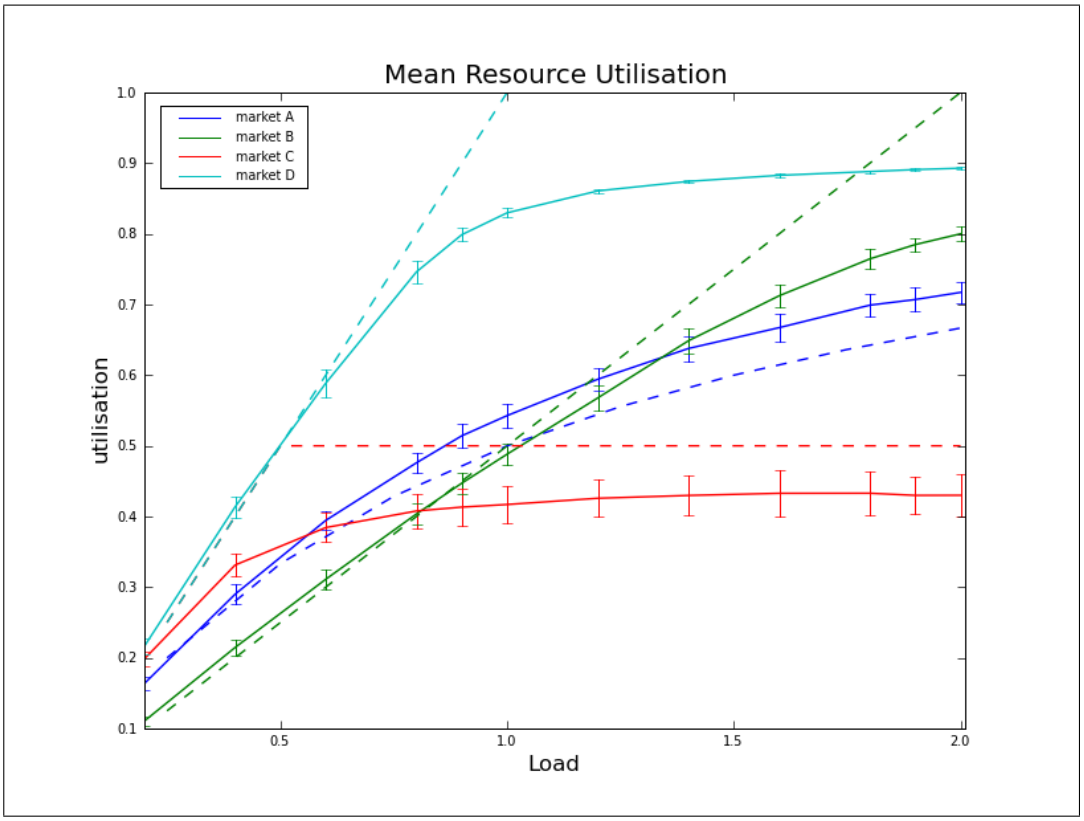

Figure 5.2: Resource utilisation for different markets by load. The solid lines indicate the measured utilisation of the grid for 100 runs, with error bars indicating the standard deviation from the mean. The dashed lines indicate the corresponding theoretical estimated maximum utilisation for each type of market. These theoretical levels are based on the assumption of a uniform distribution of job sizes and durations. See text for full discussion.

continues to increase slightly. This is because of the non-divisible nature of our job's sizes. As more jobs come into the system, there are more smaller sized jobs available to fill in the small "holes" still left in resources' capacities, so overall utilisation goes up.

An obvious factor here is that the measured utilisation actually exceeds the indicated estimated theoretical optimum. This is the case in markets B, C and D at the lower loads only, whilst in market D it is typically around 3-5% above the optimum. The reason for this is that the calculation for the estimation is misleading. The calculation is based on the market supply/demand curves described in Section 4.7.3. It assumes a uniform distribution of supply quantity values (i.e. job sizes and durations), which is not the case, as we use a gamma distribution for these values. Thus although the mean value for job size is about 20 (which is what the calculation for the optimal value uses), the modal value is somewhat less (about 16-17). Likewise, while the mean job duration is 100, the mode is closer to 75. The fact that the majority of quantity values ($size \times duration$) are under the mean value, and thus will be able to fit in the resource "holes" better and not take as long to finish, explains the ability of the utilisation to be above the theoretical optimal, as more jobs can be executed. Additionally, our resource utilisation is measured from after the first 100s of execution, to avoid calculating the first job duration interval when the grid has by necessity not reached the desired load value.

Now that we understand why market A can apparently perform super-optimally, it is clear that given market A conditions (which are the closest to potential real world conditions), the system performs very well in terms of allocating quantities. The other market conditions perform less well, although all achieve a significant proportion of the theoretical optimal (above 75% in all cases). The results for market D, given its lack of price constraint, indicate the fact that our system does not achieve an optimal allocation when based on quantities alone due to its decentralised nature. Section 5.2.6 highlights the benefits decentralisation provides.

To further understand the system behaviour, we look at the different reasons a job can fail to be allocated, and examine them for each market. A Job can fail because of quantity constraints (i.e. finding enough spare capacity) or price constraints (i.e. the Buyer's limit price is too low). Additionally, the fact that traders can only access a limited subset of the potential trades affects both these reasons.

5.2.3.1 Job Size

Given that our Buyers' desired Job sizes are indivisible, a Buyer can fail because there is currently not enough free capacity with the network of traders that can be reached. This is especially true when the job size is large, or the load is high.

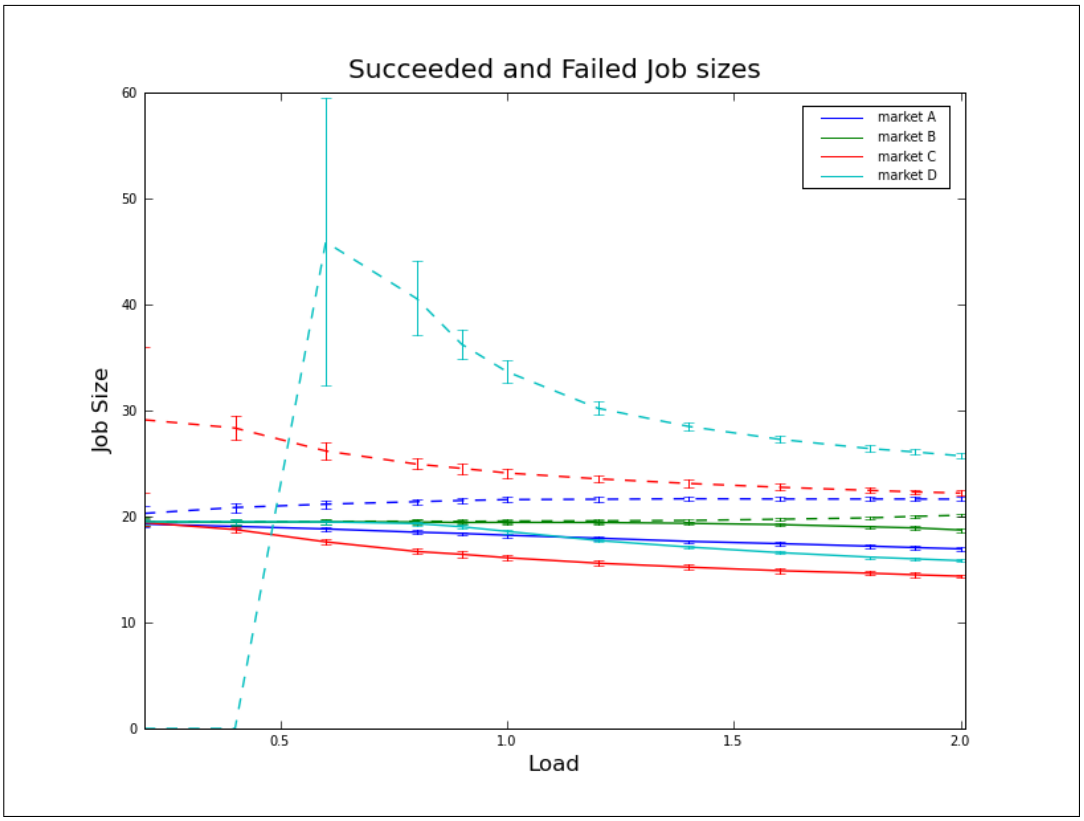

Figure 5.3: Mean Job size for failed/succeeded Jobs for different markets by load. The solid line indicates the successful jobs, whilst the dashed line indicates the failed jobs.

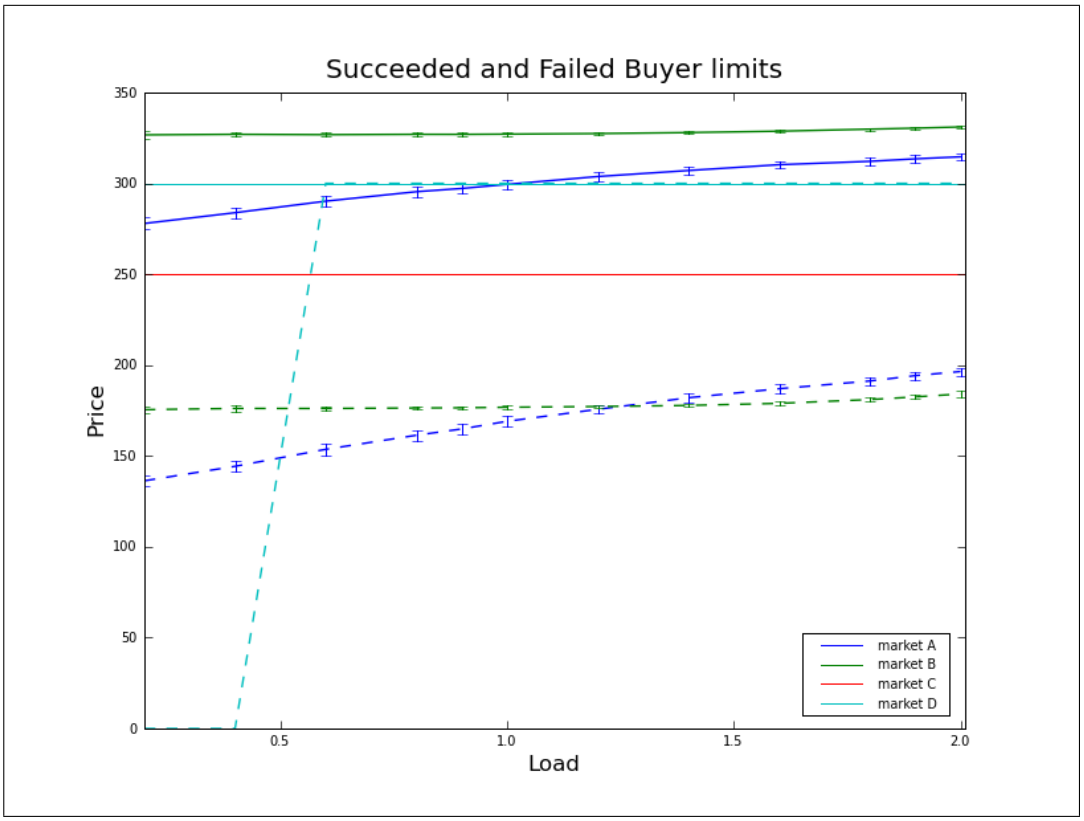

Figure 5.4: Mean Buyer limit prices for failed/succeeded jobs for different markets by load. The solid line dotted line indicates the successful jobs, whilst the dashed line indicates the failed jobs.

Figure 5.3 shows the sizes of successful and unsuccessful Jobs for each of the markets. From this figure we can see a pattern of behaviour. In general, mean failed job sizes are significantly higher than succeeded sizes, as we would expect. However, as the different markets provide different price constraints, the impact of size on job failure varies. In general, we would expect impact of size on job failure to be greater as load increases, with succeeded sizes decreasing as high loads. This would be indicated by a divergence of succeeded and failed sizes with load.

- Market A shows a clear divergence between succeeded and failed sizes. The succeeded sizes continue to drop as load increases, but the failed size levels out at around 22 at high load. This indicates that Job size is not as key a factor in allocation failure in this market as it is in others (e.g. market D), but still has some impact. This is understandable given the supply/demand curves of Market A.

- Market B is similar to market A, but with much less divergence and only at higher load levels. This implies that Job size has very little impact on job success in this situation, which given that Market B is the harshest price-wise, is intuitive. It is only at very high loads that size makes any appreciable difference.

- In Market C failed and succeeded sizes do not diverge, they are close to constant. Additionally, the difference is much greater than markets A and B. This indicates that size has a larger impact on failure, with large jobs finding it harder to get successfully allocated.

- Market D has the least price constraints, and as expected size is a key factor of success. The difference between succeeded/failed prices is the largest, and there is some convergence at higher loads. Note that at the lowest load level (0.25, 0.5), no jobs failed. This indicates that when supply is less scarce market D has no size constraints, as expected.

The effect of size on allocation success has most impact on market D, then C, A and then least impact on B. Markets A and B have a similar behaviour, as do C and D. The main difference between the two behaviours is that limit prices for Buyers are flat in markets C and D, but sloped in A and B.

5.2.4 Limit Price

Figure 5.4 indicates the Buyers' limit prices for successful and failed jobs. Given that in markets C and D Buyer limits are fixed, we only show markets A and B. The figure

shows that for both markets A and B, price is a significant factor in success of Job, with a large difference between the limit prices of successful and unsuccessful jobs. For market A, the price generally increases as expected given the increased competition at higher loads. However, the price for market B is flat as to be expected from the flat supply prices of market B. The small increase at loads 1.75/2.0 is because at loads approaching 2.0, we can see from the supply/demand curves of market B (see section 4.7.3) that the equilibrium price can be higher than $250, given the random nature of the jobs' sizes. This gives rise to slightly inflated succeeded/failed prices in this case.

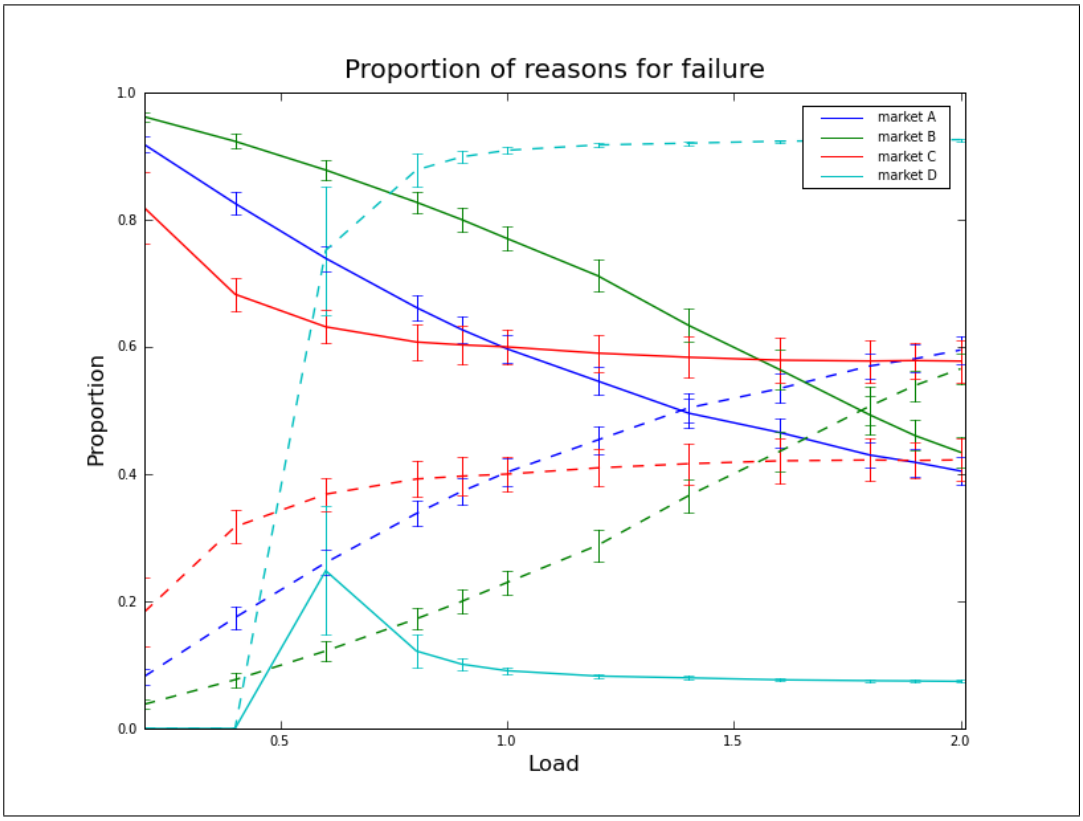

Figure 5.5: The mean proportions of reasons for Job failure. The solid line indicates the proportion of failures due to the Buyer's limit price i.e. the proportion of asks received that were above the Buyers limit. The dashed line indicates the proportion of potential asks that were lost because the Seller's resource was too busy.

However, this is not the full picture of the impact of limit price on failure or success of allocation. In markets C and D particularly, given the flat Buyer limits, the figure does not grant any insight. Figure 5.5 shows the proportions of different reasons why a Job failed to generate any valid asks. This can either be from a Seller not sending asks due to not having enough capacity, or an ask being rejected because its price is higher than the Buyers limit price. Again markets A and B show a similar behaviour, in that for low

loads, price is the key reason for failure, but as load increases size becomes more of a factor. Markets C and D show very different behaviour, however. In Market C, price is always the main reason for failure, although the proportions converge as load increases, implying size does become more important at higher loads, as expected. In Market D, the opposite is true - size is the dominant reason for failure throughout. Another factor for markets C and D is that the simple ZIC algorithm has been shown to be unable to reliably find equilibrium in these market settings, which contributes to the lack of price adaption.

5.2.4.1 Node degree

If a Buyer is at a node with a low degree, this reduces the trading opportunities and increases the likelihood of failure from either price or size constraints.

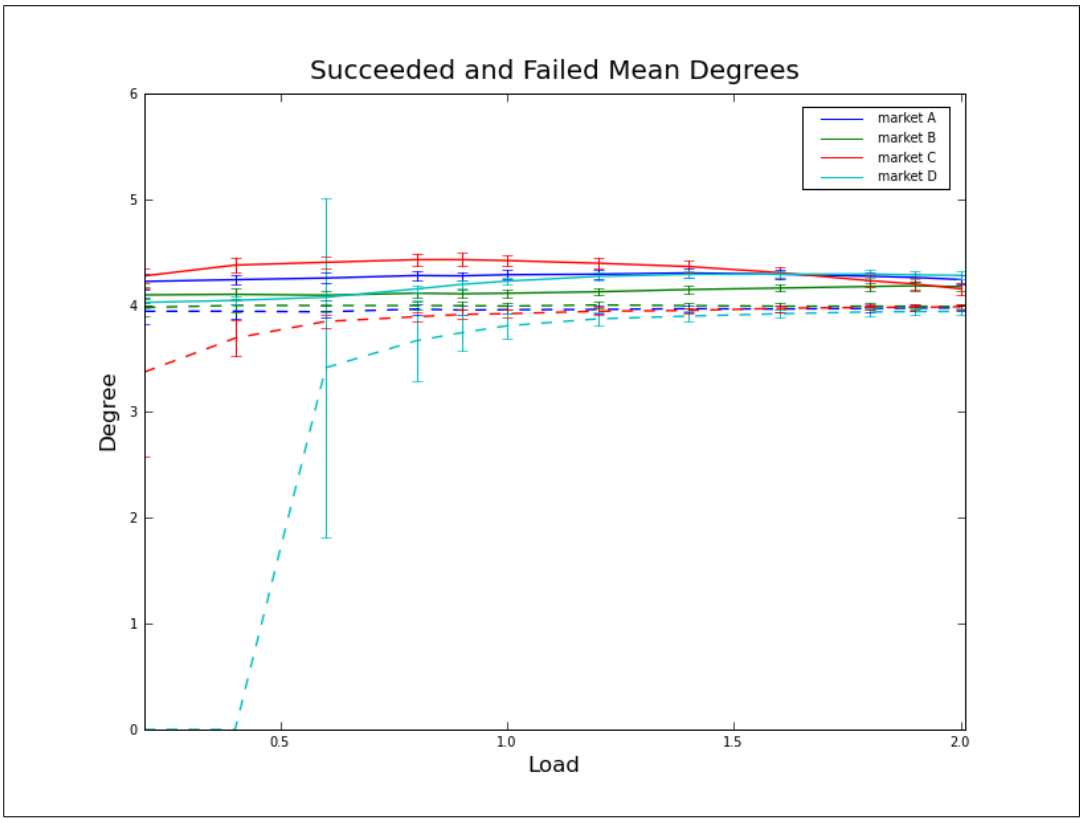

Figure 5.6: The mean degree of the nodes for failed and successful Jobs. The solid line indicates successful jobs, whilst the dashed line indicates the failed jobs.

Figure 5.6 shows the mean degree of nodes for successful and unsuccessful jobs. While it is clear that successful Jobs have higher node degree, there is not a significant difference (around 1 additional connection on average), especially at higher loads levels. At lower load levels, the slight drop is due to there being relatively few failures for markets

C and D at the lower load levels, so the Buyers that did fail were particularly limited.

5.2.5 Economic Performance

Our concern so far has been primarily with allocation of quantities. In an economic system, quantities are only half the story. In this section the economic system as a whole is considered, including price and quantity.

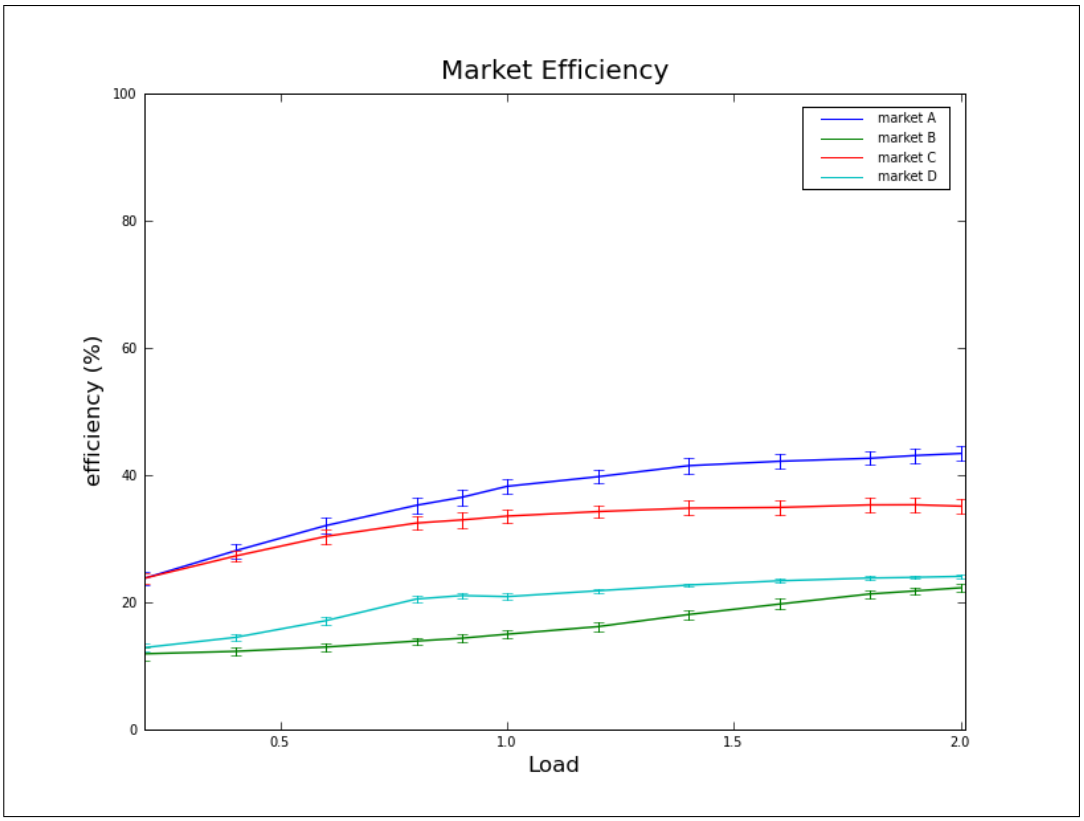

Figure 5.7: The mean economic efficiency of markets by load for markets A to D.

Figure 5.7 shows the economic efficiency of the system for markets A-D, calculated as the percentage of total surplus, and overall efficiency levels are low. Lower efficiencies are to be expected given the fact that our system is decentralised and based on the simple ZIC traders.

One factor that leads to this is our fixed indivisible sizes. Trades that might be possible from a price point of view are not achieved due to the quantity constraints. Additionally, given that larger quantities are more likely to fail, and thus add larger amounts of lost surplus. The smaller jobs that do succeed add less surplus respectively to the achieved values. The general slight rise in efficiency as load increases is due to the increased opportunity to utilise the capacity that's still free by the over demand of jobs.

As market D has the largest potential surplus given its lack of price constraints, we would expect for its efficiency to be lower than that of Market A, whose supply/demand curves dictate a much lower achievable surplus.

Interestingly market B is lower than market D. This is due to the much reduced ask price window (between $250 and $500) when Seller limits are flat and high, making it impossible for 50% of traders to trade, and more difficult for the other 50% who might be able to trade.

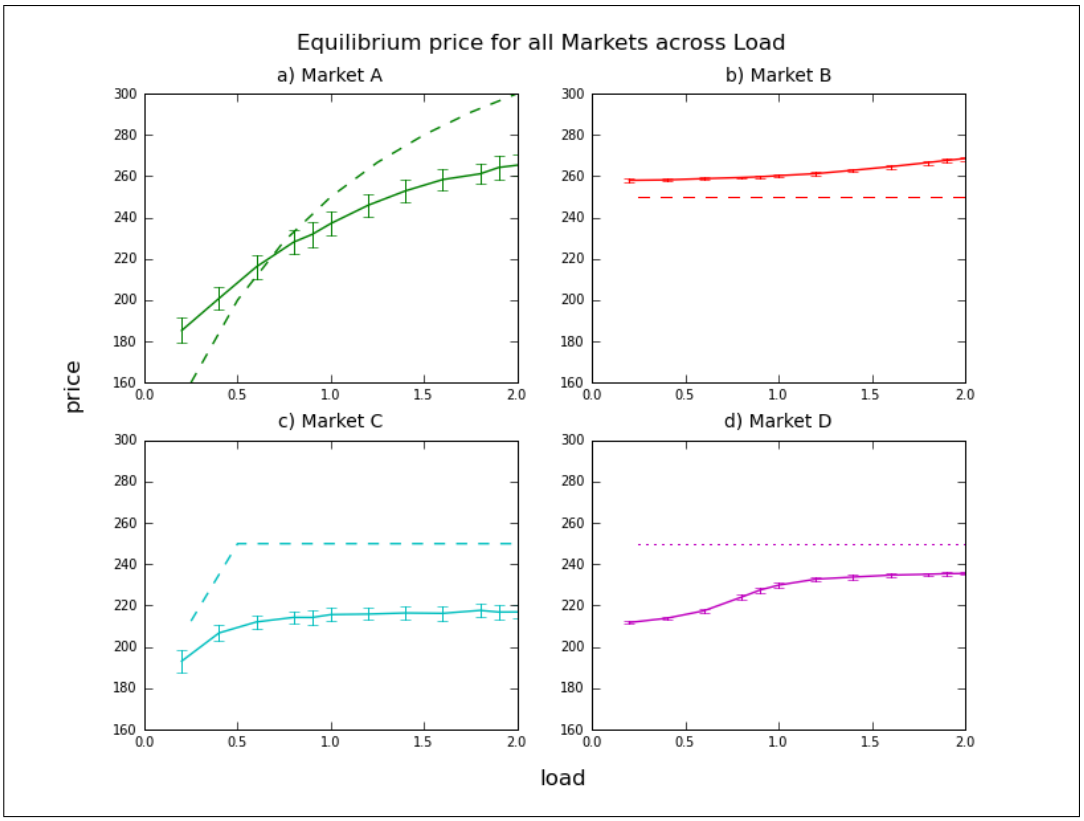

Figure 5.8: The actual equilibrium price achieved by load for markets A to D. The solid line indicates the achieved price, whilst the dashed line indicates the theoretical equilibrium price. Note that the theoretical price for markets D is the same as market B.

Figure 5.8 shows the mean achieved equilibrium prices along with the theoretical equilibrium prices for comparison. For all markets the increased demand from increasing load does drive the price of resources up to some degree, showing that even with ZIC traders, the market does adapt to different conditions. The level of adaption however is variable. Market C does not react to the under-supply situation at high loads, keeping the same equilibrium price as a load of 1.0. This is due to the fact of the flat buyer prices for market C, as once all possible supply is sold (which is around a load of 0.5 for market C), increasing the quantity does not bring pressure to increase price.

Markets C and D are consistently below the theoretical equilibrium. This is to be expected of market C, as the maximum tradable price is the theoretical equilibrium price. For market D however, this is likely an indication of the Buyers having an unequal effect on the price, due to there being many more buyers. Additionally, Market D is the type of market that ZIC traders have been shown to under-perform in, as there is no guidance towards the equilibrium from the supply/demand curves. Market B is as expected constantly above the equilibrium price, as the minimum price is the equilibrium price.

The most accurate is market A, with the "standard equal" and opposite supply/demand curves it provides the most guidance to the market towards equilibrium price. It provides the closest fit to the theoretical equilibrium price.

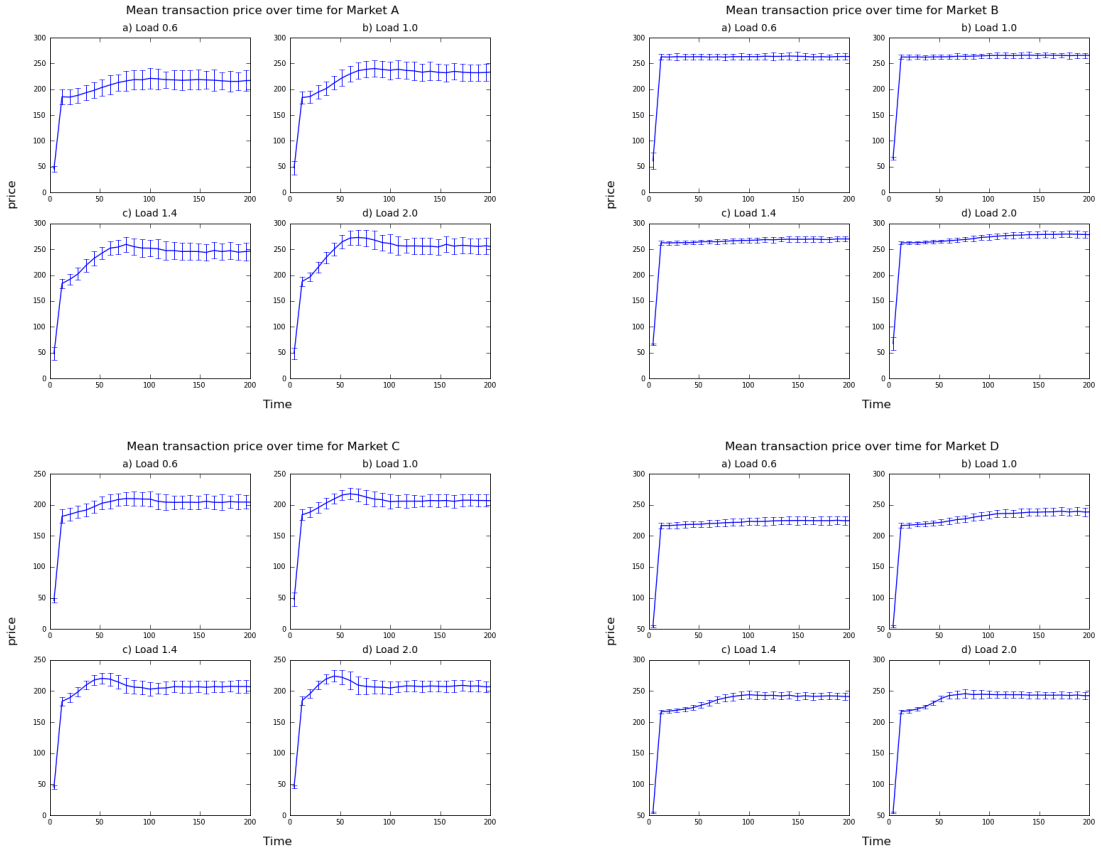

Figure 5.9: The periodic average transaction price over simulation time for a range of load values across all markets.

Figure 5.9 shows the mean transaction prices over time for selected loads for each market, as another view into the system's equilibrium state. Here we can see the initial period of price adaption, and then the prices roughly stabilise. Figure 5.10 shows the time series variation of Smith's alpha values for a range of load values for each market. It

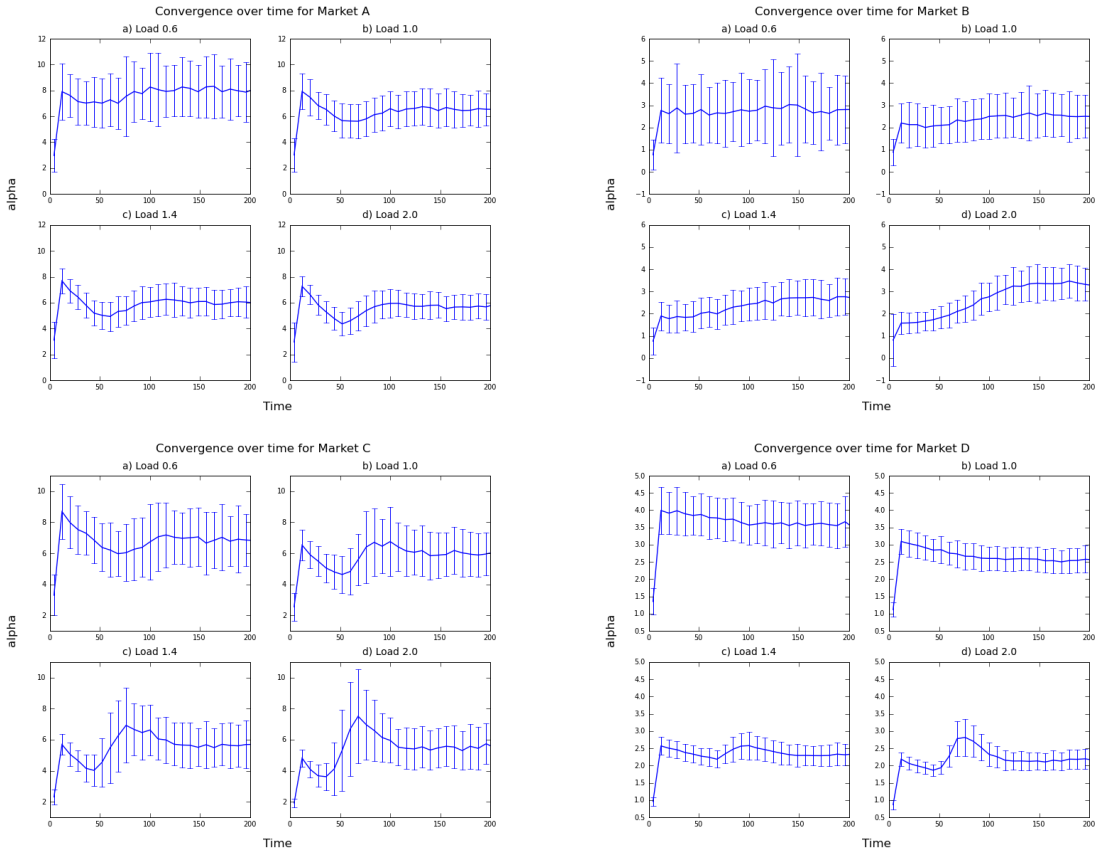

Figure 5.10: The periodic averaged alpha values over simulation time for a range of load values across all markets.

confirms that all markets do converge over time towards the equilibrium price, reaching a stable point after about 100s on average, this being one "period" of load on the system.

Interestingly, the alpha values drop below the final level before returning up to it. The reason for this is that the first 100s period is then initial "seeding" period. As the Buyers arrive in the system sequentially, the first 100s are always in an over-supply situation initially. This decreases as we approach 100s at which point the system should be stable with the specified load value and market type dictating over/under supply. This initial supply situation over exaggerates the initial convergence, and it is consequently readjusted as the system "fills up" to the requisite load.

In general, high load values mean a slightly lower alpha value. This is due to the increased number of trades that are happening after the price has stabilised. For higher load levels, the initial period where the price is adjusting has less trades occurring due to the initial interim period, and this reduces the number of trades that are further from the equilibrium.

The notable exception to this is market B, in which the higher loads produce a worse (higher) alpha value. However, the standard deviation for all alpha values for market B is much greater than that of other markets, which suggests that market B does not reach as stable a state as the other markets.

5.2.6 Scalability

One of the key goals of this work is developing a scalable system, and it is that aspect we explore in this Section. Having set the values of mean degree and shout radius at 4 and 2 respectively, we now vary the size of the network against the same load values. We test two market conditions here, markets A and D. In Market A, likely trades are more price-constrained than in Market D, as we see in their relative allocation performance measures.

Figures 5.11 and 5.12 show the results for a variety of network sizes for these markets. It is clear that the system does scale as it was intended to, both in terms of the resource utilisation and the grid infrastructure. Similarly, Figure 5.13 shows that size does not effect the market efficiency either.

While this is a simple result to show, it should be noted that no current grid allocation systems scale to this level, in simulation or reality. This strongly suggests that the scalability benefits of a decentralised economic systems are worth further investigation. Given the scalability characteristics of the system, we use a network size of 256 for the rest of our investigations, as smaller sizes produce lightly more variance in the results, and larger

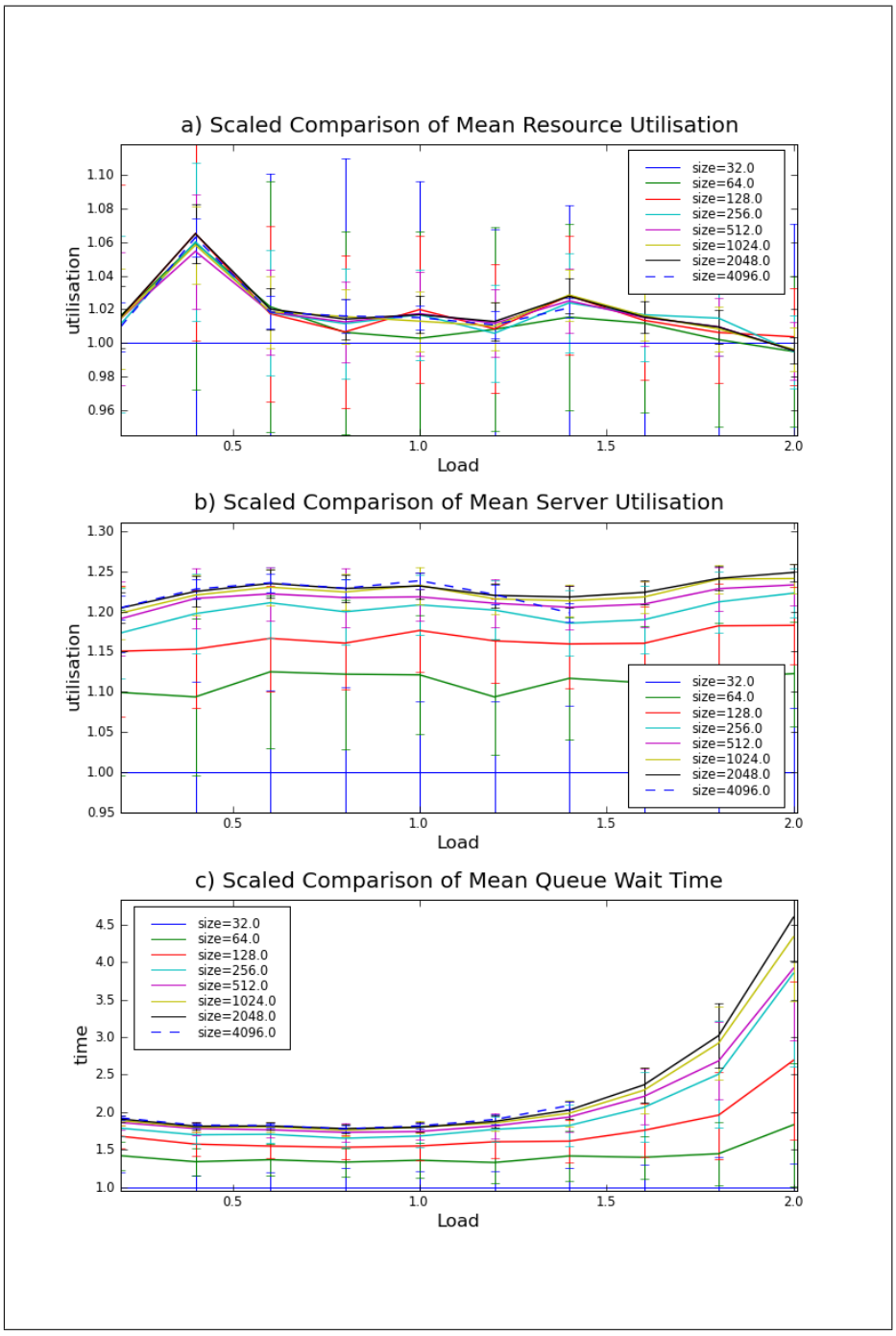

Figure 5.11: The mean resource and server utilisation for the system against load and by size in market A. The results are scaled to compare against a size of 32 as the base level for each load value. The main observation here is the scalability of the resource and server utilisation, as all sizes are indistinguishable in resource utilisation. A Kruskal-Wallis ANOVA on the resource utilisation on sizes show the utilisation levels were were statistically identical ($n = 100$, $0.1 < p < 0.9$ for all loads). However, using the Mann-Whitney U test to compare each size sample to each other indicated that for a size of 32 utilisation was significantly lower at lower load values ($p < 0.05$ for $size = 32$ and $load < 0.6$). This is due the fact that the performance suffers from the very small network size (and consequently increases the issues with discrete non-divisible job sizes).

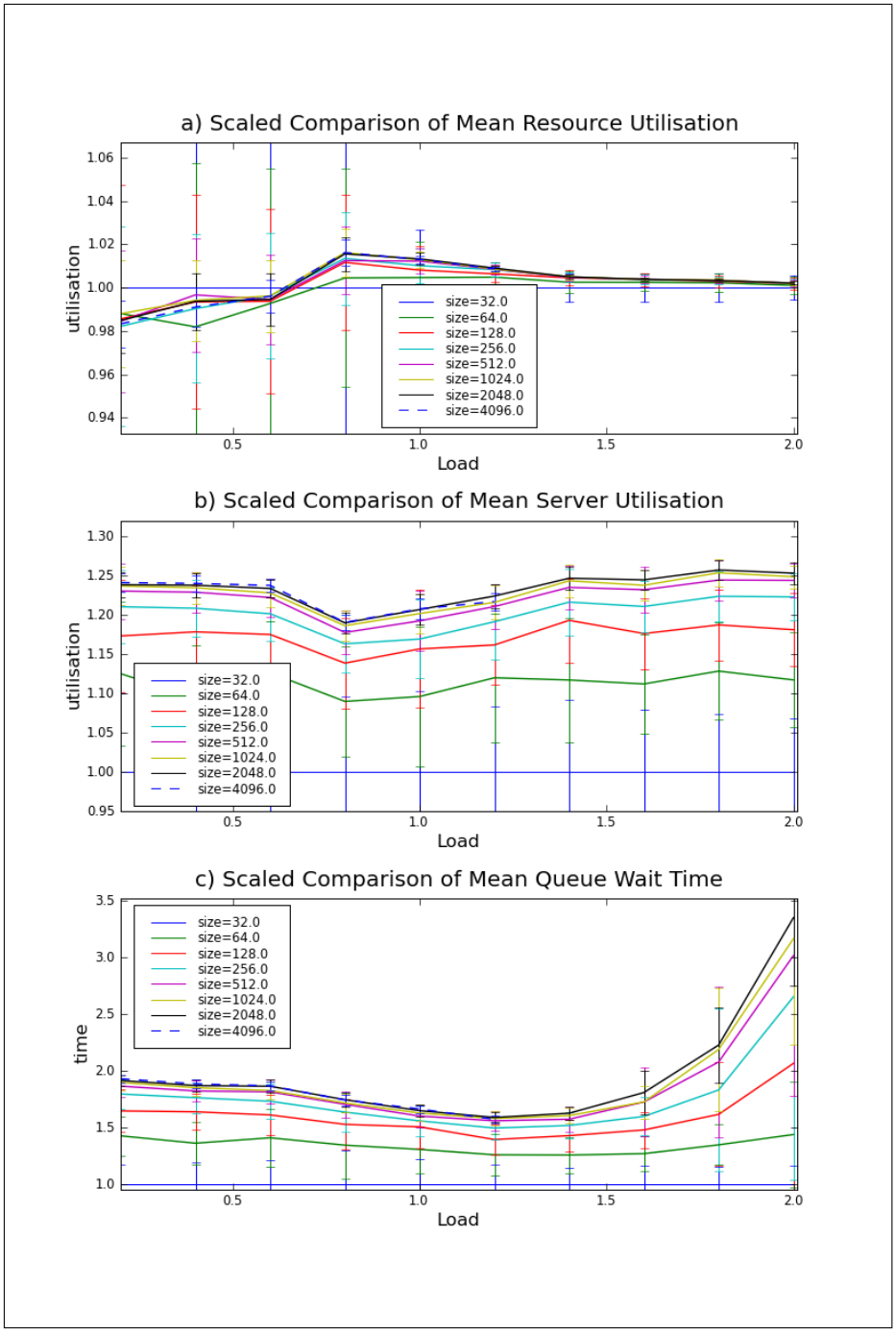

Figure 5.12: The mean resource and server utilisation for the system against load and by size in market D, scaled as in the previous figure. Again, the main observation here is the scalability of the system. Interestingly, larger sizes do perform slightly better in this scenario at higher load levels. A Kruskal-Wallis ANOVA shows utilisation is identical for low load values ($n = 100, 0.2 < p < 0.8$ for $load < 1.0$) but statically slightly higher for higher loads ($p < 0.05$ for $size = 32$ and $load >= 1$). Again this slightly higher utilisation (less than 1%) is due to the discrete Job sizes being less likely to hit limits due to the increase number of Resources.

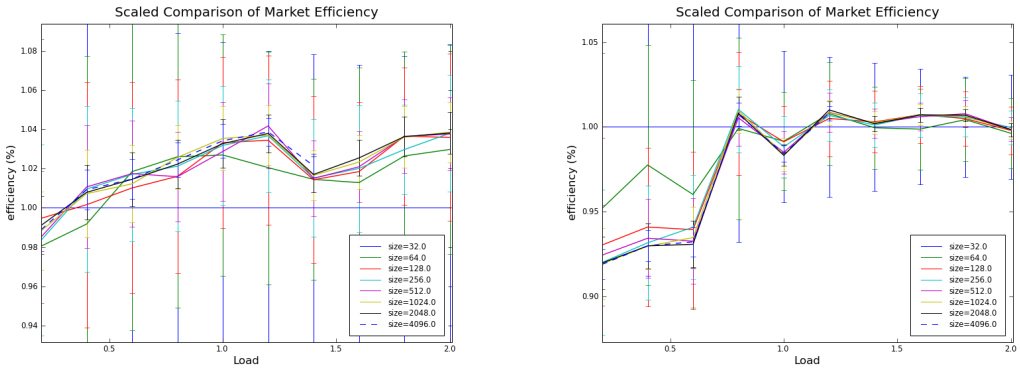

Figure 5.13: The mean market efficiency for the system against load and by size for Market A on the left and Market D on the right. Kruskal-Wallis ANOVAs show efficiency is similar for all sizes when load is low ($n = 100, 0.1 < p < 0.5$ for $load < 1.0$), however individual Mann-Whitney comparisons indicate that that higher loads, a size of 32 once again is lower by 2-3%($p < 0.05$ for $size = 32$ and $load >= 1.0$).

sizes take longer to execute.

5.2.7 Summary

The basic results indicate that the system provides a reasonable trade off between an efficient allocation and a scalable allocation. A high degree of scalability is achieved at the cost of a slightly less than optimal allocation.

5.3 CDA Auction

In this Section, we develop a decentralised CDA auction mechanism, and explore the use of ZIP traders in such an auction.

5.3.1 CDA Implementation

Implementing a decentralised CDA is more complex than a sealed bid auction, due to public quoting. Each trader must "shout" its quote out to its neighbours, which is then forwarded on up to the shout radius. Additionally, both Buyers and sellers can quote a price, whereas only Sellers quote in the sealed bid implementation. However, unlike actual shouting, where it is impossible to selectively shout to some and not others, our electronic shouting is over a point-to-point link, so it can be selective.

For all traders, we implement the shouting of quotes using a repeating timeout, or shout period. That is, every shout period, a trader who is capable of trading shouts his current price to all neighbours. Sellers commence this regular shout after a short random delay, to avoid synchronisation, so they are all shouting regularly but out of phase. Buyers, however, do not shout as soon as they enter the system - they wait for one shout period in order to ascertain the state of the market. At the end of this period, they can either commence shouting, or accept any asks they may have heard during this initial period. This delay is added to that a new naive trader that joins the market does not accept the first acceptable price to come its way - it waits until it has a more accurate picture of the current market. It does however increase the allocation time somewhat compared to the sealed bid auction.

When a trader hears a quote, it checks to see if the quote is valid, i.e. the price and quantity are acceptable to the trader. For Buyers, this means the quote price is below their current price, and has enough size to perform their job. For Sellers, this means the quote price is above their current price, and they have spare capacity with which to do the job. If the quote is valid, it sends an accept else it ignores the quote. If a Buyer accepts an ask, he becomes inactive, not responding to any other asks, and ceases to shout out his quote, although continues to observe the market. If the accept is rejected or times out, the Buyer becomes active again and starts shouting and trading. If a seller accepts and bid that would leave it no spare capacity, it becomes similarly inactive, until an executing job completes or the accept is rejected or times out.

Accepts are handled in the same manner as the sealed bid auction. That is, a private message is sent between the traders. Usually in CDA, the acceptance of a quote is public knowledge, either by the other traders hearing the acceptance shout in an open outcry

auction, or by the information presented by a central auction mechanism. In a P2P setting, this is difficult to achieve without adding some degree of centralisation, as we cannot trust the traders to shout their acceptance - they may signal acceptance just to the other trader, or even shout out false accepts to influence the market. Whereas with a private acceptance, no trust is needed between traders to behave honestly. This is another trade-off of removing any centralised component.

5.3.2 ZIP implementation

We explore the CDA with ZIC traders, and compare to the sealed bid mechanism. Unlike the sealed bid model, a CDA has public quoting, so we can also attempt to use a simple ZIP algorithm to improve the trading process.

However, there is a difficult problem to overcome. The ZIP algorithm (like all other trading algorithms discussed in Section 3.2.2) relies on the knowledge of the success or failure of a particular quote in the market. Usually, this is provided by a centralised mechanism, which we do not have. In theory, traders could shout accepts in the same manner as an open outcry CDA. This would allow other traders to observe the success or failure of a quote, and use it to update their ZIP model. However, this is difficult to enforce in a decentralised fashion, as discussed above.

To attempt to solve this problem, we make use of Despotovic's decentralised ZIP trader [37] (see Section 3.2.2.2), which we label ZIPD. ZIPD first uses ZIP's delta rule to observe the price of buys and asks separately. This is the estimate bid price and ask price, indicated at time t by $bids_t$ and $asks_t$ respectively.

When observing a quote, it first calculates an estimated completion price p as the mid-point between the quote's price, and the observed values. For a bid, $p = \frac{bids_t+q}{2}$ (where q is the quote's price and $bids_t$ is as above). For an ask $p = \frac{asks_t+q}{2}$ (q is the quote's price and $asks_t$ is as above).

If the quote's price is better than the currently estimated opposite quotes price (for a bid, if $q > asks_t$, or for an ask, if $q < bids_t$) then the trade is estimated to have succeeded at the estimate trade price p, else failed at price p. This estimated success and price p are then used to update a standard ZIP algorithm to track the price changes.

By removing the need for explicit knowledge of success or failure of a quote, ZIPD allows the use of some ZIP-like intelligence in a completely decentralised manner. The fact that success or failure is estimated rather than observed will likely mean that the ZIPD does not perform as well as centralised ZIP agents, but it does actually add some intelligence beyond the basic ZIC agent used so far.

In the next section look at both ZIC and ZIPD in a CDA market.

5.3.3 CDA with ZIC Traders

We examined the CDA auctions under the same conditions as our sealed bid investigations. That is, a mean degree of 4, shout radius of 2, a random network of size 256 and ZIC traders. The accept timeout parameter is the same (10s), as is the number of migrations (3). A new parameter was introduced, the shout period, being the time between regular quote shouts on the network. We tested a variety of values for this value in simulation, a lower period meaning better allocation performance, but more message load on the system. A value of 10s was used for these experiments, as that value was the lowest that would not overload the messaging infrastructure at high load levels. As before, the CDA was examined over varying load levels and market situations, for 100 runs.

Figure 5.14 compares the system allocation performance of the CDA auction. In general, the CDA does not perform as well as the sealed bid. In markets A, B and C, the resource utilisation is significantly lower than the sealed bid mechanism. Market D is comparable, but the utilisation is slightly less. The mean server utilisation is higher for the CDA, as would be expected given both traders are shouting regularly. The queue time increases significantly at higher loads, as many more messages are being shouted.

Figure 5.15 compares the CDA's economic performance with that of the sealed bid auction. Interestingly, the equilibrium price is higher in the CDA than in the SB, although it follows the same pattern across load values. This is understandable, given a more equal exposure of price information from both Buyers and Sellers. Market efficiencies are lower with the CDA for all markets except D.

Figure 5.16 shows the allocation times for the sealed bid auction and the CDA. Clearly the CDA takes a lot longer to perform the allocation process. This explains the lower measured resource utilisation to some degree, as well as the lower market efficiency, as jobs are taking much longer to allocate in the CDA, thus taking longer to "fill up" the grid, which lowers the utilisation results.

There are several factors that underlie the CDA's performance. Firstly, the time taken to allocate is very long, especially compared to the actual run time of the job itself. This means that jobs persist in the system much longer, so increasing the load as there are a larger number of active Buyers. Clearly, this is contrary to the argument for using a CDA in the first place - that of fast adaption.

The second factor is a key difference between the two auctions mechanisms. In the sealed bid auction, Buyers receive a number of quotes before responding to one of them.

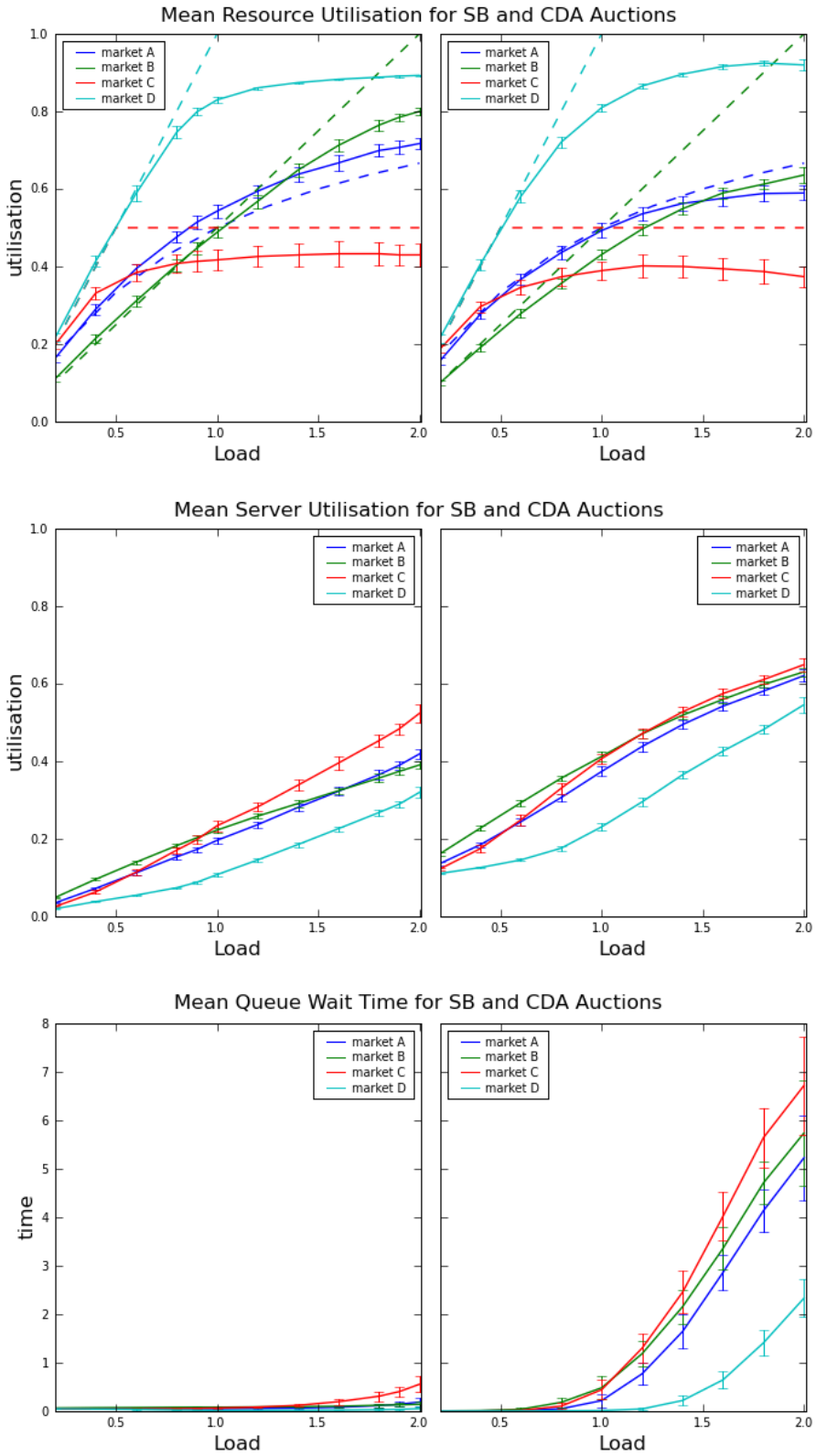

Figure 5.14: The above figures show results from the sealed bid mechanism on the left and the CDA on the right. The top figure shows resource utilisation, the middle shows server utilisation and the bottom figure shows mean server queue time. Presentation is as previous figures for these values.

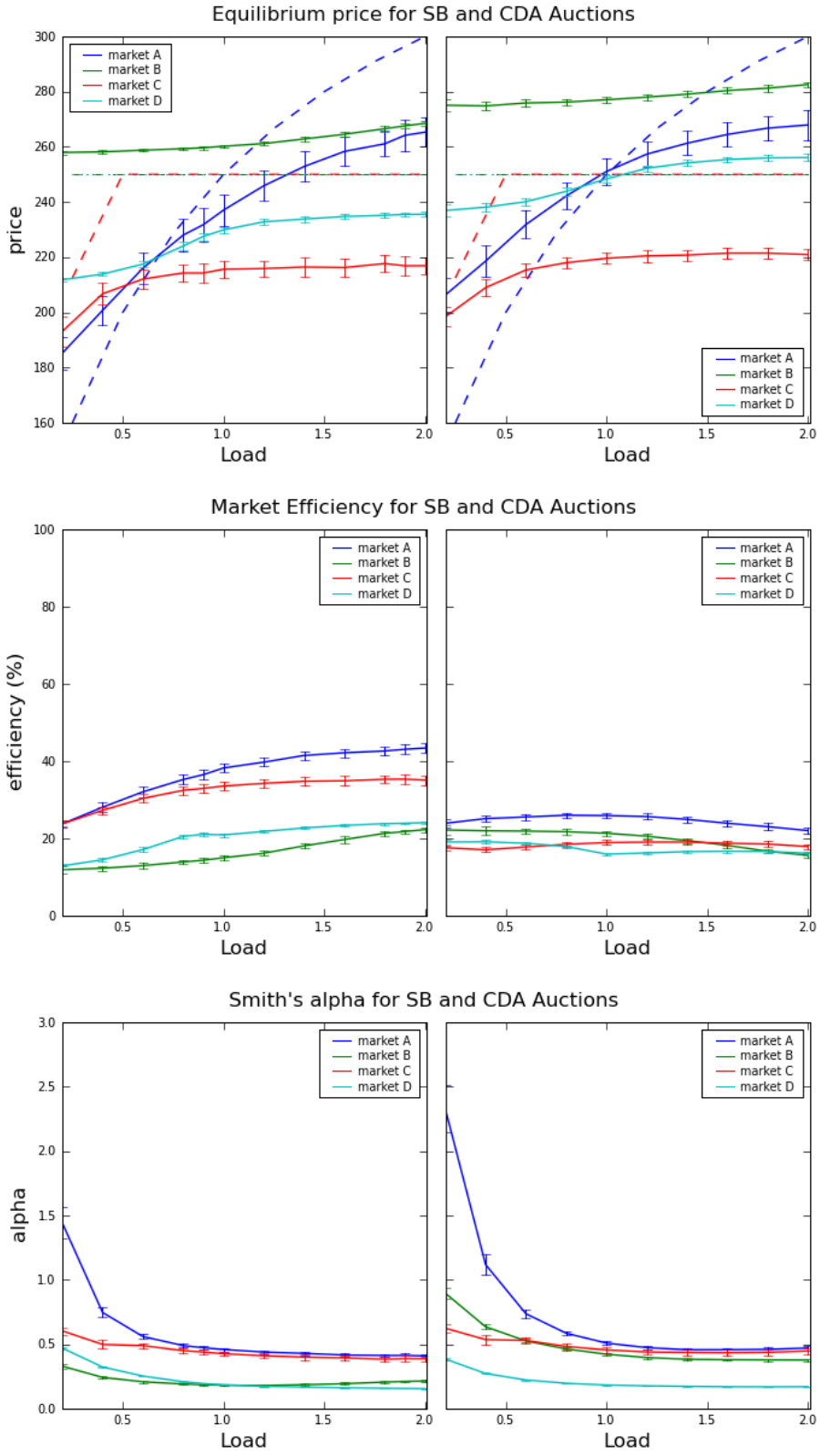

Figure 5.15: The above figures show results from the sealed bid mechanism on the left and the CDA on the right. The top figure shows the equilibrium price, the middle shows market efficiency and the bottom figure shows the alpha values. Presentation is as previous figures for these values.

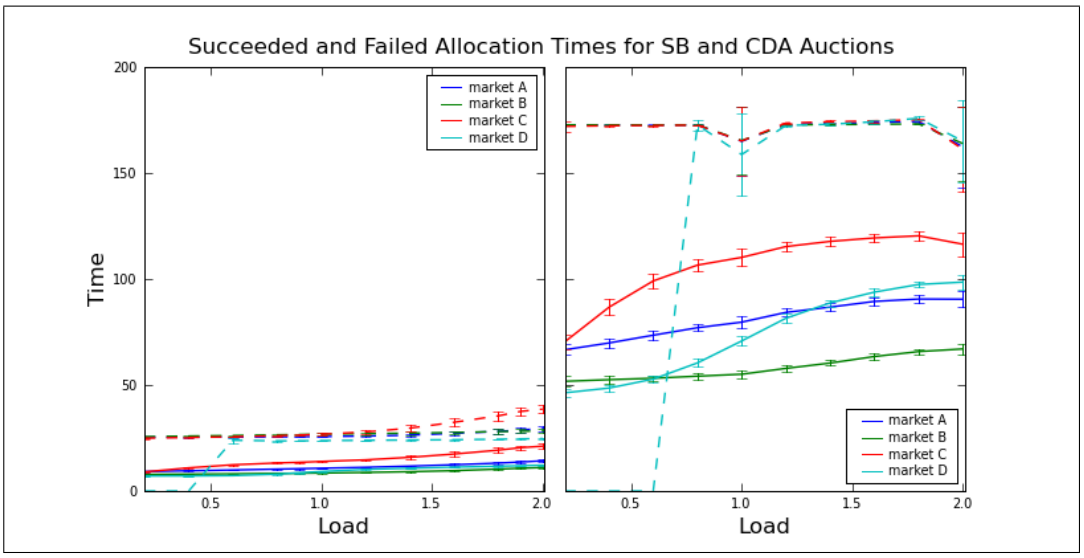

Figure 5.16: The allocation times for sealed bid auctions and CDAs. The solid lines show the succeeded times, dashed lines show the failed times.

In the CDA, all traders respond as soon as a quote is heard that is satisfactory. This eager bidding will likely result in more traders accepting the same bid at roughly the same time. For example, suppose a seller shouts a quote, and five Buyers accept this quote as soon as they hear it. The first Buyer's accept to arrive at the Seller will win the quote, and the other four will lose out. In the mean time, the losing four Buyers have not been actively trading, waiting for a response, so may have missed other opportunities, and have to wait for another acceptable quote to arrive. This partial synchronisation of the bidding is detrimental to the process, and adds directly to the greatly increased allocation times.

5.3.4 CDA with ZIPD Traders

Here we investigate the ZIPD agent strategy. We set the parameters for the internal ZIP agent to be the same as the initial ZIP work [26].

To verify the convergence behaviour of the ZIPD traders we look at the behaviour of ZIPD Buyers with constant Sellers, and of constant Buyers with ZIPD Sellers. Figure 5.17 shows the price and trader values for a single run, for both situations. The ZIPD traders observation of asks and sell price accurately follows the curve of quote prices. The ZIPD Sellers converge well to the Buyers' fixed price, however the ZIPD Buyers underestimate the Sellers price and offer below the fixed price. Figure 5.18 shows the same metrics when both Buyers and Sellers are ZIPD agents again for a single run. Here we see that the agent's prices are diverging rather than converging towards the equilibrium. They reach a steady state that is quite far from the equilibrium price.

Figure 5.17: The mean price for ZIPD agents in the CDA over time for a single run of 512 resources, a load of 1.0 and Market A. The left figure shows the Buyers as ZIPD agents with the Sellers at a fixed price of $200, while the right image shows ZIPD Sellers with the Buyers at a fixed price of $300.

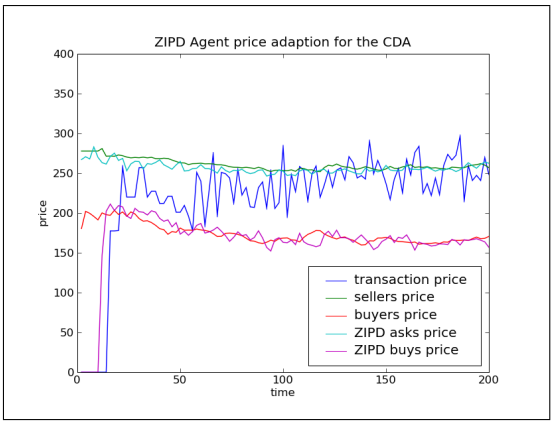

Figure 5.18: The ZIPD trader mean price factors for ZIPD agents over time for a single run of 512 resources, a load of 1.0 and market A.

The ZIPD traders' estimation of market success seems to be inaccurate in our CDA market. As little further details were published in the original work, it may be that some assumption was made that we are not aware of. It is likely that the lack of sophistication in our CDA mechanism as discussed above is influencing the ZIPD algorithm's performance to some degree.

Building a ZIP derivative that can function in decentralised markets is a challenge and requires further study. We discuss possible directions for this study in Chapter 7.

5.3.5 CDA Summary

Currently, our CDA mechanism lacks the sophistication needed to achieve reasonable performance. A possible improvement might be to collect quotes and respond to the best one heard over a period of time, thus improving the price of the final trade, and avoiding the "race condition" imposed by a synchronised immediate response. This idea and others for improving our CDA mechanism are discussed further in Section 7.3.3.

Additionally, our implementation of ZIPD was not able to adapt to the market conditions. This may be a factor of our inefficient CDA process, but more work is needed to investigate further. However, the ZIC traders still provide basic economic functionality that provides reasonable performance.

For the rest of the study, we use the sealed bid mechanism, given its superior performance.

5.4 Network Topologies

In this Section we investigate the effect of varying the topology of the network, and its effect on the performance of the simulation.

5.4.1 Implementation Details

In order to modify the characteristics of the topology or our network, we modify the basic generation algorithm described in Section 4.4.2. We maintain the initial round of giving every node a random link, and then adding random links until the requisite number of links has been achieved. However, we alter the choice of random links using a variety of parameters.

We add the concept of local vs global links (small world) after [113], preferential attachment after [7], and add a further novel mechanism for generating more social (higher transitivity) topologies.

For the local vs. global aspect, we first introduce the idea of locality. As each node in the network has a 2D coordinate, we use this to split the nodes into a number of regions, and a "local" link is defined as a link to a node in the same region.

We then introduce the parameter p_global, which denotes the probability that the link will be selected from the global population as opposed to the local population. This is based on the random re-wiring of a regular network from Watts and Strogaz [113], and is a similar method as used in our previous work [87], albeit with less regular locality. Our random networks used so far therefore have a p_global of 1.0, as all links are selected globally. Given that only small numbers of these "long range" global links are needed for small world-type behaviour, we use the values of $\frac{1}{64}, \frac{1}{32}, \frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}$ and 1 to explore this parameter space. We do not use a value of 0 (or close to 0), as for our model of locality, this would not produce a single network component, rather one component for each region.

For exploring scale free topologies, we use the preferential attachment algorithm proposed in [7]. When choosing a link to connect to (whether from a pool of local or global links, as above), preference is given to those with a higher degree according to an exponent, p_pref. A roulette selection method is used to then select the node. That is, each valid node choice is given a score equal to the degree of the node to the power of the exponent p_pref. A uniform random variate between 0 and the sum of the nodes': scores is used to select the node with which to link. Thus, those nodes with a higher degree are more likely to be chosen. Note for our random networks, p_pref is 0, and node choice is egalitarian. We investigate this space using the same values as in our previous work,

using $0.2, 0.4, 0.6, 0.8, 1.0$ and 1.2.

We add a social network inspired method of modifying the chosen node. We introduce the parameter *p_social*, which represents the probability that a node to link to must be chosen from a node's neighbours' neighbours, or "friends of a friend" (FOAF). If a link is "social", a list of all the current node's neighbours' neighbours is produced (a FOAF list), excluding any that are already neighbours of the node. If the link is local, the FOAF list is limited to local neighbours, otherwise it is global across all neighbours. The FOAF list is used to select a node to link to, using the preferential model outlined above. Thus each additional "social" link forms a new triangle in the network. In the early days of generating the network (when only a handful of links are generated), this is not always possible, so if there are no valid social links to be made, a random choice is made.

We utilise some of the measures described in Section 3.4.2.1 to analyse the characteristics of the network. This allows us to examine the expected characteristics of the network topologies generated.

5.4.2 Performance

We use the sealed bid mechanism for ZIC traders to explore each parameter, as this is the most robust solution we have explored. These experiments were run with our standard setup of size 256, a mean degree of 4 and shout radius of 2. We use Market A as an example of a common realistic market condition. We examine the effect of the topologies on resource and server utilisation, and market efficiency and convergence, over varying load levels, as before.

5.4.2.1 Small World Networks

Figure 5.19 shows the impact on the resource utilisation and the infrastructure for varying the *p_global* across the range of values discussed. A *p_global* of 1.0 is the same as the base random network we have been using so far. Table 5.3 shows the measured characteristics of the different topologies generated.

Interestingly, low values of *p_global* ($\frac{1}{64}$ and $\frac{1}{32}$) give rise to a slight improvement in overall resource utilisation, but add a slight increase to the load on the infrastructure. In these situations, most of a node's links are to other nodes in its own region, and thus are able to communicate at a reduced latency. This suggest that communication latency is a factor in overall performance, and should be taken into account when designing network mechanisms for these systems. The infrastructure load increase is interesting. Note that for a *p_global* of $\frac{1}{16}$ to $\frac{1}{2}$, the load on the infrastructure is actually lower that a *p_global* of

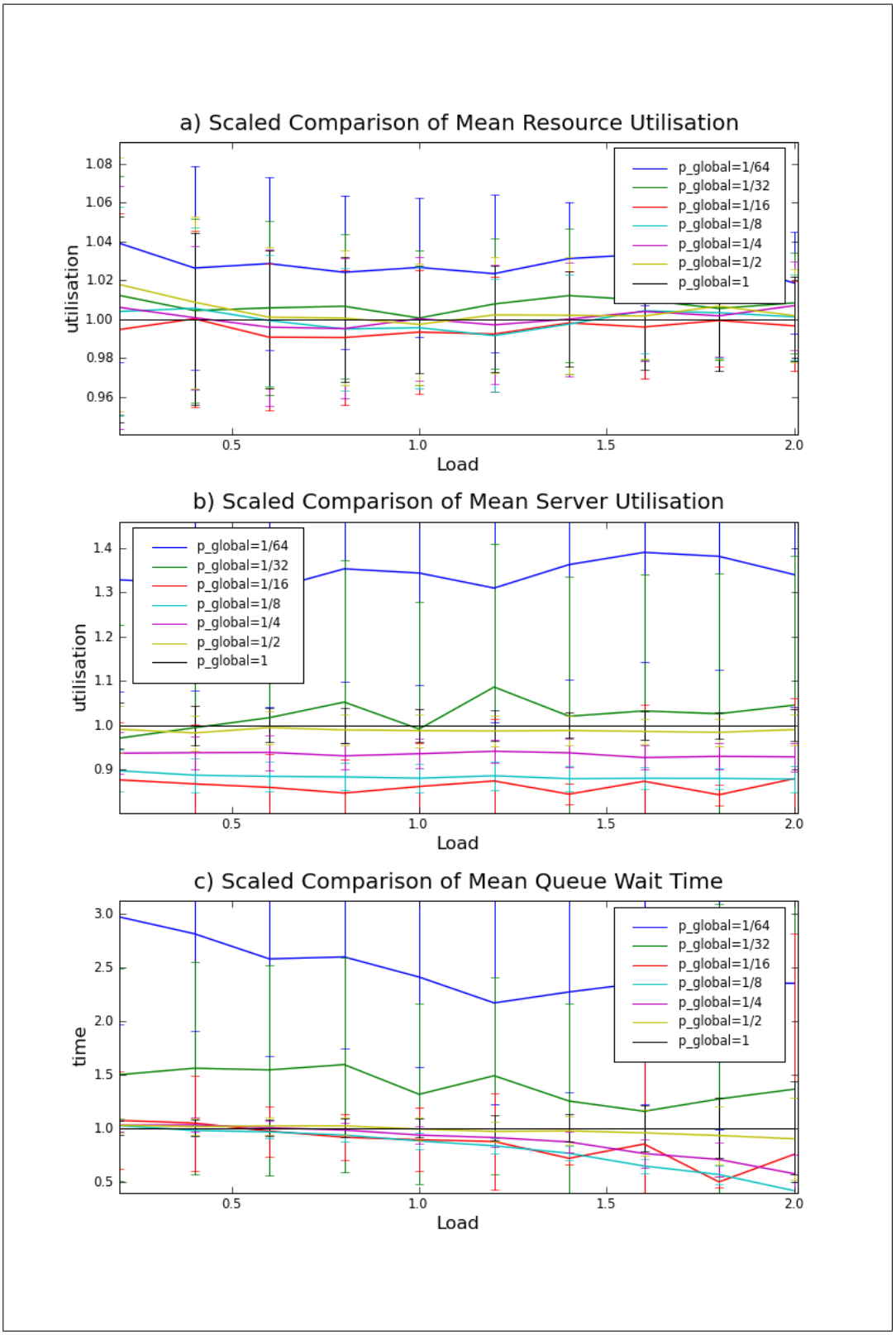

Figure 5.19: a) The mean resource utilisation b) mean server utilisation and c) the mean queue time, over load for a variety of small world networks determined by the parameter p_global. Results are shown scaled relative to $p\_global = 1$, which is a standard random network. We can see here that lower values of p_global perform better than values. Specifically (using Mann-Whitney test), p_global values of $\frac{1}{64}$ and $\frac{1}{32}$ are significantly greater by up to 30% ($n = 100, p < 0.05$ for $p\_global <= \frac{1}{32}$ for all loads), whilst greater p_global values have no real effect ($0.1 < p < 0.4$ for $p\_global > \frac{1}{32}$).

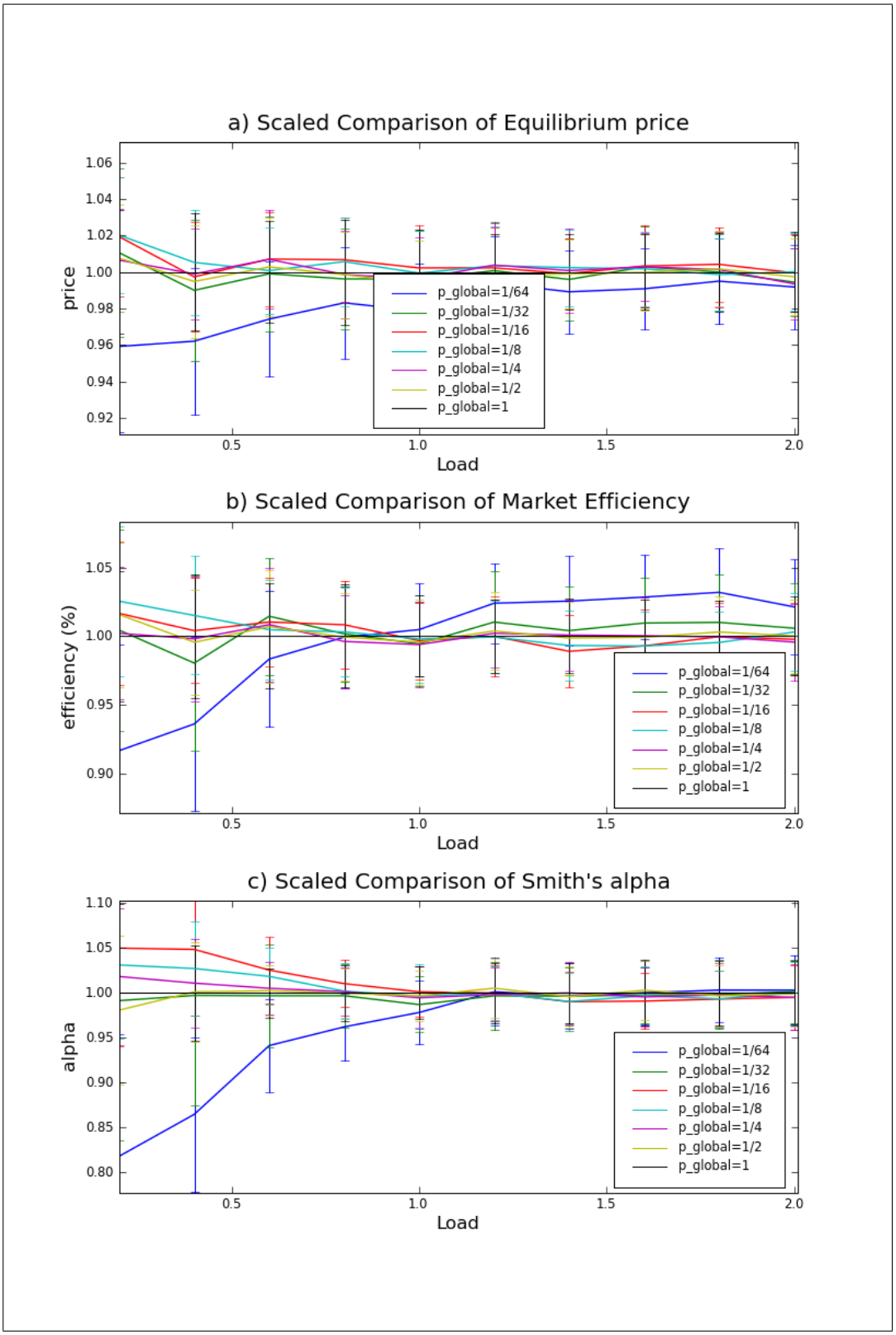

Figure 5.20: a) The equilibrium price b) market efficiency and c) Smith's alpha value, over load for a variety of small world networks determined by the parameter p_global, scaled as previous figure. Interestingly, efficiency for a p_global values of $\frac{1}{32}$ or less achieve slightly lower efficiency at lower loads (Mann-Whitney, $n = 100$, $p < 0.05$ for $load < 0.8$), and slightly higher efficiency at greater load levels ($p < 0.05$ for $load >= 1.2$). Other values of p_global performed comparably ($0.1 < p < 0.5$ when $p\_global > \frac{1}{32}$ for all loads).

p_global	Avg Path Length	Degree Correlation	Degree Skew	Transitivity
1/64	6.117 (1.350)	-0.020 (0.033)	0.301 (0.176)	0.279 (0.096)
1/32	7.017 (1.546)	-0.025 (0.043)	0.437 (0.176)	0.158 (0.067)
1/16	5.968 (0.455)	-0.034 (0.041)	0.510 (0.175)	0.112 (0.021)
1/8	5.088 (0.161)	-0.030 (0.043)	0.510 (0.158)	0.090 (0.012)
1/4	4.515 (0.062)	-0.027 (0.045)	0.545 (0.163)	0.059 (0.009)
1/2	4.222 (0.029)	-0.033 (0.041)	0.544 (0.162)	0.022 (0.006)
1	4.184 (0.027)	-0.029 (0.042)	0.532 (0.159)	0.013 (0.004)

Table 5.3: Key network parameters for small world networks of size 256 and varying *p_global* values (standard deviation in brackets, for 100 runs). As expected, the average path length decreases with increasing *p_global* values, as does transitivity. Interestingly, the degree skew (as an indicator of degree distribution) also increases slightly with *p_global*, showing a move towards scale free properties. This is a result of our initial link creation round, where we give every node a link, which skews the distribution slightly.

1.0. A plausible reason for this variation is due to the reduced latency for local links. For very local topologies (low *p_global*), the lower latency means messages arrive quicker, building up more load in the servers. It could also make the bidding more sensitive to message arrival times - "you snooze, you lose" - and increase the need for further negotiations, thus increasing load.

As locality is reduced, some messages arrive quickly, but others are slower, evening out the spread of messages out, which explains the reduction in message load. The increased load found when *p_global* is 1.0 is interesting. Given that in this case there will be the most variance in latencies across a node's links, this increases the load slightly, as some synchronisation is lost.

Figure 5.20 shows the effect of varying *p_global* on the economic market. Again, low *p_global* values induce the largest effect, most notably a slightly lower equilibrium price. This loses efficiency at low load levels but gains some at higher loads, and overall improves the convergence of the market towards equilibrium.

This suggests that a regular topology with a few global links provides the best performance for this system. These results for *p_global* are similar to the values found to be best in our previous work [87].

5.4.2.2 Scale Free Networks

In this Section we investigate the effect of preferential attachment on the system. Figures 5.21 and 5.22 show the system performance, and Table 5.4 shows the network characteristics.

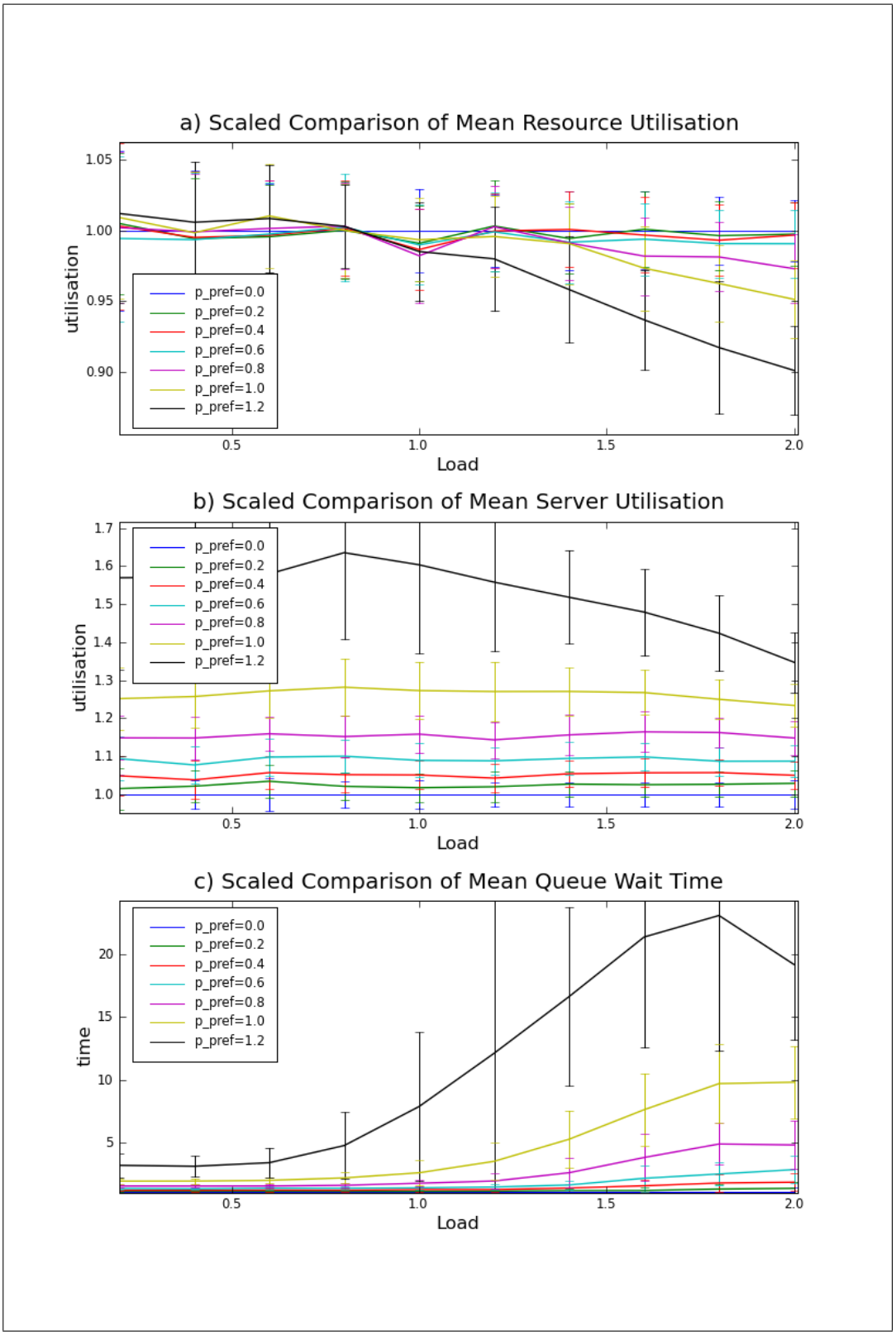

Figure 5.21: a) The mean resource utilisation b) mean server utilisation and c) the mean queue time, over load for a variety of scale free networks determined by the parameter p_pref. Results are shown scaled in comparison to $p\_pref = 0$. For lower loads, only a p_pref of 1.2 is significantly greater (Mann-Whitney, $n = 100$, when $load < 0.6$, $p < 0.05$ for $p\_pref = 1.2$, $p > 0.1$ for $p\_pref < 1.2$). The load allows the social "hubs" to perform well and do their job as a mediator for other nodes. However, when load is higher, p_pref values greater than 0.4 all perform worse (at load of 2.0, $p < 0.05$ for $p\_pref > 0.4$), while values less than 0.4 perform similarly to a p_pref of 0.

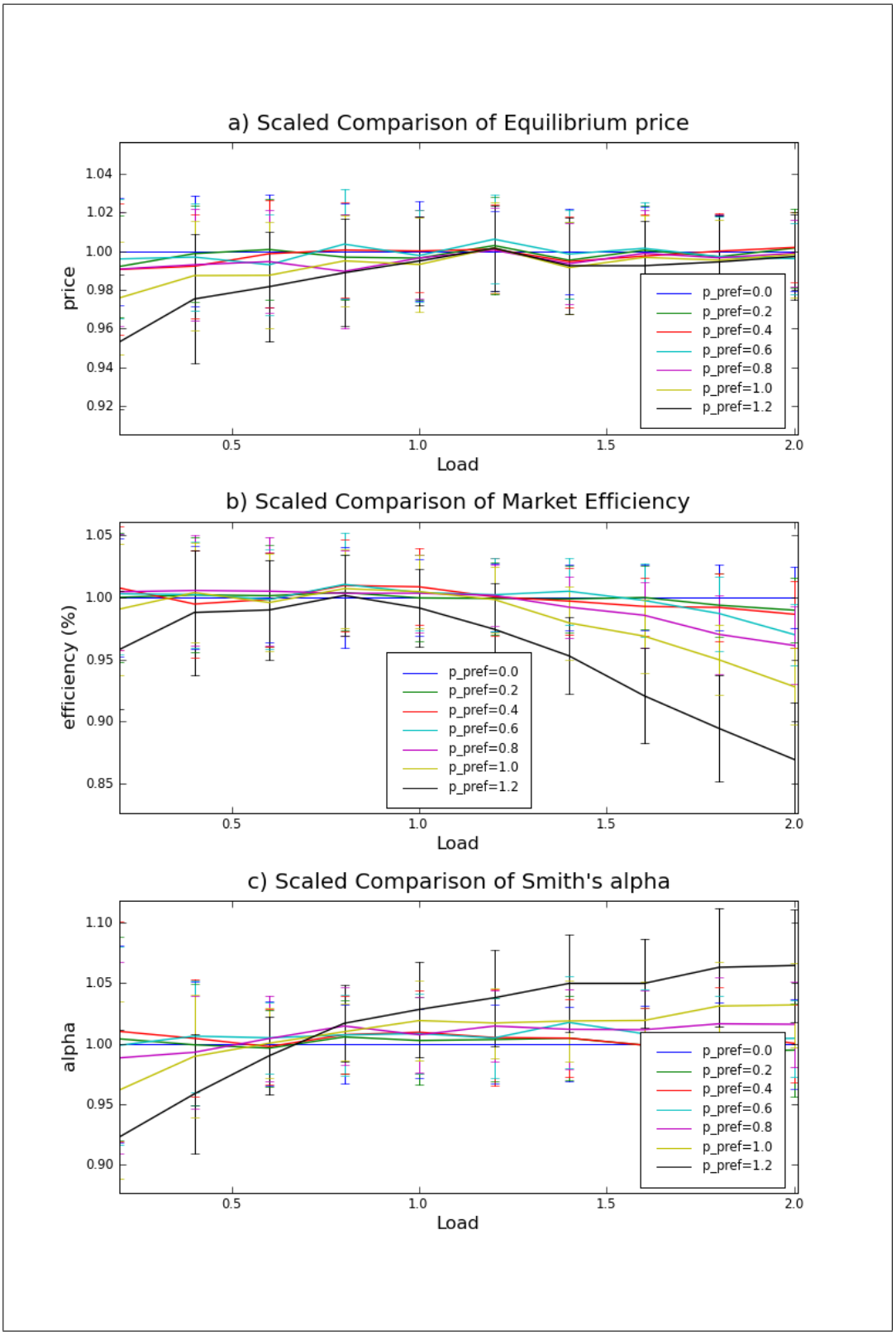

Figure 5.22: a) The equilibrium price b) market efficiency and c) Smith's alpha value, over load for a variety of scale free networks determined by the parameter p_pref, scaled as the previous figure.

p_pref	Avg Path Length	Degree Correlation	Degree Skew	Transitivity
0.0	4.183 (0.027)	-0.031 (0.037)	0.560 (0.162)	0.014 (0.004)
0.2	4.155 (0.029)	-0.047 (0.045)	0.643 (0.171)	0.014 (0.005)
0.4	4.114 (0.029)	-0.070 (0.041)	0.782 (0.203)	0.015 (0.004)
0.6	4.056 (0.040)	-0.099 (0.043)	1.017 (0.211)	0.016 (0.005)
0.8	3.972 (0.041)	-0.134 (0.036)	1.475 (0.301)	0.017 (0.005)
1.0	3.853 (0.051)	-0.168 (0.031)	2.395 (0.751)	0.019 (0.004)
1.2	3.617 (0.121)	-0.197 (0.034)	4.504 (1.928)	0.025 (0.005)

Table 5.4: Key network parameters for small world networks of size 256 and varying *p_global* values (standard deviation in brackets, for 100 runs). Here we see decreasing degree correlation as well as increasing positive degree skew, which are the indicators of scale free networks. The slight increase in transitivity is due to the increased chance that any two linked nodes will be more likely to link to the same high degree node, thus forming a triangle. The decrease in average path length are due to the popular nodes acting as central hubs for paths, and are lower than those in table 5.3.

Clearly the higher levels of *p_pref* have a detrimental effect on every aspect of performance, especially at high load levels, and do not perform any better than a *p_pref* of 0. This is due to the modelling of server performance, as there is no guarantee that a node with a a high number of links will have a faster service time. Thus the popular nodes are being swamped beyond their capabilities, increasing the load on the whole system. A worthwhile extension to this aspect of the investigation would be to change the preferential attachment to be towards faster servers rather than higher node degree. This would have the effect of faster servers having more connections, but they would be able to support these connections, which may allow the scale free nature of the topology to have more impact. It should be noted that our previous investigations found a *p_pref* value of 0 to be the most effective when communication costs were not modelled, so it is unlikely this extension would change the result. In a dynamic network, exploiting preferential attachment when load is low may provide some benefit, but would need to relaxed if load rises.

5.4.2.3 Social Networks

Here we present the results for varying *p_social*. Figures 5.23 and 5.24 show the results. Higher levels of "socialness" result in slightly degraded performance overall, particularly at high loads and low loads. This suggests that while real trading networks are formed over social lines, in a purely topological sense, random networks perform better.

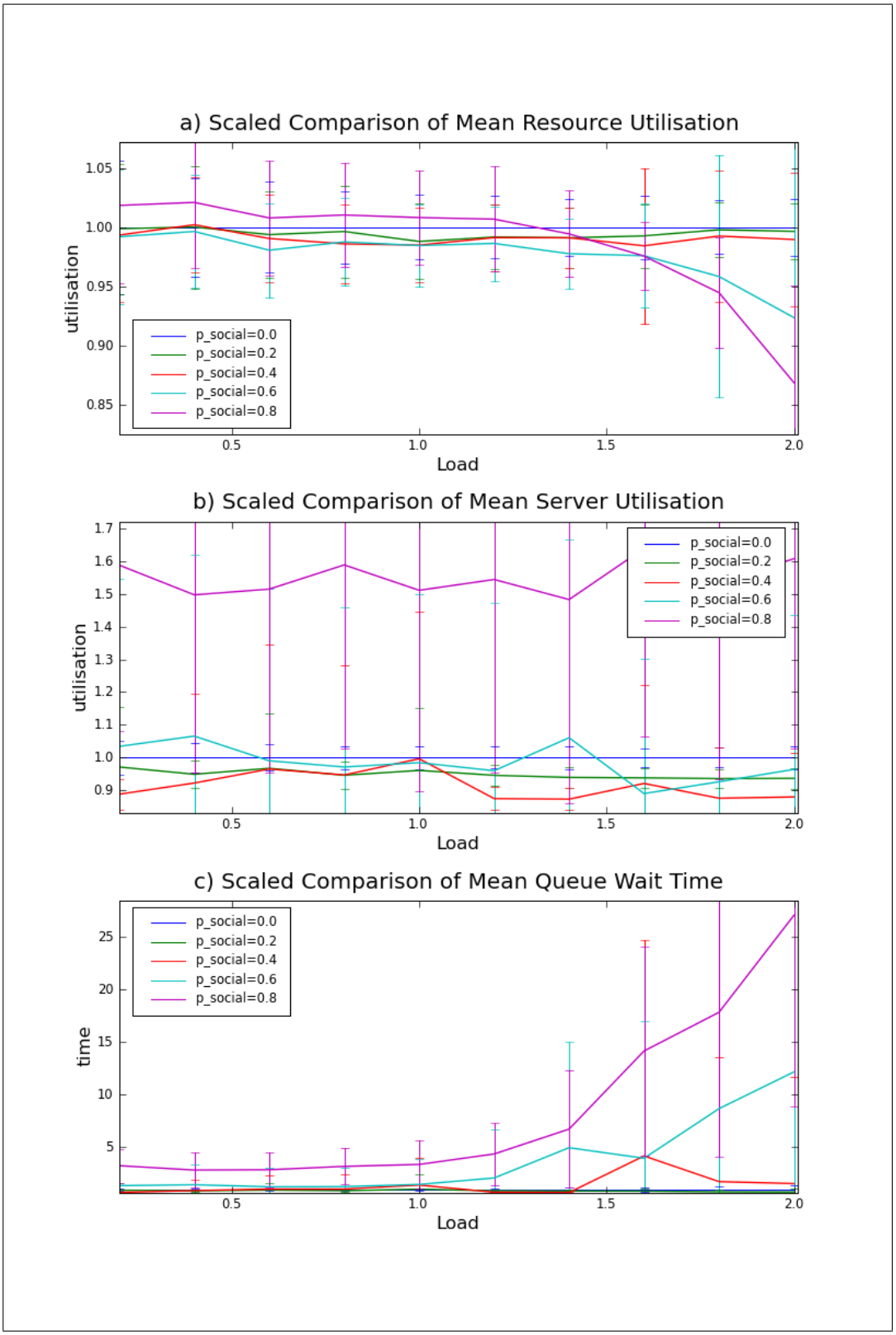

Figure 5.23: a) The mean resource utilisation b) mean server utilisation and c) the mean queue time, over load for a variety of social topologies determined by the parameter p_social. Results are shown scaled relative to a p_social of 0. For low loads, a p_social of 0.8 yield slightly improved utilisation (Mann-Whitney, $n = 100, p < 0.05$ when $p\_social = 0.8$ and $load < 1.2$). However, as load increases, the higher p_social yield significantly lower utilisation, and a p_social of 0 has higher utilisation than all others ($p < 0.05$ for $p\_social = 0$ and $load > 1.2$).

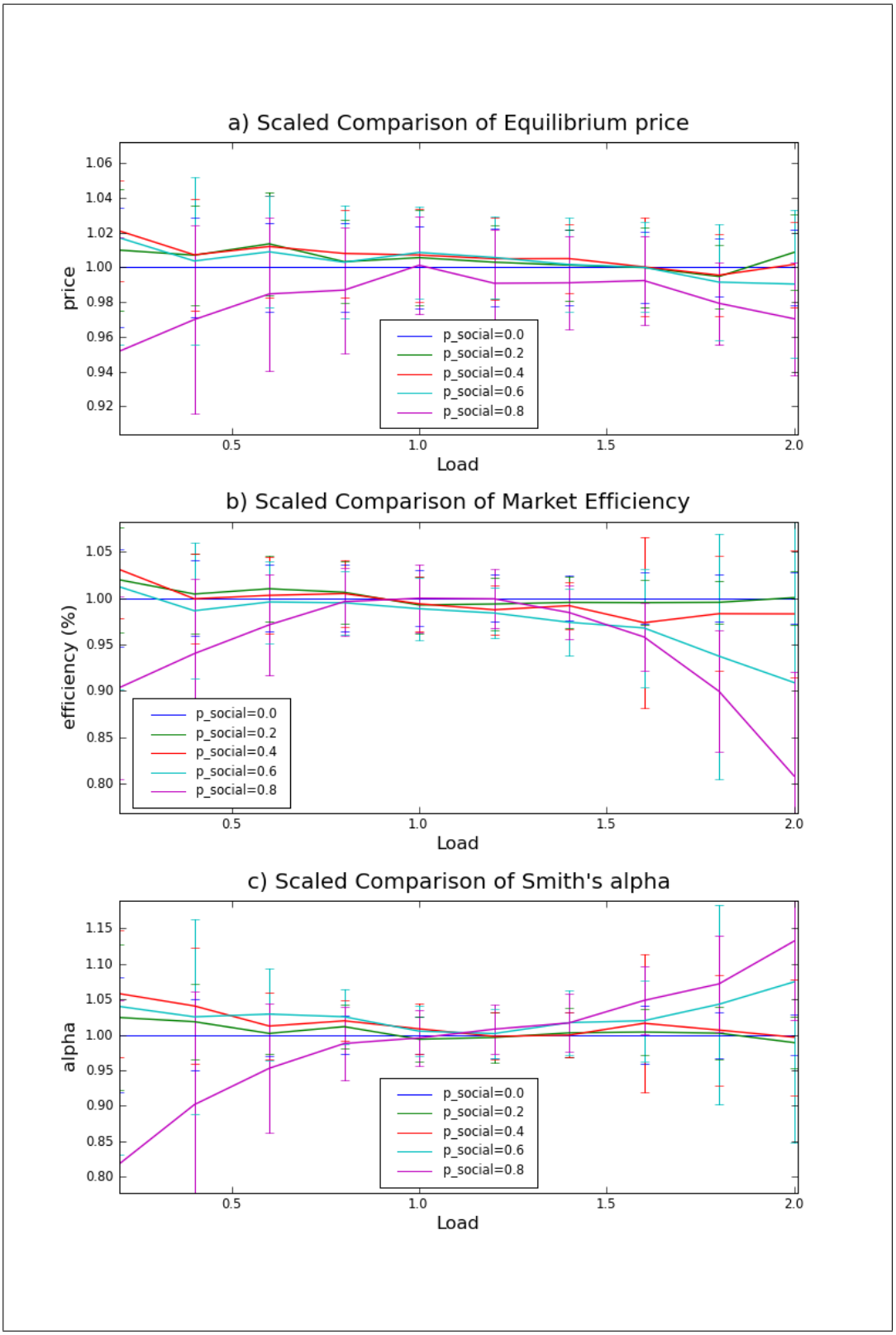

Figure 5.24: a) The equilibrium price b) market efficiency and c) Smith's alpha value, over load for a variety of social topologies determined by the parameter *p_social*, scaled as previous figure. For load values around 1.0, *p_social* make little difference to market efficiency (Mann-Whitney, $n = 100, p > 0.1$ when $0.8 <= load <= 1.2$). At higher loads however, a greater *p_social* has a negative effect on efficiency ($p < 0.05$ for $p\_social > 0$ and $load > 1.2$).

p_social	Avg Path Length	Degree Correlation	Degree Skew	Transitivity
0.0	4.186 (0.030)	-0.020 (0.044)	0.552 (0.171)	0.013 (0.004)
0.2	4.356 (0.150)	-0.019 (0.044)	0.608 (0.139)	0.103 (0.010)
0.4	4.559 (0.437)	-0.003 (0.044)	0.656 (0.187)	0.193 (0.019)
0.6	4.947 (0.690)	0.012 (0.041)	0.717 (0.170)	0.277 (0.028)
0.8	4.602 (1.672)	0.019 (0.043)	0.692 (0.184)	0.338 (0.046)

Table 5.5: Key network parameters for small world networks of size 256 and varying *p_social* values (standard deviation in brackets, for 100 runs). Here we see the increase in transitivity typical of social networks. We also see a small increase in degree correlation, which is another characteristic of social topologies. However, the average path length and degree skew are similar to our base random network.

5.4.3 Networks Summary

Overall, the topologies investigated here do not have significant impact on the performance of the system. In extreme cases, scale free and social topologies actually reduce the performance due to their uneven spreading of message load across the system. It is possible that when under low load, scale free and social topologies my provide some benefit, but not at higher load levels.

The only topology that provides any benefit at high loads is the small world topology at low values of *p_global*. As this is effectively a regular network with a few random links, topologies for this type of system would likely benefit from adopting a geographically based regular network of some kind, along with adding functionality for creating longer range links. However, the benefit is very small, and it could be that random networks provide the best real world implementation option due to their simplicity.

5.5 Summary

In this chapter we have presented our base decentralised economic mechanism, and examined its performance and behaviour. We also introduced an effort towards a fully decentralised CDA mechanism, however this did not perform as well as the sealed bid mechanism, and need further exploration. We also investigated the use of a decentralised ZIP algorithm, but found it unable to correctly converge on the market price.

We further explored our decentralised system by examining the topologies underlying the trader communications, and found some evidence that small world networks based on underlying regular networks have some benefits to our system.

In the next chapter, we evaluate our solution further, by comparing it to the current grid allocation approaches, and examining its adaptability.

Chapter 6

Evaluation

6.1 Overview

In this chapter we evaluate the performance of our grid allocation mechanism by comparing it to a centralised approach based on Globus MDS. Ideally for a fuller evaluation comparison to other economic allocation systems, such as Tycoon (see Section 3.3.2.3), would be valuable. However, the model of grid computation used by Tycoon is very different to the concept of jobs that we have utilised. Tycoon's job model is that of custom CPU-bound or IO-bound jobs that find a pool of available resources and manage a long-running job by interactively altering the bidding amounts on different machines according to their load. In comparison, our simpler model of a single job with a fixed budget, which participates in a one-off market to agree on a price for the whole job, is very different. The simulation tool used in this work is based around this model, and would require extensive modification to incorporate Tycoon's grid model. Additionally, the Parent/Child agent strategies used in Tycoon are custom built and more challenging to implement in a more generic manner.

This means that a full comparison with Tycoon is out of the scope of this work. However, comparison with an implementation of a current non-economic grid system has been investigated, and highlights some of the benefits of our economic approach.

We also examine the system adaptability. So far, the supply and demand of the system has been static for the length of each experiment. In Section 6.4, we report on effects of

varying the load on the system throughout the execution of the simulation.

6.2 A Centralised Grid Allocation Mechanism

The most common centralised algorithm for computational grids today utilises the Globus MDS system discussed in Section 2.6.1 as a resource discovery service. Various different agents can then query this service and use the results to select a suitable resource (or set of resources).

Our MDS model uses the concept of a centralised resource broker to provide an authoritative allocation. This broker maintains a view of the current global state of all the resources on the grid. Resources periodically update the broker with their current free capacity. When a new job enters the system, it queries the broker for an allocation. Our broker then returns an allocation for the job of the best resource to execute on. To determine this, it constructs a list of all resources that currently have enough free capacity to execute this job's size, and then selects the resource that would maximise utilisation, i.e. the resource with the smallest free capacity. It then reduces its record of the free capacity of the chosen resource by the job's size. In this way, the broker should maximise the resource utilisation across the whole grid. Upon receiving the allocation, the job then contacts that resource directly to request execution.

Clearly, a single broker would provide a close to optimal allocation. However one broker can only support a limited number of resources before becoming overloaded. Globus MDS supports the idea of federated servers that exchange information between each other about the current state of their resources to allow the system to scale. Thus we implement a system of federated brokers, that periodically update each other of their resources' states. We use the idea of regions to facilitate this. Each region has its own broker, and resources in that region send updates to that broker. Periodically, the brokers update each other on their resources' state. This allows the broker system to scale, as one broker is not handling the allocation for the whole grid, but introduces a further degree of error into their view of the state of the grid, as state will more likely be out of date.

The federation reduces the authority of the allocation of the broker. It may be that two different jobs get allocated by two different brokers to the same resource, which only has enough free capacity to do one of the jobs. In which case, the first job's request that arrives at the resource will execute, while the other with be rejected, and will have to query its broker again for another allocation. Jobs can only query a broker a fixed number of times (similar to the migration concept of our economic approach). Note that because of the sharing of information between brokers, each broker has a record of every resource

on the grid, thus there is no need to migrate the jobs themselves, as a broker could allocate a resource anywhere on the entire grid.

6.2.1 Performance Characteristics

The accuracy of the broker's model of the grid depends on the update frequency from its resource. The shorter the period between updates, the more accurate it will be, but it will only be able to process a limited number of updates per second before becoming overloaded. Clearly, a single broker can only support a certain number of resources (and therefore each region must be of a maximum size) before the broker cannot perform adequately. Additionally, the broker will have to handle allocation requests for all the jobs in its region as well.

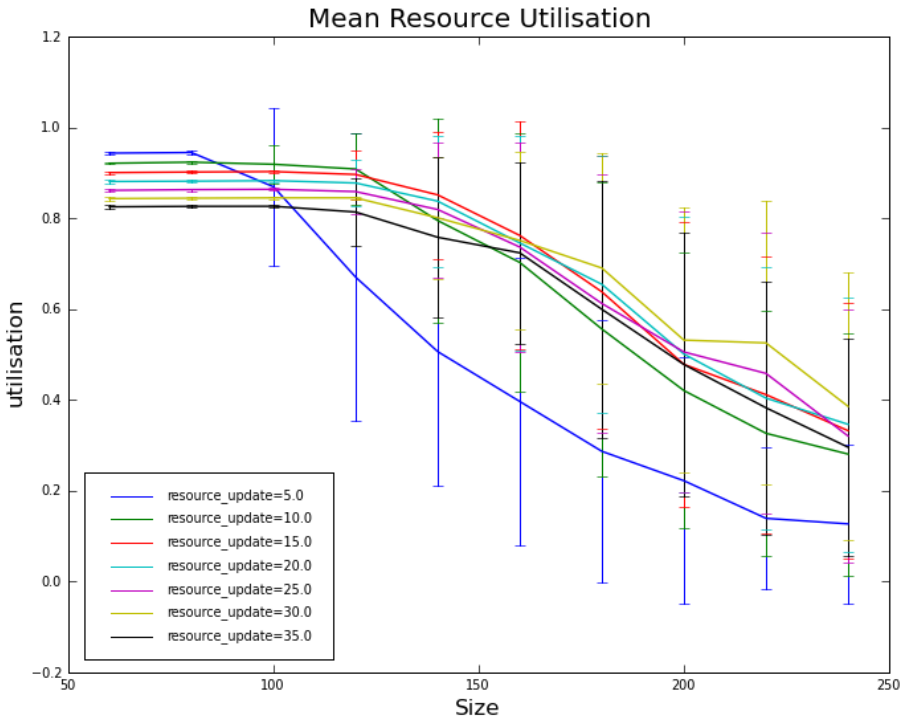

Figure 6.1: The mean resource utilisation for MDS with one broker and a load of 2.0 across a range of update periods and sizes, for 100 runs.

As the brokers would be a central aspect of the grid service, it is fair to expect that the broker's server will be faster than a normal server, so we set service time means for brokers using a normal distribution with $\mu = 0.02$ and $\sigma = 0.005$ (2.5 times faster than a normal server on average). We also ensure that each site has a faster "local" link to its

region's broker, and that the brokers have fast links to each other (see table 4.1). In terms of the maximum number of attempts before failing, we limit this to 3, the same as our maximum migrations parameter in the economic systems.

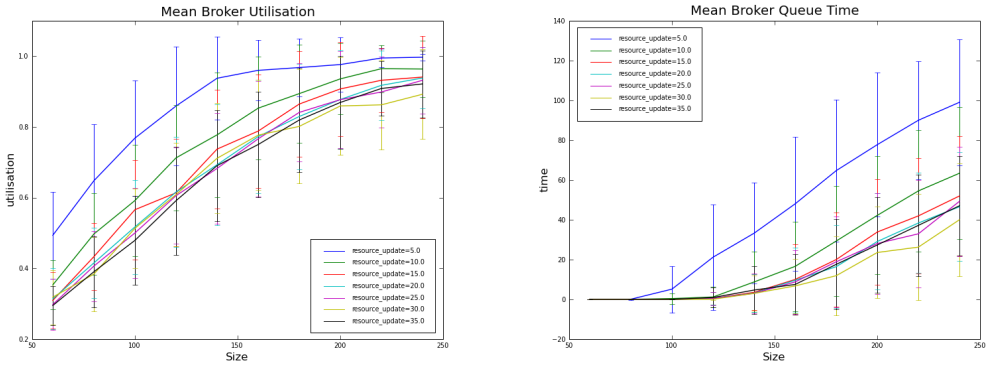

Figure 6.2: The mean resource utilisation for MDS with one broker and a load of 2.0 across a range of update periods and sizes, for 100 runs.

In order to ascertain the maximum connections a broker can sustain, we investigate the performance of the system with a single broker across a range of update periods at a maximum load of 2.0. Figure 6.1 shows the MDS resource utilisation for a variety of resource update period and sizes. Firstly, we notice from this that the mean resource utilisation is better for a lower resource update period. While a single broker keeps track of when jobs are allocated, it still needs the update from the resource to tell it when the jobs have completed. This creates a time window where a less than optimal allocation can occur due to outdated information. Of course, even with an optimal allocation scheme some utilisation is lost due to our fixed job sizes, as discussed previously.

Secondly, at bigger sizes utilisation degrades considerably. Figure 6.2 shows the server utilisation and queue time for the single broker. For each resource update period, there is a maximum number of connections before the system gets overloaded.

Given the improved utilisation of lower resource update values, we will use the lowest value of 10s for the resource update period to provide the best performance, and set the maximum region size to 100, to allow the brokers to perform adequately.

A further element to the MDS system is the broker update period, the amount of time between updates between brokers. Setting the resource update period to 10s, we examine the effect of this parameter with a fixed number of 4 brokers, across a variety of sizes. Figure 6.3 shows the results, and we can see that surprisingly, a more frequent update reduces the utilisation slightly, but not significantly. Above sizes of 400 (i.e. bigger than 4 times a region size of 100) we see performance degrades slightly. Figure 6.4 shows the

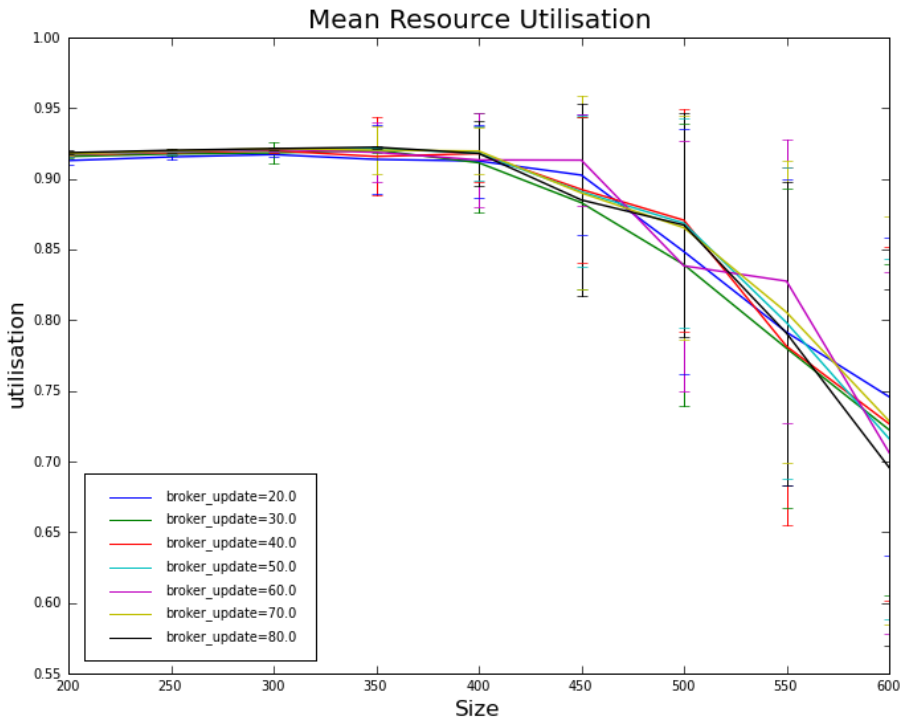

Figure 6.3: The mean resource utilisation for MDS with 4 brokers and a load of 2.0 across a range of broker update periods and sizes, for 100 runs.

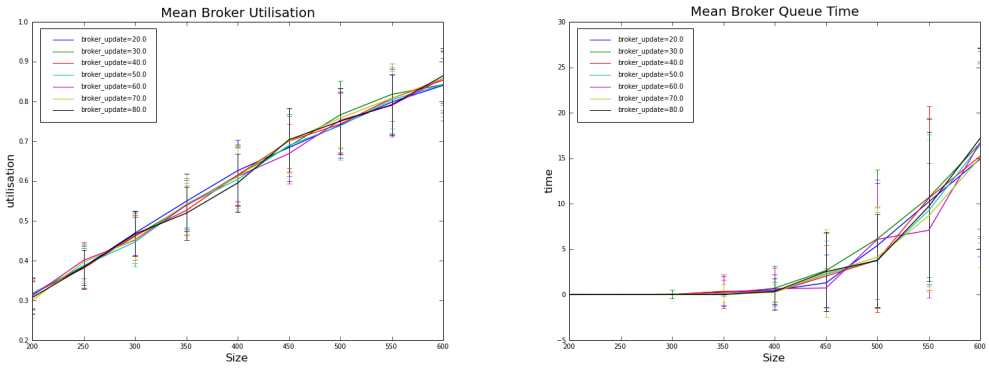

Figure 6.4: The mean broker utilisation and queue time for MDS with one broker and a load of 2.0 across a range of broker update periods and sizes, for 50 runs.

load on the brokers, and shows that broker update time has a much lesser effect on system load, and so we set the value to 20s for our experiments.

6.3 Comparison of Systems

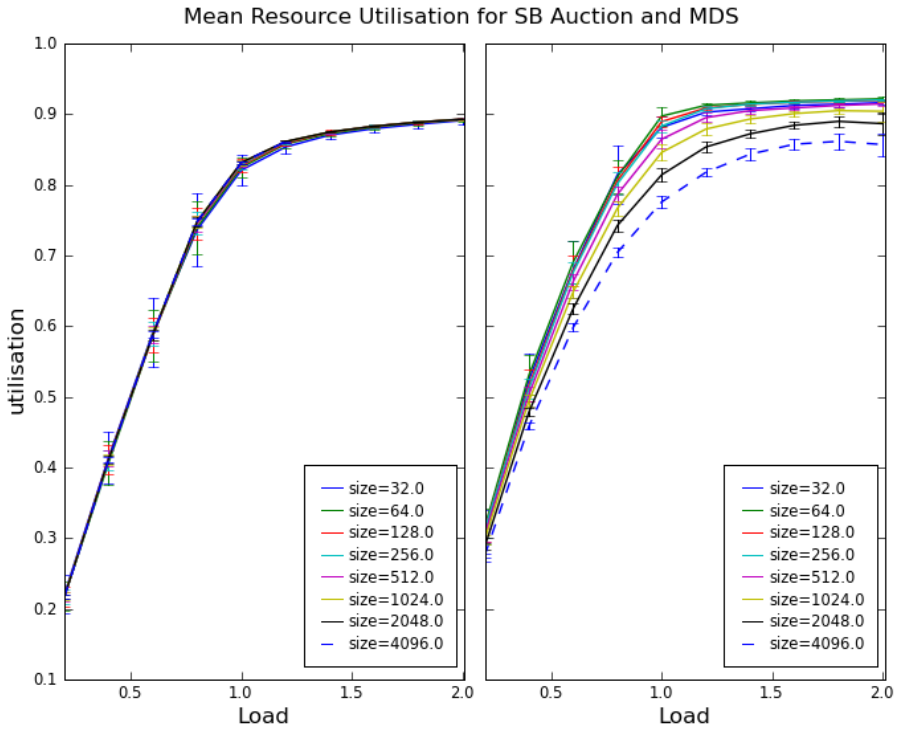

Figure 6.5: The mean resource utilisation for the sealed bid mechanism (market D) on the left and the MDS system on the right, across varying size and load values, for 100 runs.

We compare the performance of our sealed bid mechanism against that of the MDS system. As the MDS system has no concept of economic price constraints for jobs, we used market D with the sealed bid auction, with price constraints that maximise trading opportunities as Buyers and Sellers prices never intersect.

Figure 6.5 shows the resource utilisation for both systems, across both size and load. The main factor here is that while the MDS system performs better at lower loads and lower sizes, its performance degrades at high loads and sizes. Our economic system however scales identically for all loads and sizes, and out performs MDS at higher sizes. This shows the primary advantage of our economic system, that of scalability.

Figure 6.6 shows the broker utilisation and queue times across size and load. This shows that the degradation of performance in MDS at higher sizes/loads is not due to the

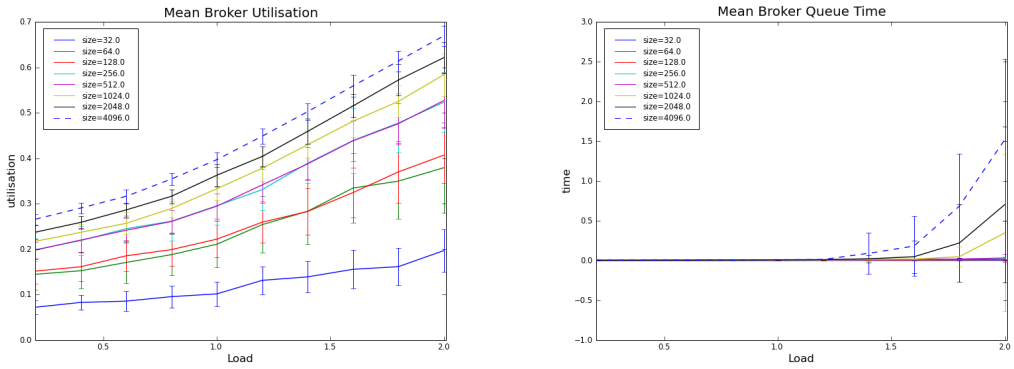

Figure 6.6: The mean broker utilisation and queue time for the MDS system across varying size and load values, for 100 runs.

overloading of the broker servers, but rather to the algorithm itself.

6.4 Market Adaption

The experiments reported in Chapter 5 were all run with static supply and demand curves for the whole simulation run. In reality, these curves are not constant, and change over time. To investigate the responsiveness and adaptability of our simulated markets, we dynamically vary the load value. We define four varying demand schemes as follows.

1. A square wave pattern, starting low at a load of 0.5, and alternating up to a load of 1.5. This simulates "market shock" situations, where the underlying market supply and demand change suddenly.

2. An inverted square wave, as 1 but starting high and then dropping low.

3. A sine wave pattern, starting at a load of 1.0, rising to 1.5, then down to 0.5. This is designed to simulate the general fluctuations of supply and demand that happen over time.

4. An inverse sine wave - as 3, but going down to 0.5 first then up to 1.5.

Every scheme's overall mean load value is 1.0, so they should provide the same overall level of utilisation, in theory, and every pattern is repeated twice.

Figure 6.7 shows the resource utilisation for the sealed bid auction with market A, over time for each demand scheme, averaged over 50 runs. The system responds well to the various schemes driving it, clearly adapting to the changes in load. For the harsher

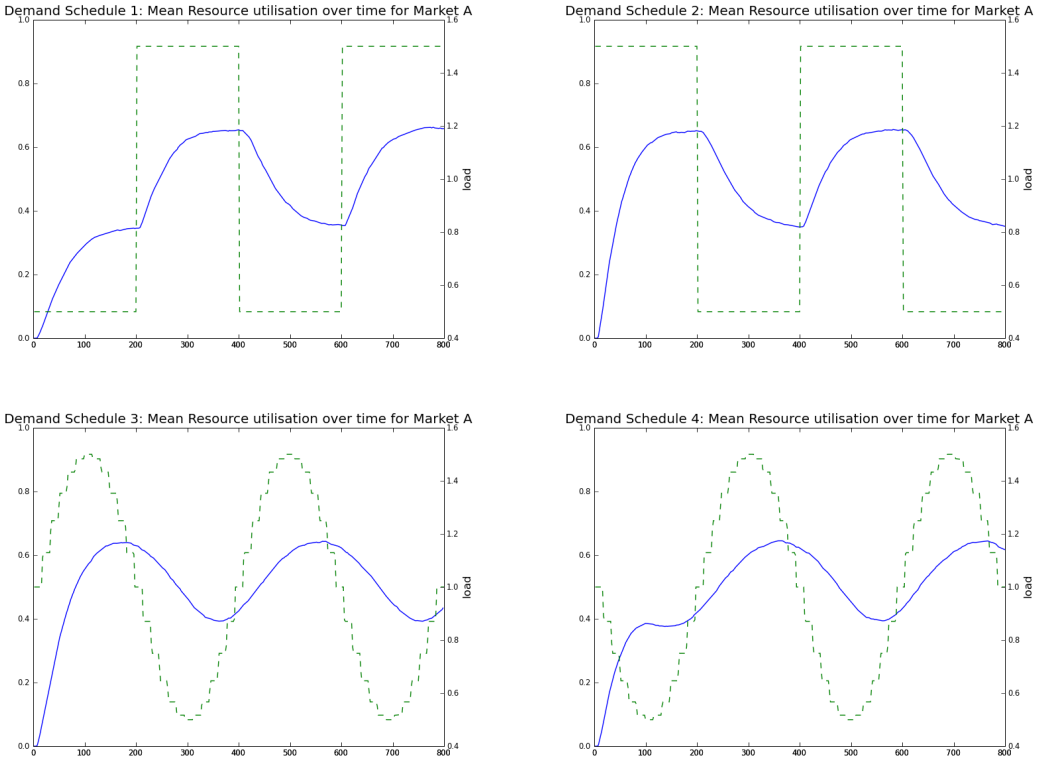

Figure 6.7: The adaption of mean resource utilisation for the sealed bid auction with market A for each demand scheme over time. The left scale indicates the resource utilisation, while the right indicates the load values driving the run. Results for the average of 100 runs.

square wave patterns (1 and 2), some time is needed to adapt. This is due to the fact that increase in load means an increase in arrival rate, which will of course impact the system over time rather than instantaneously. Of course the utilisation is not affected until the job has successful been allocated, adding the delay from the bidding process to the results. The sine wave patterns (3 and 4) show this more clearly, as there is a clear phase delay, although the adaption does follow the driving curve closely at the peaks, and less so in the troughs.

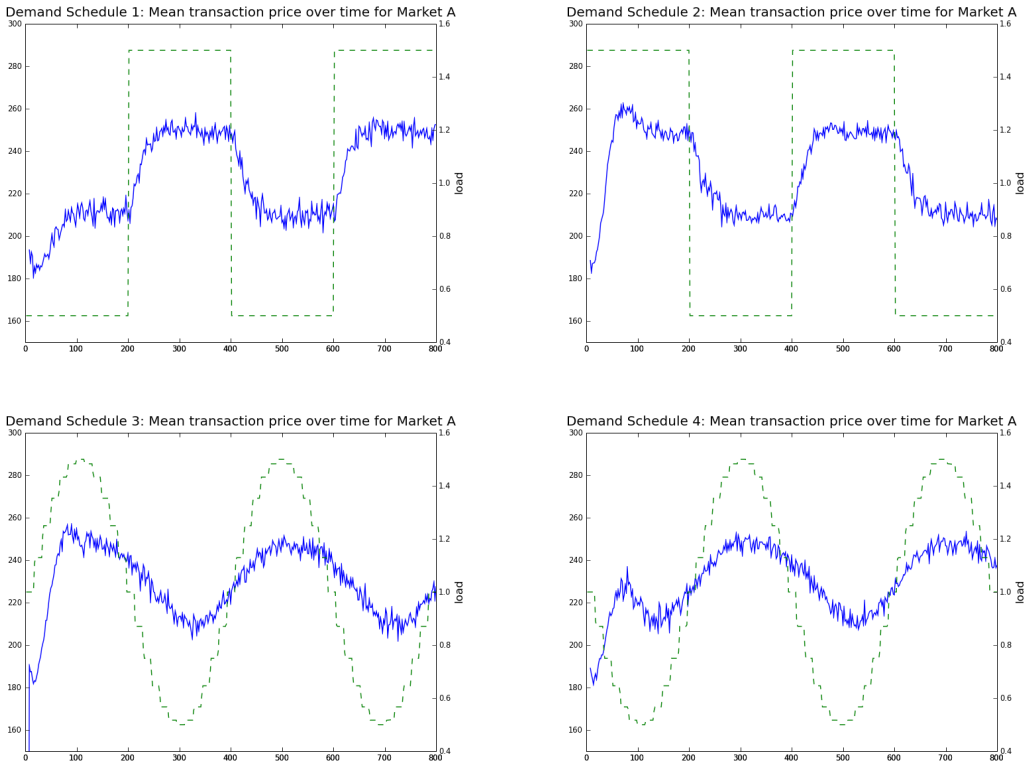

Figure 6.8: The adaption of mean transaction price for the sealed bid auction with market A for each demand scheme over time. The left scale indicates the mean transaction price, while the right indicates the load values driving the run. Results are for the average of 100 runs.

The results for mean transaction price are shown in figure 6.8, and we see a similar adaption here, the price adapts quickly to the changes in demand. With the square wave pattern, we see that the price changes to the price level usually before 100s have elapsed since the change in load value. That is, most of the change in price has occurred before the new load value has actually been achieved. This is a desirable characteristic, and shows that our sealed bid auction with ZIC traders can adapt to changes within a short time period (less than 30 simulated seconds), although we would of course expect intelligent

traders to adapt faster than this.

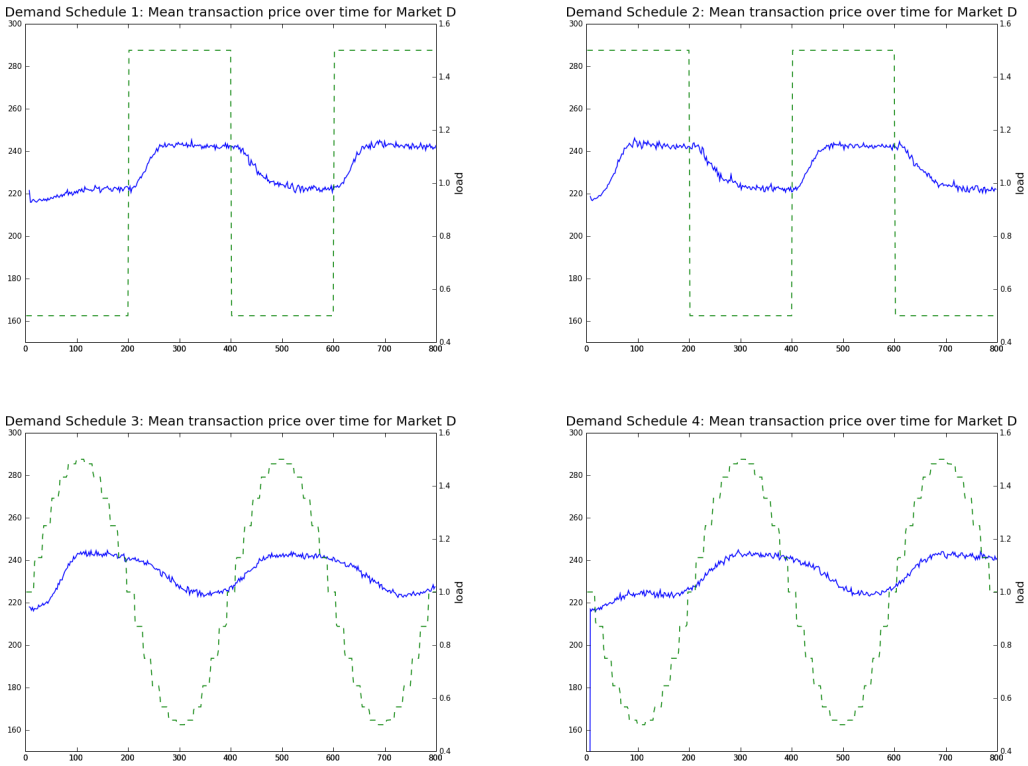

Figure 6.9: The adaption of mean transaction price for the sealed bid auction with market D for each demand scheme over time. The left scale indicates the mean transaction price, while the right indicates the load values driving the run. Results for the average of 100 runs.

The above results use Market A, which provides intersecting supply/demand curves to drive the adaption of price. For comparison, Figure 6.9 shows the results for Market D, where the supply/demand curves are flat. Here we see the same basic adaption, but without the guidance of intersecting supply/demand curves. The effect of the different supply curves can be seen in the range between upper and lower prices, which is greater for Market A than Market D. But adaption still occurs at a similar speed in both markets, if less accurately in Market D.

6.5 Evaluation Summary

In this chapter we have presented a comparison of our decentralised economic resource allocation system with an example of the centralised systems currently employed. While the centralised system does provide greater overall allocation performance, it has been

shown to have difficulty scaling to large numbers of resources. The economic mechanism in general slightly under-performs the centralised system for low sizes and loads, but scales extremely well to larger values of size and load.

We have shown that our economic system can adapt to dynamic changes in the underlying market situation, both sharp shock changes and gradual fluctuations, within reasonable time scales and performance.

Chapter 7

Conclusions and Further Work

7.1 Conclusions

7.1.1 Objectives

We have proposed and investigated a novel fully decentralised system for resource allocation on grid systems, using an economic metaphor. We have shown that this is a feasible option for a grid allocation mechanism, and in key aspects such as scalability, it has advantages over the current centralised approach. While there are many areas of improvement may be needed to make such a system a suitable solution, we have carried out initial exploratory work and laid a foundation for future development.

We also explored variations in network topology of the trading networks, which has not been extensively investigated, particularly at larger network sizes. As part of this, we develop and implement a novel mechanism for creating social network inspired topologies.

7.2 Thesis Contribution

In this thesis, we contributed the following as outlined in Chapter: 1.

- We have provided a detailed description of existing economic allocation mechanisms, and have examined their suitability for use in a grid environment.

- We have investigated our novel fully decentralised economic allocation mechanism for grid systems, and shown it to to achieve acceptable performance.
- We have examined the effects of topological features on the network used by our decentralised mechanism, and show that small world topologies have potential benefits.
- We have compared our decentralised economic system with the current approach to grid allocation, and show that our method performs better at larger network sizes.

Whilst the basic mechanism proposed here is an early effort, as a first step towards producing such systems it shows the promise of such systems. In particular, the scalability characteristics of our system would be very desirable in a real grid system. There is much that could be done to develop our system further, which is outlined in greater detail in Section 7.3.

7.3 Further Work

7.3.1 A More Accurate Model of the Grid

Our simplified model abstracts away many real world issues in grid computing. A more detailed model would increase the validity of our results.

An obviously useful addition would be the inclusion of variable job execution times. That is, a job's duration is only an estimate with some degree of error. It could therefore finish quicker or take longer by some amount of time, a more realistic model. In our current model, the costs of late completion (or savings of early completion) would be passed back to the Buyer, as they would pay for the resource actually used. However, more sophisticated mechanisms might be useful, where if a resource completes a job early, it gets to keep the difference. This would be especially important if some form of advance reservations are added.

The notion of a fixed job size could be relaxed, and jobs could be allowed to run on less resource capacity but take longer to complete, or on more capacity for a faster completion time. As not all applications are easily altered in this manner, a scalability factor could be introduced to each job to represent this. A scalability of 1.0 could indicate that a job is trivially parallel and can easily run on more or less resource for a linear adjustment in execution time, i.e. a job with a size of 10 and a duration of 100s would execute in 200s on a resource size of 5, or 50s on a resource size of 20. Correspondingly, a scalability of 0.0 would mean that a job cannot be run on more or less resource at all, much like our

current model. A value in between would modify the estimated execution in a suitable manner.

Given that different resources could now in theory execute the same job in different times, the completion time could be offered by the resource in addition to a price. For example, one resource could offer a fast execution price and a premium, whilst another offers a discount rate with a best effort execution time. Buyers could then weight their preference for a speedy completion versus cheaper resource usage, and select differing bids accordingly. This could be handled by have a multi-valued "value" element, or both monetary cost and completion time.

A further simple optimisation of the current system could be to relax the "on-demand" availability of resource to some degree. For example, if a Seller knows that a big job is due to finish in 10s, it could make an offer on a job it could not do right away, but could in 10s when the big job finishes.

7.3.2 Grid Infrastructure and Network Topology

Our model of communication and server costs could be improved in several ways.

Currently the performance of all the Buyers plus the Seller at a site is represented by a single server object with appropriate service times. A more accurate model would be to have each trader have their own "server" model in parallel with a site server. The site server would still be the main entry point for messages into the site, but would merely be a message broker, and thus could be faster than having to handle more complex state. Individual traders at a site would then be forwarded relevant messages and process them appropriately.

Another avenue of exploration would be to allow for dynamic network adaption. That is, a server could monitor its utilisation/queue levels, and either increase or decrease its number of connections to adjust to current load conditions. This would allow the system to better scale to higher loads, and increase the system performance and low loads by increasing the mean degree. A trivial implementation of this would add or release connections at random.

However, there is an opportunity here to extend the model of trader strategy into deciding who to trade with, or what market to move into.

For dropping connections, a possible method would be to keep a record of the number of messages received per second via a connection for each connection. This would be a measure of the CPU cost of maintaining that link. However this would need to be balanced against the gains that connection may have provided. Thus, a record of the profit made at

this node from transactions via each connection could be kept. A comparison of the two would allow the node with the lowest return for added load to be dropped. In theory this should provide a better overall grid/market performance than a purely random selection.

Adding a new connection is a more difficult problem, as there is no central list of nodes to choose from. However, the server could keep a record of every node that it has seen in the system, from any quote or message from anywhere, that it is not currently connected to. For each such node, it could again track a measure of the profit generated from that node. When selecting a new node, the most favourable node in the list could be selected. If no nodes have been seen, one could be chosen at random via broadcasting a request for new connections.

Note that this dynamic adaption of the network topology will mean that the network characteristics, such as transitivity, will vary over time. If the scheme proposed above is used, then growth will likely result in social topology, as new nodes will be friends of friends. If random long range links are to be preserved in such a network, some kind of central node registration would be needed.

7.3.3 Auction Mechanisms

Building on the success of the sealed bid auction, the mechanism could be further extended to an iterated version. In this case, a Buyer would advertise as normal, and then advertise the best price via a shout. Sellers could then resubmit a better bid or withdraw. This would have the advantage of providing some price information to the market, although this kind of auction is used to get the best price for the auctioneer (the Buyer in our case), so will likely drive down prices.

Clearly, a more sophisticated implementation of a CDA, which also allows for ZIP and other traders to function appropriately would be a valuable addition. Possibly exploring some form of loose bidding synchronisation may allow successful use of ZIP and other traders that have been developed in centralised markets.

An initial improvement would be to add some basic intelligence to the bidding, and allow for periodic consideration of quotes, rather than the current eager acceptance model. For example, a trader could collect quotes for a period of time, then at the end of that period, could either accept the best of any valid trades or shout their adjusted current quote if there are no valid trades.

One possibility would be to make each shout in the system include multiple pieces of information, rather than just a quote. It would still carry the trader's quote price, but it could also carry a list of quotes recently accepted or known to be accepted by the trader,

as well as recently failed quote prices. This would provide a more definite success/failure knowledge to other traders, as well as adding some synchronisation to the observation for trader strategies like ZIP and GD. However, given that traders can not be trusted to be fully honest, this would probably be best implemented as part of the site's server. It could be that only the servers shout recent accept/reject prices, the traders utilise the information at their servers only to trade.

This would allow the site servers to add a degree of centralisation to the auction mechanisms, with the aim of improving the efficiency. For example, each Site server could act as a local order book to some degree, and take responsibility for communicating transaction knowledge with other sites. Thus, the agents at a particular site could use the collective information about the success or failure of quote prices to improve their estimation of market price. It could also be used to add a level of synchronicity to the trading, which may help with the challenges of implementing decentralised CDA auctions.

This would be moving away from a central aim of this initial work, that of full decentralisation with no independent parties required. However, large numbers of Site servers adding some simple centralisation may grant significant efficiency gains whilst still preserving the desirable scalability characteristics.

7.3.4 Trader Strategy

Algorithms such as ZIP have been developed for synchronised auctions with an independent arbiter, trading in units of indivisible quantities. This is different from our market model in several key ways.

- No independent arbiter means quotes (and heir results) lack authority.
- Decentralisation means that less information is known to traders than in a centralised market.
- Our Buyer quantities are not divisible, adding an additional constraint on trading.
- Our Buyers are only interested in a single transaction - they are not persistent in the market.
- Lack of centralisation means asynchronous updates of trader's views of the market.
- There are many more Buyers in our system than Sellers.

A study of the ZIP algorithm that experimented with the effects of altering these factors would be of great value in successfully adapting ZIP to this type of decentralised market.

For example, we could alter the original ZIP auction used so that only a random subset of traders were updated with transaction data rather than all of them as in the original work. The updates could also be delayed by random amounts to different traders, to simulate an asynchronous network. We could add the idea of the quantities of one type of trader being indivisible, as well as the effect of adding new naive agents into the system for one "day" only. Observing the effect of these changes to the performance of the ZIP traders would identify the key provisions of a centralised auction that provide the ZIP traders performance, and hopefully allow for an adaption of ZIP to our decentralised setting.

7.3.5 Further Economic Evaluation

In Chapter 6 we compared the allocation mechanism against a theoretical optimal non-economic allocation scheme. This scheme was only concerned with quantities or jobs and resource, not with price. Comparison to a theoretical optimal system that considered both price and quantity would be useful.

An implementation of such a system could maintain a "magic" system-wide order book, with traders using a zero latency CDA mechanism to update it. This could be used to best allocate resources on both quantity and price in two possible ways. Firstly, it could allow trading across the whole grid system. This would potentially provide an absolute optimal performance measure of an economic system for grid allocation, and would be interesting to compare against the optimal non-economic allocation scheme.

Secondly it could restrict traders to trading with traders who are within the shout radius links apart. This would preserve the local decentralised limitations of our work, and provide a realistic optimal level for a decentralised market to aim for. This would greater aid in the analysis of the effectiveness of various auction schemes.

Bibliography

[1] D. Abramson, R. Buuya, and J. Giddy. A computational economy for grid computing and its implementation in the nimrod-g resource broker. *Future Generation Computer Systems*, 18(8), Oct 2002.

[2] G. Aloisio, M. Cafaro, E. Blasi, and I. Epicoco. The grid resource broker, a ubiquitous grid computing framework. *Scientific Programming Journal*, 10(2):113–119, 2002.

[3] Jrn Altmann, Dirk Neumann, and Thomas (Eds.) Fahringer, editors. *Grid Economics and Business Models, 5th International Workshop*, volume 5206/2008. Spinger Berlin/Heidelberg, 2008.

[4] Raghu Arunachalam and Norman Sadeh. The 2003 supply chain management trading agent competition. In *ICEC '04: Proceedings of the 6th international conference on Electronic commerce*, pages 113–120, New York, NY, USA, 2004. ACM.

[5] A. AuYoung, B. Chun, A. Snoeren, and A. Vahdat. Resource allocation in federated distributed computing infrastructures. In *Proceedings of the 1st Workshop on Operating System and Architectural Support for the Ondemand IT InfraStructure*, October 2004.

[6] Albert D. Baker. Metaphor or reality: a case study where agents bid with actual costs to schedule a factory. In *Market-based control: a paradigm for distributed resource allocation*, pages 184–223. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1996.

[7] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[8] Jim Basney and Miron Livny. Deploying a high throughput computing cluster. *High Performance Cluster Computing*, 1:Chapter 5, May 1999.

[9] F. Berman, R. Wolski, H. Casanova, W. Cirne, H. Dail, M. Faerman, S. Figueira, J. Hayes, G. Obertelli, J. Schopf, G. Shao, S. Smallen, S. Spring, A. Su, and D. Zagorodnov. Adaptive computing on the grid using apples. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 14(4):369–382, 2003.

[10] Craig Boutilier and Holger H. Hoos. Bidding languages for combinatorial auctions. In *In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1211–1217, 2001.

[11] Paulo Tibrio Bulhes, Chansup Byun, Rick Castrapel, and Omar Hassaine. N1 grid engine 6 features and capabilities. Technical report, Sun Microsystems, 2004.

[12] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger. Economic models for resource management and scheduling in grid computing. *Journal of Concurrency: Practice and Experience, Grid computing special issue*, 2002.

[13] R Buyya, M. Murshed, and D. Abramson. A deadline and budget constrained cost-time optimization algorithm for scheduling task farming applications on global grids. In *International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas*, June 2002.

[14] Rajkumar Buyya, David Abramson, and Jonathan Giddy. Nimrod/g: An architecture for a resource management and scheduling system in a global computational grid. In *The 4th International Conference on High Performance Computing in Asia-Pacific Region*. IEEE Computer Society Press, USA., 2000.

[15] Andrew Byde. A comparison between mechanisms for sequential compute resource auctions. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1199–1201, New York, NY, USA, 2006. ACM.

[16] Andrew Byde, Mathias Salle, and Claudio Bartolini. Market-based resource allocation for utility data centers. Technical Report HPL-2003-188, HP Labs, 2003.

[17] C.E.R.N. The worldwide lhc computing grid. http://lgc.web.cern.ch/LCG/public, last accessed 01/2009.

[18] Steve Chapin, Dimitrios Katramatos, John Karpovich, and Andrew Grimshaw. The legion resource management system. In *5th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP '99) in conjunction with the International Parallel and Distributed Processing Symposium (IPDPS '99)*, April 1999.

[19] B.M. Chapman, B. Sundaram, and K.K. Thyagaraja. Ez-grid: Integrated resource brokerage services for computational grids. http://www.cs.uh.edu/ ezgrid/, 2001.

[20] S-F Cheng, E Leung, KM Lochner, K O'Malley, DM Reeves, LJ Schvartzman, and MP Wellman. Walverine: A walrasian trading agent. *Decision Support Systems*, 39:169184, 2005.

[21] Brent N. Chun, Chaki Ng, Jeannie Albrecht, David C. Parkes, and Amin Vahdat. Computational resource exchanges for distributed resource allocation. Technical report, Intel Corporation, 2004.

[22] Scott H. Clearwater, editor. *Market-Based Control: A PAradigm for Distributed Resource Allocation*. World Scientific Publishing, 1996.

[23] Scott H. Clearwater, Rick Costanza, Mike Dixon, and Brian Schroeder. Saving energy using market-based control. In *Market-based control: a paradigm for distributed resource allocation*, pages 253–273. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1996.

[24] Dave Cliff. Days without end : On the stability of experimental single-period continuous double auction markets. Technical Report HPL-2001-325, HP Labs, 2001.

[25] Dave Cliff. Evolution of market mechanism through a continuous space of auction-types. Technical Report HPL-2001-326, HP Labs, 2001.

[26] Dave Cliff and Janet Bruten. Less than human: Simple adaptive trading agents for cda markets. Technical Report HPL-97-155, HP Labs, 1997.

[27] Dave Cliff and Janet Bruten. Simple bargaining agents for decentralized market-based control. In *Proceedings of the 12th European Simulation Multiconference on Simulation - Past, Present and Future*, pages 478–485. SCS Europe, 1998.

[28] B. Cohen. Incentives build robustness in bittorrent. In *1st Workshop on Economics of Peer-to-Peer Systems*, 2003.

[29] World Wide Web Consortium. W3c homepage. http://www.w3c.org/, Last visited 08/2005.

[30] World Wide Web Consortium. W3c soap technical specification. http://www.w3.org/TR/soap/, Last visited 11/2006.

[31] World Wide Web Consortium. W3c web services. http://www.w3.org/2002/ws/, Last visited 11/2006.

[32] K. Czajkowski, D. Ferguson, I. Foster, J. Frey, S. Graham, T. Maguire, D. Snelling, and S. Tuecke. From open grid services infrastructure to ws-resource framework: Refactoring & evolution. Global Grid Forum (GGF), 2004.

[33] K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, and W. Vambenepe. The ws-resource framework. Draft WS Specficiation, Global Grid Forum (GGF), 2004.

[34] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing, 2001.

[35] Rajarshi Das, James E. Hanson, Jeffrey O. Kephart, and Gerald Tesauro. Agent-human interactions in the continuous double auction. In *The Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 1169–1176, 2001.

[36] Simon Davy, Karim Djemame, and Jason Noble. The application of bioinspired engineering principles to grid resource allocation. In *UK Performance Engineering Workshop*, July 2003.

[37] Z. Despotovic, J. C. Usunier, and K. Aberer. Towards peer-to-peer double auctioning. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, pages 8 pp.+, 2004.

[38] K. Djemame and M. H Haji. Grid application performance prediction: a case study in broaden. In *1st International Workshop On Verification and Evaluation of Computer and Communication Systems (VECoS'2007)*, pages 162–173, 2007.

[39] Ian Domowitz. A taxonomy of automated trade execution systems. *Journal of International Money and Finance*, Volume 12, Issue 6:605–631, December 1993.

[40] K. Drexler and M. Miller. Incentive engineering for computational resource management. In B. Huberman, editor, *The Ecology of Computation*, pages 231–266. Elsevier Science Publishers B.V., 1988.

[41] Various UK e Science Centres. Grid markets: A market for grid services. http://www.lesc.imperial.ac.uk/markets/, Last accessed 10/2008.

[42] P. Erdös and A. Rényi. On random graphs. *Publicationes Mathematicae*, 6:290–297, 1959.

[43] T. Eymann, M. Reinicke, O. Ardaiz, P. Artigas, L. Diaz de Cerio, F. Freitag, R. Messeguer, L. Navarro, and D.Royo. Decentralized vs. centralized economic coordination of resource allocation methods in grids. In *1st European Across Grids Conference*, February 2003.

[44] Torsten Eymann, Dirk Neumann, Michael Reinicke, Bij?rn Schnizler, Werner Streitberger, and Daniel Veit. On the design of a two-tiered grid market structure. In *Proceedings of the Multikonferenz Wirtschaftsinformatik 2006, Passau, Germany*, 2006.

[45] Torsten Eymann, Michael Reinicke, Werner Streitberger, Omer Rana, Liviu Joita, Dirk Neumann, Björn Schnizler, Daniel Veit, Oscar Ardaiz, Pablo Chacin, Isaac Chao, Felix Freitag, Leandro Navarro, Michele Catalano, Mauro Gallegati, Gianfranco Giulioni, Ruben Carvajal Schiaffino, and Floriano Zini. Catallaxy-based grid markets. *Multiagent and Grid Systems, an International Journal, Special Issue on Smart Grid Technologies and Market Models*, 1(4):297–307, 2005.

[46] Torsten Eymann, Werner Streitberger, and Sebastian Hudert. Catnets - open market approaches for self-organizing grid resource allocation. In Jrn Altmann and Daniel Veit, editors, *GECON*, volume 4685 of *Lecture Notes in Computer Science*, pages 176–181. Springer, 2007.

[47] J. D. Farmer, P. Patelli, and I. I. Zovko. The predictive power of zero intelligence in financial markets. *Proceedings of the National Academy of Science*, 102(6):2254–2259, 2005.

[48] Maria Fasli, editor. *Evolutionary Optimization of ZIP60: A Controlled Explosion in Hyperspace*. Springer, 2006.

[49] Global Grid Forum. Homepage. http://www.ggf.org/, Last visited 08/2005.

[50] I. Foster. Globus toolkit version 4: Software for service-oriented systems. In *IFIP International Conference on Network and Parallel Computing*, pages 2–13. Springer-Verlag LNCS 3779, 2006.

[51] I. Foster and C. Kesselman. The globus project: A status report. In *Proc. IPPS/SPDP '98 Heterogeneous Computing Workshop*, pages 4–18, 1998.

[52] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999.

[53] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, and J. Von Reich. The open grid services architecture, version 1.0, 2005.

[54] Ian Foster. The anatomy of the grid: Enabling scalable virtual organizations. *Int. Journal of Supercomputing Applications*, Vol 15, No. 3, 2001.

[55] James Frey, Todd Tannenbaum, Ian Foster, Miron Livny, and Steven Tuecke. Condor-g: A computation management agent for multi-institutional grids. In *Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10)*, August 2001.

[56] Daniel Friedman. The double auction market institution: A survey. In Daniel Friedman and John Rust, editors, *The Double Auction Market: Institutions, Theories, and Evidence*, pages 3–25. Addison-Wesley, 1993.

[57] S Gjerstad and J Dickhaut. Price formation in double auctions. *Games and Economic Behaviour*, 22(1):1–29, 1998.

[58] Dhananjay K. Gode and Shyam Sunder. Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality. *Journal of Political Economy*, 101(1):119–37, 1993.

[59] Dhananjay K Gode and Shyam Sunder. What makes markets allocationally efficient? *The Quarterly Journal of Economics*, 112(2):603–30, May 1997.

[60] Minghua He and N. R. Jennings. Southamptontac: Designing a successful trading agent. In *ECAI 2002: 15th European Conference on Artificial Intelligence*. IOS Press, 2002.

[61] Minghua He, A. Rogers, E. David, and N. R. Jennings. Designing and evaluating an adaptive trading agent for supply chain management applications. In *Proc. IJCAI Workshop on Trading Agent Design and Analysis*, pages 35–42, 2005.

[62] R.L. Henderson. Job scheduling under the portable batch system. In D. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing (Proceedings of the 1st International JSSPP Workshop; LNCS No. 949)*, pages 178–186. Springer-Verlag, 1995.

[63] B. Huberman, editor. *The Ecology of Computation*. Elsevier Science Publishers B.V., 1988.

[64] Adriana Iamnitchi and Ian Foster. On fully decentralized resource discovery in grid environments. In *International Workshop on Grid Computing*, Denver, Colorado, November 2001. IEEE.

[65] R. Palmer J. Rust, J. H. Miller. Behavior of trading automata in a computerized double auction market. In D. Friedman and J. Rust, editors, *The Double Auction Market: Institutions, Theories, and Evidence*, page 155198. Addison Wesley, 1993.

[66] Rajkumar Buyya James Broberg, Srikumar Venugopal. Market-oriented grids and utility computing: The state-of-the-art and future directions. *Journal Of Grid Computing*, December 2007.

[67] Bernardo A. Huberman Kevin Lai and Leslie Fine. Tycoon: A Distributed Market-based Resource Allocation System. Technical Report arXiv:cs.DC/0404013, HP Labs, Palo Alto, CA, USA, 2004.

[68] Paul Klemperer. Auction theory: A guide to the literature. *Journal of Economic Surveys*, 13 (3):227–286, 1999.

[69] K. Krauter, R. Buyya, M, and Maheswaran. A taxonomy and survey of grid resource managment systems for distributed computing. *Int. Journal of Software: Practice and Experience*, Vol 32, No. 2, Feb 2002.

[70] William H. Kruskal and W. Allen Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47 (260):583621, December 1952.

[71] HP Labs. Se3d - growing animated film talent. http://www.dshed.net/SE3D, last accessed 09/2008.

[72] Dan Ladley and Seth Bullock. Who to listen to: Exploiting information quality in a zip-agent market. In *Agent-Mediated Electronic Commerce. Designing Trading Agents and Mechanisms*, pages 200, 211, 2005.

[73] Dan Ladley and Seth Bullock. The strategic exploitation of limited information and opportunity in networked markets. *Computational Economics*, 32(3):295–315, October 2008.

[74] Kevin Lai. Markets are Dead, Long Live Markets. Technical Report arXiv:cs.OS/0502027, HP Labs, Palo Alto, CA, USA, February 2005.

[75] Kevin Lai, Lars Rasmusson, Eytan Adar, Stephen Sorkin, Li Zhang, and Bernardo A. Huberman. Tycoon: an Implemention of a Distributed Market-Based Resource Allocation System. Technical Report arXiv:cs.DC/0412038, HP Labs, Palo Alto, CA, USA, 2004.

[76] A. Law and W. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, 3rd Edition, 2000.

[77] T. Malone, R. Fikes, K. Grant, and M. Howard. Enterprise: A market-like task scheduler for distributed computing environments. In B. Huberman, editor, *The Ecology of Computation*, pages 177–199. Elsevier Science Publishers B.V., 1988.

[78] H. B. Mann and D. R.Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, 18:5060, 1947.

[79] M. Miller and K. Drexler. Markets and computation: Agoric open systems. In B. Huberman, editor, *The Ecology of Computation*, pages 133–176. Elsevier Science Publishers B.V., 1988.

[80] Mark S. Miller, David Krieger, Norman Hardy, Chris Hibbert, and E. Dean Tribble. An automated auction in atm network bandwidth. In *Market-based control: a paradigm for distributed resource allocation*, pages 96–125. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1996.

[81] K. Muller. Simpy, a discrete event simulation tool in python. http://simpy.sourceforge.net, Last visited 10/2008.

[82] J. Nabrzyski, J.M. Schopf, and J. Weglarz. *Grid Resource Management: State of the Art and Future Trends*. Kluwer Academic Publishers, 2004.

[83] M. E. J. Newman. Assortative mixing in networks. *Phys. Rev. Lett.*, 89(20):208701, Oct 2002.

[84] M. E. J. Newman. The structure and function of complex networks. *Society for Industrial and Applied Mathematics Review*, 45:167–256, 2003.

[85] M. E. J. Newman and J. Park. Why social networks are different from other types of networks. *Physical Review E*, 68(036122), 2003.

[86] Walter Nicholson. *Intermeadiate Microeconomics and It's Application*. Thompson/South Western, 8th edition, 2000.

[87] J. Noble, S. Davy, and D. W. Franks. Effects of the topology of social networks on information transmission. In S. Schaal, A. J. Ijspeert, A. Billard, and S. Vijayakumar, editors, *From Animals to Animats 8: Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior*, pages 395–404. MIT Press, Cambridge, MA, 2004.

[88] David Oppenheimer, Jeannie Albrecht, David Patterson, and Amin Vahdat. Scalable wide-area resource discovery. Technical Report UCB/CSD-04-1334, EECS Department, University of California, Berkeley, 2004.

[89] International Standards Organisation. Osi reference model. http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994(E).zip, Last visited 11/2006.

[90] Boris Padovan, Stefan Sackmann, and Torsten Eymann. A prototype for an agent-based secure electronic marketplace including reputation tracking mechanisms. In *International Journal of Electronic Commerce*, pages 93–113, 2001.

[91] Globus Project. The globus project homepage. http://www.globus.org/, Last visited 08/2005.

[92] LSF Project. Platform lsf. http://www.platform.com/products/LSF/, Last visited 04/06.

[93] Maui Project. Maui scheduler. http://mauischeduler.sourceforge.net/, Last visited 06/06.

[94] The EGEE Project. Enabling grids for e-science. http://public.eu-egee.org, Last visited 10/2008.

[95] The Gridway Project. The gridway meta-scheduler. http://www.gridway.org/, Last visited 04/06.

[96] The LHC Project. L.h.c. - the large hadron collider. http://lhc.web.cern.ch/lhc, last accessed 09/2008.

[97] The TeraGrid Project. The teragrid project. http://www.teragrid.org, Last visited 10/2008.

[98] Rajesh Raman, Miron Livny, and Marvin Solomon. Matchmaking: Distributed resource management for high throughput computing. In *Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing (HPDC7)*, Chicago, IL, July 1998.

[99] Matei Ripeanu, Adriana Iamnitchi, and Ian Foster. Cactus application: Performance predictions in a grid environment. In *In proceedings of European Conference on Parallel Computing (EuroPar) 2001*, 2001.

[100] Neil Robinson. Evolutionary optimisation of market-based control systems for resource allocation in compute farms. Technical Report HPL-202-284, HP Labs, 2002.

[101] M Romberg. The unicore architecture: Seamless access to distributed resources. In *High Performance Distributed Computing, 1999. Proceedings. The Eighth International Symposium on*, pages 287–293, 1999.

[102] Thomas Sandholm and Kevin Lai. Evaluating demand prediction techniques for computational markets. In *GECON '06: Proceedings of the 3rd International Workshop on Grid Economics and Business Models*, 2006. http://gridasia.ngp.org.sg/2006/gecon.html.

[103] The National Grid Service. The national grid service. http://www.grid-support.ac.uk, Last visited 10/2008.

[104] R. Smith and N. Taylor. A framework for evolutionary computation in agent-based systems. In J. Castaing C. Looney, editor, *Proceedings of the 1998 International Conference on Intelligent Systems*, pages 221–224, 1998.

[105] Vernon L. Smith. An experimental study of competitive market behaviour. *Journal of Political Economy*, 70:111–137, 1962.

[106] Gerald Tesauro and Jonathan L. Bredin. Strategic sequential bidding in auctions using dynamic programming. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 591–598, New York, NY, USA, 2002. ACM Press.

[107] Gerald Tesauro and R. Das. High-performance bidding agents for continuous double auctions. In *IJCAI-01 Workshop on Economic Agents, Models and Mechanisms*, pages 42–51, August 2001.

[108] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maguire, T. Sandholm, P. Vanderbilt, and D. Snelling. Open grid services infrastructure (ogsi) version 1.0. Draft Recomendation, Global Grid Forum (GGF), 2003.

[109] Boston University. Brite: Boston university representative internet topology generator. http://www.cs.bu.edu/brite/, last accessed 04/2008.

[110] The Whiterose Grid University and Rolls Royce. D.a.m.e - distributed aircraft maintenance environment. http://www.cs.york.ac.uk/dame, last accessed 09/2008.

[111] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1):8–37, 1961.

[112] Carl A. Waldspurger, Tad Hogg, Bernardo A. Huberman, Jeffery O. Kephart, and W. Scott Stornetta. Spawn: A distributed computational economy. *IEEE Trans. on Software Engineering*, 18(2):103–117, February 1992.

[113] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

[114] M. Wellman and P. Wurman. A trading agent competition for the research community. In *IJCAI-99 Workshop on Agent-Mediated Electronic Trading, Stockholm, August 1999*, 1999.

[115] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1:8083, 1945.