Economy-Based Data Replication Broker Policies in Data Grids

Submitted in partial fulfillment
of the requirements of the degree of
Bachelor of Computer Science (Honours)
January 2005

Henry Lin
Supervisor: Dr. Rajkumar Buyya

Dept. of Computer Science and Software Engineering
The University of Melbourne

Abstract

Data is being produced at a tremendous velocity and volume from scientific experiments in the fields of high energy physics, molecular docking, computer micro-tomography and many others. E-Science and in particular the LHC (Large Hadron Collider), which will host experiments upon its completion in 2007, is expected to generate petabytes of data per year. There is thus an urgent requirement to obtain solutions to manage, distribute and access large sets of raw and processed data efficiently and effectively across the globe. A variant of the grid architecture for a distributed data management system named the data grid has been proposed to address these infrastructure issues of e-Science. *Data replication* is one of the key components in the proposed data grid architecture as it enhances data access and reliability. One of the key decision making tools in a data replication scheme is the *replication scheduler*; the scheduler determines when and where to perform a replication over a network in order to fulfill its goals. The other key component is the *resource broker* that determines how and when to acquire grid services and resources for higher level components. The following work introduces a novel approach to data resource broker policies for a replication scheduler under the tiered MONARC data grid structure from a market economy resource management approach. The new approach extends the *commodity market* model of the market economy to take into account other factors such as server reliability, transfer speeds, link reliability and service costs. Furthermore a detailed and realistic simulation of a data grid was implemented to empirically show the effectiveness of such scheduling policies compared with other policies and with each other on a user and grid-wide scale.

Acknowledgements

Many thanks go to my supervisor Dr. Rajkumar Buyya for advising on the path of research and offering much academic support along the way. I would also like to thank Srikumar Venugopal for numerous constructive and patient discussions that played a large part in developing the ideas presented in this thesis.

Thank you to my mother and father for their loving support and steadfastness that steeled my determination and resolve throughout the year.

This work would also not have been possible without constant moral support from my friends. A special thanks to Huishan, Thomas, Erin and Anne for their big hearts and impressive patience with me.

Contents

Chapter 1: Introduction

1.1 Background

E-Science refers to large scale scientific endeavors that are increasingly carried out via collaborations of global scale. Its application domain consists of high energy physics, medicine, molecular chemistry and astrophysics. Typically, such scientific collaborations require access to very large data collections and computing resources. The Large Hadron Collider (LHC) is the most ambitious particle accelerator project ever undertaken and it has the collaboration of many nations and more than 150 institutions. It is also one of the flagship projects to herald e-Science. After its expected completion in 2007, experiments on the LHC are forecasted to generate massive quantities of data. ATLAS [4], currently the largest of the experiments to be conducted on the LHC is projected to generate several petabytes of data per year alone. At such a vast rate and over a life time of 15 – 20 years, efficient and fair access to data from the experiments require improved solutions in dealing with large volume transfers, storage management, data consistency and wide-area cyber-infrastructure. Efficient accesses to such volumes of data may require terabyte caches and gigabit per second transfer rates over WANs. Furthermore, collaborations with the sheer number of the diverse institutions involved introduce new issues with heterogeneous administration policies and contrasting software interfaces.

As grid computing provides several solutions that meet the requirements of e-Science challenges, *grid* technologies have been proposed as an enabling infrastructure where large volume resources are required. So far there have been advances in grid computing on all levels, from its network infrastructure to user applications. The progression of grid technologies and the requirements of high volume resource management have seen the emergence of *data grids* [33]. Data grids are considered to be the leading infrastructural solution to the issues at LHC.

One of the essential optimization services of a data grid is the data replication service. This service seeks to improve the network traffic by copying heavily accessed data to appropriate locations and managing their disk usage. The replication mechanism attempts to determine when, where and what data to replicate across the grid. However in order to realize the potential of the replication and the data grid itself, there is a requirement for a decentralized replication management systems. One of the methods proposed have been to employ market forces analogous to the real world in order to manage grid resources. In economy resource management grids, resources are sold to users as a service.

1.2 Thesis Focus

This thesis will reduce the overheads of the replication mechanisms that drive the data management components of a data grid by extending the resource broker with policies that factor in user quality of service as well as service costs when replicating and transferring data. The end result is a separate data replication broker for the replication process that will locate best data server when making a local replica. The data replication broker will operate within a market economy replication management framework. It will improve the received quality of service from the individual replication transfers by factoring not just explicit cost of transfer, it can, through the price mechanism (in reference to economic theory) also implicitly improve the network traffic and increase user received quality of service so that data resource usage is sustainable and fair.. Furthermore, the work will be will be applicable to all large volume transfers common in the LHC.

The ideas are inspired from real world economies where the players involved – the producers and the consumers – are assumed to be rational entities. That is, consumers base their purchasing decisions intelligently from quantitative and qualitative analysis of the quality of the service they receive and the producers price their services according to factors such as demand, investment and risk analysis, etc. Particular attention is applied to simulating a realistic market economy data grid and introducing the concepts of server reliability and transfer migrations to improve replication decision making mechanisms and data transfers during replication.

1.3 Contribution

The contributions made by this thesis, demonstrating the effectiveness of proposed economics-based policies for data replication brokers are delineated below:

1. The exploration of the effects of the service pricing policy in economy based resource management grids. One of the key concepts that drive economy based resource management is the price mechanism and one of the main contributions to the force of the price mechanism is the manner in which producers price their

products in response to the market. This work will evaluate the effects of a constant pricing scheme and a simple variable pricing scheme on the network traffic of the grid.

2. A data replication broker policy that allows a broker or a user using the broker to transfer data with a view to reduce both the transfer time of a replica and/or the cost of the transfer. This policy gives the option to allow the user to trade between time and cost of a replication procedure.
3. The previous policy was extended to allow for replication transfers to be interruptible during transfers. An interruptible replica transfer service will allow for a more dynamic optimization of replication transfers by migrating to cheaper servers and/or faster servers instead to taking the risk of staying on the one chosen server. However there are innate sub-problems associated with determining when to interrupt a replication transfer.
4. The previous policy is further extended so that it factors the long term effects of unreliable servers and server-to-broker links. While a server may offer faster and cheaper services, an unreliable service may significantly affect the overall long term experienced quality of data transfers. The final policy will generate a list of candidate servers and then choose a server based on a Monte Carlo method that will favor reliable servers over unreliable servers.
5. A complex and realistic model of the LHC data grid was created to simulate and explore the performance of the policies. Much work has been done to done to extend the base data grid simulator in order to accommodate the requirements of an economy-based data grid.

1.4 Thesis Organization

The following will detail the structure of the document. Chapter 2 will introduce grid and the data grid concepts as well as components of data replication and data grid models. Concepts in economy based resource management will be presented in Chapter 3. Related work in scheduling with the economic data grid model will be discussed in Chapter 4. The proposed policies will be discussed in detail in Chapter 5. In order to test the effectiveness of the policies, a model of a data grid was implemented and simulated, this is used to test the effectiveness and performance of the polices which are described and discusses in Chapter 6.

Chapter 2: Data Grid Concepts

2.1 Grid Concepts

A classical grid (as opposed to a data grid) is a collection of connected resources that are available for an application to perform a task [9]. As network and distributed computing technologies evolved to become more efficient and cost effective it became apparent that one was able to effectively utilize the resources from relatively cheap small scale computer resources into a collective that rivals the resources on dedicated supercomputers. Resources can be in the form of CPUs, data storage and services from applications (see Figure 1) – in fact any device connected to the grid network is defined as a grid node and its utility can be harnessed for its available resources. Nodes are often geographically distributed, heterogeneous in their make and administrative domains connected by links of various performance and reliability.

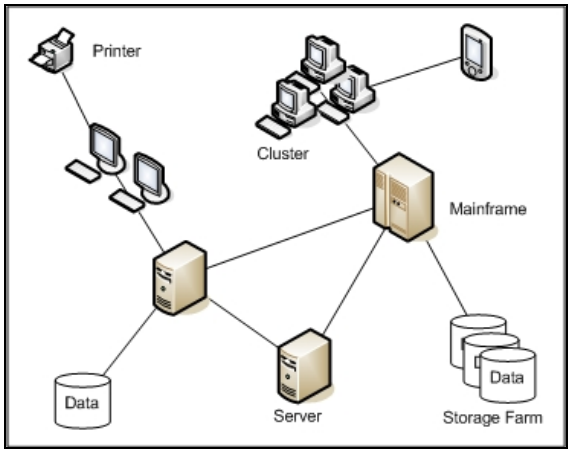


Figure 1: An example of a grid of resources.

Due to the heterogeneity of grid, there's needs to be a software layer that serves to abstract the native interfaces of the available resources and provide a common interface. This layer lies below a grid application and is composed of software components that together create an environment where all the resources in the grid are available to the application. This layer is termed the middleware and some of its services include:

- Security (Authentication, Single Sign-on)
- Resource information service
- Data transfer and communication
- Process creation and monitoring
- Remote data access and storage
- Resource management scheduling

One of the main types of currently operational grids is a computational grid. They specializes in managing and scheduling compute resources for grid applications. These applications submit *jobs* to be processed by grid resources. Jobs require mainly 3 types to grid resources: compute time for processing, temporary caching or storage for input and output data sets and bandwidth for transfer and communication. Its ability to manage large amounts of data in an efficient and organized manner was existent largely in terms of delivering application inputs and output to and from the remote node(s). Other types of grid are access grids which provide virtual presentation environment with distributed audio and visual resources; and data grids which will be detailed in Section 2.3.

2.2 Resource Management System

One of the key components of any grid is the Resource Management System (RMS) that allows resources of a grid to be accessed and utilized efficiently in a coordinated fashion. The RMS is usually built into the middleware but

should not be considered a local entity to a node but more of a grid wide service. There are usually three main services on offer: the Resource Information Dissemination, the Resource Discovery and Resource Scheduling [7].

The *resource information dissemination* distributes information regarding a resource across a grid so that resources can be utilized. The distribution process can be either a push or pull process where by the RMS can broadcast the information (push) or the RMS can publish the information where by applications can acquire it at will (pull).The implementation of the service can be either centralized or distributed as a network directory or objects.

The *resource discovery* service provides a way to search and register new resources available on the grid as well as track already registered resources that are no longer available. Similar to the way resource information dissemination is implemented; the discovery mechanism can be implemented as distributed or centralized queries.

The *resource scheduling* service uses the information gathered from the previous services in order to discover candidate resource providers[1] that can be used by an application. The scheduler will also control and monitor a running application in terms of where to run the process or whether it will be placed in a queue, the exact method is policy dependant. The scheduler monitors the use of a resource after scheduling it so that it may decide – according to policy – on whether to reschedule an application to other resource provider(s). The resource scheduler can be centralized, distributed or hierarchical.

The *centralized scheduler* is similar to those deployed on clusters in that a central process receives and schedules requests for the usage of available resources. In general, this is infeasible of for the grid due to scalability issues. A reasonably sized grid will contain a single point of failure and would not be able to grow beyond the performance limits of the node that a centralized scheduler is deployed at. The most common alternative is to deploy a *distributed resource scheduler* at every node – the implementation can be at two perspectives – first from the viewpoint of the application where it can decide where to use its required resources; or it can be from the resource provider where it can decide how to respond to a resource request. Another alternative that falls somewhere in between the previous two methods is a *hierarchical scheduler* where sub-components are deployed for to schedule resources at different levels of the grid. For example we may employ one scheduler that may decide which section of the grid to perform a job computation and a scheduler of that section of the grid will then determine exactly which node in its section to fulfill the requirements.

In a grid that utilizes a decentralized resource schedulers, the components that performs the local scheduling at a node are also called *resource brokers*. The broker is analogous to the stock market broker that works as the middle man of a transaction and the separate nomenclature is to highlight the idea that resource brokers are considered to be providing resource scheduling for a grid as a collective. On a computational grid where the main resources are compute cycles, an instance of a broker is associated with each application however it can also be the case where we may have a broker for each resource, although this is rare. The broker is responsible for retrieving the available resources for its application – usually from resource information dissemination services – it will then select the nodes that best suite the application requirements and submit them to the resource. Monitoring of submitted resource usage is policy specific and can be either the responsibility of other components of the RMS or the broker itself.

2.3 The Data Grid

In 1999, [1] introduced the concept of the *data grid* as an integrated architecture that is a specialization and extension of the classic grid. The data grid was designed to satisfy the requirements of the combination of large dataset sizes; geographically distant distribution of users and resources; and computationally intensive analysis. The architecture also was developed to suite operations in a wide area, multi-institutional and heterogeneous environment.

A data grid, like a classic computational grid, is designed to be mechanism and policy neutral, to have uniform access to grid resources and to be compatible with other types of grid infrastructures.

The requirements of the design led to a layered system architecture for the middleware that extents the classic grid middleware architecture (Figure 2) similar to the one presented in [1]. The high level view shows that there are two layers: a lower layer for core services and higher layer services built on top. The lower level core services seek to abstract heterogeneous storage systems to have uniform ability to read, delete, create, and modify file instances. The core services also contain a metadata service that works to manage information about the data as well as replicated data. The key difference between a data grid and a classical grid was the proposal of the replica system and its

[1] The terms “resource provider” and “node” will be used interchangeably.

subsystems: the *replication selection* and *replica management* components. These systems sit on top of the previously discussed resource management system. The replica management controls the process of creating and deleting replicas – which are essentially 1:1 copies of data. The replication selection component is responsible for choosing specific data or file to be replicated (note this is different to choosing the data *source* at which a copy of a file or data maybe reside).

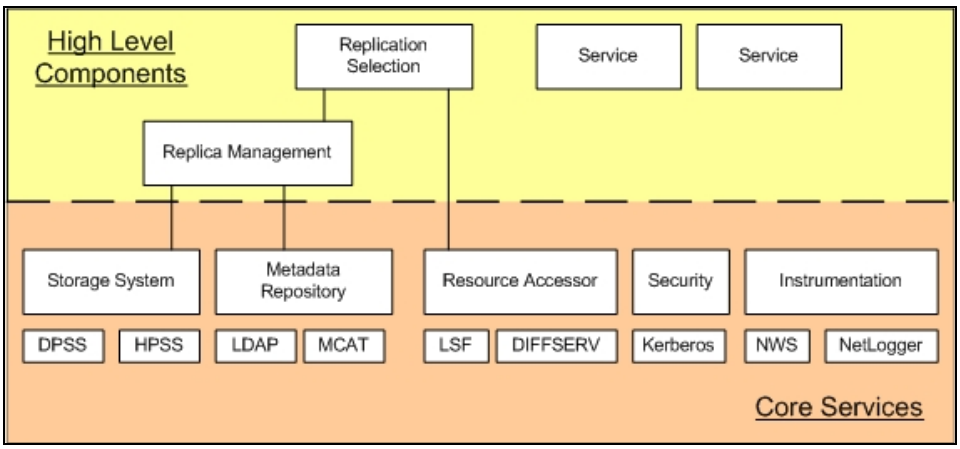


Figure 2: Structure of the Data Grid architecture.

Although data grids focus on the management of data, today's grid infrastructure couples job scheduling with replication scheduling. In order to predict user requirements for job placements and therefore data set requirements, replication pre-placement based on usage patterns have been proposed. In [6], algorithms and concepts for a decoupled data scheduler are introduced to address pre-placements. The premise is the decoupling of replication resource requirements from application resource requirements; this then requires the grid applications to send jobs to the external schedulers. The external schedulers then inform dataset schedulers of dataset usage, which enables the dataset schedulers to automatically create replicas across the network.

2.4 The Role of Replicas and Replica Scheduling in Data Grids

The replication of files is motivated by the desire to provide communities of researchers with faster and better data access performance characteristics. The combination of the Replica Management and Replication Selection components allow for multiple replicas of a data entity to exist in the grid and therefore a much larger potential download bandwidth where a community of peer nodes have a wide selection of possible servers. A storage failure at the original node could be recovered by the replicas and transcontinental transfers can be reduced by creating localized sets of replicas.

However, despite the benefits, the justifications of using replicas are largely determined by their storage and communication overheads. There are also the problems brought about by real time data changes to any of the replicated files or the source file which involve large scale data consistency issues. The overhead in transferring and storing replicas deserve close scrutiny and optimization to maximize the gains in using replicas. Often the problem lies in replication policy. Given a decoupled data scheduler, we need to determine the three dimensions of *what, where* and *when* to replicate data. These considerations are determined wholly by grid policy. We also need to decide before hand how replicas are to be created. For example a data grid can allow nodes to push replicated data into each other or one can have each machine decide when to pull or replicate data onto itself. The later is generally used so that the node owner does not lose control of his/her own storage resources.

The question of when to replicate data is one of the key aspects to replication since static replication or pre-replaced replicas cannot adapt to the fluctuating demands of a dynamic grid. The problem can be further segmented into two questions: when does one check for whether there is the requirement of replications; and when does one actually transfer the data. When we check for replication need, and assuming a pulling type of replication, we also need to fulfill the questions for which files that contain the data are to be replicated locally followed by the need to know where the file will be retrieved from since there may be more than one source.

2.5 LHC Data

As discussed in Chapter 1, the LHC experiments are expected to generate vast volumes of data. The experiments will create two types of data [13]:

- Experimental data collected by the experiment run. This will typically have a single creator. These files are modified as new data are added during an initial production period lasting several weeks. After data production is complete, files are not modified. Sizes are expected to be 2-10GB in size.
- Metadata that contain information about the experiment like the number of events and results after analysis. Metadata files may be created by multiple individuals and may be modified over time, even after the initial period of data production. These types of files are expected to have sizes of approximately 2GB. The volume of metadata is typically smaller than that of experimental data.

2.6 LHC Computing Model and Regional Centers

The tiered regional center model is a top level infrastructure proposal for the LHC data grid venture. It is envisioned that there will be very large, multi-service facilities to specialized facilities such as PC farms that will concentrate key grid resources at a geographical point. The tiered model, also referred to as the MONARC model [12], involves the arrangement of these regional centers in a hierarchical tree structure (see Figure 3).

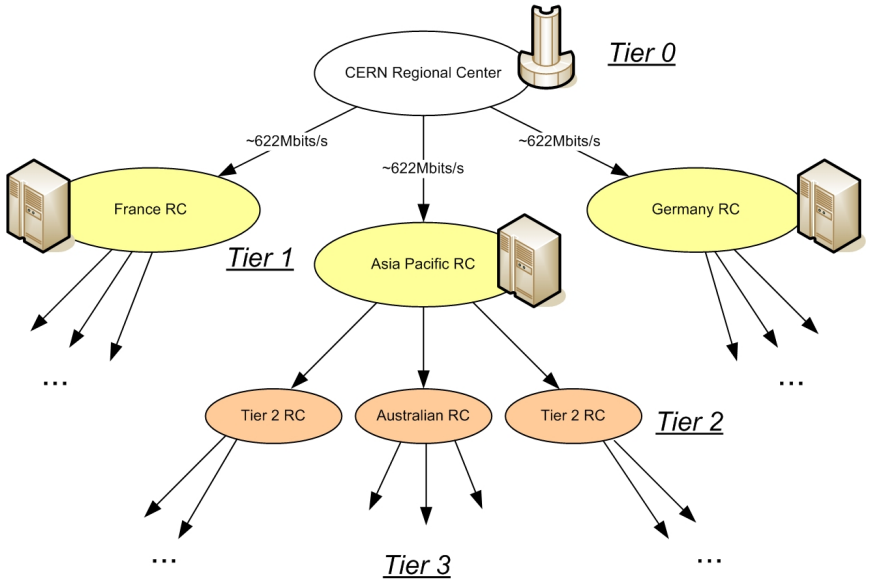


Figure 3: Hierarchy of Regional Centers.

The powerful facility at CERN is referred to as the tier-0 center with support regional centers classified by the level of coverage and service they provide:

- Tier-1: They provide a large range of services and facilities. Tier-1 regional centers serve a large geographic region.
- Tier-2: this centers have less services and facilities available than the tier-1 centers and serve a smaller geological region or country.
- Other special purpose centers focus on providing for a limited amount of computing requirements and users.

This structure was proposed to maximize the intellectual contribution of physicists from around the world without the necessity of their physical presence [12]. The regional centers take into account the realistic implications of network bandwidth and costs. By centralizing computing infrastructure one is able to reduce cost and achieve higher effective bandwidth compared to a set of geographically spread out facilities. As well, it is claimed that the regional center model allows for greater flexibility and efficiency in data access and transfers by monitoring and/or controlling the where data goes, where it residues and how the users are to use its resources.

Chapter 3: Economic Models for Grid Resource Management

3.1 Concepts

A grid that uses an economic model for managing resources is one that utilizes the real world analogy of a market economy to limit, manage and optimize the utilization of resources used. In real world markets there are many models for setting the monetary value of goods and services based on supply and demand. It has been shown that the forces of demand and supply iterate the value of a good or service towards an equilibrium value, this process is caused by the price mechanism [15]. The equilibrium value represents an idealistic value where neither demand nor supply are in excess or shortage. The price mechanism is essentially a dynamic self optimizing resource management tendency in a market economy that moves the market towards a sustainable balance between the demand and supply of a good or service. Many grid projects (Distributed.net, SETI@Home, Condor pool, Distributed ASCI Supercomputer, GUSTO, and eGrid) have relied on the good will of the public, prizes, fun, fame or collaborative advantages in order to sustain the number of participants - who are often the largest pool of resources providers - on their grid. However without a more flexible and exchangeable reward such as real currency, the participation rate is more likely to drop [14], hence leading to reductions in grid performance and/or unfair usage of the remaining resources within the system.

All systems involving goals, resources, and actions can be viewed in economic terms [16]. If one considers the use of resources on a data grid on offer as various types of services then grid resource providers can be considered analogous to service suppliers and applications (and ultimately grid users) as service consumers[2]. In grid systems, brokers and service providers are free to choose their own private strategy, which cannot be imposed from outside.

Economic perspectives have already been applied to manage fair resource usage allocations in current grid brokers such as Nimrod-G [14][17] and Gridbus [18][34]. A data replication strategy from an economic model is proposed in [22].

In an economic grid model there are two key types of players: *service providers* and *service consumers* [14]. Service consumers are users and applications that interact with their resource brokers to demand the use of resources (a service). Nodes that make their resources available for use are service providers. The motivation for service consumers in a data replication perspective with a decoupled data scheduler, for the purpose of replication, is to satisfy the requirements of a replication transfer (or file transfer) at the lowest cost within an optional constraint such as time. The motivation for service providers is to maximize the investment of their resources (such as bandwidth) so that it attracts the most amount revenue from its customers. Furthermore, the two types of players are not necessarily exclusive entities. A consumer may become a provider with its idle resources in order to earn back spent credits and vice-versa. Players in this system are assumed to be non-cooperative and strategic [2] without giving consideration in explicitly optimizing for the grid as a whole. However, like its analogy in the real world, market forces will implicitly guide service usage – in this case bandwidth – towards a balanced and sustainable global equilibrium.

The medium for valuation and purchasing services can be real currency or it can be abstract grid tokens or credits as long as they can be redeemed for future services. The act of purchasing of services or *transactions* can be mediated through a market directory. The interaction between the mediator and players involved in the transaction are determined by a particular economic model with its associated *transaction protocol*. These models, discussed in the following section, will specify the process in determining the price for the use of a grid resource(s), and transaction protocols prescribe how to trade, methods in accounting, billing and other analogous fiscal services.

A complete implementation of an economic grid requires the following services [14]:

- A market directory for publicizing participating grid players.
- Models of establishing the value of services so that suppliers are able to create a pricing scheme. This includes negotiation protocols for the transfer of services for credits (or currency).

[2] The use of the terms “service” and “resource” can be confusing due to the shifting perspectives - from economic to computer science - of various research publications. One can say that resources are used in delivering services in the grid. To call a node a service provider *and* a resource provider is not technically incorrect. In terms of grid entities, service providers and resource providers are the same entity. However, a resource broker technically brokers for a service and not the ownership of a resource. Therefore resource brokers are technically service consumers not resource consumers.

- Service pricing schemes and publishing methods.
- Regulatory agency for establishing currency standards and service value.
- Accounting, billing and payment methods

In Figure 4 we see an abstract scenario where software components that provide the previously listed services may relate, a similar diagram was first presented in [14]. It has been applied successfully to brokers via Nimrod-G [17] and Gridbus [18].

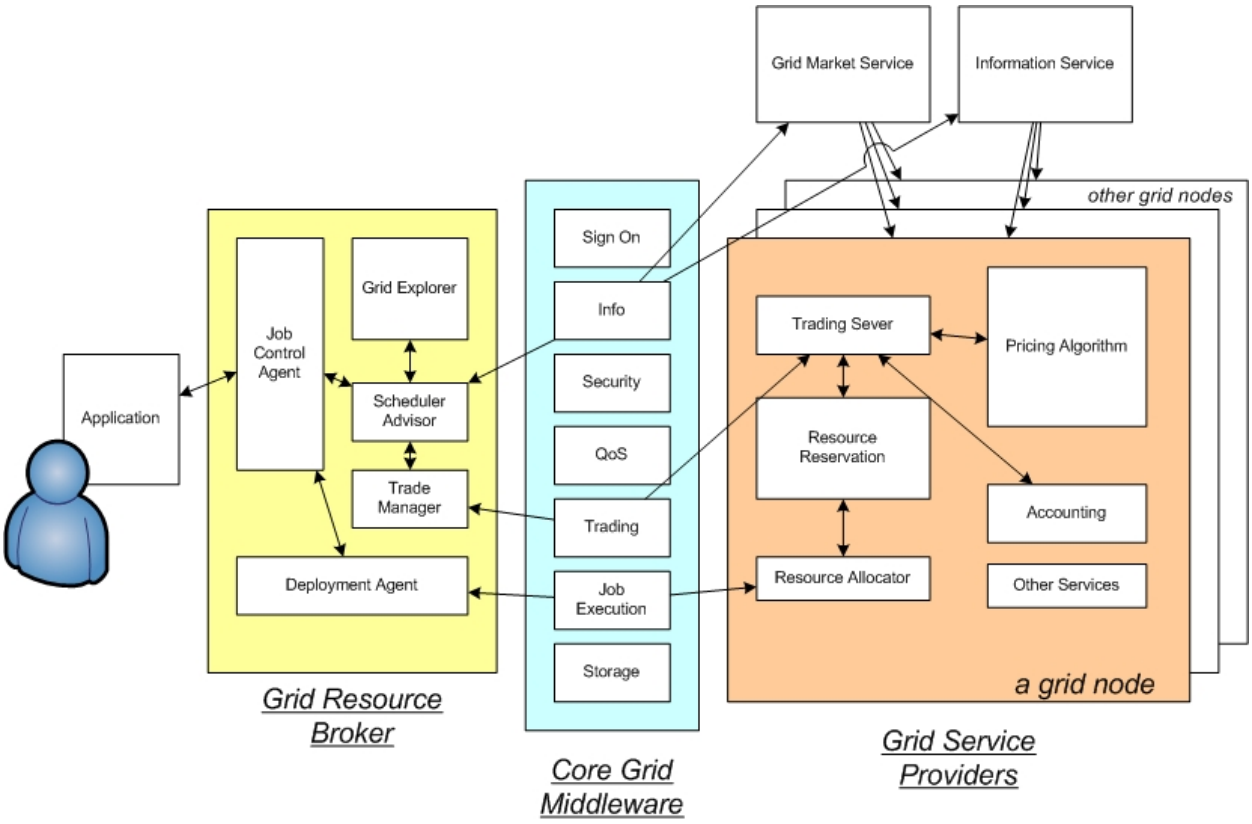


Figure 4: An abstract scenario of interacting grid components under an economic model [14].

The benefits of economy based resource management include [24]:

- The system motivates resource owners to contribute their idle resources.
- Provides a fair way to access grid resources.
- Helps balance supply and demand of services in the grid.
- Distributed management of resources and makes the grid much more scalable to larger sizes (however small grids will have issues with trading monopolies).
- Uniform trading of all resources via currency.
- Based on the field of economics so that trading policies can be derived from analogous methods that worked well in the real world.

When economic resource management models are applied to replication management it translates the question of where (and when to some degree) to replicate data into the sub-problems of costing and valuation of services. How does a service provider decide on the cost of a service that best fits their motivation and in a related issue how does a service consumer value a service that one receives?

3.2 Economic Models and Transaction Protocols

Many models derived from microeconomic and macroeconomic principles have been proposed for research not just limited to grid based systems, some of them are listed below:

- Commodity market model
- Posted price model
- Bargaining model
- Auction model
- Bid-based proportional resource sharing model
- Bartering model

The choice of selecting a protocol for a domain lies in its ability to maximize the total and/or mean social welfare[3] (according to a set of criteria) of the grid, while satisfying the rational strategy of the individual. In data replication, welfare can be measured in terms of:

- The quality of service a consumer experiences during a transfer.
- The average time it takes to complete a transfer.
- The buying power of a broker (the amount of data it can afford).
- Any other means of quantitative performance measurement or qualitative feedback.

Measures such as stability, communication overheads in terms of software and networking should also be considered as well as the distribution of welfare to ensure fairness and reduce symptoms of bad resource management such as starvation.

The transaction protocol of economy schedulers is analogous to grid resource scheduling policies. In Appendix C I have presented three of the most widely applicable economic models in terms of their transaction protocols and how they are related to data replication scheduling.

[3] The term "welfare" is essentially an economic qualitative measurement of how well off an individual is.

Chapter 4: Related Scheduling Policies

4.1 Scheduling with Economic Models

The following sections will present related scheduling algorithms that utilize economic models discussed in the previous chapter. While the focus here is on replication scheduling schemes, economic approaches for scheduling other resources will also be discussed for coverage.

4.1.1 Data Replication Scheme Using an Auction Model

In [21], a dynamic replication scheme was proposed based on a variety Vickery auction for an economy based replication solution. This proposal attempts to solve all the questions of when and where regarding replication management in data grids. The policies proposed methods in:

1. Determining the best file for target replication when given a request by a job for a particular file.
2. Decide when to delete and replicate files.

The scheme was simulated as a coupled job and data scheduler in a data intensive computing grid. From the component architecture diagram (Figure 5), computational jobs that require large amounts of data are submitted to the local resource broker over time. The resource broker would schedule jobs to the computing element of a remote node with the motivation to improve the overall throughput of the grid. The resource optimizer is a part of the replica manager of a data grid middleware. The optimizer is the key component that selects files for replication and deletion of local replicas.

The replication process is triggered by jobs requiring data not found locally at the remote execution site. The broker will then start a reverse Vickery auction to obtain files for that job. In [21] the roles of resource brokers and service providers are reversed. The broker of a service consumer initiates the bidding process and multiple service providers bid for their patronage. In short, the service providers become the bidders instead of the consumers and the auction winner is the bidder bidding the lowest cost.

The replica optimizer (RO) initiates a bid by sending a propagated auction message where the propagation rate is limited by a parameter. The RO then waits for a limited time for bids to arrive locally then sends a notification to other nodes informing them of the result of the auction. When a grid node receives an auction initiation message, they will bid only if they have the file locally. If they don't have a local copy of the file then they may, in turn, conduct a nested auction to obtain a local file replica. Once they've obtained the local file, they will then bid in the initial auction call.

There are important motivational distinctions between the initial auction call and the decision to undertake nested auctions. In the former, it is to acquire the cheapest file transfer service to a job execution site. In the prior it is to invest in a replica for its revenue earning potential. The decision made by a node to replicate in the nested auction results in a long term data grid optimization in order to recognize and diffuse demand hot spots. This decision is made on the basis of a revenue prediction function on a per file basis where:

Predicted Revenue = func(predicted demand within a window of time)

The predicted demand can use either a random walk or Gaussian random walk for file demands. This prediction is then matched to its initial cost and whether there is enough local storage capacity. If there isn't any capacity all local replicas are compared in their predicted revenue and if there exists a file with lower predicted revenue then it is replaced with the new replica.

It is claimed that the strength of Vickery auction lies in the fact that bidders are motivated to bid their true valuation rather than to over or under bid. Bidding too little would result in lower then acceptable revenue for the bidder and bidding a higher pricing would run the risk of losing the auction. In fact, the Vickery auction would more likely to incite the bidder to take greater risks in bidding lower prices as long as they win the bid, they would only need to charge the consumer at the second lowest cost.

The motivation of this scheme does not assess nor consider the quality of service that a job or transfer may receive. By not assessing the quality of service one receives from a provider, the consumer cannot be considered completely

economically rational. From an economic perspective this blind side to quality of service allows for lower quality services to be offered at lower costs – slowly forcing a global reduction in grid service quality over time as higher quality services become ignored and unprofitable. Factors of transfer quality such as node-to-node link reliability, actual transfer times, and server reliability have been ignored so that future broker decisions will make the same choices.

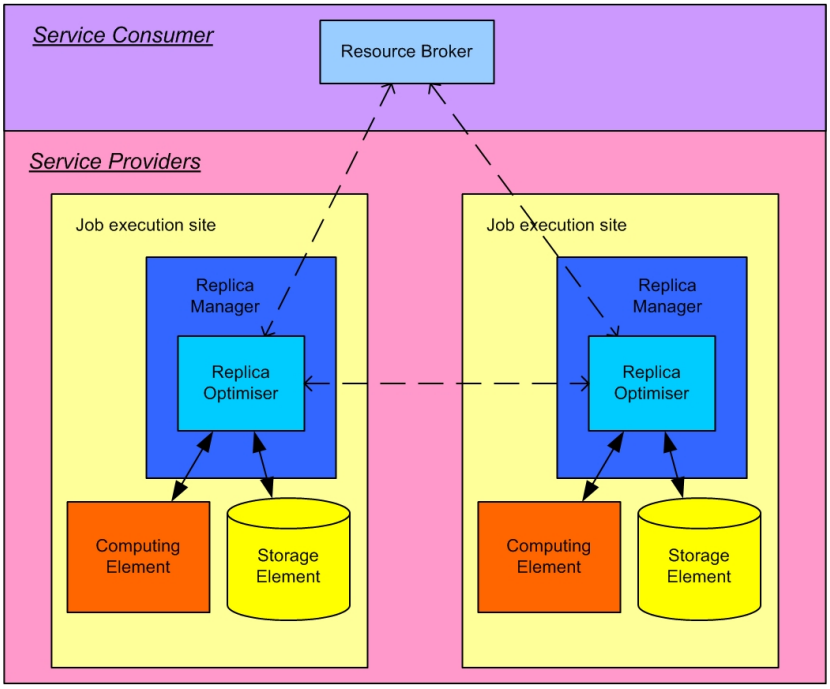


Figure 5: Data grid component interactions.

4.1.2 The Gridbus Broker

The Gridbus broker presented in [18][34] extends Nimrod-G [17], a computational resource broker, to cater to data grids under an economic model. The broker considers a data-intensive computing environment as a collection of resources hosting data, surrounded by compute nodes. The broker essentially schedules to a job to the computing resource so that the total time for a job is minimized – it assumes that data transfer takes a significant portion of the total turnaround time and factors the transfer times of all data requirements of the job to a remote node.

The scheduling mechanism that is focused here is the *budget and deadline constrained scheduling algorithm* first proposed for Nimrod/G. This scheduler works under the assumption that there exists a centralized costing matrix (Figure 6) that maps consumers and service providers to their associated service costs. For example a service for consumer 1 from service provider 4 will incur a cost of 3.

	1	2	3	4	5
1	1			3	
2					
3					
4	2			1	
5					

Figure 6: The Nimrod/G costing matrix.

The scheduler is given a set of jobs to schedule within the given constraints of time and/or budget per job. The scheduler applies the following heuristic for scheduling jobs:

1. Enumerate and identify the lowest-cost set of services able to meet deadlines by querying the cost matrix and the service directory.
2. Unscheduled jobs (which are maintained in a pool) are allocated to the candidate providers from 1.
3. The completion of the job is monitored and job execution rate is established for each resource.
4. The rate information in 3 is used to update estimates of typical execution times of on different providers and hence the expected job completion time. The refinement process may be repeated at 1, 2 or to drop existing candidate servers.

The heuristic is continued until either budget is exceeded or deadline is met. The deadline can be modified if it's too restricted by budget or simply cannot be met by available services. In relation to the Gridbus broker, the scheduler is extended to account for data transfer times as well as job execution time.

The underlying assumption for the Nimrod-G and the Gridbus brokers is that one can reasonably predicted the completion times of transfer and execution times. On a grid that can reserve local resources for maintaining a guaranteed standard of service by say reserving a percentage of compute cycles this is a reasonable assumption. However, with data replication where the main issue is the transfer of large volumes of data through a network, one cannot easily predict the completion times of transfers. In a grid, transfer speeds can be dictated by the link bottleneck which can occur at the uplink of the service provider, the downlink of the consumer or anywhere in between.

While the brokers presented here are not data replication specific brokers or schemes, it serves to highlight other grid related scheduling heuristics as well as to underline the different directions from which optimization in the grid can be achieved. From both Sections 4.1.1 and 4.1.2 we see how grid performance can be improved by scheduling internally by the broker and externally as an implied effect from the price mechanism of economic models.

4.2 Scheduling with Non-economic Models

Early in the development of data grids and until this day there are still many proposals for data replication schemes that do not include utilizing an economic resource management scheme. These schemes often required centralized policy that ignores the individual requirements and motivations of the grid nodes.

4.2.1 Early Studies

In [25] several simple data replication strategies were suggested and simulated for their data saving properties on a grid model that resembles the hierarchical network structure of LHC:

Best Client:

In this method, all nodes in the grid maintained a detailed history for each file that is contained locally, summarizing the number of requests for every file and the originating node of the request. At a set time interval, each node would check for any files with requests that have exceeded a predefined threshold. For those files that have exceeded the threshold, the node will actively push a replica of that file to the node with that had registered the most requests and the request records for the node-file association is reset.

Cascading Replication:

In this method the network is perceived as layers or levels in a tree structure (Figure 7). Each node has a threshold for requests for a file per node, like in the best client scheme. Only in cascading replication, when the threshold is exceeded, the replica is pushed onto an ancestor of the best client which is one level removed from the current node. This will eventually put replicas closer and closer to the best client (if demand rate continues). In this strategy the storage space of all intermediate nodes from the original data provider to the terminal nodes of the network tree can be utilized for replication. A further advantage is that it takes advantage of geographical locality as data is slowly brought closer and closer to the source of the demand.

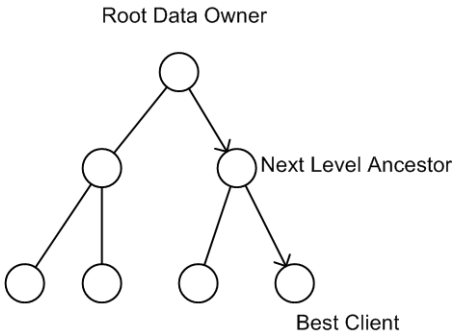


Figure 7: Data replication flow for cascading replication.

Plain Caching: Requests temporarily store files locally.

Caching plus Replication Caching:

This combines plain caching and cascading replication where requesters always cache files and the server periodically pushes replicas for popular files to the next level when a popularity threshold is exceeded.

Fast Spread:

This method stores a replica at each intermediate node along the path to a requesting node.

All the methods above use same file replacement policy when storage is completely utilized. Periodically, the node will check its replicas and if there's are files below a threshold of popularity then it will remove the oldest of these files from storage.

Under simulation, it was determined that Fast Spread performed consistently better than its rivals in terms of bandwidth savings and response times for data requests in a grid. This is to the cost of increased storage usage and excessive initial data transfers. Cascading replication and its caching variant was found to use around 50% less storage space but less bandwidth savings where made. The Best Client method performed the worse and this was particularly exacerbated by random file demand patterns.

The methods proposed are an interesting variant of replication scheduling where replicas are pushed by remote nodes instead of being pulled by local policy. This may present some problems especially in LHC where many heterogeneous organizational and national interests may not allow for their resources to be utilized without being able to exercise local control. Furthermore, a Fast Spread technique may become too taxing on network bandwidth as each file, ranging between 2 to 10 gigabytes, will be cyclically replaced all over the grid. As storage runs out of space, it will remove a replica, only to have to replica it again at a later date.

4.2.2 More Static and Adaptive Data Replication Algorithms

Two replication algorithms were proposed in [26] were not constrained for use in a grid but also for information networks in general. The static algorithms are designed to run once a day after collecting read and write statistics and compute a good solution for replication allocation in a network. The results were to be computed and disseminated from a central source. The assumptions are that there is always one primary copy of a set of data – defined as an object – for each object in the network and that the immutable primary site of the original object is responsible for maintaining the list of sites that the replica is contained in. Every site x also keeps a record for every object and the object's associated primary site p and the "nearest" site q that contains that object relative to x.

The main goal of both algorithms is to minimize the total network transfer cost (NTC) due to object transfers. The model used to define NTC is a function of two components: read and writes to and from objects. The total NTC is the sum of all read and writes from all sites for all the duration under consideration, given by the value D.

The model is further refined to translate the problem statement so that given that there is M sites in the network and there are N objects in the system. And given an M x N matrix that represents the configuration of all the objects or

object replicas on each site where a 1 represents that an object *n* is stored at site *m* and where a 0 represents that *n* is not stored on *m*; the Data Replication Problem (DPR) is defined as:

Find the assignment of 0, 1 values in the M x N matrix that minimizes D
Subject to storage constraints and primary site copy condition

Static Algorithm: A Greedy Method

The greedy method first defines a benefit value *B* for replicating an object *n* to site *m*. *B'* is computed by using the difference between the NTC occurred from current read requests and the NTC which would be non existent if there existed a local replica of *n* plus the NTC from updates to *n*. *B* is then computed by dividing *B'* by the object size of *n* so that we consider the benefit per storage unit. The goal is to maximize the total value of *B* within the DRP constraints for all *M* x *N* assignments.

The algorithm for the static replication algorithm (SRA) is as follows:

1. In each step a site S is chosen from the set of candidate sites that still conform to the DPR constraints in a round-robin fashion.
2. And for the set of all the objects L that can be replicated on S within constraints, find the object n with the highest B value.
3. The n is associated with S in the M x N matrix.
4. Repeat from1.

Dynamic Algorithm: An Adaptive Genetic Replication Algorithm (AGRA)

The SRA has shortcomings when there a large number of reads and storage capacity is too limited. Furthermore while the SRA is claimed to be fast enough for dynamically computing complete replication allocations for the entire network – that is, updating the replication allocations on a more frequent basis – results show that it sometimes produces results that are worse than the current allocation. To overcome this issue a hybrid genetic algorithm was proposed. AGRA is a compromise between a full genetic algorithm (GA) implementation and the SRA method. It does not under go the full epoc of a GA but instead computes results incrementally.

A deeper coverage of genetic algorithms introduced by Holland [27] is outside the scope of this thesis. However, on a superficial level, the *M* x *N* matrix of the object allocations is essentially the solution to the DPR. By mapping this solution to a one dimensional data structure (an array of bits), one can combine and mutate a whole population of candidate solutions to evolve, on average, a population of better solutions. Better solutions are recognized by comparing solutions via a fitness function. This fitness function is basically the *D* function.

For the GRA an initial full run of the genetic algorithm is run to create a population of good initial solutions *R* for replica allocation. Then, each time the read/write pattern of an object changes above a threshold value in favor of reads or writes, the AGRA will attempt to compute a new replica allocation. The AGRA will take in the new pattern; generate set of replication schemes *r* which is added to *R* to make *R'*. *R'* is then evolved for a few more generations to derive a better set of solutions. The solution with the highest fitness is then chosen out of *R'* and disseminated into the network.

The two algorithms for general network replication obviously have similar weaknesses that the algorithms in Section 4.2.1 have. They are all centralized replica management schemes and hence suffer from scalability issues. Compared to cascading replication and fast spread, SRA and AGRA are computationally intensive which will only serve to highlight scalability issues when there are many service providers and many data sets available for replication. All the algorithms in Section 4.2 explicitly attempt to optimize for global network benefits and largely ignore the desires of the individual nodes.

For all the algorithms and schemes that have been covered in this chapter, none of them considered improving the method in which replication transfer should occur. In fact, the method of transferring data will both affect the performance overheads of creating replica and the way all data is accessed in the grid even with near optimal replica pre-placements.

Chapter 5: Data Replication Broker Policies

5.1 Policy Overview

The problem of finding an optimal replica allocation at a given state of the grid, i.e. an allocation that has minimum transfer cost for a given read-write pattern, has been shown to be NP-complete [28]. Furthermore, various centralized pre-computed allocation methods are slow and quite infeasible in grid environments where participants number from the hundreds to the thousands. The distributed nature of grid environments and its issues favor the price mechanism driven resource management scheduling polices. When individual players in the node make decisions based on economic rationality, economic theory indicates that the market economy will iterate towards a balanced and sustainable system of service demand and supply [15].

This chapter will detail improvements to data replication schemes for a data grid that utilizes a *commodity* market economy (see Appendix C for details) for the purposes of resource management. The improvements will be implemented as policies into a data replication broker[4] for transferring replicas. If used on its own the improvements are equivalent to a replication scheme that is driven by user initiated data replication requirements, so that given a selected file, this scheme will deduce how and where to retrieve the file from. The reasons for not implementing automated replication selection and internal queuing for transfer requests is twofold:

1. In shared facilities such as the regional centers for LHC experiments, there will be instances where centers with limited capacity do not require automated file selection for replication. Tier 2 and tier 3 centers may simply utilize human judgment for selecting popular data requirements. This will be especially apparent for desktop users who require data for their application.
2. Secondly, the improvements will not only aim to improve replications, it will serve to improve the manner in which large amounts of data is moved throughout the data grid. Therefore it is expected that the improvements can be incorporated into more complete replication schemes that automate replica selection.

While the software architecture for data grid middleware and especially the replication system is still in constant flux, Figure 8 describes the services that are most commonly used in proposed data grid data replication schemes and shows where the resource broker is situated. The file transfer functionality of the data replication broker is accessed in three scenarios:

1. A user application that requires data.
2. An automated replication scheme through the replica selection service has initiated replication.
3. Site owner that has foreknowledge of expected data demand.

In both situations the resource broker is given the identity of the file to be replicated and provided there is enough credit to complete the transfer service, the broker will select the “best” resource provider to initiate the transfer.

[4] Note that in the following chapters the terms ‘data replication broker’, ‘broker’ and ‘resource broker’ will be used interchangeably, they will all refer to the former. Additional clarifications will be made if there are any specific distinctions to be made between a generic broker and the data replication broker.

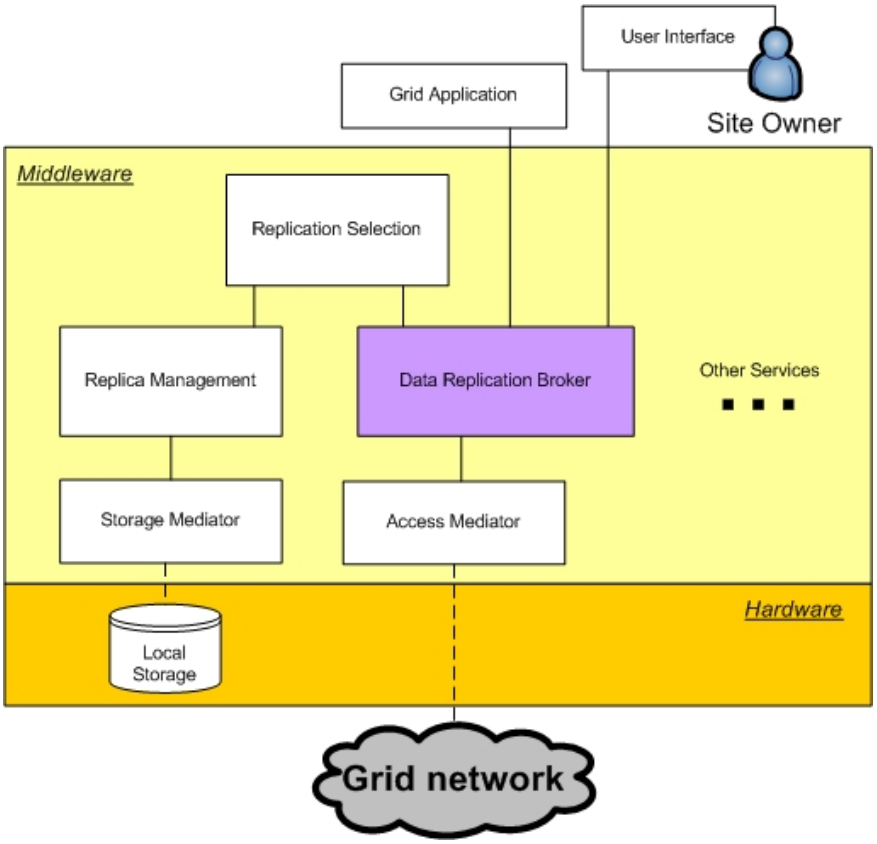


Figure 8: Top level data replication services.

5.2 Broker Operating Environment

The performance of the broker for data transfers will be simulated and assessed in a complex environment so that the benefits of the broker policies and market economy behaviors emerge. The operational environment during simulation for the broker policies are outlined below.

5.2.1 Networking

The smallest conceptual granularity of data sent through the network for file transfers are *fragments*. The sizes of the fragments are a parameter of simulation set at a default maximum size of 10MB, to minimize waste, smaller sized fragments are allowed for the last fragment of a file transfer. The larger than normal maximum size of the fragment is used to improve the efficiency of the simulation.

The grid fabric uses a fair share policy for bandwidth. That is if there are x number of transfers on a link between site a and site b, and if both sites have the same bandwidth y both up and downstream respectively then the maximum effective transfer speed (ETS) of every ongoing transfer is:

$$\mathrm{ETS} = \frac{y}{x} \quad (Mbit/s)$$

In the more complicated for the case where there are many transfers from multiple sites to and from the one local node, we will consider the perspective of how the local node will divide up its own bandwidth for individual transfers. For simplicity assume that the maximum bandwidth is shared by both up and downstream traffic. When an incoming data transfer is to be introduced, the local node will attempt to adjust the transfer speed of all ongoing transfers so that every transfer receives an equivalent proportion of the total bandwidth. There are two consequences of allocating that proportion of bandwidth:

1. If the actual speed of transfer is smaller than the bandwidth portion to which it is allocated to (i.e. the network link or the remote node is the transfer bottleneck), then the unused remainder of that proportion is evenly distributed to the other transfers.
2. If the speed is greater (the local node is the transfer bottleneck), then the transfer speed is simple limited to the allocated portion of the bandwidth.

5.2.2 Service Errors

During a transfer three types of transfer service errors analogous to real world networks can occur:

1. *Error fragments*: This can be detected immediately by the service consumer and corrected by the server[5] by resending the fragment.
2. *No connection*: This stops the file transfer from starting and can be detected by the consumer after a connection timeout set to 60 seconds. The consumer node may try a different server or retry connection for the same server depending on broker policy.
3. *Dropped connection: This* occurs during the middle of a transfer, but always after the completion of a fragment. Dropped connections are dealt by the consumer node in the same manner as in the event of no errors.

Errors of these three types can come from varies parts of the network, often due to a busy server. Error fragments can come from a corrupted file or errors in the network fabric. Dropped connections and no connections may also arise from fragment acknowledgements from the transfer receiver that are lost. Dropped and no connections are collectively termed as *bad connection errors*.

The modeling of characteristic error patterns for simulation is outside the scopes of this thesis, errors occur on a random basis specified by simulation parameters. No connection errors occur according to a failure probability parameter e_n that is specified for each grid node (set to 1% as default) of every incoming transfer connection request. The probability for an error to occur during transfer, that is an error fragment or a dropped connection to occur is specified by a parameter e_t for every fragment sent. During transfer, if a fragment is selected to be an error then there is a 15% chance that that error will cause a dropped connection and 85% chance it will cause an error fragment.

If a connection is dropped during transfer the service consumer can resume the service in the same condition as it was before the connection was dropped.

5.2.3 Service[6] Pricing Policy

The service provider's ultimate goal is to try to maximize its revenue. Therefore, in order to achieve the most amount of revenue over time from file transfers, the service provider will attempt to increase the chances of return customers by maintaining a standard of service while setting a competitive price. Since the grid network uses a fair share system, and the fact that LHC regional centers will have high speed links all through tier 0 to 2, the main bottleneck for transfers will likely to be at the server end. Furthermore, the bottleneck will be dictated by the number of concurrent transfers.

Hence, all the service providers will use a variable pricing policy that depends on the average number of concurrent transfers over a window of time. This policy is used to approximate the supply and demand behavior in real world markets where increased demand will push the price of a service upwards (given that serving ability for the supplier does not change). This will have the effect of reducing demand on heavily loaded machines as economically rational consumers look for cheaper alternatives. The pricing function for a price per megabyte p on a service provider j is defined as:

---

[5] The terms "service provider" and "server" will be used interchangeably

[6] Here the term "service" refers to transfer services. The terms "transfer service" and "service" will be used interchangeably and should not be confused with other grid services. When discussing services for the grid in general, the context will be made clear.

$$p(t_j) = \begin{cases} b_j, q_1 \le t_j \le q_2 \\ \frac{3b_j}{2}, q_2 < t_j \le q_3 \\ 2b_j, q_3 < t_j \le q_4 \\ 3b_j, t_j > q_4 \end{cases} \quad \text{where} \quad \begin{array}{l} b_j = \text{base price for node } j \\ t_j = \text{number of transfers for node } j \\ 0 \le q_1 < q_2 < q_3 < q_4 \text{ are contants} \end{array}$$

The constant base price of the service is essentially multiplied by the level of the demand; the function above is left intentionally non-continuous to reduce price fluctuation and to reflect the manner in which pricing decisions are made after a period of observation and analysis. The constants q_1, q_2, q_3, q_4 are constants that define the boundary of the pricing classes.

A node will periodically upload the price of its transfer service along with a list of served files to the nearest service directory.

The pricing algorithm in this thesis is kept simple as the costing of service is a complex problem where a more complete solution should incorporate many quantitative variables such as investment costs, fixed maintenance costs, depreciation, seasonal demand; and qualitative variables such as consumer perception. However, the main motivation for implementing a dynamic costing policy was to simulate the resource balancing effects of an economy market and to this end, a simple function should be sufficient.

5.2.4 Transaction Protocol

The broker policies will be simulated on a grid where services are offered for grid credits; in this grid every service provider is offering the service to transfer files that they have. The cost of a service is measured in how much data is transferred on a per megabyte basis. By posting a price on the service directory, the service provider is agreeing to a contract to charge a service consumer this price until the price has timed out at the service directory. Furthermore the per megabyte price of the service is confirmed between the broker and the service provider at the beginning of transfer. Therefore, even if the service provider updates its service rate during a transfer, the consumer will still be charged at the initial confirmed rate.

The general protocol for file transfer is as follows:

1. A replication request of size $f > 0$ MB is received by the broker.
2. The broker communicates with the nearest service directory and retrieves the *service list* (list of service providers) that is able to serve the file. This list also contains the cost of transferring data per megabyte p_j associated with some server j. This cost is the agreed service charge by the service provider.
3. The broker will input the service list, the file request and other statistical information required into the transfer policy. The policy algorithm will output the "best" service provider j^*.
4. The broker will confirm the transfer and the cost per MB of transfer p_{j^*} directly with j^*.
5. The broker will place the transfer request into the *transfer queue*.
6. Transfer will begin immediately and is monitored throughout the transfer.
7. When the transfer is complete, the service provider will bill the broker for the total service cost T and inform the data grid bank, where:

$$T = f p_{j^*}$$

8. The broker pays the total service cost through the data grid bank.
9. Transfer details are registered and stored for use in future server selection decisions in the broker.

5.2.5 Additional Grid Services

In addition to standard data grid middleware services we assume there are network services that return basic network statistics such as:

- The number of transfer connections currently active on a server for grid traffic
- The maximum available bandwidth between two grid nodes
- The currently available bandwidth between two grid nodes

Actual network sensors such as NWS [29], which is a distributed network traffic forecasting system, would be able to collect these statistics. NWS works by deploying server software on the remote nodes to probe CPU and network connections on the grid. These servers then send NWS clients live and stored data regarding the grid.

5.3 Broker Transfer Policies

The following sections will detail the broker algorithms that determine the "best" server when given:

- A file replication request from the user (or replica selection service)
- A list of hosting servers from a service directory query

5.3.1 Least Cost Policy

The least cost policy will be used as a benchmark for the other transfer policies. In a market economy, this policy is the most basic of decision making heuristics.

The least cost algorithm finds the best server by finding the server with the least cost in the server list. If during the search, the current server has the same cost as the best server found so far, then we consider the server that is closest to the service consumer (the local node) as the new best server.

Current data grid replication schemes do not explicitly factor in one of the fundamental characteristics that enable real life consumers to make rational economic decisions. That is, they do not place emphasis on the quality of service that service consumers have received or are currently receiving in order to make better purchasing decisions in the future. The next 3 policies will gradually introduce policies that will increasingly take advantage of quality of service (QoS) statistics in order to improve data transfer performance.

5.3.2 Minimize Cost and Delay Policy (MCD-1)

The minimize cost and delay policy (MCD-1) is a heuristic that uses a scoring function to determine the "best" server which is the server with the highest score. The function is given the cost per MB of a server and the expected delay time for a file to finish transferring. The delay time begins from the start of a transfer for a file until its completion. The minimum delay time is a QoS estimation based on the current available bandwidth between a potential service provider and the broker. The fitness function is given below:

$$s_{ij}^{Dp} = -w_D D_{ij} - w_p p_j \times f$$

where:

s_{ij}^{Dp} is the score based on delay and cost between nodes ι and j

$w_D > 0$ is the weight for D_{ij}

D_{ij} is the estimated transfer delay in seconds between nodes i and j

$w_p > 0$ is the weight for p_j

p_j is the cost per MB for service provider j

f is file size of the requested file

The best server is decided as a compromise between choosing the servers with the shortest time required to fully transfer the file and the ones with the lowest total estimated service cost. The function has placed negatives in front of the weights since it makes intuitive sense that the best server has the best score. The weights w_D and w_p are parameters that determine the negative significance of increased cost and increased transfer delay. If a consumer values the time to transfer a file over increased costs the value w_D of should be increase and similarly if the consumer wishes to save credits and does not hold the delay time to be as important then w_p should be increased.
The algorithm for the MCD-1 policy is as follows:

```
bestServer ← Ø

for aServer in serverList

        if bestServer = Ø
                bestServer ← aServer
        else-if score_Dp(aServer) < score_Dp(bestServer)
                bestServer ← aServer
        else-if score_Dp(aServer) = score_Dp(bestServer)
                if distance(localNode, aServer) < distance(localNode, bestServer)
                        bestServer ← aServer
        else continue

return bestServer
```

Similar to the least cost algorithm, if two servers are found to be equally as good, then the algorithm will resolve the tie by choosing server that is closest to the broker.

5.3.3 Minimize Cost and Delay with Service Migration (MCD-2)

For job executions on computing grid it is quite easy for the service providers to control and reserve a standard for computational services. A job that pays for a compute service that agrees to provide x percentage of CPU cycles per unit of time will expect to receive that throughout the duration that its broker has negotiated for. Therefore it is a safe decision to commit to a service once it has begun.

However, in the case of data replication and data access, where the transfer of the file from one node to another is the service, a constant standard of service[7] is difficult to maintain in a busy network. This is because the transfer speed of a file through a busy network is difficult to predict over the short term, i.e. for the duration of a file transfer. As an improvement to the MCD-1 policy, the MCD-2 variant will allow transfers to be interrupted and migrated to a difference service provider when the current standard of service is deemed interruptible and there exists a better service provider for the transfer. In that case of connection failure, as with all other polices where there is a drop out or a no connection, the server will restart the transaction protocol and find a better server.

In order to implement transfer interruptions several new mechanisms need to be introduced to the current framework:

1. Modification to the transaction protocol.
2. Periodic monitoring of the current standard of service per transfer. For transfers, the most important short term quality of service measurement is the speed of the transfer.
3. A sub-policy to decide when it is appropriate to interrupt transfers and migrate to another provider.

Modifications to the Transaction Protocol:

Two terms are clarified:

1. The *transfer* refers to the complete transfer of a file from receiving the first fragment of the file until the last service provider receives the acknowledgment for the last fragment of the file.
2. A *transfer session* refers to act of sending data from a service provider to a consumer before a transfer migration occurs. A transfer is made up of one or more transfer sessions (see example in Figure 9). Each session is associated with a file, a broker and a service provider. A transfer may have multiple transfer sessions with the same service provider but not concurrently nor immediately sequential. A broker may not have concurrent sessions of the same file transfer – this is outside the scope of this thesis.

The transaction policy with service migration is given below:

1. A replication request of size $f > 0$ MB is received by the broker.
2. The broker communicates with the nearest service directory and retrieves the service list.
3. The broker will input the service list, the file request and other statistical information required into the transfer policy. The policy algorithm will output the "best" service provider j^*_1.
4. The broker will confirm the transfer session and the cost per MB of transfer directly with j^*_1.
5. The broker will place the transfer request into the transfer queue.
6. Transfer session will begin immediately and is monitored throughout the session
7. If the session is a candidate for interruption and there is a better service provider than the current. Then:
    a. Send interrupt acknowledgement to j^*_1 and place the current transfer request on the *suspended queue*.
    b. j^*_1 will stop the session and bill the broker for the $f' < f$ MB of data transferred during the duration if the session.
    c. Broker pays for the service and repeats the protocol from step 2.
8. When the transfer is complete, the final service provider will bill the broker for its session cost and inform the grid bank [34].
9. The broker pays the total service cost through the grid bank.

The transaction protocol is also slightly different to previous policies in terms of the way the broker deals with dropped connections and busy servers. If a transfer is dropped, the broker will respond as if the broker's transfer interrupt policy had interrupted the download, that is it automatically informs the service provider and suspends the transfer request and identifies a better server (Figure 9).

[7] Note the use of the term "service" is subtly different from "service provider". The discussing a service it includes the link and the service provider relative to the service consumer.

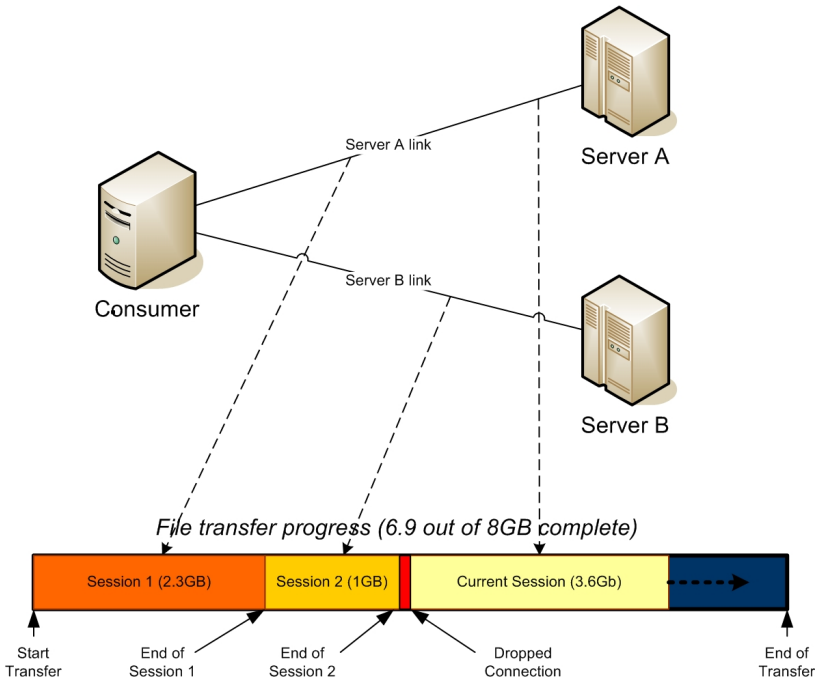


Figure 9: Transfer migration scenario of an 8GB file transfer. During session 1 the broker found a better server and migrated to server B. During the middle of session 2, the service drops the connection. The broker finds server A has improved and is better and resumes transfer.

Transfer Session Monitoring

During a session the broker will update the session transfer statistics after receiving every fragment. Therefore there is a requirement for this statistics to be computationally efficient to update. The session statistics include:

- Current file offset which represents the total amount of the file transferred.
- Current transfer speed in Mbits/sec.
- Current running average change in transfer speed with respect to successive fragments received. This value is needed to evaluate the quality of a transfer session, not just for the current fragment. This is achieved by giving previous changes in transfer speeds a geometrically decreasing significance while maintaining an output value that is positive for increasing speed and negative for decreasing transfer speeds:

  Let:

  $S_n \geq 0$ be the speed for transferring the nth fragment

  $n \geq 0$ be the fragment number of the current fragement

  $n-1$ = the previously received fragment

  R_n = the running average change of the session speed at the nth fragment

$$\because R_n = (S_n - S_{n-1}) + R_{n-1} \Big/ 2$$

Let:

$$\Delta S_n = S_n - S_{n-1}$$
$$\Delta S_0 = 0$$

$$\therefore R_n = \frac{1}{2}(\Delta S_n + R_{n-1}) = \frac{1}{2}(\Delta S_n + \frac{1}{2}(\Delta S_{n-1} + \frac{1}{2}(\Delta S_{n-2} + \ldots))$$
$$= \frac{\Delta S_n}{2} + \frac{\Delta S_{n-1}}{4} + \frac{\Delta S_{n-2}}{8} + \ldots + \frac{\Delta S_0}{2^n}$$
$$= \sum_{i=0}^{n} \frac{\Delta S_{n-i}}{2^{i+1}}$$

From the expansion of the R_n formula one can verify that the numerical significance of successive changes in transfer speeds to the R_n value is geometrically reduced. The formula has the added benefit of being simple to compute as it only requires three numerical operations.

Interrupt Sub-policy:

It is important to define an interrupt sub-policy that benefits a transfer. This is because interrupting a transfer has both a marginal computational overhead and a significant communications overhead. The communications overhead consists of re-querying the service directory for the server list to find better servers and if an interruption does occur, the communications costs of the transaction protocol may be imminent. There are two scenarios that cause an interruption to occur: 1) From a bad connection 2) the algorithm below which is run periodically by the broker to trigger for interruptions.

```
minimumDuration ← a
maximumDuration← b
allocatedBWFraction ← u
minimumRn ← r

for aSession in transferQueue

        if aSession.sessionDuration < minimumDuration
                continue
        else-if aSession.usedBWFraction < allocatedBWFraction ∪ aSession.sessionDuration > maximumDuration
                if aSession.Rn < minimumRn
                        bestServer ← findBestServer(aSession)
                        if bestServer ≠ aSession.currentServer
                                interrupt(aSession)

return
```

In order to reduce the amount of communications overheads, the interrupt policy first checks a session to determine whether it is a candidate interruption without querying for a better server.

- A session is not a candidate if it has not been transferring for at least a minimum duration of 30 seconds. This is to given a session a chance to stabilize its transfer speed and R_n value.

- A session is a candidate if it is using less than 80% of its originally allocated bandwidth in the fair share networking policy and its R_n value is lower than the minimum R_n value of -10. Using less than the allocation signifies that the transfer is being bottlenecked by either the link between the remote node and the local node or by the server. However, we need to give significance to the original policy that determined that this was the best server so unless the transfer speed has also been dropping over the session then we do not consider the session a candidate.
- A session is also a candidate if it has been transferring for longer than the maximum duration of 120 seconds, this is so in the case that a better server has emerged in the system which is deemed much more likely after 2 minutes and furthermore the overhead experienced by the best server query is limited by the 2 minute cycle.

If it is a candidate, the algorithm then determines whether it should interrupt the candidate by querying for the "best" server and if a better server is found than the current, the broker will send a signal for the transfer to be interrupted.

5.3.4 MCD-2 with Monte Carlo Reliability Selection (MCD-2R)

One further enhancement to the MCD-2 policy is to factor the long term effects from experiencing many faults. From the consumer's perspective, bad fragments, dropped connections and no connections do not form a large penalty and do not explicitly affect how much they pay for a service. For example if a service transfers a total of 1.2GB of data in one session and of that data, 0.2% of that was bad fragments then 2.4MB of the data sent was wasted bandwidth and extra time required for transfer. However the broker will not be billed for those fragments with bad data. The amount of dropped connections and no connections also introduce a constant time penalty for transfer requests, reducing the overall quality of service that a broker and ultimately, the user experiences. On a larger scale and over time the effects of errors can be cumulatively significant on a grid wide scale.

The differences in the amount of errors experienced by a consumer from different service providers over time can be largely associated with external reliability of the link to the server and the server itself. Therefore it can be a viable option for a policy to choose a server which, although may not be best server according to some fitness or score function, but may be a more reliable one. In this section we propose a long term dynamic MCD-2 policy that reduces the usage of services that has historically generated more errors than its competition.

The MCD-2R policy will generate a list of candidate providers and then randomly select a "best" server using a biased roulette wheel selection technique from the candidate list (Figure 10). The conditions for candidate selection utilize the MCD-1 scoring function to calculate the best service providers within some Δ points (default $\Delta = 200$) from the best service provider limited to a maximum of x_{cmax} servers (default $x_{cmax} = 3$). This serves to limit the amount of candidate server to the most elite scoring servers. Then from the candidate list, the policy will choose the best server from a biased roulette wheel selection using the historical reliability of individual server services as experienced by the broker as weights.

The biased roulette wheel selection method is a Monte Carlo method for selecting, on average, the server with the most reliable service relative to the local node. This method first assigns slots for every candidate. The size (or numerical range) of the slot is equal to the reliability score *Rs* of the server (Figure 10). Successive slot ranges are contiguous, where the first slot for server *s1* ranges $[1,R_{s1}]$, then second slot for server *s2* ranges $[(1+R_{s1}),(R_{s1}+R_{s2})]$ and so on. A random number between 1 and the total *Rs* score of the candidates is generated. The server associated with the slot in which this random number's value falls between is selected as the service provider for the transfer session.

This stochastic method is used since the reliability measurements are a rough estimation of the reliability of the servers at best. In order to further refine the reliability statistics on a server, the broker must risk using its service. These measurements will fluctuate when events such as servers becoming unavailable or the network is improved. Therefore, by using a stochastic method the broker will still take the risk of using less reliable services in return for the usage of a possibly better server, gathering further reliability statistics and in the hope that the server has improved.

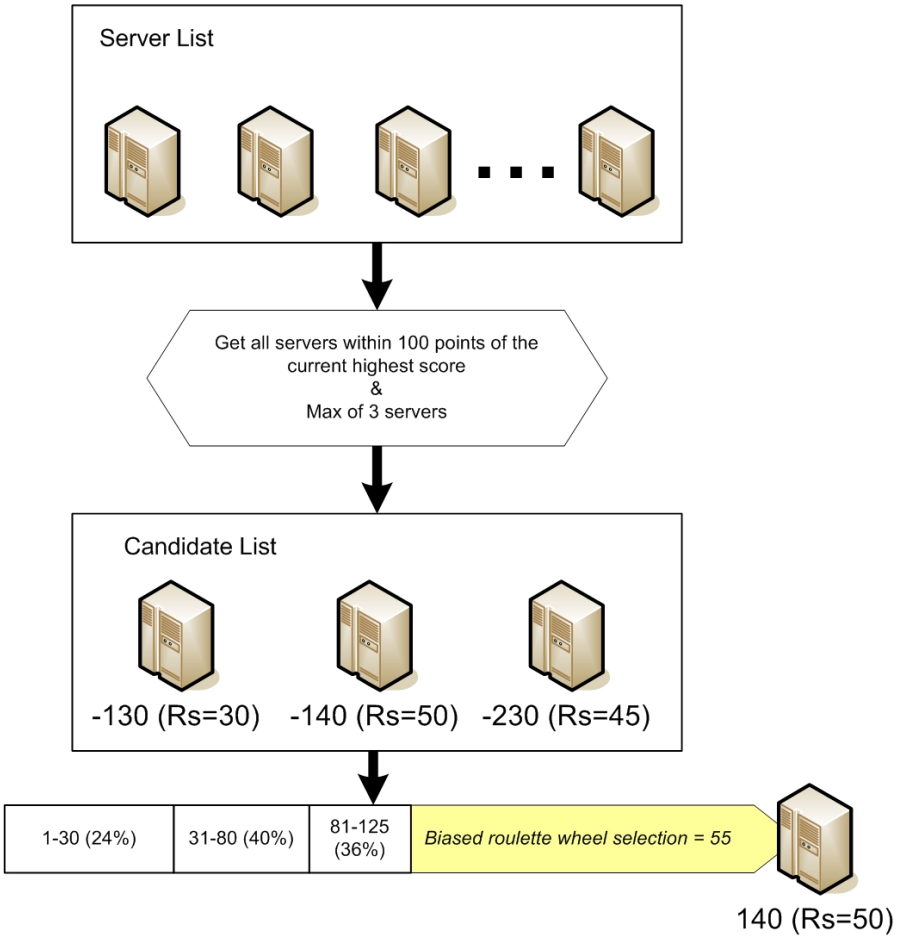


Figure 10: Server selection scenario using the MCD-2R policy. The chances of selecting the leftmost candidate server are 24%, the middle server 40% and the rightmost 36%. The roulette picks 55 and the middle candidate server is selected.

Scoring service reliability:

In order to estimate the relative reliability of a server's service, the error statistics are kept in a *service detail list* which is updated at the end of every session. The total amount of fragments, error fragments, dropped connections and no connections for a session is added to their totals experienced by the local broker. The statistics updated by the broker for all the servers that it has received services from are:

- Total number of received file fragments.
- Total number of received bad file fragments.
- A windowed list of the amount of data in MB transferred before a bad connection (dropped or cannot connect) occurs. This window is limited to the last 50GB transferred.

When the broker policy queries the service details list with a server IP, the statistics are combined into a reliability score *Rs* using the formula derived below. The intention for the *Rs* value was to compare the reliability of service offered by a service provider to another through the roulette wheel method. Therefore the reliability score of a server must have a significant effect on its chances on being selected by the broker. The actual modeling of server reliability is outside the scope of this thesis.

Let:

$Rs \geq 0$ be the raw inadjusted reliability score

$g_{bad} \geq 0$ be the total number of bad fragments received

$g_{total} > 0$ be the total number of fragments received

$E_{total} \geq 0$ be the number of bad connections within the last 50GB of service

$50,000 \geq N_{total} > 0$ be the total amount of data received up until the last bad connection

Have:

$g_{ratio} = 1 - \frac{g_{bad}}{g_{total}}$ be the good fragment ratio

$\lambda = 1 - \frac{E_{total}}{N_{total}}$ be the inverse bad connection intensity

Let:

$w_g \geq 0,\ w_e \geq 0$ be constants where $w_g + w_e = 100$

$w_{sen} = 20$ a sensitivity constant for Rs

And:

$Rs' = w_g g_{ratio} + w_e \lambda$ be the raw reliability score

$$Rs = \max\left(\frac{w_{sen}}{100 - Rs'}, 200\right)$$

The λ value is windowed by the amount of data transferred and is also relative to that amount. The intention was to estimate the reliability of the service during the actual service, using units such as time will include time that wasn't used during the process of a service. Both λ and g_{ratio} are limited to (0,1].

The default values of the weights are $w_g = 30$, $w_e = 70$. The comparatively low w_e was to take into consideration the much higher values that the λ value would return. Compared to λ, the g_{ratio} function will have a higher numerical g_{bad} value and hence produce a lower g_{ratio} value. Furthermore, $w_g + w_e$ must be equal to 100.0, this is so that the raw *Rs*' values never evaluate to more than 100.0. However, *Rs'* values as they are are not effective the roulette wheel selection. This is because the numerical difference between a reliable server (around *Rs'*=99) and an unreliable server (around *Rs'*=90) is not effective enough to compensate for the qualitative difference between the two scores.

Therefore, the *Rs* formula is used to exponentially exaggerate the difference. Furthermore, we limit the maximum value of *Rs* to 200. This is so since at *Rs*=200, *Rs'*=99.9, which is judged to be statistically very reliable. Furthermore, since the *Rs* is exponential as *Rs'*→100 we do not want servers with overly large and possibly outlier values of *Rs* to crowd out the chances of other servers being selected. The effect of the *Rs* formula and different w_{sen} values are discussed in more detail in appendix A.

The next chapter presents a) the modeling and simulation of data grid network and b) the result of performance evaluation of the above proposed policies.

Chapter 6: Simulation and Performance Evaluation

6.1 The MONARC 2 Simulator:

A Java™ based data grid simulator, MONARC 2, was used to simulate the effects of the policies discussed in the previous chapter. This is a data grid simulator based on the computational grid simulator MONARC [12]. The original simulator was designed to provide realistic simulation of large distributed computing systems and allow for the evaluation of different optimization procedures for data intensive distributed environments such as a data grid.

Since the policies described in the thesis required the simulation of complex interactive components that reside conceptually on many different levels – from a grid node to the fragment monitoring level – an efficient and flexible simulation architecture was required. The second version of the MONARC simulator was chosen as it was designed to improve on the original's ability to handle a large number of threads. Furthermore, MONARC 2 has incorporated the general components for a data grid including classes that model the behavior of regional centers and databases.

6.2 Policy, Broker and Data Grid Environment Implementation

MONARC 2 was designed to run and simulate a data grid without modifications to the source code. Java class extensions and configuration scripts are sufficient for extremely low level work up to the level of simulator event messages. In order to realize the policies however, several main components were built from the ground up:

- A general class to model the transfer requests had to be implemented at the message level. This is because the transfers changed states form being originally requested from the user to being analyzed by the broker, then to being a part of the transfer sessions to being suspended by interruptions, etc.
- The broker had to be constructed as the default brokers did not support an economy based resource management grid.
- As a requirement form the previous point, the basic services such as the service directory for a commodity market economy model, the pricing function for service providers, billing, session accounting and payment services were implemented.
- The broker's scheduling requirements, both for scheduling to external service providers and broker internal scheduling of transfer service requests and transfer sessions.
- Centralized simulated data collector used to simulate the broker's support services that monitor and update service quality and transfer sessions.
- Multi-threaded download and upload classes to implement transfer protocols and parts of the transaction protocol.
- Automated logging system that calculated and logged policy specific data involving the quality of service experienced by the brokers and network performance summaries.
- Policy plug-in classes.

Due to the frugality of space and time, more details of the simulation implementation will not be discussed here.

6.3 Data Grid Configuration

The MONARC 2 simulator allows configuration of networks through configuration scripts including LAN and WAN settings, TCP configuration, router configurations, and latency. For simplicity, latency in the grid was disregarded as it would introduce asynchronous acknowledgements for each fragment into the fragment based data transfer; which is work that is beyond the scope of this thesis. Furthermore, the effects of TCP configuration was left as default. The fair networking sharing policy was enabled via a configuration option in the configuration scripts.

The data grid configuration that forms the basis of testing is shown in Figure 11 and the associated per regional center configurations are summarized in Table 2.

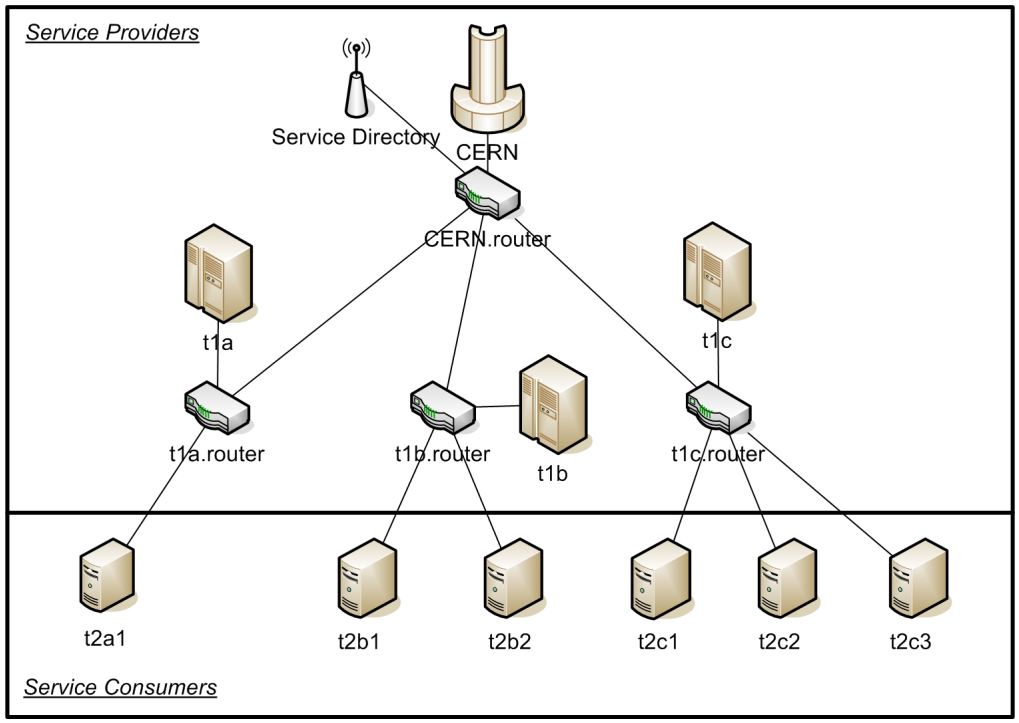


Figure 11: Data grid configuration in the LHC/MONARC structure.

Regional Center Name	***WAN speed (Mbps)***	***LAN speed (Mbps)***	***Has Transfer Service***	***Requires Service***	***Files Demanded***	***Probability of error during service e_t***	***Base cost per MB b_j***
CERN	1000	600	Yes	No	-	0.1%	2
t1a	1000	550	Yes	No	-	2%	1
t1b	1000	500	Yes	No	-	1%	1.5
t1c	1000	450	Yes	No	-	0.5%	1.7
t2a1	1000	300	No	Yes	15	-	-
t2b1	1000	300	No	Yes	15	-	-
t2b2	1000	300	No	Yes	15	-	-
t2c1	1000	300	No	Yes	15	-	-
t2c2	1000	300	No	Yes	15	-	-
t2c3	1000	300	No	Yes	15	-	-

Table 2: Basic data grid node parameter configuration.

The purpose in the basic configuration above was to simulate the scenario where a popular set of LHC experimental data files have been initially been released into the grid. It includes tier 0 (CERN), tier 1 (t1a, t1b, t1c) and tier 2 (t2xx) nodes where tier 0 and 1 nodes typically have higher bandwidth available to them (≥450Mbps). The tier 2 nodes then rush to access or replicate the data files. The demand patterns for the files are randomly distributed amongst 20 files and the service requests are normally distributed to occur between 100 and 300 seconds. The close distribution of the service demands helps simulate busy data traffic in a global data grid as well as to produce a variable global data demand behavior which will be explained later.

The default policy parameters are summarized in Table 3, parameter values that have * denote values that will be varied from one test run to another.

Parameter	*Description*	*Value*	*Comment*
q_1	Service pricing policy class boundaries	0	Small boundaries due to limited amount of consumer nodes.
q_2		2	
q_3		3	
q_4		4	
f	File size ranges	2,000 - 10,000MB	From suggestions by [13]
w_D	MCD-1 constant weight for delay D_{ij}	1*	
w_p	MCD-1 constant weight for cost p_j	0.5*	
I	Interrupt sub-policy is triggered every I seconds for MCD-2	5 seconds	
a	Interrupt candidate session minimum duration for interrupt sub-policy	30 seconds	Session must have transferred for at least 30 seconds
b	Interrupt candidate session maximum duration for interrupt sub-policy	120 seconds	Session transferred over 120 seconds must be a candidate
u	Minimum % utilization of locally allocated bandwidth for a session for interrupt candidate sub-policy	80%	
r	Minimum *Rn* value for interrupt sub-policy	-10	A session with an *Rn* value of 10 implies that the transfer speed has been dropping by about 10Mbps for a significant amount of previous fragments
Δ	The maximum difference between the highest MCD scoring server and the lowest for MCD-2R candidate selection	200*	
x_{cmax}	The maximum amount of candidate servers selected for MCD-2R	3	
w_g	Weight for the good fragment ratio gratio in the *Rs'* function	30	$w_g + w_e$ must = 100
w_e	Weight for the reverse bad connection intensity ratio λ in the *Rs'* function	70	
w_{sen}	Sensitivity constant for the *Rs* function	20	See appendix A

Table 3: Default policy parameters.

Due to the large file sizes and limited bandwidth for every server and WAN that it is connected to, there will exist all three types of transfer bottlenecks as discussed. For example if all the consumers are using services from server t1a, then the maximum bandwidth demanded by all t1a's consumers (1800Mbps) will exceed the 1000Mbps maximum limit on the t1a router. In fact demands from all the other consumers except t2a1 will overwhelm CERN's router first.

By setting the distribution of request occurrences on consumers close so that the next replication request will often occur before the previous file finishes will cause the maximum bandwidth available per transfer to vary over time.

Early in the simulation, transfers will still finish relatively quickly. As time passes, there will be instances of concurrent sessions with more and more transfer on all nodes until the requests run out and a drop in the number of transfers running concurrently on a node occurs.

6.4 Experiments

6.4.1 Pricing Policy Experiment

The interactions of the nodes in the grid are complex especially due to the existence of the variable priced services, file request (binomially distributed via time of request arrival) and file sizes. The first experiment will show the effect of introducing of a variable pricing scheme on to the data grid compared to a static pricing scheme. This experiment will only use the cheapest policy for its brokers.

From Figures 12 and 13 it is obvious that the average global performance of the constant pricing scheme with cheapest cost policy causes consumers to swamp a single provider. This results in a very cheaply priced, but also a very low quality of service. Average transfer speed of the constant price policy was only 22% of that of the average speed of variable pricing policy.

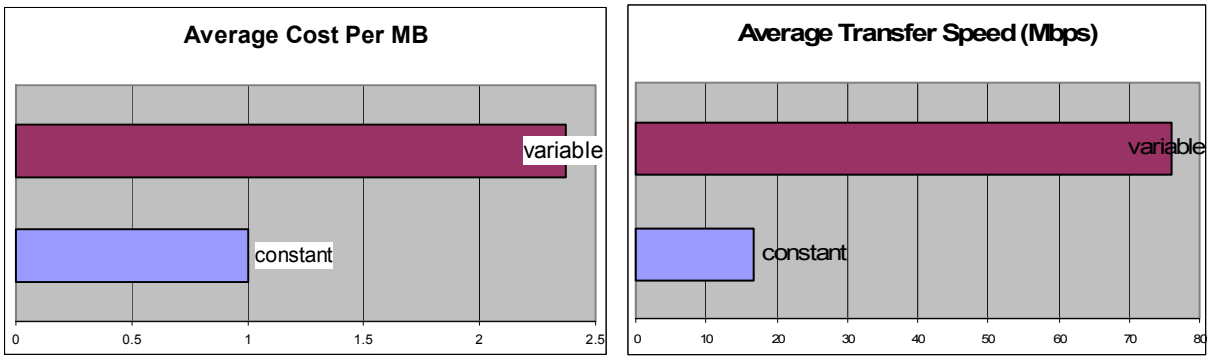


Figure 12 & 13: Average Cost per MB and Average Transfer Session Speed.

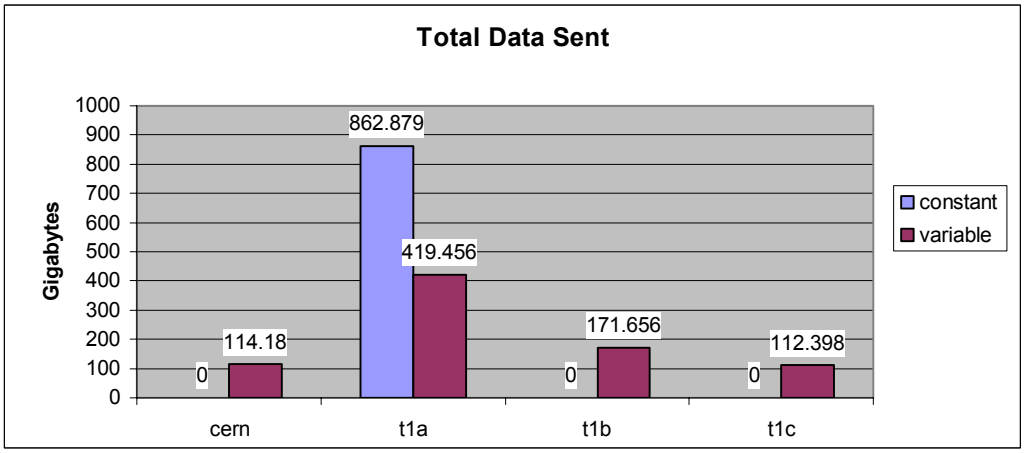


Figure 14: Spread of global transfer service load.

From Figure 14, we see how much data is uploaded by each service provider. In the constant pricing policy all uploaded data originated from t1a. The resource management effects of the price mechanism caused by variable pricing can be seen with a less extreme distribution of data demand on server t1a. However, in line with the cheapest cost policy, the majority of the data upload is still derived from the cheapest provider.

Note that in this comparison of policies, the difference in the average cost per file between the two pricing policies cannot be used to compare the true buying power of a broker in the constant pricing and one in the variable pricing policy. This is because these policies create two different economic models that place different valuations on their

respective currency. That is 1 grid credit in one closed economic model is unlikely to be equivalent to 1 credit in another.

6.4.2 Mixed Policy Experiment

This experiment compares the performances of the 4 policies on an individual node basis in a data grid with variable pricing scheme. Only the performance of the individual nodes are analyzed, not the global grid. Analysis ranges from the quality of the service experienced by the node to the behavior of individual file transfers. Nodes on the grid have different policies and several different node configurations are run to isolate differences in performance between one policy and another.

Four grid simulations use the following policy configurations for the brokers of its service consumers:

Simulation ID	*t2a*	*t2b*	*t2b2*	*t2c1*	*t2c2*	*t23*
Sim-1	Cheapest	MCD-1	MCD-2	MCD-2R	Cheapest	MCD-1
Sim-2	MCD-1	MCD-2	MCD-2R	Cheapest	MCD-1	MCD-2
Sim-3	MCD-2	MCD-2R	Cheapest	MCD-1	MCD-2	MCD-2R
Sim-4	MCD-2R	Cheapest	MCD-1	MCD-2	MCD-2R	Cheapest

Table 5: Mixed policy experiment configuration. For example the broker of regional center t2a uses a cheapest cost policy.

All policies are installed and simulated once on each consumer node. This to factor in the performance effects that the relative network position of the node in a data grid may have, if any.

Characteristics of sample transfer sessions from this experiment of the MCD-1 and MCD-2 algorithms will first be presented below to validate the difference during transfers with and without transfer service migrations.

MCD-1:

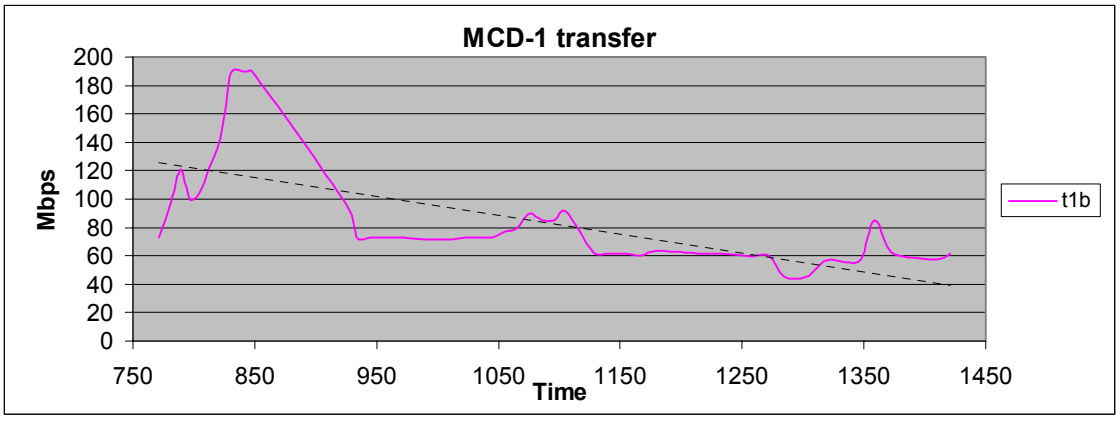


Figure 15: Chart for a typical MCD-1 transfer session with linear trend lines.

This particular transfer service was taken from early in the simulation where average transfer speeds are expected to decrease (average simulations ran for approximately 100+ simulated minutes). The transfer began with relatively high speeds then drops and maintains a low speed for the duration of the transfer. The speed reduction maybe due to various factors such as an increase of transfer requests, a reduction in service quality at the server end or the increase in traffic on the link. From the trend line, the gradient of the decrease in transfer speed over time is more distinct.

MCD-2:

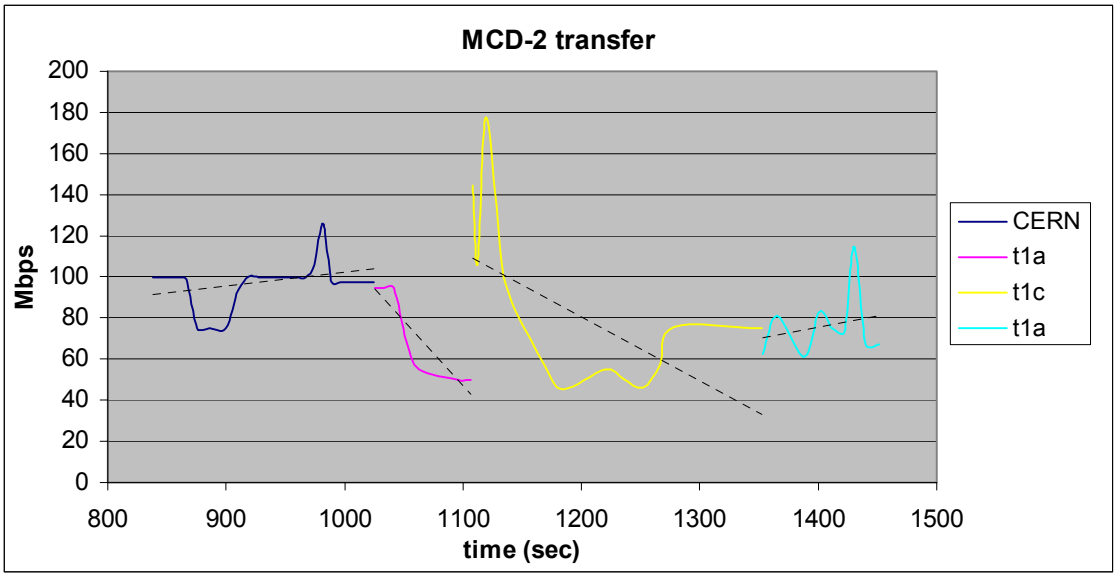


Figure 16: Chart for a typical MCD-2 transfer session with linear trend lines.

This transfer is typical of a broker transfer using the MCD-2 policy. This transfer is similarly taken from an early segment of the simulation. The entire transfer takes place under 4 distinct transfer sessions starting at the central CERN server from 837th second to the 1025th second. The reason for migration attempt at the 1025th may most likely be due to finding a better server in t1a, recall that CERN is the most expensively priced server and t1a is the cheapest server. However over the second session at t1a, the speed of transfer is drastically dropping over 1025th second to the 1107th second. The migration to t1c occurs at the 82nd second of the second session and migration occurs. The quick migration will be due to the transfer session being an early interrupt candidate due to constant decreases in transfer speeds and being able to discover a better server. And indeed, after the migration to t1c the speed gain is clear, however for the third session the transfer speed also drops drastically. Different to the second session, the broker decides to stay with t1c even though speeds have dropped as much as that in t1a. This may be due to the overall decrease in grid traffic and increased local transfer demands where the session may still be using > 80% of its allocated local bandwidth. Eventually, the broker migrates back to the cheapest server t1a to complete the transfer.

Also note that while average speed should not be compared from just 2 sample transfers, the transfer speeds for the MCD-2 are characteristically more erratic than those in MCD-1.

We will now compare the average policy performances of brokers in mixed policy grid environments. Recall that the results are averaged over the 4 simulation runs where each network position will use every policy over the runs.

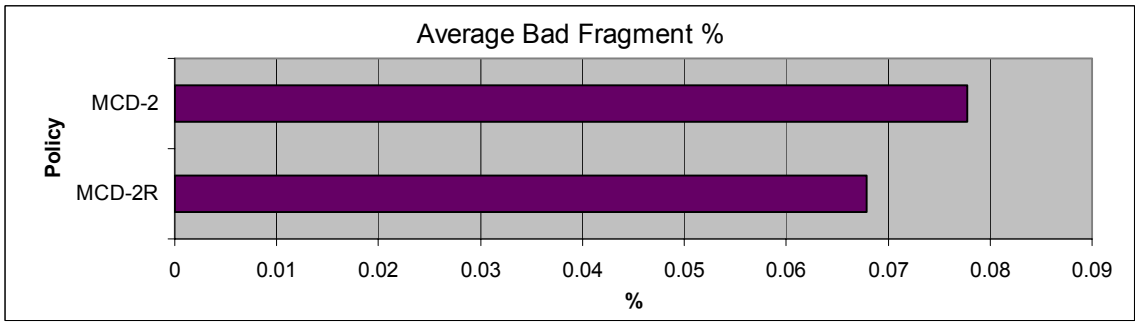


Figure 17: Graph for the average percentage of bad fragments out of all the fragments received.

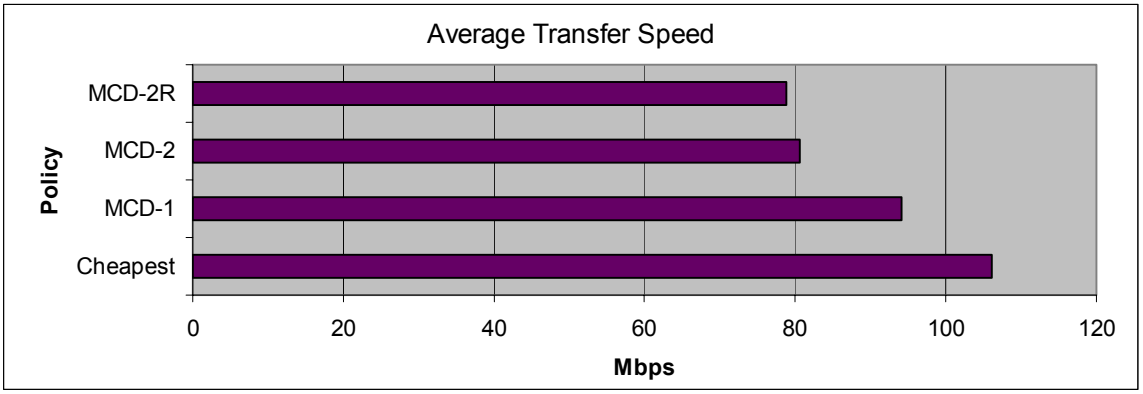


Figure 18: Graph for the average transfer speed.

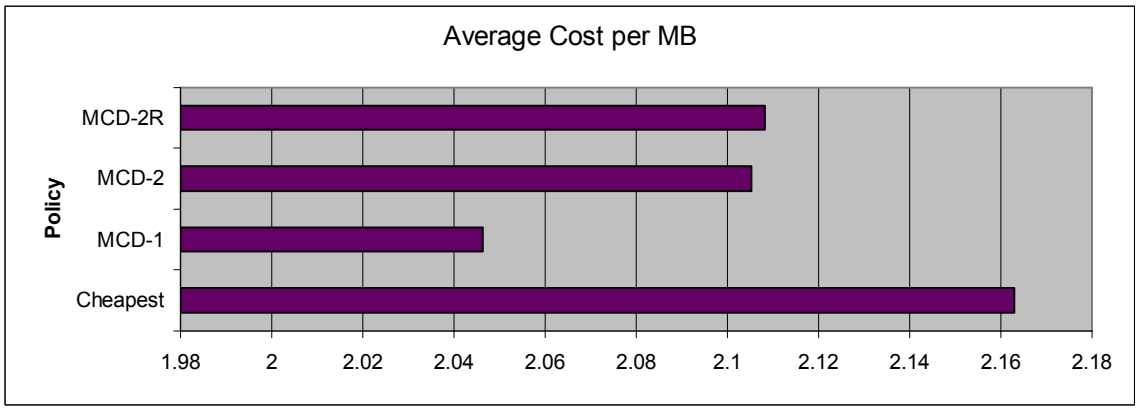


Figure 19: Graph for the average cost per MB.

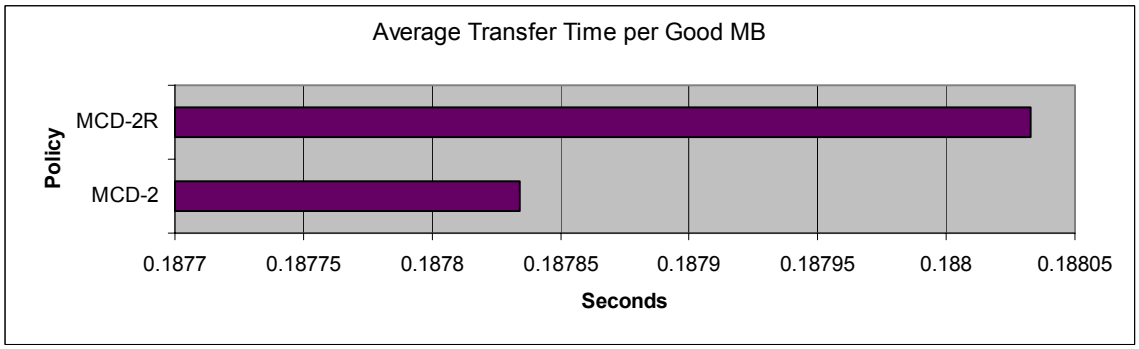


Figure 20: Graph for the average transfer time per good MB.

The results for this experiment were mixed. Figure 17 indicates an expected decrease in the average number of bad fragments received by the nodes that used the MCD-2R compared to the nodes that used MCD-2. Service reliability of the MCD-1 and cheapest policies are not compared due to the effects that the configuration of the servers would have (Figure B.1 in appendix B). Reliability comparisons should only be used between the MCD-2 and MCD-2R to show the improvement of MCD-2 to MCD-2R.

In terms of the average percentage of fragments received and the average number of bad connections (Figure B.3 in appendix B) MCD-2R chose to utilize more reliable servers. The difference in bad fragment percentages is small, and this can be attributed to two reasons a) the small values of errors set for the servers to begin with (0.5% - 1%) and b) the long term stochastic behavior of the MCD-2R which is not as prominent in the short term. However, even in a data grid scenario that transfers only 90 LHC files, the 0.01% difference can amount to about 90MB of extra traffic.

The average time required to transfer a good MB is a per transfer measurement of:

$$\mu' = \frac{totalTimeSpentTransferring}{totalFileSize \; - \; totalBadFragmentData}$$

This μ' is averaged over all transfers for an individual node, and then averaged over all nodes in the 4 simulations that used a specific policy. The transfer time to transfer a good MB is a performance measurement that factors in the delay time experienced from bad connections and bad fragments. In Figure 20 there is a 0.0002 second difference between transferring a good MB under the MCD-2 and under the MCD-2R policy. This difference can translate to a 0.4-2 second delay for per LHC experimental data file. If we refer to Figure 18 and 19, we can also confirm that the average transfer speed is slowest and most expensive for the MCD-2R. Due to these two factors we can conclude that while the MCD-2R does receive services of a higher reliability, it does so at the cost of lower transfer speeds and higher costs.

The cheapest algorithm used more credits per MB on average than any of the other policies. This is due to the tendency for cheapest policy servers to respond slowly to pricing changes in the servers, compared with other policies that factor in speed of transfer and have migration. What is perhaps more surprising is the higher speeds and lower transfer costs achieved by the MCD-1 (Figure 18, 19) compared to the policies that used migration. This is due to several subtle factors however. The weaknesses of the non-migration polices where they tended to stay on a server even if the service degrades. They also implicitly choose to stay with the cost that was confirmed at the beginning of transfer. For policies with migration they tend to move around more often and hence acquire servers which, although will have a better server score, might also have either a worse transfer speed or a higher cost of service. The relative quality of service for non-migration policies is also improved when used in conjunction with migration policies as the swarming effect is reduced and brokers using the MCD-2 and MCD-2R policy move to different servers and there-by improving the transfer speed of the server it just left. It can be said that the migration policies

6.4.3 Dedicated Policy Experiment

This experiment compares the global performance of grids with nodes that use a policy exclusively. For example, a grid that exclusively utilizes the MCD-1 policy will be compared to a grid that only utilizes the MCD-2 policy, etc.

Four more simulations will take place, once for each type of policy:

Simulation ID	***t2a***	***t2b***	***t2b2***	***t2c1***	***t2c2***	***t23***
Sim-1	Cheapest	Cheapest	Cheapest	Cheapest	Cheapest	Cheapest
Sim-2	MCD-1	MCD-1	MCD-1	MCD-1	MCD-1	MCD-1
Sim-3	MCD-2	MCD-2	MCD-2	MCD-2	MCD-2	MCD-2
Sim-4	MCD-2R	MCD-2R	MCD-2R	MCD-2R	MCD-2R	MCD-2R

Table 6: Dedicated policy experiment C configuration.

The performances of data grids are compared below:

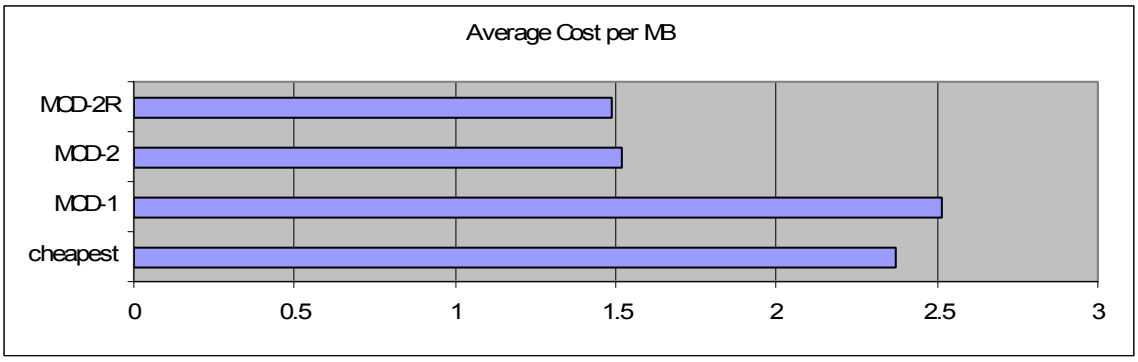


Figure 21: Graph for the average cost per MB.

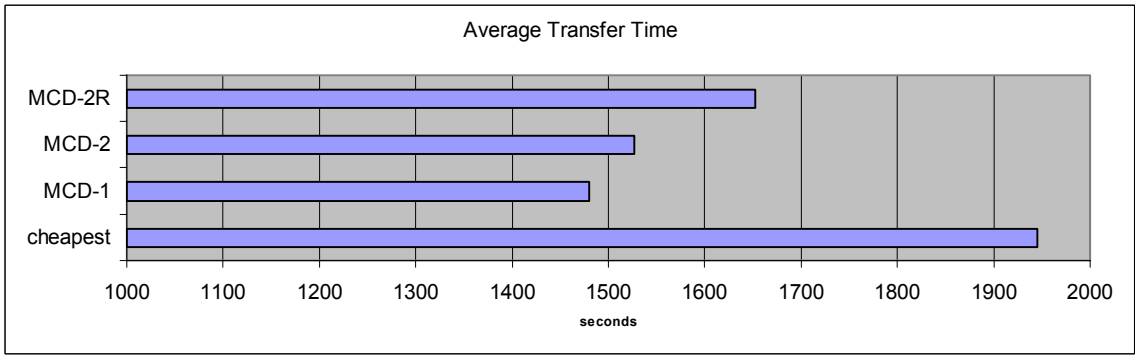


Figure 22: Graph for the average transfer time.

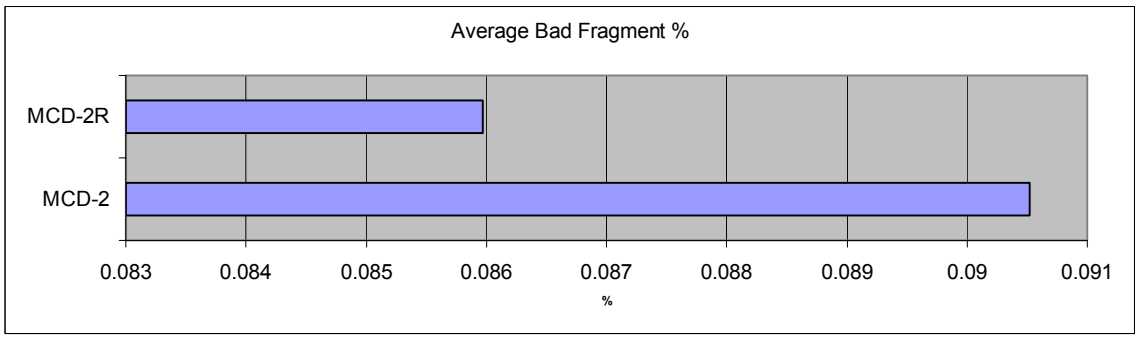


Figure 23: Graph for the average percentage of bad fragments.

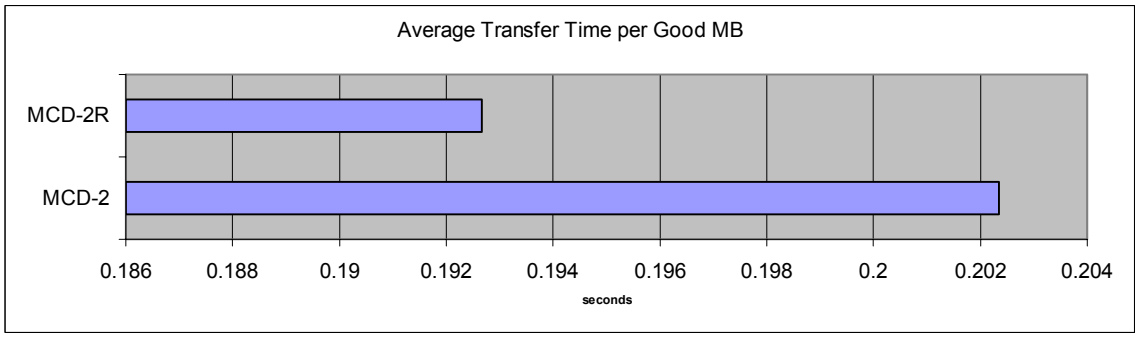


Figure 24: Graph for the average transfer time per good MB.

With grids that exclusively use a particular broker policy, one can see the beneficial effects of service migration. In Figure 21, the migration policies use significantly less credits on average than even the "cheapest" policy (by about 37%). As well both the MCD-2 and the MCD-2R policies use less transfer time than cheapest policy (Figure 22). The MCD-1 policy uses marginally more credits per MB than cheapest policy while using some 465 seconds less per transfer. Compared to MCD-1, the other MCD variants using slightly more transfer times, while some of the extra time used can be attributed to migration overheads, the Monte Carlo selection technique of the MCD-2R prefers more reliable servers over non-reliable ones, and can hence risk selecting slower servers.

The effect of the reliability selection is however, still not obvious (Figure 23). The average percentage of bad fragments received was 0.005% less in MCD-2R than in MCD-2, so although there is agreement between the mixed policy and dedicated policy experiments that reliability is increased, the level of the effect is slight. Similarly, if we compare the average transfer time per good MB, we see an approximate 0.01 sec faster speed for MCD-2R in receiving a good MB, while this is the reverse in the mixed policy experiment, the margin is still a small one.

6.4.4 Parameter Experiment

This experiment explores the global effects of varying the MCD-1 weights with the MCD-2R policy installed on all nodes. Table 7 details the various parameter configurations for each simulation type.

Simulation ID	w_D	w_p
Sim-1 (default)	1.0	0.5
Sim-2	1.5	0.5
Sim-3	0.5	0.5
Sim-4	1.0	0.25

Table 7: Parameter experiment configuration.

The results of the experiment are presented below:

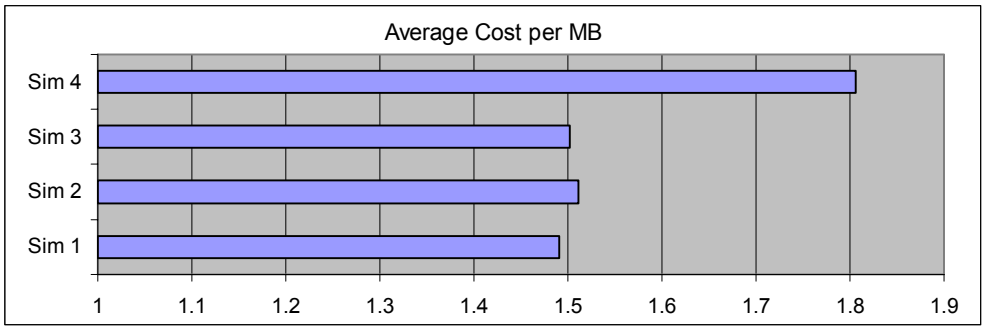


Figure 25: Graph for the average cost per MB.

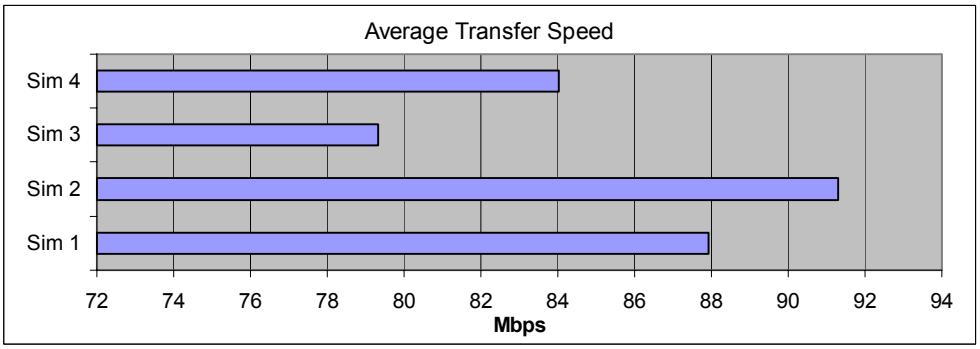


Figure 26: Graph for the average transfer speed.

By increasing the weight w_D for delay in Sim-2 we see that more significance in the transfer speed of a server has been placed (Figure 26). With a 0.5 increase in w_D value there is a 4% increase in speed with a small 0.02 increase in cost per MB (Figure 25). Conversely, in Sim-3, a 0.5 decrease in the delay weight led to a large decrease in transfer speed from 88.92Mbps to 79.32Mbps, a 10.8% drop in speed. For Sim-3 we also see a small increase in cost (0.8%) even though relative to the delay weight, its numerical contribution to the score function has been increased.

When one decreases the w_p by 0.25 in Sim-4 there was a decreased significance placed on servers with lower service costs, the average cost per MB dropped by 0.32 which is significant when the average amount of data received by a consumer in this simulation was over 95GB (saving of over 30,000 credits on average). The drop was offset by a small decrease in transfer speeds from 88.92Mbps to 84.02Mbps, a 4.4% drop.

6.5 Summary

The results show significant evidence that the MCD-2 and its MCD-2R variant are able to improve the global quality of service received by a broker compared to a cheapest cost policy or a non-migration policy such as MCD-1. This improvement in quality translates to a direct reduction in data replication overheads. However, the statement is limited to a grid where all brokers use the same policy. In a mixed policy grid, there was a tendency for non-migration transfer policy brokers to force migration brokers to other servers, thereby giving the non-migration brokers the advantages of staying on the "best" servers. Furthermore, the migration policies caused increased network traffic fluctuations, however its effects on service quality globally has not been analyzed, increasing the constraints on interruption candidate selection may ameliorate the worst of the traffic spikes as witnessed in Figure 16.

The weights of the server scoring function significantly affect the behavior of the broker. By increasing the delay weight w_D one is able to trade increased transfer speeds for increased service cost and vice versa. Similarly with the cost weight w_p, one is able to increase the importance of a cheap service provider and implicitly decrease average replication costs by trading it for lower transfer speeds. There is an unidentified relationship in the exact numerical effects of varying the weights, this is not easily unidentified and if identified is not generalisable to other simulations, or even a neighboring node in the same simulation. This is due to the fact that in order for the relationship of transfer speed and service cost to be established, one needs to quantify and price a service consumer's individual valuation of transfers speeds in terms of saved credits. While this value may exist in simulation, it may not have applications in an actual economy data grid.

There is also strong support for the distributed resource management abilities of an economic resource scheduler in a data grid using the commodity market model with a variable pricing policy. The existence of an intelligent pricing policy in the data grid is crucial in its price mechanism effect.

The difference in terms of quality of service made by the MCD-2R algorithm compared to the MCD-2 algorithm is not as apparent in context of the simulation scenarios explored in the four experiments. While evidence shows a reduction in the average amount of errors received per node, the long term savings on a data grid where there is a trade-off between transfer speed, service cost and reliability are not clear.

6.6 Limitations

The experiments were limited in its ability to truly test the long term effects the proposed policies. This is mainly due to the computational complexity of the simulation objects during simulation. Furthermore a long term simulation should incorporate traces of real live data grid network activity from LHC as this would allow for a more accurate modeling of the effects of server reliability.

Another limitation for these experiments is that the simulation is unable to explicitly pinpoint the exact reasons why certain micro-events occur – such as the decisions for a broker to migrate to another service provider. This is due to the great amount of competing network and local factors that influence the traffic characteristics perceived by the broker. However, accurate inferences can and were made through network conditions, available instrumentation and simulation traces.

Chapter 7: Conclusion and Future Work

7.1 Conclusion

This thesis explored the effectiveness of economy-based resource management in data grids and proposed polices for a data replication broker that improved replication.

The economic data resource management system were shown to perform with marked improvements when using a variable pricing scheme that is based on the expected actions of a rational agent compared to a fixed pricing scheme. This finding can be generalized to all services on a data grid that use a common currency for transactions in a commodity market.

The specific focus of this work was to improve the replication service of the data grid however, the broker policies also has applications for standard large volume data transfers in data grids. The policies displayed an effective means of improving the performance of the grid network traffic and are indicated by the improvement of speed and cost of transfers by brokers. The policies allows flexibility in that users may specify the degree in which to trade between transfer time optimization versus the minimizing the cost of the transfer service. This flexibility will allow low priority applications to operate in cost saving modes while high priority applications can achieve close to maximal transfer speeds relative to network conditions. In terms of replication, this means that users can decide the high level long term strategy in replication transfers.

The interrupt and migration sub-policies proposed in MCD-2 ad MCD-2R take advantage of multiple replica/data sources that would exist on data grids. They allowed brokers to minimize the risk of using services that do not guarantee service quality over time. While results showed that there were increased network traffic oscillations, the sub-policy behaved in a rational manner so that service interruptions are not frequent and resulted in significantly reduced average transfer cost and time in the data grid compared to data grids that did not use brokers with migration policies. The migration of services can be applicable to other types of services such as computation, as long as the service itself can be decomposed. Migration is most effective when applied to services where the quality or standard of service vary over the duration of the service.

The performance gain and service quality gain in the long run from the MCD-2R policy was not as obvious as the migration and minimization heuristics of the MCD-2 policy. An attempt was made to account for the long term effects of server/service reliability and from the initial results, there does appear to be gains in accounting for service errors which should become significant over the lifetime of the data grid.

The proposed policies for data grid brokers can form effective extensions to existing replication schemes and allow for those schemes to take into account the bandwidth used while transferring large sized replicas across the grid. By accounting for the quality of the service received – not just the cost of the transfer but the speed and reliability of the service – the brokers are able to choose fast, cheap and reliable services in the future and hence reduce the overhead of replication schemes.

7.2 Future Work

There is much scope and depth on improving the replication service for the data grid. The sub-problems involved in optimizing data replication by the way of a market economy approach are first and foremost complicated by the lack of a truer and more complete testing solution. Whether by using existing grid test-beds or by utilizing a simulator, the most challenging aspect is to capture the expected complexity and magnitude of a data grid. Numerous interacting components and entities will affect the performance of a distributed resource management scheme to varying degrees. Current simulations still require a large initial effort in developing required data grid components and current test beds are extremely limited in size and capacity compared to the scale of the LHC. In terms of simulations, future works that plan to use data grid simulators should consider:

1. Extending the duration of simulations over periods of weeks and months to observe the long term trends of network demand and supply cycles. From economic theory, it is well known that macroeconomic performance such as GDPs are cyclical. Similar emergent behaviors should exist in an economy-based data grid. Other factors such as the accrued effects of server failures and connection errors over time on consumer demand and service quality can also be more thoroughly determined.
2. With extended simulation durations, network traces and live data demand patterns can be used to further aid in determining the long term effects of server reliability. The MCD-2R algorithm may show a greater effectiveness as the stochastic algorithm stabilizes.

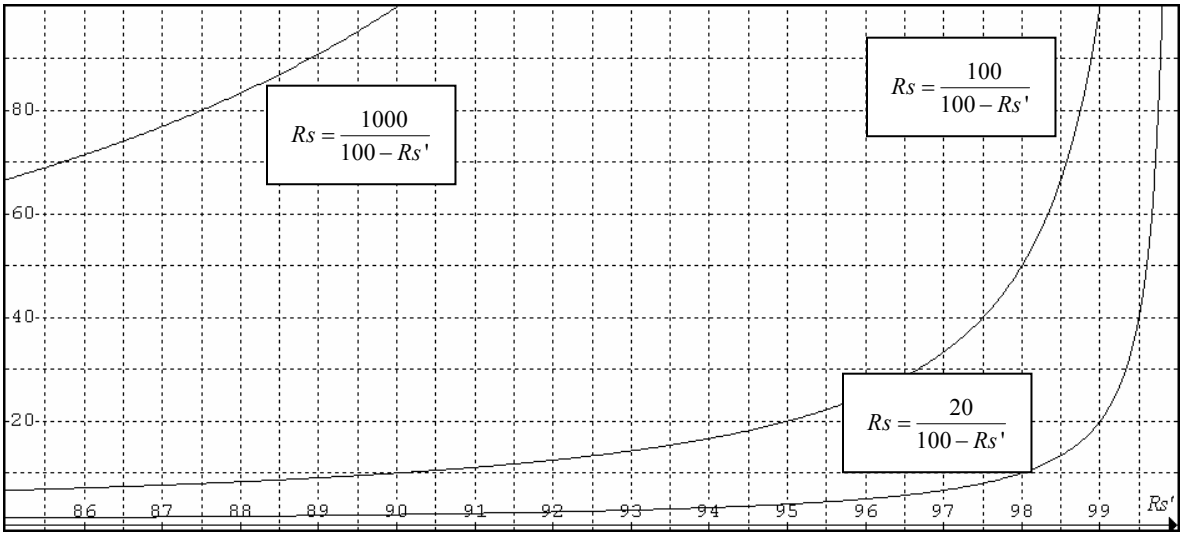
There is also much work in determining the roles of the economy resource management in grid environments. It has been shown that even a simple variable pricing policy can have a great effect on the ability for a resource management scheme to optimize network traffic. It is expected that fundamental economic theory will play an important role in improving the economy-based resource management schemes. Studies of the following nature should be considered:

3. The analysis of network resource usage between an open and closed economic system. That is, what will be the effects of having a grid economic system tied to the economy of the real world versus a close system where only grid credits, which are not refundable for monies, are circulated? In open systems, the issues of wealth distribution may have undesirable effects on the distribution of grid resources.
4. A better pricing policy for the servers should also be explored. Pricing is a complex issue in the real world; it has to deal with both quantitative and qualitative analysis of items for sale and includes understanding of risk, investment and long term versus short term or seasonal planning. An interesting avenue maybe found from employing game playing algorithms for strategic pricing.
5. A related avenue for research would be to employ different pricing schemes for different classes of users in order to entice desirable customers. For example, customers from oversea maybe charged for a greater cost than for local customers. One can artificially improve the resource management scheme to take advantage desirable attributes such as geographical locality.

Lastly, the policies proposed in this thesis demand further investigation to improve data replication and transfer services in data grids.

6. One major direction is to determine how to queue transfer demands as they arrive from a user or some software agent, internally at the broker. The transfers maybe ordered by time or by score. This queue should of course take into account user/application desired quality of service such as deadline, budget and priority. A dynamic priority queue [31] such as one employed by eMule [32] for client queuing was originally intended for this thesis but was excluded due to time limitations.
7. The network model was simplified by using a fair share policy. In future works, simulations of policies should explore networks that allow for dedicated classes of network usage. For example one service may use 50% of the available bandwidth while other services use the remaining 50% according to some policy. With dedicated bandwidth/transfer speed, one can better predict the time of transfers and quality of service. With more accurate prediction, better transfer queuing mechanisms (see point above) can be developed.
8. Another direction is to refine migration tactics by the brokers. From simulation, it was apparent that in a mixed broker policy grid environment, it was sometimes beneficial for consumers to "camp" on a good server. Migration based policies would "give way" to non-migration based policies. In order to improve the performance of migration based policies in a competitive environment, a more intelligent interrupt sub-policy is required.
9. The parameters w_D and w_p used in the MCD policies to define the cost/transfer time function are given relative values. That is in order to increase emphasis on faster transfer speeds; a comparatively higher w_D value is given. In the future an explicit weight value may be mapped to some specific value for the gain in speed or the time of transfer. Or if a relationship between time and cost can not be determined, then a ratio between the two values may be established to simplify parameter value choices.
10. A final completion up is to merge with existing replication schemes such as ones described in Section 4.1.1 in order to determine which files to replicate since the where and when to transfer the replica can be established by the data replication broker.

Appendix A: The Formula



This graph above show the graphs of the *Rs* function with 3 w_{sen} values. As *Rs'* increases, the *Rs* values increases exponentially and as *Rs'* decreases, the *Rs* value drops off quickly and then tapers off at around *Rs'* = 97.

Furthermore as we raise the w_{sen} value, *Rs'* values close to 100, say at 99.9 with w_{sen} = 1000, *Rs* = 10^4. As we decrease w_{sen}, values close to 100 of *Rs* decrease dramatically. At w_{sen} = 20, *Rs'* = 99.9, *Rs* = 200.

Appendix B: Auxiliary Charts and Graphs of Results

B.1 Mixed Policy Experiment:

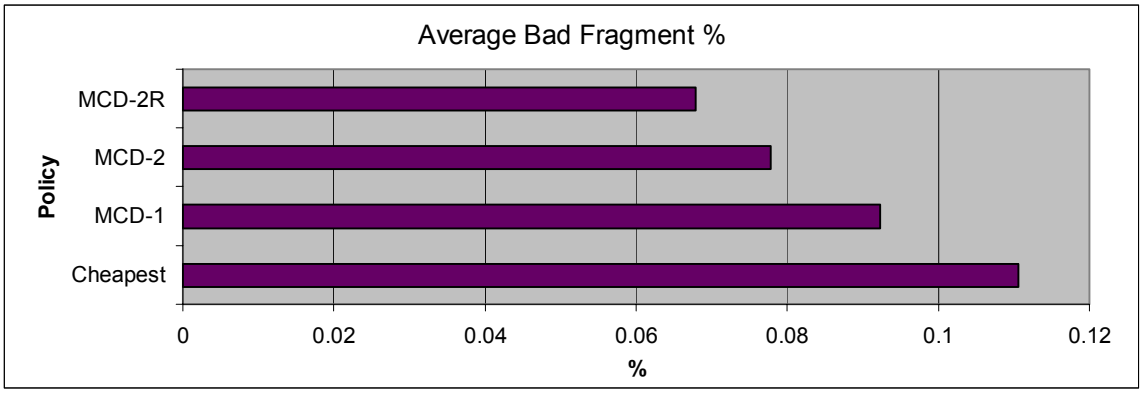


Figure B.1: Graph for the average percentage of bad fragments out of all the fragments received.

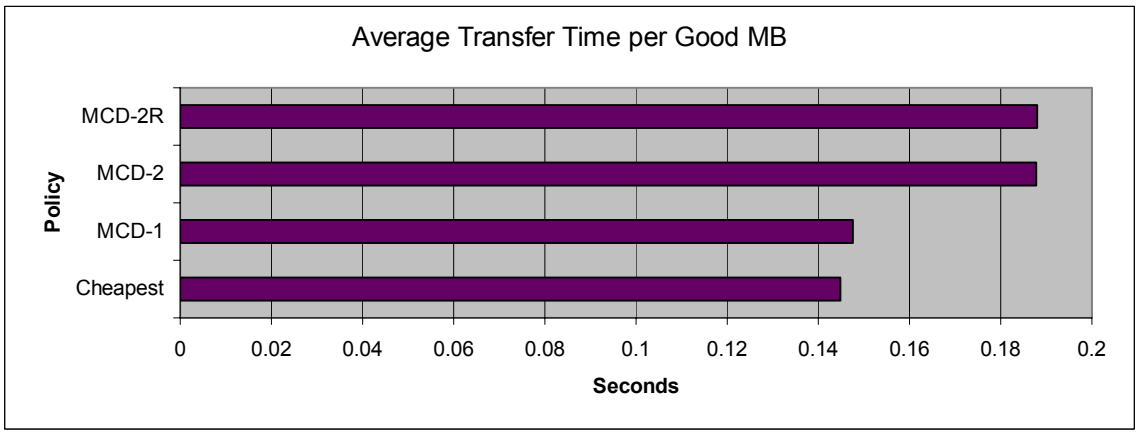


Figure B.2: Graph for the average transfer time per good MB.

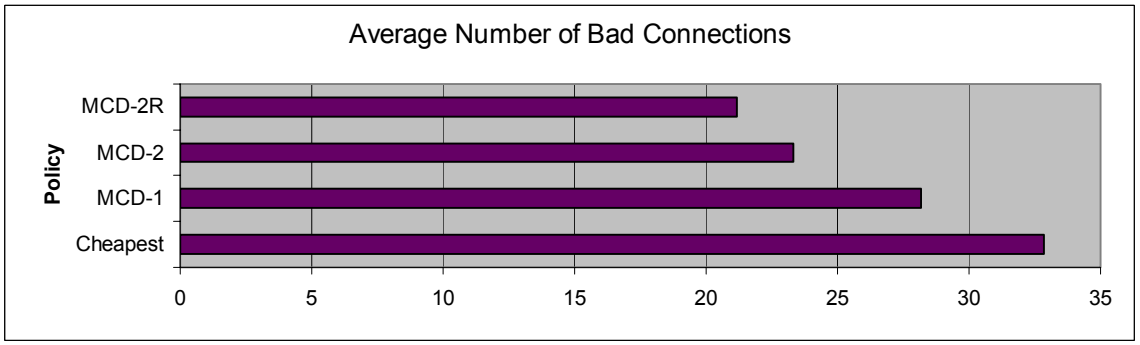


Figure B.3: Graph for the average number of bad connections per node over a simulation run.

B.2 Dedicated Policy Experiment:

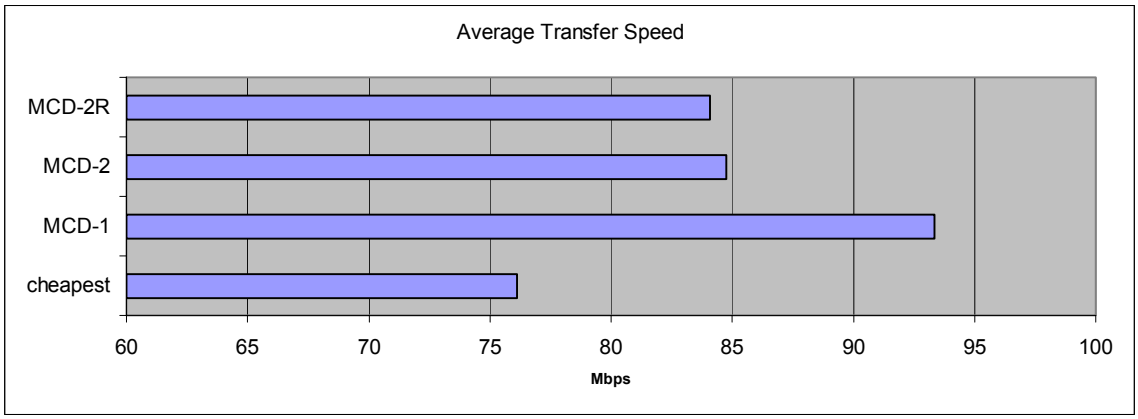


Figure B.4: Graph for the average transfer speed.

Appendix C: Economic Models

C.1 Commodity Market Model

The commodity market model is also called the *supply and demand driven pricing model*. In this model the service providers advertise the price of a file transfer service according to the amount they use. The pricing scheme of the provider may be flat or variable. A flat pricing scheme would affix the price of a service once the valuation of the service has stabilized. Generally, a price is considered stabilized if it does not change for a period of time. The price is constant irrespective of the quality of the service the provider may deliver. In a variable pricing scheme, the pricing will change in response to a rise or fall in service demand – the provider will shift the price up if there is an increase in demand and vice-versa [20]. The shifting will result in prices that iterate toward an equilibrium value at that the given time. Other pricing schemes can be subscription based and usage duration based.

Prices can be published to a market directory service in the grid. This service may be centralized or decentralized. In a variable pricing scheme, the service provider will evaluate the price on offer traditionally in accordance to production cost and desired profit margin [15]. In a grid environment we may use parameters to create a costing function:

Service Cost = func(initial investment, fixed costs, current demand, future demand, consumer perceived value, preferences)

Values for parameters such as future demand and consumer perceived value can be difficult to extract from the system unless consumers offer clues. Fixed costs are derived from maintenance and utilities paid (e.g. electricity). Note that costing is a complex issue – many financial theories attempt to provide solutions to costing, however because the value of a service is largely determined by qualitative factors and consumer perception, they can be difficult to translate into quantitative values.

A resource broker may carry out the following steps for replicating a selected file:

1. Identify candidate service providers from service directory.
2. Identify available file transfer services and evaluates their costs.
3. Selects the service providers that meet broker policy objectives.
4. Uses the services and pays for the service through a regulator or fiscal institution such as a bank.

This model is simple and compared to other models require less communication per transaction, while it does not offer room for price negotiation if consumers are able to continuously reassess received service of quality from a service provider and its associated service cost then the pricing mechanism will motivate service providers to revalue their prices over the long term[8].

C.2 Auction Model

An auctions use market forces to negotiate a clearing price for a service. This model is a method for one-to-many negotiation. The auctioneer sets the conventions of the auction as specified by the service provider and the many consumers bid for the file transfer service until a single value and a single consumer for the service is agreed upon by all participating parties.

In the real world the auction process is used to sell a good or service within a deadline, the most obvious example of an auction system is eBay where the site serves as the mediators/auctioneers for the auction of goods. In the grid, both the seller and the buyer may be automated via a policy. Auctions can be open – where the value of the bids are made public to all participant, or closed – where the value of the bids are hidden to all except the bidder and the auctioneer. In an open system, repeated bids and counter-bid are allowed. Depending on the bidding process, auctions can be classified into five types:

- English auction (first-price open cry)

[8] The phrase "long term" is from a data grid perspective which may be significantly different to its sense in econometric terms where the long term may span years to decades. Due to fast information spread, grid economic trends should mimic but occur much faster than its real world analogy.

- First-price sealed-bid auction
- Vickery (second-price sealed-bid) auction
- Dutch auction
- Double auction (continuous)

The general protocol in the auction process is:

1. A service provider publicizes their services to attract bids.
2. Brokers offer bids.
3. Bidding continues until no broker will offer a higher price or if a minimum price specified by the provider is not reached.
4. The service provider offers the service to the winner of the bid, if any.
5. The broker uses the resource.
6. Bill the consumer.

The key differences, advantages and disadvantages of using auctioning, as a way of allocating resources, lies in the specific type of auction. In Table C.1 below, I summarize the key characteristics between the various types of auctions.

Type	***Open/Closed***	***Bids Updatable by Brokers***	***Price Updatable by Auctioneers***	***Bidding Process***	***Bidding Termination Rule***
English	Open	Yes	No	Brokers keep bidding higher prices.	Bidding deadline is reached or no more bids. The highest bidder wins.
First-price sealed-bid	Closed	No	No	Each bidder bids once.	Highest bidder wins.
Vickery	Closed	No	No	Each bidder bids once.	Highest bidder wins at the price of the second highest bidder.
Dutch	Closed	No	Yes	The price is lowered gradually by the auctioneer. Bidders only have the chance of bidding once.	First bid wins.
Double	Open	Yes	Yes	A continual process of matching what is offered by the service providers and the bids of the consumers. Analogous to share trading.	Auctioneer matches specific bid with a price offered by a provider.

Table C.1: Characteristics of various auction models.

One of the key disadvantages is that auction session need to contain and limit the amount of bidders at one time. With a high number of servers in a large grid, complete coverage of auction calls to all potential bidders will not be feasible or scalable. Communication overheads for auction types such as English and Dutch can be exacerbated by the successive broadcast of current price/bid value. However if one limits the coverage of an auction, certain service providers may be starved from potential consumers.

C.3 Tender/Contract-net Model

One of the most widely used models for service negotiation in a distributed problem solving environment, the contract model is a many-to-one negotiation. However, unlike the auction model the situation is reversed where there are many service providers negotiating for the patronage of one customer. The general procedure for a transaction is as follows:

1. The broker announces its requirements for a file replication and invites bids from service providers that have the file.
2. Service providers receive tender announcements from the service directory.
3. Interested service providers respond by submitting their bids to the associated broker.
4. The broker evaluates and awards its patronage to the most appropriate service provider(s).
5. Accepted providers deliver the file.
6. Bill the consumer.

The service consumer may illicit the service of many providers in the case that a service provider cannot fulfill its contract and/or the services of a single provider is not enough.

This model has the disadvantages of taking the risk of awarding potentially less capable service providers if more capable service providers are busy. There is also a problem with establishing enough service provider bidders – a call for bidding process may receive no bids due to system wide service congestion. This can be nullified by implementing quick response messages to bidding with that server's current status, this allows brokers to re-evaluate its own requirements and set constraints on its budget and deadline for the file transfer.

References

[1] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury and S. Tuecke, *The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets*, http://www.globus.org/, 1999. 132.

[2] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, *Economic Models for Resource Management and Scheduling in Grid Computing*, The Journal of Concurrency and Computation: Practice and Experience (CCPE), Wiley Press, May 2002.

[3] Belle Collaboration, http://belle.kek.jp/

[4] ATLAS at the University of Chicago, http://hep.uchicago.edu/atlas/

[5] Large Hadron Collider (LHC) at CERN, http://lhc-new-homepage.web.cern.ch/lhc-new-homepage/

[6] K. Ranganathan and I. Foster, *Decoupling computation and data scheduling in distributed data-intensive applications*, In Proceedings of the Eleventh IEEE Symposium on High Performance Distributed Computing (HPDC), Edinburgh, Scotland, July 2002.

[7] K. Krauter, R. Buyya, and M. Maheswaran, *A taxonomy and survey of grid resource management systems for distributed computing.* Software Practice and Experience, 32(2):135-164, February 2002.

[8] W. Hoschek, J. Jaen-Martinez, A. Samar, H. Stockinger, K. Stockinger, *Data Management in an International Grid Project*, 2000 Intl. Workshop on Grid Computing (GRID 2000), Bangalore, India, December 2000.

[9] I. Foster, C. Kesselman (eds.). *The Grid: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann, 1999.

[10] K. Holtman, *Object Level Replication for Physics*, Proceedings of 4th Annual Globus Retreat, Pittsburgh, July 2000.

[11] I. Foster, C. Kesselman, S. Tuecke, *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*, IJSA 2001.

[12] MONARC Collaboration, *Models of Networked Analysis at Regional Centres for LHC experiments: Phase 2 report*, Technical Report CERN/LCB-001, 2000.

[13] Allcock, W., Bester, J., Bresnahan, J., Chervenak, A., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D. and Tuecke, S. *Data Management and Transfer in High-Performance Computational Grid Environments*. Parallel Computing. 2001.

[14] R. Buyya, J. Giddy, and D. Abramson, *A Case for Economy Grid Architecture for Service-Oriented Grid Computing*, 10th IEEE International Heterogeneous Computing Workshop (HCW 2001), In conjunction with IPDPS 2001, San Francisco, California, USA, April 2001.

[15] J. Sloman, K. Norris, *Macroeconomics*, Prentice Hall 1999.

[16] M. Miller and K. Drexler, *Markets and Computation: Agoric Open Systems, The Ecology of Computation*, B. Huberman (editor), Elsevier Science Publishers, The Netherlands, 1998.

[17] Abramson, D., Giddy, J., and Kotler, L., *High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?*, International Parallel and Distributed Processing Symposium (IPDPS 2000), Cancun, Mexico, May 2000.

[18] S. Venugopal, R. Buyya and Lyle Winton, *A Grid Service Broker for Scheduling Distributed Data-Oriented Applications on Global Grids*, Technical Report, GRIDS-TR-2004-1, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, February 2004.

[19] K. Ranganathan, A. Iamnitchi and Ian Foster, *Improving Data Availability through Dynamic Model-Driven Replication in Large Peer-to-Peer Communities*, Global and Peer-to-Peer Computing in Large-Scale Distributed Systems, 2002,

[20] McKnight. LW, Boroumand J., *Pricing Internet Services: Approaches and Challenges*. IEEE Computer 2000; 33(2): 128–129.

[21] W. H. Bell, D. G. Cameron, R. Carvajal-Schiaffino, A. P. Millar, K. Stockinger, and F. Zini. *Evaluation of an Economy-Based File Replication Strategy for a Data Grid*. In Proceedings of 3rd IEEE Int. Symposium on Cluster Computing and the Grid (CCGrid 2003).

[22] M.Carman, F.Zini, L.Serafini, and K.Stockinger. *Towards an Economy – Based Optimisation of File Access and Replication on a Data Grid*. In Workshop on Agent based Cluster and Grid Computing at Int. Symposium on Cluster Computing and the Grid (CCGrid2002), Berlin, Germany, May 2002. IEEE – CS Press

[23] Berman, F. *High Performance Schedulers*, In the Grid: Blueprint for a Future Computing Infrastructure, pages 279 - 309. Morgan Kaufmann Publishers, 1999.

[24] Buyya R, Abramson D, and Giddy J, *An Economy Driven Resource Management Architecture for Computational Power Grids*, International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000), Las Vegas, USA.

[25] K. Ranganathan and I. Foster. *Identifying Dynamic Replication Strategies for a High Performance Data Grid.* In Proc. of the International Grid Computing Workshop, Denver, CO, November 2001.

[26] Thanasis Loukopoulos and Ishfaq Ahmad. *Static and Adaptive Data Replication Algorithms for Fast Information Access in Large Distributed Systems*. ICDCS '00: Proceedings of the 20th International Conference on Distributed Computing Systems (ICDCS 2000), IEEE Computer Society 2000.

[27] J. H. Holland. *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Mich., 1975.

[28] O. Wolfson and A. Milo, *The multicast policy and its relationship to replicated data placement.* ACM Trans. Database Syst., vol. 16, no. 1, pp. 181--205, Mar. 1991.

[29] R. Wolski. *Dynamically Forecasting Network Performance to Support Dynamic Scheduling Using the Network Weather Service*. In 6th High-Performance Distributed Computing, Aug. 1997.

[30] I. Legrand, C. Dobre, C. Stratan, MONARC Collaboation, *MONARC 2 – Distributed Systems Simulation*, Technical Report, 2003. http://monarc.cacr.caltech.edu:8081/www_monarc/Papers/

[31] Jackson, J. R. *Queues with Dynamic Priority Disciplines*, in Management Science, Vol. 7, No. 1, 1961.

[32] The eMule Project, http://www.emule-project.net/

[33] Heinz Stockinger. *Distributed Database Management Systems and the Data Grid.* 18th IEEE Symposium on Mass Storage Systems and 9th NASA Goddard Conference on Mass Storage Systems and Technologies, San Diego, April 17-20, 2001.

[34] Srikumar Venugopal and Rajkumar Buyya, *An Economy-based Algorithm for Scheduling Data-Intensive Applications on Global Grids*, Technical Report, GRIDS-TR-2004-11, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, Dec 8, 2004.

[35] Alexander Barmouta and Rajkumar Buyya, GridBank: A Grid Accounting Services Architecture (GASA) for Distributed Systems Sharing and Integration, Workshop on Internet Computing and E-Commerce, Proceedings of the 17th Annual International Parallel and Distributed Processing Symposium (IPDPS 2003), IEEE Computer Society Press, USA, April 22-26, 2003, Nice, France.