Negotiated Resource Brokering for Quality of Service Provision of Grid Applications

by

Richard Edward Kavanagh

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

The University of Leeds
School of Computing

April 2013

Abstract

Grid Computing is a distributed computing paradigm where many computers often formed from different organisations work together so that their computing power may be aggregated. Grids are often heterogeneous and resources vary significantly in CPU power, available RAM, disk space, OS, architecture and installed software etc. Added to this lack of uniformity is that best effort services are usually offered, as opposed to services that offer guarantees upon completion time via the use of Service Level Agreements (SLAs). The lack of guarantees means the uptake of Grids is stifled. The challenge tackled here is to add such guarantees, thus ensuring users are more willing to use the Grid given an obvious reluctance to pay or contribute, if the quality of the services returned lacks any guarantees.

Grids resources are also finite in nature, hence priorities need establishing in order to best meet any guarantees placed upon the limited resources available. An economic approach is hence adopted to ensure end users reveal their true priorities for jobs, whilst also adding incentive for provisioning services, via a service charge.

An economically oriented model is therefore proposed that provides SLAs with bi-criteria constraints upon time and cost. This model is tested via discrete event simulation and a simulator is presented that is capable of testing the model. An architecture is then established that was developed to utilise the economic model for negotiating SLAs. Finally experimentation is reported upon from the use of the software developed when it was deployed upon a testbed, including admission control and steering of jobs within the Grid. Results are presented that show the interactions and relationship between the time and cost constraints within the model, including transitions between the dominance of one constraint over the other and other things such as the effects of rescheduling upon the market.

Acknowledgements

I would like to thank my supervisor Karim Djemame for his guidance during my studies at Leeds University and similarly my secondary supervisor Natasha Shakhlevich. Karim has been a great help during my studies imparting a great deal of knowledge and advice, regardless of his busy schedule, while Natasha has shown me aspects of operations research and how it reflects upon my work. I would like to acknowledge the guidance and help that Django Armstrong gave during the setting up of the experimental testbed, which was most appreciated, as part of his role as its administrator. I would also like to thank Oliver Waldrich for his assistance via email during the early stages of my usage of WS-Agreement for Java, which was an essential software component of my work.

Technical issues aside a PhD is an emotional experience I would therefore like to thank the members of the Distributed Systems and Services Research group for their much valued companionship and discussions during my course. In particular I would specifically like to mention the coffee group from my office who were some of my closest friends during my time at Leeds: Django Armstrong, Silvana De Gyves, Mariam Kiran and Asif Sangrasi. We often ventured across to the coffee shop the Opposite, for much needed socialising and relaxation.

I would like to show my gratitude and appreciation for my family, for their tireless support throughout my life, my parents Stuart and Denise Kavanagh and my brother and sister Christopher and Michelle. I would also like to thank my girlfriend Beth Williams for her steadfast support during my studies. She has tirelessly driven to Leeds from my home town to be with me and has been there for me when I have needed her the most.

I would also like to thank the Engineering and Physical Science Research Council for funding the Intelligent Scheduling for Quality of Service (ISQoS) research project (EPSRC: Reference EP/G054304/1) which has allowed this work to be made possible.

Declarations

Some parts of the work presented in this thesis have been published in the following articles:

Journal Articles:

R. Kavanagh and K. Djemame. An Economic Market for the Brokering of Time and Budget Guarantees. Concurrency Computation Practice and Experience, (submitted).

Contributing to Chapter 6.

Conference and Workshop Papers:

R. Kavanagh and K. Djemame. A state based grid. In 26th UK Performance Engineering Workshop, page 8, Warwick, 2010. The University of Warwick.

Contributing to Chapter 4.

R. Kavanagh and K. Djemame. A grid broker pricing mechanism for temporal and budget guarantees. In Nigel. Thomas (editor), 8th European Performance Engineering Workshop (EPEW 2011), Lecture Notes in Computer Science, vol. 6977, Borrowdale, The Lake District, UK, 2011. Springer.

Contributing to Chapters 3 and 4.

R. Kavanagh and K. Djemame. Negotiated economic grid brokering for quality of service. In Yeo S, Pan Y, Lee YS, Chang HB (editors) Computer Science and its Applications, pp.87-96. 2012.

Contributing to Chapter 5.

R. Kavanagh; K. Djemame. The ISQoS Grid Broker for Temporal and Budget Guarantees. In Vanmechelen K, Altmann J, Rana O (editors) Lecture Notes in Computer Science, vol. 7714, pp.76-90. Springer Berlin Heidelberg. 2012.

Contributing to Chapter 6.

All material in these articles is the candidate's own work, under the supervision of the co-author Karim Djemame.

Contents

1 Introduction **1**

1.1 Introduction 1

1.2 Motivation 1

1.3 Research Context 5

1.4 Aim and Objectives 5

1.5 Methodology 6

1.6 Contributions 7

1.7 Thesis Overview 9

2 Background **10**

2.1 Introduction 10

2.2 Grid Computing 10

2.2.1 What is the Grid? 10

2.2.2 History of the Grid 12

2.2.2.1 First Generation: Coupling Super Computers Together . 12

2.2.2.2 Second Generation: Rise of Middleware 12

2.2.2.3 Third Generation: Start of Standardization and Interoperability - Meta-Data, Self* and Automation 13

2.2.2.4 Fourth Generation: Common Middleware Standards . . 14

2.2.2.5 Fourth Generation: Quality of Service Provision 14

2.2.2.6 Fourth Generation: Economics and User Orientation . . 15

2.3 Quality of Service 16

2.3.1 What is Quality of Service 16

2.3.2 What effects Quality of Service 16

2.3.2.1 Task Heterogeneity 16

2.3.2.2 Resource Heterogeneity 18

2.3.2.3 Network Heterogeneity 19

2.3.2.4 Scheduling Requirements 19

2.3.3 QoS Provision Mechanisms 20
2.3.3.1 Advance Reservation 20
2.3.3.2 Check Pointing and Replication 21
2.3.3.3 Modelling Grids 23
2.3.3.4 Scheduling 23
2.3.3.5 Admission Control 24
2.4 Service Level Agreements 24
2.4.1 SLAs in Grids 25
2.4.2 WS-Agreement 26
2.4.2.1 Flow of Actions in an Agreement 28
2.4.2.2 Agreement States 29
2.4.2.3 Service Term States 30
2.4.2.4 Guarantee Term States 31
2.4.2.5 Negotiation States 32
2.4.3 Requirements for Negotiation 33
2.4.3.1 Negotiation Initiator 33
2.4.3.2 Negotiation Responder 34
2.4.3.3 Negotiation Terms 34
2.4.3.4 Negotiation Requirements due to Economics 34
2.5 Grid Economics 35
2.5.1 Overview of Grid Economy 35
2.5.2 Economic Concepts 37
2.5.3 Economic Approaches 40
2.5.3.1 Auctions 40
2.5.3.2 Commodity Markets 43
2.5.3.3 Bargaining Model 45
2.5.3.4 Tender Contract 45
2.5.3.5 Catallexy 46
2.6 Scheduling in Grids 47
2.6.1 Aspects of Scheduling in Grids 47
2.6.2 Schedule Fragmentation 48
2.6.3 Information Availability and Dynamics of the Grid 49
2.6.4 Network Oriented Scheduling 49
2.6.5 Data Storage 52
2.6.6 Bag of Tasks, Workflows and Divisible Load 52
2.6.7 Batch vs Online Scheduling 54

2.7 Summary 55

3 Pricing Policy **56**

3.1 Introduction 56

3.1.1 Aims and Objectives 57

3.2 Related Work 57

3.2.1 First Price 57

3.2.2 First Reward & Risk Reward 58

3.2.3 First Profit, First Opportunity & First Opportunity Rate 59

3.2.4 LibraSLA 60

3.2.5 Aggregate Utility 61

3.3 Pricing of the Broker's Job Execution Service 62

3.3.1 Input Parameters 62

3.3.2 Overview of the Model's Sequence of Events 65

3.3.3 Calculating the Service Price 66

3.3.4 Overall Effect of the Market 68

3.4 The Simulator 69

3.4.1 Simulators 69

3.4.1.1 Primary Candidates 70

3.4.1.2 Other Simulators 73

3.4.1.3 Summary of Simulators 74

3.4.2 ISQoS Pricing Simulator 75

3.4.2.1 Discrete Event Simulation 75

3.4.2.2 Parameter Study Mode 76

3.4.2.3 Discrete Event Simulation Mode 77

3.5 Experimental: Parameter Variation Study 80

3.5.1 Resource and Service Market 80

3.5.2 Service Market 82

3.6 Experimental: Discrete Event Simulation 90

3.6.1 Experimental Method 90

3.6.2 Configuration 91

3.6.3 Results 92

3.6.3.1 Job Acceptance and Slack 92

3.6.3.2 The Effects of Gradient Based Deadlines 96

3.7 Comparison Study 99

3.8 Summary 100

4 ISQoS Broker 102
4.1 Introduction 102
4.2 Related Work 103
4.2.1 Best Effort 103
4.2.1.1 Condor & Condor/G 103
4.2.1.2 Portable Batch System (PBS) 103
4.2.1.3 Oracle Grid Engine 104
4.2.1.4 Globus 104
4.2.2 Quality of Service 104
4.2.2.1 OpenCCS (Paderborn Center for Parallel Computing) . 105
4.2.2.2 GridWay Metascheduler 105
4.2.3 Economic 106
4.2.3.1 Nimrod/G 106
4.2.3.2 GridBus + Aneka 106
4.2.3.3 GEMSS 107
4.2.3.4 Resource Aware Policy Administrator 107
4.2.3.5 GridEcon 108
4.3 Requirements Analysis 109
4.3.1 Requirements 109
4.3.2 Development Choice 109
4.4 Architecture 111
4.4.1 Overall Architecture 111
4.4.2 Agreement Structure 113
4.4.3 Data Representations 115
4.4.4 Grid State Representation 117
4.4.5 Offer Generation 119
4.4.6 Scheduling Algorithms 122
4.4.7 Offer Ranking and Selection 126
4.4.7.1 Offer Sorting Mechanisms 127
4.4.7.2 Offer Filtering Mechanisms 128
4.4.8 Pricing Mechanisms 130
4.4.8.1 The Available Pricing Mechanisms 130
4.4.8.2 The Effects of Dynamic Pricing Mechanisms 131
4.5 Feature Comparison 133
4.5.1 Cost & Time Guarantees 133
4.5.2 Advance Reservation Support 134

4.5.3 Queuing vs Scheduling 134
4.5.4 Negotiated Job Submission 135
4.5.5 Monitoring, Adaptation and Rescheduling 136
4.5.6 Advanced Discovery, Scheduling API & Flexible Pricing 136
4.5.7 User vs System Centric Scheduling 137
4.5.8 Open Standards 137
4.5.9 Comparison Study 137
4.6 Summary 140

5 Job Admission and Profit **142**
5.1 Introduction 142
5.2 Experimental Research Objectives 143
5.3 Experimental Setup 143
5.3.1 Experimental Method 143
5.3.2 Experimental Configuration 146
5.4 Results 147
5.5 Summary 151

6 Dynamic Pricing And Offer Prioritisation **152**
6.1 Introduction 152
6.2 Experimental Research Objective 153
6.3 Experimental Setup 153
6.3.1 Experimental Method 153
6.3.2 Experimental Configuration 154
6.4 Results 155
6.4.1 Transition to Economic Constraints Dominance 155
6.4.2 Rescheduling 161
6.4.3 Causes of Price Instability 163
6.4.4 Price Stability 164
6.5 Discussion upon Price Stability 176
6.5.1 Maintaining Selection Pressure 177
6.5.2 Effectors of Price Stability 177
6.6 Recommendations for Countering Price Instability 179
6.7 Time and Cost Constraints Compared 180
6.8 Summary 182

7 Conclusion **183**
7.1 Summary 183
7.2 Contributions 185
7.3 Future Work 186
7.3.1 Cloud Computing Deployment 186
7.3.2 Economics 187

Bibliography **189**

List of Figures

1.1 An Overview of the Grid 2

2.1 Overview of WS-Agreement Structure [141] 27
2.2 Overview of WS-Agreement Negotiation Structure [144] 28
2.3 Agreement States in WS-Agreement [141] 29
2.4 Service Term States [141] 30
2.5 Guarantee Term States [141] 31
2.6 Negotiation States [144] 32
2.7 Scheduling in Grids from a Broker's Perspective [107] 48

3.1 Aggregate Utility's Function For Rewarding The Completion Of A Set Of Tasks 62
3.2 The Resource Cost in Relation to the Pricing Model 63
3.3 Overview of the Model's Sequence of Events 65
3.4 Charging of the Guaranteed Time Service 66
3.5 Charging of the Guaranteed Time Service - Temporal Aspects 68
3.6 The Service Market - An overview of Temporal and Budgetary Aspects . 68
3.7 An Overview of Discrete Event Simulation 75
3.8 An Overview of the Simulator 78
3.9 An Overview of the Resource and Service Market Interaction 81
3.10 Broker's Profit - Varying Deadline and Markup (completion time = 15,000) 82
3.11 Broker's Profit Shown as a Surface - Varying Deadline and Markup (completion time = 15,000) 84
3.12 Penalty Fee Exploration Shown as a Surface - Varying Due Date, Deadline and Penalty Fee 84
3.13 Penalty Fee Exploration at 25% Mark-up Varying Penalty Fee, (Due Date = 15,000, Deadline = 18,000 85
3.14 Resource Cost Exploration: Fixed Penalty Fee at 0 Varying Resource Cost, Markup and Due Date 87

3.15 Resource Cost & Penalty Fee Cap Exploration: Varying Penalty Fee, Resource Cost, Markup and Due Date 89
3.16 The Effects of Slack upon Job Acceptance with Varying a) Markup, b) Arrival Rate, c) Task Count, d) Task Size and e) Reference Machine Speed 93
3.17 The Effect of Slack and Deadlines on Job Acceptance 97
3.18 The Effects of Intelligent Filtering 98

4.1 Overview of the ISQoS System 111
4.2 ISQoS SLA Structure 114
4.3 Job, Task and Action's Internal Representations within an ISQoS Provider 115
4.4 Resources 117
4.5 State 117
4.6 The Elements of the Scheduling System 119
4.7 The Flow of Events in the Offer Maker and Committer Component 120

5.1 Average Slack 147
5.2 Average Number of Jobs Accepted 148
5.3 Overall Broker Profit 148
5.4 Average Start Delay 150
5.5 Completion Time Trace - Run 1 - Highest Profit Based Selection 150

6.1 Transition to Budget Prioritisation - With Round Robin Scheduling 156
6.2 All Constraint Violations - With Round Robin Scheduling 157
6.3 Constraint Violations with a Budget of 12,000 only - With Round Robin Scheduling 157
6.4 Constraint Violations with a Budget of 15,000 only - With Round Robin Scheduling 158
6.5 Start Delay - With Round Robin Scheduling - For All Due Dates 159
6.6 Start Delay - With Round Robin Scheduling - For Due Dates 18, 20 and 24 159
6.7 Start Delay - With Round Robin Scheduling - Shown By Due Date 161
6.8 All Constraint Violations - With Rescheduling 162
6.9 Effect upon Service Price of Rescheduling (Due Date = 20, Average of all Runs) 162
6.10 Instability Demonstration - Round Robin 163
6.11 Instability Demonstration - Rescheduling 164
6.12 Average Time Between Billing Events for each Given Due Date) 165

6.13 Standard Deviation for the Time between Billing Events for each Given Due Date) 165
6.14 Average Time Between Billing Events for Due Date = 24) 166
6.15 Average Time Between Billing Events for Due Date = 8) 167
6.16 Average Time Between Billing Events for Due Date = 16) 167
6.17 Average Service Price (For Each Due Date) 168
6.18 Standard Deviation of the Average Service Price (as shown in Figure 6.17) 169
6.19 Standard Deviation as Multiple of Smallest Deviation Value 169
6.20 Service Price over Time (Due Date = 24) 171
6.21 Final Overall Revenue 171
6.22 Constraint Violations - Re-Priced (All Budgets) 172
6.23 Job Acceptance and Budget Selection Preference for re-pricing 173
6.24 Job Acceptance and Budget Selection Preference for Rescheduling 173
6.25 Rescheduling - Job Completion Time 174
6.26 Round Robin - Job Completion Time 174
6.27 Rescheduling Re-priced - Job Completion Time 175
6.28 Job Acceptance 176
6.29 Price and Schedule Dependence 176

List of Tables

3.1 EBNF Used in description of user input 66
3.2 The Machine Cost Pairs used in the Resource and Service Market Example 81
3.3 Parameters for Figures 3.10 and 3.11 83
3.4 Parameters for Figures 3.12 and 3.13 85
3.5 Parameters for Figure 3.14 88
3.6 Parameters for Figure 3.15 90
3.7 The Minimum and Maximum Tasks per Job 92
3.8 The Relative Speed Difference Between the Reference Machine and the Slowest machine 95
3.9 Comparison of Related Pricing Models 100

4.1 Queuing vs Scheduling [18] 135
4.2 Feature Comparison Study - Best Effort 138
4.3 Feature Comparison Study - QoS 139
4.4 Feature Comparison Study - Economic 140

6.1 Summary of Standard Deviation of Service Price 170
6.2 Summary of Standard Deviation of Service Price Ignoring Due Date 20 and 24 170
6.3 Cost and Time Constraints Compared 181

List of Algorithms

1 Nimrod/G's Cost Optimize 123
2 Nimrod/G's Time Optimize 123
3 Nimrod/G's Cost Time Optimize 124
4 Cost Roulette Wheel 125
5 Round Robin Rescheduling 126
6 ISQoS Hybrid Offer Filter 128

List of Abbreviations

API	Application Programmers Interface
BEinGrid	Business Experiments in GRID
BOT-R	Bag of Tasks Rescheduling
BREIN	Business objective-driven REliable and Intelligent grids for real busiNess
CCR	Communication-to-Computation Ratio
CPU	Central Processing Unit
CVS	Concurrent Versioning System
DAG	Directed Acyclic Graph
DiffServ	Differentiated Services
distributed hash tables	DHT
DRMAA	Distributed Resource Management Application API
EBNF	Extended Backus Noar Form
EGI	European Grid Initiative
EGI-InSPIRE	Integrated Sustainable Pan-European Infrastructure for Researchers in Europe
EMI	European Middleware Initiative
FAFNER	Factoring via Network-Enabled Recursion
FCFS	First Come First Serve
FIFO	First in First Out

G-QoSM	Grid Quality of Service Management
GARA	General-Purpose Architecture for Reservation and Allocation
GB	GigaBytes
GEMSS	Grid-Enabled Medical Simulation Services
GES	Grid Economic Simulator
GLUE	Grid Laboratory Uniform Environment
GNB	Grid Network Broker
GNRB	Grid Network-aware Resource Broker
GPU	Graphical Processor Unit
GRACE	GRid Architecture for Computational Economy
GT4	Globus Toolkit 4
HECIOS	The High End Computing I/O Simulator
HPC4U	Highly Predictable Cluster for Internet-Grids
IaaS	Infrastructure as a Service
ISQoS	Intelligent Scheduling for Quality of Service
JSDL	Job Submission Description Language
JSS	Job Submission Service
LJF	Longest Job First
LL	Load Leveller
MB	MegaBytes
MHz	MegaHertz
Middleware for Activating the Global Open Grid	MaGoG
MPI	Message Passing Interface
NASA	National Aeronautical and Space Administration

NQS	Network Queueing System
NRSE	Network Resource Scheduling Entity
OGE	Oracle Grid Engine
OGSA	Open Grid Service Architecture
OGSI	Open Grid Service Infrastructure
PaaS	Platform as a Service
PBS	Portable Batch System
QoS	Quality of Service
RAM	Random Access Memory
RAPA	Resource Aware Policy Administrator
RSL	Resource Specification Language
SaaS	Software as a Service
SETI	Search for ExtraTerrestrial Intelligence
SJF	Shortest Job First
SLA	Service Level Agreement
SLO	Service Level Objectives
SORMA	Self-Organizing ICT Resource Management
Time to Live	TTL
VM	Virtual Machine
VO	Virtual Organisation
WQR	Work Queue with Replication
WQR-FT	Work Queue Replication - Fault Tolerant
WS-Agreement	WS Agreement
WSAG4J	WS-Agreement for Java

WSLA	Web Service Level Agreement
WSRF	Web Service Resource Framework
XML	eXtensible Markup Language

Chapter 1

Introduction

1.1 Introduction

This is the introductory chapter to the thesis. It starts by expressing the motivational scenarios for this work in Section 1.2 and places this motivation within the context of the Intelligent Scheduling for Quality of Service (ISQoS) project in Section 1.3. From the motivational scenarios the aims and objectives for the thesis are derived (see Section 1.4). The discussion then focuses upon how the research was conducted in the methodology (Section 1.5) and then onto the contributions of the research in Section 1.6. This chapter then finishes with an overview of the thesis' structure in Section 1.7.

1.2 Motivation

Grids [76] link a large set of computers together for the execution of jobs in a coordinated manner. Thus they enable the execution of applications in a distributed fashion using many computers. Their focus is upon sharing "direct access to computers, software, data and other resources, as is required by a range of collaborative problem-solving and resource brokering strategies emerging in industry, science and engineering." [79].

To enhance the understanding of the Grid the following flow of actions in a standard Grid job submission are described. This description that is given has an intermediary broker that steers jobs in the Grid to local schedulers, which is the case in the majority of centralized Grid environments [117].

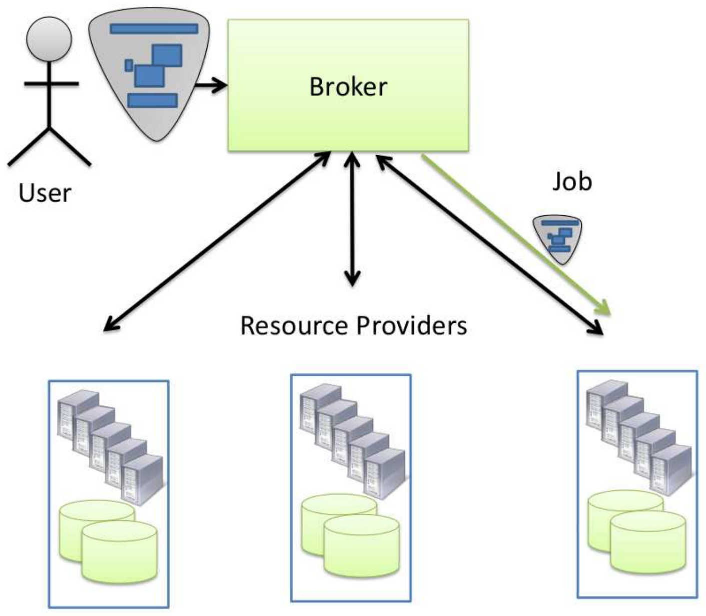


Figure 1.1: An Overview of the Grid

A Grid broker would be used to contact several resource providers, across a network infrastructure, as illustrated in Figure 1.1. Each of these resource providers makes available to the broker a set of compute resources, that provide both storage and compute facilities. A user utilises the broker to submit work to the providers. The broker takes the job of the user and selects the best provider to perform the execution of the job and submits it to that provider. A resource provider is required to execute any job it is given by the broker. This may either be via queuing the job for execution or by performing scheduling. Jobs comprise of several tasks, that have to be executed. These tasks require that data be transferred to the provider, which may also include the executable to be run. The Grid application is then run on the provider's resources, after which a set of results are generated. At the end of execution the resource provider then sends the computed results back to the end user, over the network.

A brokered scenario is not the only one possible. It is also possible that a client would submit jobs directly to a given provider. It therefore means there is a need for the broker to offer added benefit in order to justify its existence within the Grid environment. This may be as simple as the knitting together of different environments such as Condor with Condor/G [178], but in later years others such as AssessGrid [18] offered assessments in regards to trust and offered the provider most likely to fulfil the SLA. AssessGrid achieved its brokerage in one of two ways either acting as a full intermediary in which it acts as a virtual provider and subcontracting to the best provider available or acting merely as an advisory service pointing users onto the best provider.

In Grids it is common that applications are served in a best effort approach only, with no guarantees placed upon the service quality. This is primarily due to the system centric nature of existing production Grids. The emphasis being upon the queuing of jobs ready for computation, with the sole intent of maintaining high resource utilisation, rather than user satisfaction. User-centric behaviour [54, 99] on the other hand is oriented toward providing the greatest utility to the end user by providing guarantees upon performance and are usually aided by a system of rewards and penalties [37]. It has also been known for some time that guaranteed provision of reliable, transparent and quality of service (QoS) oriented resources is the next important step for Grid systems [18, 90, 106, 111, 124, 130, 134].

In many commercial and scientific settings, guarantees that computation is going to be completed on time are required. It is therefore important to establish during the submission of a job the requirements of the users in terms of completion time and the priority they hold for the work.

The motivation for this work in providing time and cost guarantees can be illustrated in the following three scenarios:

- The first is a commercial scenario, such as animation, where frames may be computed overnight before the animation team arrives the next day, as with any overnight batch jobs partial completion of the work delays or stops the team from starting the next day's work [17]. This scenario presents a commercialisation of the Grid, where the use of best effort services, which have no attached guarantees limits the economic importance of Grids. This is due to users reluctance to pay, directly or indirectly (e.g., by contributing resources to the Grid), for the service they receive, if there are no guarantees on performance particularly when computed results are needed for a commercial process [111]. A further problem in this scenario is that current Grid markets are system centric selling physical resources, i.e. CPU hours. This however, is not relevant to enterprise customers who have deadlines to meet and have limited ideas about how many physical resources are required to meet such a deadline [130], thus a more user centric approach must be realised.

- The second scenario is in an academic environment where it is common before conferences for Grids to become overloaded [95]. This requires on an academic Grid for jobs associated with the conference to be prioritised and the results obtained in a timely manner before it becomes too late to submit to the conference. The emphasis in this scenario is upon the limited availability of resources and the requirement to prioritise work, to ensure the most valuable jobs are completed first. In existing infrastructure jobs are queued and results are returned as and when they are ready. If the Grid becomes overloaded jobs that are of a high priority get delayed by jobs that could otherwise have waited. In order to determine priority however users need to be incentivised to reduce their priority, over others. An economic approach is a means to achieve this [37], as value-oriented approaches are not sufficient, as all participants have to be willing to report their priorities and values honestly [130].

- The third scenario is taken from the field of Urgent Computing. The aim of Urgent Computing is to aid decision making in events [127] e.g. earthquake prediction [80]. In such cases it is not useful to waste time waiting for jobs to complete in a queue, as is the case in the best effort approach. This is principally because Urgent Computing jobs are computationally demanding jobs which are started with a low level of predictability and which need to be completed before a given deadline [127]. In this scenario the time criticality is key. If deadlines are missed then the value of the results gained by computation is negligible. Thus further emphasis can be brought upon ensuring time guarantees.

To further focus our work and enhance its relevance the types of application running upon the Grid was considered. The types of tasks to be considered was narrowed to Bag of Task based applications, which are the predominate form of workload in Grids [95]. In this form of application a collection of similar tasks are submitted, which are wholly independent and do not need communications between tasks belonging to the same job [122].

Given the finite Grid resources available (including the end user's budget), it is wise to prioritise jobs based upon the importance to the end user. In order that this prioritisation is provided correctly an economic approach is used to ensure end users give more truthful indications of their priorities [37, 118]. The adding of the cost requirement means the planned broker has to deal with the bi-criteria problem of ensuring both the budget and completion time are met and that its justification for participating within the Grid is that guarantees can be placed upon such criteria.

In delivering these time and cost requirements, it presents the added advantage that Grids can be moved away from the best-effort service which limits their importance, as users would be more willing to pay or contribute resources if results are returned on time, as late results are of limited use [111].

1.3 Research Context

The work discussed in this thesis was part of the ISQoS project (2009-2013). This project was funded by the Engineering and Physical Science Research Council (EPRSC). It was multidisciplinary in nature, with the aim of producing collaborative work between the Grid and Operations Research communities. The focus of the project was to improve QoS provision in Grids via advanced scheduling techniques. The focus of the development work was therefore directed towards methodologies that could foster this possibility and hence introduced its own requirements, while also focusing the project towards the area of scheduling.

The aim of the ISQoS project in particular was to develop a resource broker that combines the features of both system-centric and user-centric brokers [161]. It was to transparently meet users' requirements, whilst also optimising the usage of Grid resources for multiple resource providers. This was to be achieved by scheduling while ensuring the runtime for users' jobs and resource usage cost was balanced.

1.4 Aim and Objectives

The aims and objectives of the work presented in this thesis are to:

- provide a brokering architecture that can be used to support QoS in terms of time and cost constraints.

  This architecture should also support:

  - Reservation Based Allocation: This is a central requirement of planning using techniques used in operational research. In meeting this requirement it allows domain experts in the field of scheduling to contribute to the Grid community.
  - Service Level Agreements (SLAs): SLAs can be used to offer guarantees on QoS, it is proposed that this can be based upon a plan of action created by a scheduling phase.
  - Negotiation: so that QoS may be achieved while respecting the current environment's ability to provide such service levels. This should hence respect both user and service provider's needs.

- provide a market model for job submission that distinguishes between jobs in terms of QoS levels. The market should be tested via simulation and via an experimental testbed.

- ensure that the market model is constructed to allow multiple providers to compete for jobs and to manage the steering of jobs between providers.
- provide a platform so that experimentation may be performed using the proposed brokering architecture as an exploration of the possibility of taking expertise from operations research into Grids.

1.5 Methodology

In order to perform an investigation that meets the aims and objectives of this work, the following methodology was applied.

The first stage was to assess existing economic models against the envisaged ISQoS requirements, which were derived from the motivating scenario. After this assessment was performed the pricing model was designed, ensuring it binds cost and time factors and generally meets the requirements specification.

After an initial model was designed it was required to be tested. There are three principle ways this might be achieved, of which two methods were chosen. They are described as follows:

Simulation: This requires the model to be represented in a simulator and for experimentation to be performed. In this respect it is reliable in that conditions within the simulated Grid are likely to be reproducible. It can also be performed quite quickly and is responsive to changes should they be required. The simulation alone however requires further verification, to ensure that the simulation matches reality. Examples of the simulation approach been taken include [35, 42, 49, 94, 125].

Direct Experiment - on a Grid: This method of testing the model is the most accurate should the conditions match that of a typical infrastructure. It is however very resource intensive to prepare requiring substantial preparations. It also requires interfering with a working Grid infrastructure. A more reasonable and related approach is to use a testbed, that is a scaled down version of such a Grid infrastructure. In doing so there is no interference with existing infrastructure and greater control can be granted over the testbed's infrastructure. Examples of such an approach been taken include [32, 150, 184, 185, 191].

Mathematical Modelling: This requires the construction of a precise mathematical model that must match the real environment as closely as possible. This proves difficult in the Grid situation which is subject to randomness caused by the distributed nature of the Grid and underlying non-Grid workloads. Examples of mathematical modelling being used include [26, 102, 153, 183].

The approach taken was therefore to test the model via simulation and by direct experiment upon a testbed.

In order to ensure the model was correctly implemented when testing via simulation, a discrete event simulator was constructed. In doing this it ensured the model was implemented without taking into consideration any shortcomings of other simulators due to the simulator's authors own assumptions and research focus, such as auctions in GridSim [42].

The initial investigation of the proposed model was performed using simulation. This was done to ensure viability. Discrete event simulation also acted as a way of providing fine grain control of the test environment, while also ensuring the development risk was low, by not committing to building an experimental prototype based upon an untested model.

Finally an experimental prototype was constructed and used to perform experimentation upon a testbed. A testbed within the school was chosen for this purpose, due to both its availability and capacity to configure the test environment, particularly in regards to that of the broker and internals of the resource providers.

The experimentation was performed in stages initially without dynamic pricing or rescheduling. Dynamic pricing and other advanced features were finally added, thus testing the model with all the complexities that it allows.

1.6 Contributions

This thesis has the following major contributions:

- An Architecture, for the ISQoS Broker. This is a broker and middleware that establishes a tender market by which service providers may bid for jobs to perform. It incorporates a negotiation mechanism that is used to define quality of service constraints upon the services for both completion time and cost. This contribution can be found in Chapter 4.

- An Economic Model: The Pricing Model is the central part of the broker, it binds the price to the end user with the quality of service provided by the service provider. It ensures that if the SLA obligations are not met that the broker is incentivised by losing money. In order to ensure the broker remains profitable it must perform only the admission of jobs that are likely to have their SLAs met, hence admission control also features as part of the economic models contribution. This contribution can be found in Chapter 3 and has been implemented within the broker, aspects of the models behaviour can be seen in the next three contributions.

- A Simulation Study of the Pricing Mechanism of the Offers/Job Service Market: A simulation was performed with the main aim of establishing a repeatable experiment that tests the viability of the brokering mechanism, ensuring that it may be designed to have budget balance and remain incentive compatible [155]. In achieving this a viable broker for a job execution service can be established. This contribution can be found in Chapter 3.

- Recommendations on Selection Strategies: From the experimentation performed both with a testbed and by simulation, recommendations are presented on selection strategies that may be used by any person wishing to implement a similar economically orientated Grid. This contribution can be found in Chapter 5, as part of the experimentation performed.

- Recommendations in Regards to System Stability: The experimentation performed will show how economic stability can be achieved within a Grid market given various factors such as rescheduling. The impacts upon the market will be demonstrated and recommendations on structuring the pricing mechanisms will be made to ensure overall system and price stability. This contribution can be found in Chapter 6 as part of the experimentations performed.

1.7 Thesis Overview

The remaining part of the thesis is laid out as follows:

- Chapter 2 will establish the background and introduce the current state of the art in Grids. It will indicate the current trends in Grid research towards QoS and outline where this thesis fits within this trend.
- In Chapter 3 the pricing mechanism within the brokering system will be introduced. This model drives the decision making within the broker and ensures that the quality of service parameters of time and cost can assessed by the broker.
- The broker is then introduced in Chapter 4. This chapter discusses the architecture in detail and discusses recommendations on economically oriented Grid brokers.
- Chapters 5 and 6 discuss experiments using the broker. Chapter 5 focuses upon static pricing and what can be learned while Chapter 6 focuses upon dynamic pricing and rescheduling.
- Finally Chapter 7 concludes this thesis and suggests areas for future work.

Chapter 2

Background

2.1 Introduction

In this chapter the background to the work presented in the thesis is discussed. This starts with the topic of Grid computing followed by a history of the Grid, which places the thesis work in the context of developments within the Grid community. The next stage is to discuss Quality of Service (QoS) provision and to discuss in particular what has an effect on QoS and how these issues might be mitigated. The next area discussed is that of Service Level Agreements (SLAs), this covers their usage in Grid and also the Open Grid Forum standard called WS-Agreement [141], after this is discussed the concept of the Grid Economy is covered. An overview of Grid economics is given first, detailing the benefits that it offers. Then key requirements for Grid economics to work are discussed, followed by a discussion of the possible approaches that may be taken. Finally scheduling within Grids is discussed.

2.2 Grid Computing

2.2.1 What is the Grid?

The Grid [76] is a form of distributed computing in which resources are shared. The resources are used for computationally difficult and/or data intensive jobs that require multiple resources in order that the job is completed. There are fundamental requirements for a distributed system to be considered a Grid. These requirements are to [76]:

1. ”co-ordinates resources that are not subject to centralized control”
2. use ”standard, open, general purpose protocols and interfaces”
3. ”deliver non-trivial qualities of service”

In order to explore the meaning of the Grid these points are discussed further:

Point 1 declares there to be no centralized control. In Grids this comes about from the notion of sharing resources, where multiple institutions may collaborate together to form a Grid. There is therefore a strong emphasis on sharing resources, between multiple different groups and on collaboration. This requirement to coordinate resource sharing and problem solving across multiple organisations is hence fundamental to the Grid problem [79].

Virtual organizations (VOs) are a concept in Grids that captures this concept clearly. VOs are collections of individuals, institutions and resources that have been brought together to meet a common goal. VOs in Grids are therefore aimed at supporting the Grid’s goal of cooperation by enabling the sharing of resources. This might include access to applications, compute cycles and storage facilities etc. and that this sharing is non-trivial in nature [79]:

> ”The sharing that we are concerned with is not primarily file exchange but rather direct access to computers, software, data, and other resources, as is required by a range of collaborative problem-solving and resource brokering strategies emerging in industry, science, and engineering.”

This requirement for collaboration as previously discussed tends to detract from the technical aspects as being fundamental to the definition of the Grid. This may be argued because the collaboration of multiple organisations gives rise to point’s 1 and 2 of Foster *et al.*’s original definition. Organisations could not be expected to change their entire infrastructure so centralised control of resources cannot easily be considered. Further to this, questions such as, ”who should be given control of the Grid?” are difficult in collaborative ventures.

The second point relates to the Grids complexity. Due to the Grid size in order that it be established abstractions are required, in order to make the task simple and practical enough to be undertaken, common and open standards are a means to make this happen.

Kesselman and Foster [108] wrote about a computational Grid that it ”is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities.” This definition at first sight discusses more infrastructure issues in the Grid. It shows however the need for dependable and consistent resources and ”high-end computational capabilities” which both give a leaning towards computational capacity in that Grids should be large but also to a need for quality of service. This is in keeping with point 3, which requires non-trivial qualities of service for example job completion time and cost.

2.2.2 History of the Grid

2.2.2.1 First Generation: Coupling Super Computers Together

The Grid concept started in the early to mid-1990s as a series of efforts to link supercomputing centres together [57]. This resulted in the linking of a number of US national supercomputing centres together. Typically, the objective of these early projects was to provide computational resources to a range of high performance applications [57]. Two representative projects of this was [57]: FAFNER [57] and I-WAY [57, 59]. FAFNER acted as the forerunner of desktop computing projects such as SETI@home [112] and I-WAY middleware was a forerunner of Globus Toolkit [77, 181].

2.2.2.2 Second Generation: Rise of Middleware

In the next generation of Grids, the middleware became prominent as a mechanism to deal with the complexity of Grids. It allowed Grids to become more ubiquitous than merely a few specialised data centres coupled together [57], examples of such middleware include Globus, Unicore [154] and NorduGrid/Arc [166].

The main issues the middleware was required to handle was:

- Heterogeneity: of the resources, they had to be made uniformly accessible, even if they were vastly different in terms of their capabilities.
- Scalability: The Grid had to scale, the principle aims of Grids as a means of collaboration required that many computers would be used together concurrently. Scalability hence became a fundamental element of the Grid.
- Adaptability: The scale of Grids meant failures could be expected, given the quantity of resources used, it was most probable that at least one would fail.

To look at early efforts at resource scheduling such as: Condor/G [178] PBS [31] and OGE [177] their primary focus is batching and resource scheduling and that they principally started life as systems for managing jobs or tasks on cluster platforms i.e. locally distributed computing platforms. This can be seen in cases such as Condor, with the use of Condor/G as a means to couple together local Condor pools. This factor of being adapted to suit the Grid is important in the later development efforts.

2.2.2.3 Third Generation: Start of Standardization and Interoperability - Meta-Data, Self* and Automation

In order to build upon prior successes it became desirable to be able to reuse existing components and assemble them in a flexible manner. This meant there was an increasing adoption of a service-oriented models [57].

The use of Web Services principally starts with the Open Grid Service Architecture (OGSA) [140] as proposed in [78] and the Open Grid Services Infrastructure (OGSI). In the Globus toolkit this is shown in v3 (GT3).

OGSI was then refactored [182] to create Web Service Resource Framework (WSRF) [92, 114, 136] and in doing so WSRF more or less replaced the old OGSI [159]. In the Globus toolkit this is shown in v4 (GT4). This move to web services can however, be seen as not being entirely successful. Globus Toolkit Version 5 dropped the use of Web service components that had existed in GT4 [159]. In what has been classed here as the fourth generation of Grids, this standardisation effort continues.

De Roure, *et al.* also indicate that there is a strong sense of automation in the third generation systems [57]. This is principally down to the complexity becoming too difficult to handle, in regards to the scale and heterogeneity. The concept of the virtual organisation [79] also becomes more prominent in how organisations can collaborate together, in a distributed fashion [57].

The fourth generation has been split into several themes. The first of these themes is the improving of integration and common standards between the initial variations of middleware. The second is the adding of additional services and involves the making of the software more commercially robust, in considering risk, trust, general quality of service provision and economics. Economics given the ISQoS project's nature is divided into its own section for a more in depth discussion of related Grid projects.

2.2.2.4 Fourth Generation: Common Middleware Standards

This section discusses further the on going efforts for common standards in middleware and the driver towards this requirements. The fact that there was several different main-stream middlewares with their own specific interfaces affected the ability to collaborate. This is because incompatible middlewares limited accessibility of resources as well as application portability.

In terms of accessibility once a particular Grid infrastructure has been committed to, it limits users to using resources controlled by that distribution, thus limiting the ability to collaborate, to which the Grid was intended, by construction islands of incompatible middleware. This damages the portability of applications in that each Grid specifies its own unique interfaces, causing applications to become distribution specific [10].

Adapters in some cases could be written to provide partial solutions and breakdown of some of these islands. This is particularly common in meta-schedulers such as GridWay [92, 93], however the requirements for multiple adaptors scale poorly, since if there are N different kinds of middlewares, $O(N^2)$ different adapters are needed [10].

Standards however take time to develop and sometimes have limited but common capabilities, in comparison to legacy interfaces. After development it also takes time to retrofit Grid middlewares, with the emerging standards. This includes notable standardisation efforts such as the KnowARC project (2006-2009) [14, 165] where the ARC middleware was redeveloped so that the newer standards could be followed. This includes a surveying efforts of existing standards and their adoption which can be found in [110].

The main standardisation effort in Europe came from the European Middleware Initiative (EMI) [68]. The principle aim of this was from existing middleware, namely UNICORE [154], ARC [166] and gLite [63] to create a single common middleware for Europe. The general theme of increasingly following standards is one of integration and collaboration. The integration in Europe of middleware included the rise of the European Grid Initiative (EGI) [64] where the integration of resources from providers around Europe into a seamless and secure e-infrastructure could be coordinated. This includes the EGI-InSPIRE project (Integrated Sustainable Pan-European Infrastructure for Researchers in Europe) which started on 1 May 2010, which has the aim of establishing a sustainable Europe wide Grid infrastructure.

2.2.2.5 Fourth Generation: Quality of Service Provision

Another significant enhancement made to Grids is in regards to Quality of Service provision. It became more apparent to further the uptake of Grids that Quality of Service provision would be required and that only the guaranteed provision of reliable, transparent and quality of service (QoS) oriented resources would be useful, in the step towards successful commercial use of Grid systems [18, 106, 124, 149]. Given that businesses would require guarantees regarding the technology.

The concept of quality of service would take many forms including the resilience of the resources in the HPC4U [90] project and additional features such as risk analysis as performed by AssessGrid [18]. The enhancing of Grids also followed economics which is discussed in the next section.

2.2.2.6 Fourth Generation: Economics and User Orientation

The use of economics in Grids derives from the drive towards self-management, quality of service provision and the commercialisation of Grids. This is because concepts in economics were particularly well suited to the distributed nature of the Grids, towards its commercialisation and quantification of QoS, or lack thereof, in terms of rewards and penalties.

Grids without economics have largely been restricted to the context of academia and in order to expand the use of the technology commercialisation has been seen to be key [90, 106, 111, 130, 134]. The discussion on economics is expanded further in Section 2.5, the discussion here limits itself to a discussion of European projects that have introduced economics into Grids.

In Europe this drive towards economics came from projects such as GridEcon [8, 9] and SORMA (Self-Organizing ICT Resource Management) [130] as well as commercial exploitation from the BEinGrid project [24].

GridEcon [8, 9] aimed to define a model for managing complex market-based service oriented Grids, introducing new stakeholders such as risk brokers, SLA monitors and price modellers [127], which provided value added services such as capacity planning and insurance contracts [9].

SORMA [131, 133] was a project that aimed to create an Open Grid Market, based upon a set of use cases. The primary focus was to develop methods and tools for the efficient market-based allocation of resources in Grids. SORMA features concepts from autonomic computing, for the self-organisation in the resource management system, ensuring it was both automatic and autonomous.

BEinGrid (Business Experiments in GRID) was a project aiming to facilitate business applications such as: business process modelling, financial modelling and forecasting via the use of Grid technologies [151]. This was therefore aiming to commercialise the Grid and ensure it had higher commercial uptake.

2.3 Quality of Service

In this section the concept of Quality of Service on the Grid is discussed. The discussion first covers what could be meant by QoS and then goes on to what can affect QoS and then how it can be provided for.

2.3.1 What is Quality of Service

Quality of Service (QoS) is the notion of how good a service is and that of its fitness for task. It is therefore difficult and most likely impossible to express QoS as a single scalar value. This is largely because the ”goodness” of a service is a matter of perspective and is context specific. It may cover many different areas such as Performance, Dependability, Reliability, Scalability, Capacity, Robustness, Manageability, Exception Handling, Accuracy, Integrity, Accessibility, Availability, Interoperability, Security, Confidentiality, Traceability, Accountability [89, 101].

QoS is very context oriented for example data transfers in networks may consider things such as maximum latency, minimum bandwidth and the continuous data transfer rate available. In the example of data transfers the context matters greatly, such as real-time data communications need a minimum continuously available amount of bandwidth whereas a file transfer might not be as fussy [27].

Given this variation the focus of the discussion upon QoS will be within the context of the project. Quality of service will therefore take the meaning of anything that delays tasks completion time or makes them cost more. This may include things such as data transfers, which can cause variations to the expected completion time of a task, due to the transfer of data to and from the resources that perform computation in the Grid.

2.3.2 What effects Quality of Service

In Grids there are many things that effect the performance of an application and hence the quality of service that can be offered. This can effect either the computation, communication or data storage requirements of an application. A listing of factors that influence the provision of services is listed in the next sections:

2.3.2.1 Task Heterogeneity

Tasks have varying properties that once placed on the Grid greatly influence the overall performance obtained. Factors that are directly affected by variations in the tasks are discussed below:

Task File Sharing The degree to how much tasks share the same input files is important. If tasks use the same data then they can be allocated to either the same resources or to resources close together, hence allowing input files to be reused. Input file affinity is a measure that quantifies this degree of sharing of input files among tasks belonging to a job [163]. A key benefit claimed of using such a metric is that applications scale better, as it ensures the network is less likely to become overtaxed thus preventing it acting as a bottleneck.

Data Locality The locality of data is important, files may be very large and hence cannot easily be moved [37]. In situations where languages such as Java are used the common runtime environment presents the ability for the application to be moved towards the data [57]. In cases where this is not applicable, applications may be compiled for a particular operating system and CPU instruction set which places constraints upon the resources selection. Files may also become over utilized when multiple tasks require the same dataset simultaneously, which causes a bottleneck in the system [37]. The files therefore may be replicated but this brings issues of keeping the copies synchronised.

Communication-to-Computation Ratio (CCR) This is the ratio between how much an task performs data transfers as compared to computation [22] and is an aspect of a tasks heterogeneity. This is important because high communication costs results in data locality issues and increased effort in data transfers. Schedulers such as one presented in Liu and Baskiyar [124] makes uses this ratio during the sorting of task queues. This is done by sorting by an applications priority value then by the CCR ratio and finally by a deadline. The aim of this is to ensure that the most jobs complete in time. This sorting however creates a bias towards the highest priority jobs that involve only computation and are closest to the deadline and hence the applications fairness could be questioned, though it remains a good demonstration of the ratios importance.

Communication Patterns It is important to understand both when and how much data is going to be transferred in order to maintain QoS. This is because if this is not understood transfers critical to the completion time of a job may be delayed. Substantial effort has been undertaken to perform analysis on traces taken from Grids. This has shown that batch submissions accounted for up to 96% of the CPU time consumed and that 75% of these batch jobs that were parameter sweeps [96]. This means most jobs involve a transfer at the start followed by execution and then a transfer back, with no communication during execution.

Task Parallelisability Tasks depending upon how the application is written can have varying degrees of parallelisability. If the task is written to use multiple threads concurrently it can benefit from having available to it multiple cores. This therefore means some tasks will benefit more from resources that can provide access to multiple cores.

2.3.2.2 Resource Heterogeneity

Resources are variable in their nature and this is required to be managed. Factors that are directly affected by variations in the resources are discussed below:

Reliability Resources in Grids are subject to failures and given the scale, failures are the rule rather than the exception [57]. They fail at different rates and being able to determine which resources are the most reliable becomes of great use, as is the recovering from failures when they happen.

Library Availability/Runtime Environment The selection of suitable resources in regards to the runtime environment, e.g. Java, or the presence of library X,Y or Z and runs on Solaris etc, with 2GB free disk-space has its issues. Different environmental configurations causes heterogeneity in the resources which varies demand for them. In doing so it complicates the allocation problem. It requires an initial resource filtering mechanism [157] such as Condor Class Ad's [178].

CPU Speed, Count and Architecture Optimisations Tasks can be optimized for certain instruction sets (i.e. SSE, CUDA) and may only be able to run on specific architectures (i.e. x86, x86-64, ARM), issues extend with variations such as number of cores, hyper-threading and access to GPU based computing. This is especially the case when runtime environments such as Java or other virtualisation technologies are not in use. The main issue with CPU architecture optimizations is that it can give rise to further execution time variations between resources and that such variations are heavily task dependant [7]. It also means that the selection process if it utilises these optimisations can be made more efficient, while causing demand variations based upon CPU type.

Existing Workloads Resources will have background processes which may slow current working Grid tasks. Tasks can also be either allocated to a single resource exclusively, or many tasks can be concurrently run on the same resource [72].

2.3.2.3 Network Heterogeneity

The passing of data needs bandwidth; this means it needs to be controlled as it both takes time to transfer data and is a limited commodity. Factors that are directly linked to network issues are discussed below:

Reservability Bandwidth may be reserved so that the transfer of data occurs [45]. It must also be noted that common protocols such as TCP/IP are not QoS aware [27, 48] and that data is sent on at a best effort attempt, where redundancy is key and not timely and predictable arrival. It can therefore be seen that it may not be possible to give complete reservations of network capacity when the exact capacity available to the Grid may fluctuate.

Traffic Predictability & None Grid Traffic It is not always possible that the Grid is solely responsible and in control of all of the network traffic, as there is often none Grid processes which compete for the available bandwidth [44, 45]. This is particularly prevalent in cases involving multiple sites that require the use of wide area networks. The time taken to transfer data is often unreliable and various techniques are needed such as Differentiated Services [30], traffic shaping [111] and reservations in order to meet these issues [1–3, 5, 6, 27, 44, 45, 75].

Real-time Requirements Some tasks need real-time data communications, which means a minimum amount of bandwidth that is continuously available needs reserving [27]. An example of this would be a conference call in which breaks in the amount of data transmitted cannot be tolerated.

Reliability This is much like the case for resources, not only can resources prove unreliable but network connections can also fail or otherwise under perform.

2.3.2.4 Scheduling Requirements

Scheduling is a key requirement for quality of service, not only does it have to take into account the variations in tasks, resources and the network capacity, it has its own issues. Factors that are linked to scheduling are listed below:

Task Duration Estimation Advance reservation requires estimates upon how long a task is going to run for, it is quite common for these predictions to be poor [72, 98, 167]. This can lead to reservations lasting for insufficient time for the task to complete or for reservations to last for too long.

Preemption Tasks may have the possibility of being paused or moved to a different resource midway through execution. This allows for the possibility of re-optimising the schedule without necessarily losing the work currently having been achieved. It also means the Grid may recover from errors without loss of substantial amounts of processing via checkpointing. This however mean files would need to be moved in order to continue the process on another resource and the correct trade off needs to be determined.

Schedule Fragmentation It has been shown that when advance reservations are in use the overall throughput of a Grid can drop [50, 71, 72]. This is caused by the fragmentation of the schedule where tasks cannot fill the gaps between reservations causing the space to go unused.

2.3.3 QoS Provision Mechanisms

In this section discussion focuses on the various techniques that can be used to provide reliable Grids that are resilient to failures and can provide computed results on time.

2.3.3.1 Advance Reservation

Advance Reservation is the process by which resources may be booked in advance for tasks to complete their work. These bookings ensure resources are reserved for execution and in doing so QoS parameters such as a tasks runtime can be projected in advance.

Advance Reservation is a key technique for offering guarantees upon the response time of submitted tasks and has been described as been expensive in terms of performance in a system that does not support advance resource reservation to offer such guarantees [98]. There is however a trade-off between QoS guarantees and performance, as the stronger the QoS guarantees that are made the lower the overall system performance [98]. This is largely because gaps form between the reservations that cannot be utilised lowering efficiency. Several authors have proposed scheduling algorithms that consider such workload fragmentation, with the aim of mitigating its negative effects [44, 45, 50, 71, 72, 184].

A substantial requirement of advance reservations is that it tends to rely upon, user-estimated execution times. These estimates are often error prone and execution times for applications may vary from one run to another [72, 98, 167]. This leads to under use due to over estimation of execution time or overrun if under estimated [169].

2.3.3.2 Check Pointing and Replication

Checkpointing is a process by which the state of the Grid can be saved into stable storage so that when failure occurs the saved state may be restored [4]. The main aim of checkpointing is therefore to increase reliability. In large scale systems this is important because the rate of failure increases with the system size [33]. Checkpoint can also be used to assist task migration between machines with the use of virtualisation, with the benefit of preserving work already computed [11, 19, 20]. The virtual machines (VM) are used in such a situation to provide a homogeneous environment capable of supporting such whole-scale migration of task [19, 20].

Replication is a simple process to increase the reliability of a task submitted to the Grid. Tasks ran in duplicate, can be stopped once at least one of the replicas completes. This mechanism is of course wasteful of resources but ensures high priority tasks get completed.

In distributed systems that involves messages passing, checkpoints are required upon several machines, in order to record fully an applications state. This creates the need for a recovery line which is a set of checkpoints that provide a consistent application state across all processes [4]. They are two main forms of checkpointing in distributed systems coordinated and uncoordinated, which are described as follows:

Uncoordinated Checkpointing: means processes autonomously take checkpoints, without regard for other processes. The can lead to checkpoints being wasted if messages are passed between programs because there is no guarantee of consistency with the other processes' checkpoints. This means multiple checkpoints have to be maintained by each process because the latest checkpoint for any given process may not be consistent with other processes' checkpoints. When restoring the systems state it is then required to roll-back to a consistent copy of the system state, thus potentially losing work already computed [113].

Coordinated Checkpointing: means all processes collectively record a consistent global state/snapshot together thus no checkpoint. This has become the preferred way of checkpointing by the high performance massively parallel systems community [113]. It however be difficult to coordinate this effort efficiently.

In [4] they concluded that checkpointing algorithms should exhibit little rollback propagation (i.e. avoid wasting of checkpoints) and that piggybacking of control information on every message, as these were the sources of the highest overhead. Coordinated checkpointing protocols, were seen to be better than checkpointing induced by messages as they satisfied these conditions. They also stated that the protocols behaviour was highly sensitive to conditions like error rate, checkpoint time length etc so there was no perfect checkpointing protocol.

There exists two main strategies for implementing checkpointing, these are sequential and forked checkpointing. In sequential checkpointing the process suspends normal activity in order to save the current state, while in forked checkpointing, the process forks and one fork carries on while the other records the current process state [4]. The strategy chosen affects the overall impact upon the Grid application. Forking allows the running program to run continuously thus reducing downtime, but can led to applications slowdown and is more complicated to implement compared to sequential checkpointing.

Checkpointing and replication always adds overhead to the Grid so there is a trade off between the effort lost due to the strategy used and the benefit gained. Checkpointing for example when ran sequentially adds delays to the completion time of a task, due to the pauses needed for checkpointing, but should the task crash some computational effort will be saved and will not need to be recomputed. Replication ensures resources are more heavily utilised so this means resources will get wasted.

A selection work covering both checkpointing and replication is presented below:

Highly Predictable Cluster for Internet-Grids (HPC4U) [90]: This project was aimed at providing a resource management system that provided checkpointing facilities. This followed the principle that if the underlying local resource management software did not support QoS then nor could the Grid, following the argument that focusing only on Grid middleware was not enough. HPC4U was then used to support SLA-aware Grid middleware, such as AssessGrid [18].

Work Queue with Replication (WQR) [162]: uses replication as a means to both increase the chance that one of the replicas completes and to increase the chance of placing work upon a fast compute resource, where no information about CPU speed is known. This presents a trade off between wasted cycles and the chance of finishing quickly given no information about the resources.

Work Queue Replication - Fault Tolerant (WQR-FT) [13]: is a similar algorithm where tasks are replicated and checkpoints are made to ensure that if a replica crashes that its work is not lost and in [11] WQR-FT is extended so that checkpoints are used to make sure that replicas do not have to commence work from the beginning of a task every time a new replica instance is created.

2.3.3.3 Modelling Grids

In some Quality of Service literature modelling has been used to make predictions about the future environment. Nou *et al.* for example describes modelling as mandatory. ”To prevent inadequate QoS, resource managers must be able to predict the system performance for a given resource allocation and workload distribution” [134]. The modelling of Grids has also been shown in an economic context and modelling of future workload profiles has been performed in order to accurately predict cost [160]. This can be useful in future markets where access to resources may be sold well in advance of their actual use (further details may be seen later in Section 2.5.3.2).

These predictions generally rely upon a reasonable knowledge or best estimation of the workload profiles for a given job. Wieczorek et al. show that the schedulers performed better when estimates of task length are provided even when these lengths were inaccurate [194]. There are many papers that rely upon this knowledge [43, 49, 189, 193], others make provisions to recover from lack of knowledge [111, 134] and others are knowledge free [13, 162].

2.3.3.4 Scheduling

Scheduling is a means to place tasks upon particular resources within the Grid. The correct placement of jobs on resources can greatly enhance the Grids performance and ensure that quality of service is maintained.

Scheduling can be as simple as ensuring that requirements for executing a job are met e.g. CPU type, disk space and RAM available. Besides ensuring that the hardware requirements are matched it can also require the right software such as OS, libraries and other resources are available to run application. Scheduling however can also include more subtle factors such as:

- scheduling tasks to allow for fast communication in cases where this is needed, usually by ensuring they are placed close together.
- ensuring tasks that share the same input data are placed on the same or upon resources that are close together.
- ensuing data is not overly utilised, should data be shared between tasks then it is possible that a single copy of the data could cause bottlenecks when requests for it are made in close succession.
- Planning the use of replicas of tasks to ensure redundancy and guaranteeing that at least one replica finishes successfully.

- If replicas are not used then rescheduling may be used to mitigate failures. This requires that plans be flexible to change and adds complexity to the scheduling process.
- Ensuring that data transfers are minimized is also of high importance as in data intensive applications it can be a significant part of the execution time of an application on the Grid.
- Planning bandwidth usage to ensure that applications can have their data transferred in time. This can include real-time applications that require a minimum continuous amount of bandwidth available.

2.3.3.5 Admission Control

The process by which jobs are admitted to the Grid is a key factor upon QoS. In particular it changes the time a job has to wait in order to get access to resources on the Grid and hence can have a strong impact upon the completion time of jobs. In terms of economics admission control therefore can have a strong impact upon the ability to meet a given SLA [99, 148] and in particular can effect overall profitability [148].

2.4 Service Level Agreements

Service level agreements are a data structure and process that is used in order to create an agreement between two or more parties. This can be augmented by negotiation as it allows for the convergence towards an agreed set of parameters for a given service.

The agreement defines a relationship between two or more parties, with the aim of delivering a service within a defined context. This is achieved by agreeing upon the roles, rights and obligations of each of the parties. These obligations may include who is to pay whom at under what circumstances and for what service the payment is made. The agreement may include descriptions of functional requirements such as CPU and memory needed, but also non-functional properties such as time and cost guarantees [141].

This section first of all introduces where SLAs have been used within a Grid context and for what purpose. It then discusses the standards WS-Agreement [141] and WS-Agreement (Negotiation) [144], which are modern standards that are the most applicable to the negotiation context within Grids. The final part of the discussion on SLAs focuses upon the requirements for negotiation.

2.4.1 SLAs in Grids

In this section a description of where SLA's have been used within a Grid context is discussed. Two major implementation efforts for SLAs are WS-Agreement [141] a standard produced by the Open Grid Forum and Web Service Level Agreement (WSLA) [105] produced by IBM. Though WSLA has not been widely adopted and seems to be superseded by WS-Agreement to some extent now [174]. Another early implementation to note for SLAs is SLAng [120]. A notable survey of SLAs in Grids has been performed in [174] and is summarised here with updated discussion, on the subject.

VIOLA MetaScheduling Service (MSS) [123] This implementation is used for the scheduling of jobs. In particular it is used for the co-allocation of jobs on a UNICORE [154] based system. negotiation is used to find a potential start time for a job and request reservations for both network capacity and resources [174]. WS-Agreement is therefore focused upon the co-allocation problem rather than, specific QoS goals.

AssessGrid [18] and OpenCCS [18, 20, 146] is aimed at introducing risk management and assessment into Grids. The aim of WS-Agreement is to present offers that provide a price, risk of failure and compensation clauses for the computation of jobs [174].

ASKALON [194] is a service oriented architecture for the development and optimisation of Grid applications, primarily looking at workflow based applications. The use of WS-Agreement is for agreeing for a specified time the availability and capacity of resources, including capacity such as CPUs available etc, which is achieved by the client initiating the agreement process for resources. The aim is to maximise the resource and client utility [174].

Community Scheduler Framework (CSF) [201] is a meta-scheduler. It primarily uses agreements for co-scheduling across multiple resource providers and operates on a Globus toolkit [77, 181] 4.0 based infrastructure [174]. It is noted however that this though claiming to implement ws-agreement the authors in [174] were not aware of an implementation that actually did so.

AgentScape [137] is aimed at providing mobile agents access to resources. Agreements are used to compare multiple providers for the services they offer. Negotiations are used to request CPU time, communication bandwidth, memory, disk space and access to web services including a limited count of calls to web services [174].

CATNETS [69, 70] is a project to create an economic market for Grid resources, via a catallexy i.e. a decentralized "free market". WS-Agreement is used to submit requests to buy and sell resources within the computational market [174].

Job Submission Service (JSS) [66, 67] is a broker aiming to minimize the total time to delivery for jobs submitted to the Grid, working both with Globus Toolkit and the ARC middleware [62, 166] as well as support for the Maui scheduler. JSS provides as part of this job submission process the ability for reservations. These include as part of the agreement process reservations for: number of CPUs, duration, earliest start time, latest start time and a malleability flag, that indicates if the scheduler is allowed to use the range of possible start time options. The aim of this variable start time is to diminish the loss in resource utilisation caused by reservations. Though JSS supports such a mechanism its underlying local schedulers were not able to, so this capacity cannot be used [174].

The SLA@SOI [53, 164, 172] project created a SLA reference architecture [172] aimed at service-oriented infrastructures such as cloud. It also created a modelling language SLA* [104], which allowed SLAs to be described away from the context of a specific SLA language such as WSLA or WS-Agreement. SLA* has the advantage that it can be used to describe multiple layers in a commercial SLA scenario such as the infrastructure, software and business layers [53], thus achieving a generic way to describe the elements of an SLA.

mOSAIC [152] was aimed at building a SLA oriented cloud application for the management of security features and related user authentication and authorisation within an Infrastructure as a Service (IaaS) cloud provider.

The Business objective-driven REliable and Intelligent grids for real busiNess (BREIN) [65,138,176] project was aimed at making Grids more business centric, by using artificial intelligence, multi-agent systems, and semantics. Part of the business orientation was to use SLAs as means to negotiate QoS provision. The QoS provision was performed by a semantically enhanced resource allocator, which used WS-Agreement and elements of WSLA as a means of forming SLAs. It used an ontology to describe tasks and resources as a means of assisting the resource allocation process.

2.4.2 WS-Agreement

In this section the standard WS-Agreement [141] and WS-Agreement (Negotiation) [144] are discussed. Together they form the most relevant current standards for negotiation within Grids.

A diagrammatic overview of the document that describes an agreement is shown in Figure 2.1. The parts of it are described below [141]:

Context maintains the information about the parties within the agreement, the agreement's lifetime and optionally a reference to the template from which the agreement was created.

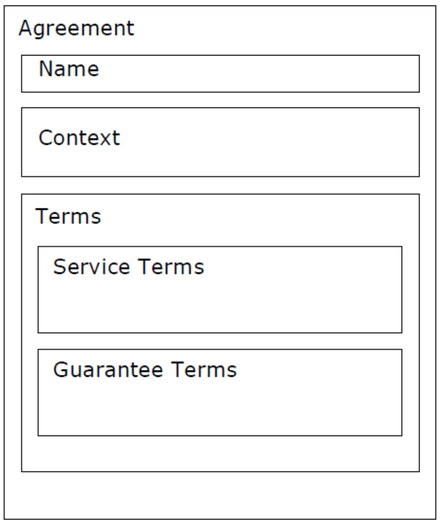


Figure 2.1: Overview of WS-Agreement Structure [141]

Terms Agreement terms define the content of an agreement and are largely domain-specific. Terms fall into two classes, namely service terms and guarantee terms:

Service Description Terms describe the functionality that will be delivered under the agreement. An example of this could be a description of a task to executed, which might therefore be written in Job Submission Description Language (JSDL) [139].

Service Properties These are definitions of values taken as measurements that can be used to enforce/help define the guarantees made in the agreement.

Guarantee Terms Guarantee terms define the assurances made within the agreement. They refer to the service description and in particular to the service properties and define thresholds on acceptable values of the properties. These are known as the service level objectives (SLO) . This might include constraints such as deadlines for completion or response times etc. Guarantee terms may also express the business value of an objective which indicates the importance of the SLO.

Agreement Constraints In agreement offers before the agreement is formed, constraints may be used to describe the various options and limitations placed upon the agreement. This is done by providers to ensure obviously incompatible agreements cannot be asked for, by laying down the rules on how the agreement template should be filled in.

Negotiation extends the agreement process by allowing convergence towards a middle ground that is acceptable to all parties in an agreement. This is achieved using WS-Agreement (Negotiation).

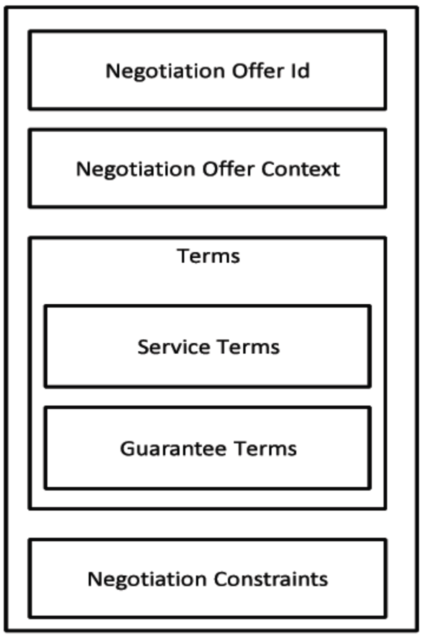


Figure 2.2: Overview of WS-Agreement Negotiation Structure [144]

The overview of its document description is given in Figure 2.2. The parts of it are then described below [144]:

The negotiation document adds to an agreement the following extra parts, that enable its capacity to perform negotiation:

Negotiation Id An id used to identify the negotiation offer in the negotiation process.

Negotiation Offer Context this specifies the lifetime of an negotiation offer, as well as indicating the position the offer has in regards to the negotiation process i.e. if it is a counter offer and to which offer it countered etc.

Negotiation Constraints These are structurally the same as agreement constraints but may change during the lifetime of the negotiation process in response to terms selected in the negotiation offer.

2.4.2.1 Flow of Actions in an Agreement

The first stage in an agreement process is when an agreement initiator requests a template from an agreement responder. The initiator then fills in the agreement template and then submits the offer to the agreement responder. The responder then can either accept or reject the offer. If the offer is accepted, the agreement is formed and the service it describes is executed.

This however is quite simple and gives only a request reply based system, the initiator can be none the wiser for the exact reason/s the provider had to reject the offer. This brings about the need for negotiation that brings both the initiator and responder closer to an acceptable middle ground.

The negotiation process differs in that the agreement initiator requests a template from an agreement responder, but limits this to negotiable offers. A negotiation offer is then created and sent to the responder. The responder then replies with a counter-offer. If this offer is acceptable to both sides then the initiator can accept the offer and an agreement is formed. If it does not then the process of offer and counter-offer may carry on until either the process is abandoned or an agreement is formed.

The agreement and its terms all have states, that indicate the progress made upon the agreement and are discussed in the next sections.

2.4.2.2 Agreement States

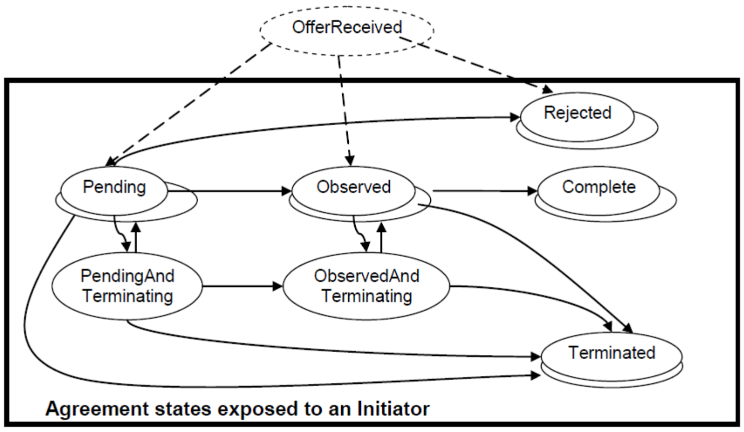


Figure 2.3: Agreement States in WS-Agreement [141]

In WS-Agreement an agreement has a state indicating the place the agreement is within its lifecycle. The possible set of states and their transitions are described in Figure 2.3. A summary of the specification's [141] description of agreement states is given below.

"Offer Received" is an initial transition state that is not exposed to the initiator so is represented by the dashed lines. The initial exposed states can be either *"Pending"*,

"Observed" or *"Rejected"*. Each state can also be extended with one or more sub-states for domain specific use.

Pending The offer has been made but has been neither accepted nor rejected.

Pending And Terminating The offer has been made and it has been terminated by the agreement initiator, before been accepted or rejected.

Observed This means that an offer has been made and accepted.

Observed And Terminating The offer has been made and accepted. Furthermore, a terminate operation has been issued from the agreement initiator and is being processed by the agreement responder. It is possible for a termination request to be rejected by the service provider.

Rejected This means that an agreement offer has been made and rejected.

Complete The offer has been received and accepted, and that all activities pertaining to the Agreement are finished.

Terminated The offer has been terminated by the agreement initiator and that the obligation no longer exists.

2.4.2.3 Service Term States

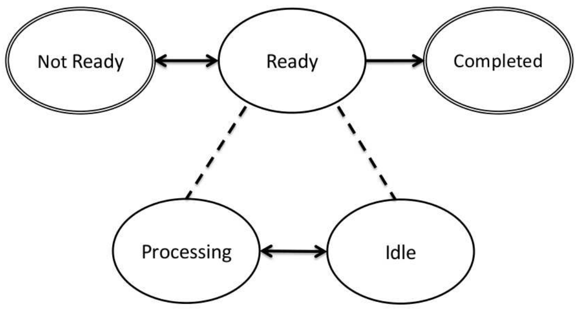


Figure 2.4: Service Term States [141]

In WS-Agreement an agreement has service terms describing the functionality that will be delivered under the agreement. These service terms have states, indicating the progress towards completing individual terms. The service term states and transitions are shown in Figure 2.4. A summary of the specification's [141] description of service term states are given below.

Not Ready This indicates that although the agreement has been formed the service denoted by the service term cannot be used yet. This also acts as the default entry state of an agreement. This could be seen for example in cases where resource reservations have yet to be established.

Ready This means this term of the service level agreement is ready to be acted upon, it has two sub states, idle and completed.

Processing This means this term of the agreement is actively been worked upon.

Idle This means this term of the agreement is not actively been worked upon.

Completed This means the term of the agreement has reached a stage of completion. It does not express whether the term was successfully met but merely indicates after all terms reach this stage that the agreement should proceed to its closing stages.

2.4.2.4 Guarantee Term States

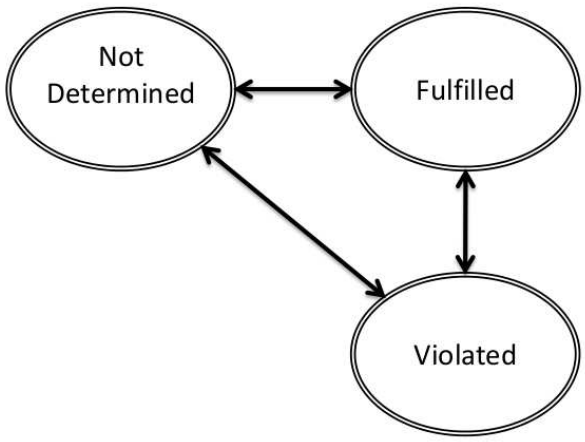


Figure 2.5: Guarantee Term States [141]

In WS-Agreement as shown in Figure 2.5 the guaranteed terms may start in any state. In terms of meeting deadlines this could well be expected to be *"Not Determined"*. This is because the temporal guarantees made will only be violated once the deadline has passed.

Fulfilled This will be when all tasks of the job have had their result sets returned.

Violated This will be for when the due date and deadline is passed. The penalty for passing the due date will be expressed as part of this term.

Not Determined As the guarantee is dependent upon work completing by a set time this will be the default state for temporal guarantees. It may eventually be the case where milestones for certain amounts of work to be completed during the agreement will be added, in which case these milestones could be evaluated earlier then the completion of all work submitted.

2.4.2.5 Negotiation States

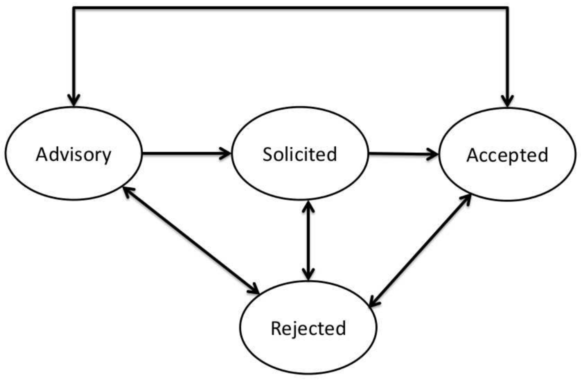


Figure 2.6: Negotiation States [144]

The negotiations also have states, these are shown in Figure 2.6.

The negotiation process is WS-Agreement (Negotiation) is non-binding so when offers are made there is not necessarily a need for an agreement to form. Only at the agreement formation stage are obligations to both parties conferred. The states that represent the flow of events in a negotiation are described below [144]:

Advisory State There are offers that are yet to be able to be used to form an agreement. They usually contain unspecified items, that have yet to be filled in, which means further negotiation is required.

Solicited State The solicited state is used to create instigate final convergence towards an agreement. A participant in the agreement can set an offer to this state, so that any counter offers may be accepted as is. The possible following states are therefore the accepted or rejected states.

Accepted State This state indicates that a participant accepts the negotiation offer. This therefore means that the negotiation offer specified is complete and no further negotiation is required. An offer in this state can be used to form an agreement, though additional rounds are possible, as is the ending of the negotiation without an agreement.

Rejected State If a participant wants to reject a negotiation offer this state is used. This means it cannot be used to create an agreement. It can still however be used to continue the negotiation process by addressing the reason the offer was rejected.

2.4.3 Requirements for Negotiation

In many negotiations only two parties exist within the negotiation, the initiator and the responder. The initiator in such a case can be seen to be asking for a service while the responder is considered to provide such a service. The requirements for such a negotiation are therefore discussed next.

2.4.3.1 Negotiation Initiator

The initiator of a negotiation must be able to provide a mechanism to:

- construct an offer and to decide upon its requirements, which may in part be taken from the end user.
- be able to change the requirements if the negotiation responder cannot fulfil the request. Though the agreement responder may provide negotiation constraints or agreement creation constraints as a guide to the initiator.
- monitor the agreement and ensure it is satisfied with compliance. This is important as if major terms of the agreement cannot be monitored or watched over by a trusted third party then the initiator is at the mercy of the responder.

Additional requirements may be required to be fulfilled if the initiator is using the SLA undergoing formation, as a means so that it in turn may honour an agreement/make an offer. The following therefore would be required:

- The aggregated offers lifetime must not last longer than the offers it relies upon to fulfil the promises made.
- Each offer has the prospect of liability, i.e. payment out, a requirement to provide a given service. This therefore means these needs must be at least balanced or over provisioned.
- If the process involves multiple rounds of negotiation then the previous offer is revoked, hence an aggregated offer may first be viable but not ideal, so having an additional SLA negotiation round, to construct an aggregated offer may render the aggregated offer unviable.

2.4.3.2 Negotiation Responder

Negotiation responders must be able to:

- provide a mechanism for generating offers. This may include making a plan to fulfil the offer but not perform any work. It may however have to reserve resources to be able to honour any offer made.
- act upon an agreement and execute the terms it has agreed to.
- monitor the agreement and to try and maintain compliance, should terms be violated/approach violation. It needs mechanisms in place to control compliance with the terms made in any offer and to have strong influence over the properties measured to test SLA compliance.

2.4.3.3 Negotiation Terms

The elements within an agreement need to be decided upon, this means:

- The service description terms, or the description of what needs to be achieved/asked for in the agreement needs deciding.
- The properties that are going to be monitored and used to decide if the SLA succeeds or fails needs deciding upon.
- The guarantee terms, the description of the thresholds on the properties being monitored, i.e. if $X < Y$ then term Z is broken needs deciding upon.
- These terms need to be able to be monitored sufficiency i.e. they need to be measurable. This is in order to detect violation of the agreement or otherwise, mitigating action may be required to be taken if this is the case.
- If adaptation is going to take place then the set of actions available must be able to influence the values undergoing measurement.

2.4.3.4 Negotiation Requirements due to Economics

Additional requirements are often needed given economic scenarios. This stems from anything that deals with cost and may incur liability itself through an agreement, requires it to follow economic principles.

- A way to resolve its liabilities:

1. if a negotiation responder agrees to something and has the chance for expenses, such as compensation, then it must have a revenue stream to offset its requirement to make payouts.
2. if a negotiation responder has to pay compensation then it needs to be able to payout reliably, a reserve of money would be required for this in order to generate a buffering effect or it cannot make an outright loss on a given SLA and therefore can only offer a discount.

- Participants must have incentive to participate in the process i.e. gain a service or gain money [130, 155].
- The mechanism must break-even at the very least, any long term deficit means the mechanism would need an outside source to remain viable [130, 155].

2.5 Grid Economics

In this section the concept of Grid economics is introduced. It starts by giving a general overview of this concept and discusses the benefits it provides. The discussion then goes on to basic concepts and requirements for the Grid economy, followed finally by the different methodologies used to implement the economy.

2.5.1 Overview of Grid Economy

Economics in Grids has been studied since the early days of the Grid, principally due to the advantages it offers. The reasons given for economic approach are therefore discussed next:

Using economic principles allows research to be drawn from the field of economics, and is useful in predicting emergent behaviour from using such models [198]. The concept of efficiency is also well defined in economics and although different from typical computer performance evaluation [198].

The use of economics is intuitive and the assumptions upon which it is based give familiarity to users, given most people understand fairly quickly that increases in resource price will diminish demand for it [198].

It is also in keeping with the Grid, in that ”Economists describe market participants as competing for limited resources and coordinating themselves through pursuance of their own goals.” [131]. This is in keeping in terms of both the limited resources available and also in that different elements act in their own interest, which gives some notion of a dispersed nature of Grids. Turning to markets can be seen as a traditional way of dealing with scarce resources. Buyya et al. [37] in this regards state that markets helps in regulating the supply and demand for resources.

Grids are often on a very large scale, which gives rise for the need to manage such a system. Economics provides a way to create self-organisation which has been described as going hand in hand with markets [131]. In regards to self-organisation and scale markets can help in building a highly scalable system as the decision making process can be distributed across all participants of the market [37].

Market based approaches have strong similarities with well established methods of resource such as proportional share. Markets however offer greater flexibility in resource allocation and can be used to limits the possibility of abuse of the system, by users constantly trying to unfairly overuse the resources [118]. Simple value oriented approaches like proportional share relies upon users indicating their priorities for jobs, but give no incentive for users to report their needs truthfully, market mechanisms however provides a way to achieve this [130]. The incentives for users to reduce their priority can be derived in markets from ensuring users incurring a lower expenditure for doing so, which aids in solving time critical problems first [37].

Markets can also be described as assisting user orientation. This is because participants can make their own decisions upon how they maximize their utility or profit, as it provides a simple means for offering differentiated services for applications [37]. This can assist commercialisation and wider spread adoption of Grids in that the pure selling of physical resources, i.e. CPU hours is not relevant to enterprise customers who have deadlines to meet and are not always sure of job resource requirements in order to meet a given deadline [130].

Markets can also be useful for the commercialisation of Grids and their wider adoption. It can offers incentives for resource owners to contribute by charging for resources [37]. It has been argued that Grid computing has not yet been adopted in commercial settings due to the lack of viable business models [130]. This also extends to the quality of service provided as uncharacterized, unguaranteed resources have no value and even poor QoS provided it is precisely characterized can have value [106], therefore so long as the correct model is used to characterise resources they can gain commercial value. Kenyon and Cheliotis [106] indicate that the building of an economically functioning market for Grid resources/services will help diminish commercial resistance to Grids, while also stating that best effort, or queuing priorities, are insufficient to support wide-scale commercialization.

Finally it can assist in issues regarding comparison making, in particular in regard to decision making, as it provides a common assessment mechanism for comparing conflicting users requirements in monetary terms. This common mechanism for comparison extends to resources in that it offers uniform treatment of all resources, even if they have different properties such as CPU speed, memory and storage space available [37].

2.5.2 Economic Concepts

In Chapter 3 an economic model is developed for the Grid, as part of this it is useful to understand what properties would be desirable in such a model. This understanding also assist generally in regards to understanding the Grid economy. A list of desirable properties for Grid economic models are therefore listed below:

Allocative Efficiency [130, 155] A central goal of an economic model is the efficient allocation of jobs to resources. A key part of this would be in achieving pareto optimality. Pareto efficiency is a state that occurs when an allocation of jobs to resources exists, where there is no other allocation, that makes at least one participant better off without making at least one other worse off. If utility is transferable among all participants, a mechanism that maximizes the sum of individual utilities is considered to be allocative efficient [155].

Incentive Compatibility/Truthfulness [130,155] This is a general requirement for participants in the system to have a preference towards acting truthfully, this is especially in reference to reporting their relative worth of tasks and resources. A situation where participants have an incentive to untruthfully report their preferences should be avoided [155] as it complicates the market mechanism. A market mechanism can be described as incentive compatible if it is within participants interest to reveal private information that the process requests (e.g. such as the maximum amount they are willing to pay), though it is not strictly necessary but it does speed up the market mechanism [198]. An example of this can be seen in a strategic participant in a market, which continually undervalues resources in an attempt to lower prices, other participants would have to be resilient to such a strategy, should this action be prevalent in the system. This concept can extend to providing poor evaluations such as in the English Auction. English auctions tend to provide overestimates of the price, as bidders try to outbid each other and the bids then straggle the actual cost so the highest bid is always overly high [198].

Individual Rationality [130, 155] This principle requires that there should be benefit for participants to take part in the market mechanism by receiving a greater utility.

If an element of the mechanism gains no benefit, either utility or financial reward then there is little incentive for its participation.

Budget Balance [130, 155] A computational market should 'merely' redistributes the payments among the participants in the market. Funds should therefore neither be added to from outside the system, nor removed. This is because, if any one participant continually makes a loss, it must be subsidized by some outside source, making it infeasible in the long term. Hence the sum of all trades given all participants should sum up to zero. The principle in a non-closed system still applies and requires participants to not require continual subsidization.

Computational Tractability [130,155] Any system that is performing the resource allocation must be in itself efficient. It must therefore not be so complex that computing which allocations that should be made takes a long time.

Perfect Competition [198] This is a situation where all participants both consumers and produces are price takers. This generally requires that no one participant can sufficiently influence the market such that the stability of prices is risked. This is important when participants are expected to work for themselves and are inherently going to act in a greedy fashion. The concept of "Tragedy of the Commons" [88] shows how scheduling decisions made to maximise individuals profit influence upon the entire system and its efficiency and that there is a tradeoffs between individuals efficiency and social welfare/global efficiency [37]. This property therefore lends itself towards allocative efficiency.

Price Stability Stability of the price is an important requirement in terms of fairness to users [118] and also in order to maintain the allocative efficiency/scheduling stability of the market [198]. If a particular participant hoards currency on the Grid and then suddenly releases a large quantity of that money then demand is likely to rise as is the resource price. The large release of money means one user can cause starvation effects upon others and users, which can be seen to be unfair [118]. Essentially fairness requires a user's budgetary endowment should be translated into a corresponding share of the infrastructure [186]. The need for fairness means if users budgets are periodically replenished it must be sufficiently controlled. This avoids two situations one ensures the currency retains its value and cannot be spent frivolously thus damaging any requirement to report the true relative worth of jobs [34], while the other is the hoarding situation where one consumer can begin to dominate the market [118], thus losing perfect competition.

Price Equilibrium This is closely related to price stability and regards the degree to which prices are fair. If the market is in equilibrium then the relative value of trades on the Grid for resources is accurate. If the equilibrium cannot be reached the pricing cannot be trusted [198] and the budgetary endowment of users is less likely to be translated into a corresponding share of the infrastructure [186]. The price essentially is required to track demand or otherwise not all the profit or utility possible can be realised [118]. In regards to price equilibrium the general requirements are summarised well by Wolski et al. as [198]:

1. ”The relative worth of a resource must be determined by its supply and demand”.
2. ”The price of a given resource is its worth relative to the value of a unit called currency”
3. ”Relative worth is accurately measure only when market equilibrium is reached.”

Structured Penalties It is common to spread the losses made between two parties when something goes wrong. Penalties however are a form of loss where one side takes predominately all the loss, be it in terms of utility or monetary loss [196]. Penalties are intended to incentive participants in the market from acting against the interests of other parties within a negotiation/agreement. They however can led to a moral hazard, whereby one participant can impose a bad operation point on the other while largely being insulated from the consequences. An example of this would be if the load was purposefully underestimated for a Grid application, when the provider would have to pay out if a deadline was met. The client therefore is required to make a loss should they mislead the provider.

Self-Organisation It is useful to have a Grid infrastructure that determines the price and resource allocations automatically. Economics can be used to create markets that perform this self-organisation. In fact self-organisation and markets have been described as going hand in hand [131]. The main characteristics of the free markets that provides this self-organization and in particular that of the participants are [131]:

1. That they must act as utility maximizers.
2. That their strategies should subjectively weigh and choose preferred alternatives in order to reach maximum income or utility.
3. That they should have access to markets in order to exchange price signals, wrapped in supply and demand offers.

Now that key principles of Grid economics have been discussed the approaches that may be used to establish a market will be discussed in the following section.

2.5.3 Economic Approaches

In Grids there are various different ways to establish an economic market for control of the Grid. In this section the principle ways of performing this are listed. The first area discussed is that of the popular [156] auctions mechanism. The second mechanism is that of the commodity market, this usually acts as a more centralised way of determining the market price. This is split into several different model variations: namely: flat price, supply and demand and posted price. The final methods discussed are a bargaining model, a tender contract model and catallexy.

2.5.3.1 Auctions

Auctions can be used in Grids to auction off resource time to users. It therefore acts as a price setting mechanism that determines the user whom is most willing to pay for a given resource. In doing this it achieves some notion of determining the work that is most valued and can hence prioritise the computation of such work.

Auctions by their nature have a particular advantage in that they can be easily decentralised. An important consideration when using auctions however, is in regards to the overheads involved, such as the communications needed and if an auction type introduces any point which acts as a bottleneck, such as a co-ordinator/auctioneer [40]. Auctions can also be compared to commodity markets, where there can be a higher cost for fine grain valuations [186]. Auctions can hold many different formats, with all have different overheads. The types of auctions that are commonly used [40] are discussed below:

English Auction Bidders are expected to increase their offers price in order to exceeding other offers. This is repeated until no bidder is willing to increase their offer any further. The auction then ends and the highest bidder wins usage of the resources at the highest price bided [37, 40]. These auctions suffer from holding several rounds until the final price is reached. This therefore can led to significant computational effort to determine the correct allocation.

Dutch Auction This acts in reverse to an English Auction. In this case the auctioneer starts with a price and continuously lowers it until one of the bidders accepts the price offered. A reserve price is then used to allow the auction to terminate an auction when the price gets too low [37, 40].

Double Auction In this model ”buy orders” (bids) and ”sell orders” (asks) may be submitted at any time during a trading window. If at any time there are open bids and asks that match or are otherwise compatible in terms of price and resource requirements, a trade is executed immediately [40]. In this fashion it is similar to the Condor ClassAds [179] mechanism, which has been considered by some to be primitive [127], though it has also been described by some as having ”high potential for Grid computing” [40]. Double auctions are a popular type of auction, due to their ability to handle a large amount of participants while producing a relatively small overhead compared to other auction types [86].

The orders in this auction are ranked in the matching process, in order to generate demand and supply profiles. From the profiles, the maximum quantity exchanged can be determined by matching ”sell orders” (asks) (starting with cheapest and moving to the most expensive) with ”buy orders” (starting with highest price and moving down) [40].

First-price sealed-bid Auction In this auction each bidder submits a single bid, in a blind fashion, i.e. without knowing the value of competing bids. The highest bidder then wins the and pays the price of their own bid [40]. As with any auction mechanism the users are price setters, however it becomes difficult for a user to make an informed decision about defining a reasonable price to have their jobs computed, a blind auction exacerbates this fact.

Vickrey Auction/second-price sealed-bid The Vickrey Auction [190] is similar to the First-price sealed-bid auction however the highest bidder pays the price provided by the second highest bidder. The reason to take the second price can be explained as part of the Vickrey-principle, in that the payment of each winning job amounts to the lowest willingness to pay, that would have been required to be reported in order to still win the auction [170]. In doing this it gives a fairer price to the winner, as the estimated values by the bidders are likely to spread across the real value, meaning that the user is likely to overpay for any given resource. Vickrey auctions have also been found to achieve efficient allocations as it creates an incentive for truthful biding [156].

Auctions are a popular economic model in Grid based scheduling and most are studies use this perspective [156]. The nature of auctions however creates interesting challenges, which will be discussed next:

Winners Dilemma: The Winners Dilemma [180] indicates that given any auction it is likely that the average bid will be significantly less than the value of the auctioned resource due to bidders risk aversion and that the winning bid will exceed the true value of the resource. It therefore means the true resource price is unlikely to be found.

Complexity and Multi-Item Auctions: Auctions are considered to be simpler than the commodity markets and are popular because of this simplicity. They are however difficult to analyse globally and can arise to NP-complete problems. An example of this is combinatorial auctions (multi-items) in which the search for the optimal solution is NP-Complete [198].

Commercial Suitability: Auctions do not always lend themselves to commercial suitability and have been described by [115] as not always being suitable for enterprise wide Grid deployments. This claim is substantiated by indicating that auctions would lead to competition between multiple divisions of the same organization. The practicality of such internal competition would therefore need to be realised and might not lead to the obtaining of the greatest added value of Grid computing in the business.

Bidding Round Duration and User Burden: Auctions can impose significant burdens upon on users, they can potentially suffer from the burden of frequent interactive bidding, or given infrequent bidding suffer latency effects in the acquisition of resources [119]. There can also be seen to be a trade-off between commodity markets that price resources more generally and the finer grain pricing that auctions provide [186]. Auctions can provide the opportunity for fine-grain pricing which is not as practical in the centralised commodity markets due to increased communications cost [186]. The greater granularity can however add burdens on users by adding more complexity into the market.

Price Determination and Fairness: If users are asked to make bids, then this needs to be guided. They will need an understanding of the current market prices and what is reasonable. This therefore requires infrastructure to advise them of current market conditions i.e. by publish summaries of recent contracts, should there be a sufficient volume of similar auctions [99]. The effect of users having to determine the correct price may be seen in the early stages of creating an auction market, before price stabilisation [186]. The auction market also has to adjust again during sudden changes to the market though remains comparable with commodity markets [186].

Price Stability and Market Equilibrium: In Wolski *et al.* it is stated that "auctioneering is attractive from an implementation point of view but that it does not produce stable pricing or market equilibrium, and that a commodity market performs better from the standpoint of a Grid as a whole" [198]. Though a following study [186] shows that price stability can be achieved, in their study of Vickrey auctions and commodity markets, with them indicating that using fairly simple bidding logic to obtain stable average prices that track supply and demand. A difference that Vanmechelen *et al.* highlight as a potential cause of this difference is that their study did not need to co-allocate disk and CPU resources as had been the case in Wolski *et al.* Achieving the correct price is important and thus the correct type of auction needs to be used, for example first price auctions favour the providers, while Vickrey auctions favour the users and double auctions favour neither [84]. Vickrey auctions however create incentive for truthful biding [156], which simplifies the bidding process.

2.5.3.2 Commodity Markets

Commodity markets may act for the sale of goods (such as time on a compute resource) for either immediate or later use. They hence can be broken down into two types of market, the spot price and the futures market:

Spot Price Market This market is characterised by the sale of goods, both perishable and non-perishable with little time passing between the original sale and the use of the resource [9]. In this respect they can be used to fulfil the need for the immediate provisioning of resources [187]. They are also known as the "cash market" or "physical markets" as purchases are often settled in cash at the current price on the market, as opposed to the price at the time of delivery of the goods/resources [9]. They are often used for resources that are perishable or not easily storable such as electricity [9].

Futures Market These markets however trade in resources that may be delivered some time after they are bought. It allows consumers to avoid the bidding exposure and price risk (of fluctuation), that is associated with a spot market and can be used to complements the spot market [9, 187]. The futures are contractual obligations by buyers and sellers to make a transaction in the future for a particular resource at an agreed price [9]. Delivery of the resource therefore occurs sometime after the sale is made. A centralised broker can be used to provide such a market. Such a broker would pairs requests for resources with declarations of resource availability at set intervals [187], thus it cannot be used for resources that are needed immediately.

Buyya *et al.* [40] indicate commodity markets to be where resource providers act as price setters, publishing their prices ready for consumers. Consumers then select the most appropriate price for their resource requirements, hence acting as price takers. This differs slightly from the purely economic oriented view presented in [9], in that it focuses more upon the communication patterns of the underlying Grid infrastructure. The commonality is in the centralised nature (as per [198, 199]) of how prices are determined and the aim of optimising the market environment as whole, as opposed to a single auction.

In [118] commodity markets are placed into two categories either: flat price or dynamic supply and demand models as part of an argument towards dynamic pricing where suppliers must monitor demand and react to in relative to the available resources, in order to avoid unrealised profit or utility. In addition to this a posted price model [40] is also listed here as part of the commodity market, due to its similarities. The different variations of commodity market models are therefore discussed next:

Flat Price Model The flat price model presents a single price to the market for resource usage that does not adapt to usage patterns. In [118] such a model is criticised as it does not lead to sufficient control. The lack of price fluctuation means the effects of a true market are not realised and can lead to either:

Unrealized Utility: A situation where resources are idle as they are overpriced.

Unrealized Profit: A situation where resources are under priced so hence do not make as much profit as is possible.

Supply and Demand Model This is a variation where the suppliers dynamically alter the price in response demand, as a means of controlling it. The aim of this is to change the price to achieve equilibrium between the supply and demand. There are several ways at which such a market could be achieved, two of which are listed below:

- Buyya *et al.* [40] suggest a Walrasian Auction [186], though this requires perfect competition [198]. This ensures resource providers are not going to either undervalue or overvalue resources, which in turn can lead to unrealised profit or utility [118, 130].
- K-Pricing [23, 156, 196] K-pricing is to determine the price based upon the difference between the resource providers price and the amount the buyer is

willing to pay. It requires for a given amount of utility the buyer expressing their maximum willingness to pay and the service provider expressing for a given amount of utility their minimum sale price. The difference is then split, based upon a factor (commonly shown as 50:50).

Posted Price Model The posted price model is a variation upon the supply and demand model. It differs in that it allows for special offers to be issued by suppliers. The primary reason given for this by [40] is to attract new customers and to increase market share, by motivating users by offering cheaper resource time slots. Such offers may be used to smooth demand and the market price for resources e.g. by ensuring deals are linked to periods of lower demand. It is hence classified here as a commodity market due to its similarity and that it acts in part trading with units of resource-time that are potentially distant from the time of sale i.e. a primitive futures contract.

2.5.3.3 Bargaining Model

In the bargaining model [40] consumers are not just price takers, they are allowed to perform bargaining with suppliers. Bargaining for consumers may mean an attempt to acquire lower resource prices or longer usage durations, while providers may offer resource time slots at periods with lower demand. This method is generally employed when market supply and demand as well as service prices are not clearly established [40]. It is however no very economically efficient and produces high communications overhead [86].

2.5.3.4 Tender Contract

In this model otherwise known as Contract Net Protocol [40, 86], resource consumers place their job requirements out to tender. Resource providers then make offers on the available work at which point the consumer selects the winning bidder. This can be viewed as an auction in reverse, hence the key differences with auctions are listed below:

- consumers are price acceptors (in auctions they set the price),
- resource providers are price setters (in auctions they are acceptors),
- bids are for work (in auctions resources are what is bid for)

2.5.3.5 Catallexy

The aim of a Catallexy is to have a decentralized environment in order to create a ”free market” uses economic mechanisms for self-organisation. Prices are expected to evolve dynamically over time, from the actions of economically self-interested agents participating in the system [34, 69]. The core principles of such a market are therefore [34, 69]:

- Agents work for their own self-interest and attempt to optimize their own profit.
- Agents do not have global knowledge; they can only act on information that they current have access to and hence can only make estimates about the possible alternatives actions available to them to maximise their income/utility.
- Changes in price will indicate if an agent looks for alternative source for a resource.

Examples of the Catallexy based approach may be found in projects such as: CatNets [69, 70] and CatNet [15] and the Middleware for Activating the Global Open Grid (MaGoG) [56].

There are several key areas to consider when considering a Catallexy based model [69]:

Service Discovery Mechanism The aim of a catallexy is to be distributed, therefore there is an aim to have no centralised components. This extends to centralised registries where buyers and sellers, could register themselves. The service discovery therefore has to be established differently and is restricted to two mechanisms for distributed discovery of resources:

unstructured discovery In unstructured discovery a search request is forwarded from a node to all of its neighbours with a given Time to Live (TTL) for the request.

structured discovery In structured discovery the search does not rely upon random query propagation, but instead calculates the closest known node to the requested service instance. This mechanism however relies upon distributed hash tables (DHT), which have been claimed to lack scalability in dynamic networks, as state changes lead to high overhead.

Bargaining Protocol/Price Convergence Strategy The bargaining protocol in catallaxy can be likened to that of auctions. Auctions provide a wide set of variations and care must be taken when considering the type of auction to use, as they have varying communication patterns that can cause scalability issues [37] as well as different economic properties, such as time efficiency, price stability and ability to maintain market equilibrium [86].

Combinatorial Auctions In order to gain multiple resources it may be required to have auctions for multiple resources at once. Eymann *et al.* [69] shows that solving the resource allocation problem in this case is an instance of a complex multi-attribute allocation problem, which has been shown to be NP-complete. This situation is particularly apparent because of interrelations of several markets, such as storage and compute i.e. to use one requires the other market [34]. Vanmechelen *et al.* [186] however state from a usage model point of view that combinatorial auctions, is one of the most attractive models available for the Grid problem.

2.6 Scheduling in Grids

In this section scheduling in Grids is discussed. This first of all covers aspects of scheduling and then discusses work in scheduling based upon set themes. This includes: the fragmentation of resource availability due to reservations, information availability in a distributed system, the attempts to create network based reservations for data transfers, taking advantage of shared storage, job composition and structure and finally how the scheduling is performed in terms of being either batch oriented or performed continuously.

2.6.1 Aspects of Scheduling in Grids

Scheduling in Grids tends to start with brokering and usually within a two tired hierarchy of brokers and local schedulers [117]. To illustrate aspects of scheduling in respect to brokerage Figure 2.7 is presented and demonstrates some of the design considerations of a broker. The aspects illustrated in this figure are then discussed next.

The architecture that is being used can either be under control of a centralised scheduler, that lends itself to classical scheduling techniques or distributed and thus requires peer to peer techniques such as used by CATNETS [69, 70]. Schedulers may also be assembled into hierarchies [117, 200] to assist with scalability issues.

Scheduling requires the matching of jobs to resources, this can either use static data, or dynamic [107]. The user at broker level may be asked to select a provider, this is a user led strategy for brokering. The decision process may also be guided by historical data.

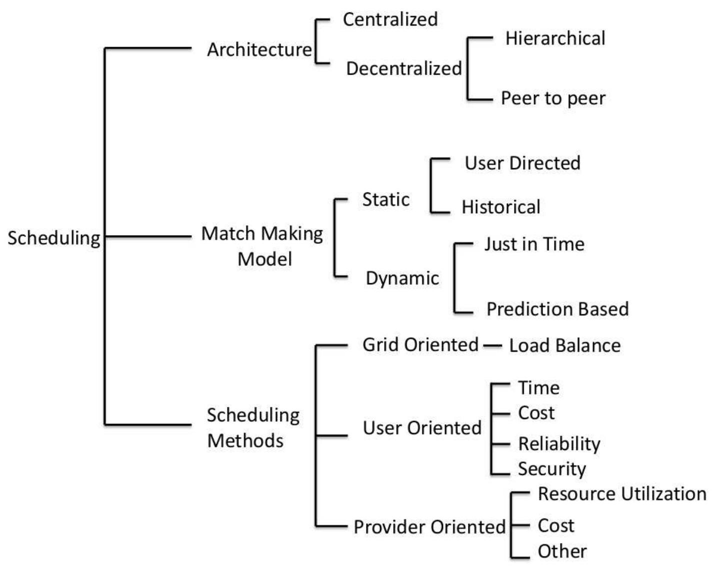


Figure 2.7: Scheduling in Grids from a Broker's Perspective [107]

Dynamic approaches use the most current information available and can take one of two strategies, either just-in-time allocations or prediction [134, 135, 169] based approaches.

The aims of the scheduling algorithm should also be considered, the objective function of a scheduling algorithm can be either user or system centric. In Figure 2.7 and [107] a third option of being oriented towards the Grid, is given with a general concept of load balancing. This can however be considered to be merely an aspect of system centric scheduling. The User's focus is often towards guarantees upon execution time, cost, or other factors such as reliability and security. This is contrasted by provider's aims which largely focus upon system centric measures such as a wish to maximise the utilisation of their resources and gain as much revenue as possible, from the provisioning of their resources to end users.

Scheduling in Grids often is very simple, but efforts focusing on scheduling often reflects aspects of the Grid problem, as presented by its dynamic and distributed nature. Efforts within Grids for scheduling are thus discussed next with a particular reflection upon what aspects of Grids is trying to be resolved. For general background however in scheduling and in particular queueing theory by Kleinrock [109] and by Harchol-Balter [87] can be recommended.

2.6.2 Schedule Fragmentation

Schedule Fragmentation can occur if advance reservations are used, it can lead to lower utilisation of the resources. Several authors aim to mitigate this issue [44, 45, 50, 184].

The focus in during scheduling becomes less upon the reservations but more upon the gaps between the reservations and the attempt to minimise small unutilisable fragments within the schedule.

Fragmentation can especially be seen when a job is scheduled in an idle period, between two existing jobs. It will create at most two new idle periods [51]: A leading idle period before the jobs starts and a trailing idle period at the end of the new job, but before the existing scheduled job is due to start. Scheduling strategies therefore exist to minimise both the leading (Min-LIP Min-leading idle period) and trailing (Min-TIP Min-trailing idle period) idle periods. Other strategies [51] exist such as best-fit, which minimizes the sum of the leading and trailing idle periods, as well as first-fit, which inserts tasks into the earliest feasible idle period, regardless of the sizes of the leading and trailing idle periods. These algorithms produce various different overall system behaviours such as first-fit offering relatively low delays to starting jobs, but at the cost of overall system utilisation in comparison to the other techniques mentioned. There is therefore a trade-off between system centric and user centric behaviour when considering this type of scheduling.

2.6.3 Information Availability and Dynamics of the Grid

It is not always possible to be able to collect good quality information about Grid resources, given the nature of large distributed environments [122, 162]. This can lead to not having information about the nature of the available resources, this can include information upon the CPU speed. Several algorithms aim to negate this issue and work without detailed knowledge. This can be achieved by running pilot jobs [121, 122] of a known size, or by running replicas of a given task and cancelling replicas once at least one copy of the results have been returned [11–13, 162].

In Grids failures are the rule rather than the exception [57], in that given the Grids scale and complexity there is likely to be a failing resource somewhere. It therefore means there needs to be strategies to counteract these failures. This can be dealt with by detecting errors and slowdown and then rescheduling [93,200] or by using replication and check pointing techniques [4, 11, 19, 20, 90, 113].

2.6.4 Network Oriented Scheduling

Data transfers in Grids must be managed, as the transferring of data within the Grid can cause substantial issues. In terms of guarantees made on completion time the transfers can account for a large proportion of the time allocated for the results of the application to be returned. It is also not always sufficient for schedulers or brokers to assume that the network capacity for scheduled work is always available. It has however been noted that designs of broker and schedulers can omit any mechanisms to deal with situations where network capacity is not available [27]. The reasons given for this behaviour is that applications tend to be run on a single-site and high-speed local area networks can negate much of this issue. The fact that distributed applications can also be expected to cope with asynchronous communication and that many applications do not require large amounts of data to be transferred across wide-area networks also assists [27]. In regards to QoS provision however it is useful to known the time it is going to take to complete a data transfer. The scheduling and the planning of network traffic can greatly enhance the performance of the Grid particularly in regard to timeliness. Various mechanisms are available to do this, such as hardware support for quality of service with differentiated services (DiffServ) [30] or flow-aware [47, 48] or traffic shaping [111] techniques.

Flow-aware techniques perform quality of service provision at the flow level instead of at the packet level [47, 48]. Instead of tagging packets and prioritising them as with DiffServ, packet headers are observed and a stream of packets with the same header attributes is managed using various prioritisation techniques. Traffic shaping can include techniques like leaky bucket that can be used in Grids [111] to ensure that a constant stream off data can be sent across the network by creating a bucket of data packets that are released at a constant rate. This ensures data transfers are more predictable and limits usage spikes, should too much data be attempted to be transferred packets are simply dropped and then resent at a later time. There are a variety of work from the literature that handle such networking issues which is discussed next:

General-Purpose Architecture for Reservation and Allocation (GARA) [3, 75] provides a uniform mechanisms for making QoS reservations for different types of resources, including computers, networks and disks [6, 27, 45] providing a guarantee that an application initiating a reservation will receive a specific QoS from the Resource Manager [6].

The network QoS in GARA is however limited and only works with a specific router, the Cisco 7507, as it requires usage of Cisco's Modular QoS Command-line interface to configure routers, as part of its establishment of differentiated services capability [6] though this constrain is common to any system that requires underlying hardware support for control of network resources.

GARA's network problems extend further to scalability when performing reservations across different domains, as GARA must exist in all the traversed domains [6, 45], which is unlikely.

Network Resource Scheduling Entity (NRSE) [27] acts as a network booking facility. An instance is placed in each network domain. The NRSE in each domain is responsible for receiving user service-requests, checking if the request can be honoured site to site and then issuing configuring instructions to the local network elements in order to provide the requested QoS. These QoS requests are sent in the form of a local service level agreements. NRSEs then perform a booking of the request in both domains. NRSEs in each domain then issue local instructions to enforce the requested QoS within the network, followed by monitoring the network QoS.

Grid Quality of Service Management (G-QoSM) [5,6] is a framework to support QoS management in Open Grid Service Architecture (OGSA) based computational Grids. It is a generic modular system that, conceptually, supports various types of resource QoS, such as computation, network and disk storage [45].

G-QoSM aims to provide support for: resource and service discovery based upon QoS properties; QoS guarantees at application, middleware and network levels and the establishment of service level agreements SLAs to enforce QoS.

The Quality of service is offered in a tiered system with three different levels of provisioned QoS: namely guaranteed, controlled load and best effort and supports adaptation strategies to share resource capacity between these three user QoS categories [45].

G-QoSM has three main components [5]:

- Application QoS Manager: negotiates SLAs with clients and passes SLA parameters to resource managers as well as performing SLA conformance checking.
- A Resource Manager: This is for example Globus toolkit and a registry service that is capable of recording QoS values.
- A Network Resource Manager: controls network SLAs so acts as a bandwidth broker. It also controls inter-domain communication and coordinates SLAs across domain boundaries. The NRM can also monitor the network to determine both its current activity and the correct network configuration.

Grid Network-aware Resource Broker (GNRB) [1,2] acts as an intermediary between the network and the Grid environment and aims at providing network resource management and network information retrieval facilities [1].

It provides facilities to aid scheduling in terms of network topology discovery allowing the networks topology to be discovered with network link information such as available bandwidth, delay, etc. This also includes a weighted topology service that provides the best path information as well. This information therefore allows for QoS provisioning. This QoS can be provisioned into categories such as: Premium services defining the peak rate, burst size and maximum latency available and better than best-effort service where a mean rate, burst size and mean latency can be specified [1]. This broker has however been criticised as been overly centralised within a given domain, hence offering a potential bottleneck to the system [45,48].

Grid Network Broker (GNB) [44,45] is a network aware Grid meta-scheduler. It aims at providing network QoS in both a single administrative domain and cross domains. It is claimed that GNB is the only broker that considers the network when performing the scheduling of jobs to computing resources [45]. The focus on the network within the broker stems from the fact that only paying attention to the load of the computing resource is not sufficient and that a powerful unloaded computing resource with an overloaded network could be chosen to run jobs, which decreases the performance received by users, especially when the job requires a high network I/O [45].

GNB has some obvious limitations given the limits of what is possible without total control of the Grids environment. It restricted in that only traffic generated in the network as a result of Grid jobs is considered [45]. GNB in been network aware also needs to have a global knowledge of the network's topology. It therefore needs providing with routing tables of all the routes within the same domain [44].

The VIOLA project is a testbed that has be used for various projects, one of which established a meta-scheduling service that provides co-allocation support for both computational and network resources. This is achieved through negotiations with local scheduling systems to reserve resources [45,192]. This meta-scheduling service has been implemented using the UNICORE middleware for job submission, monitoring and control [45]. Like in many systems however it is not always possible to reserve network resources if the network is under ownership of a different administrator [45].

2.6.5 Data Storage

The way data is stored on the Grid and can be accessed effects its efficiency. In Grids there is the potential to have shared storage. This can be reflected in the scheduling algorithms implemented. If they can take advantage of shared storage then related work that shares the same files can be transferred to resources that can access the shared storage, X-Suffrage [49] and Q-Sufferage [193] are examples of algorithms aiming to obtain such a benefit.

2.6.6 Bag of Tasks, Workflows and Divisible Load

In Grid scheduling tasks within a job can either be completely independent of each other and hence form bag of task applications [122], or they can have dependencies upon one another in which case they form workflows. The workflows [29,58,195,203] hence needs specialised algorithms that consider these interdependencies in order to arrange them as efficiently as possible.

Workflows are first built from templates that describe the flow of services to be called within an overall Grid job. This can then have specific service instances attached to realise the workflow, hence creating a workflow instance [58].

The workflow representations can hold distinct structural differences. This effects how the scheduling of the workflow can be perceived. They can be either [175]:

- Functional and Data-Driven, or Dataflow: where services that execute tasks are viewed as functions and their composition is specified by data dependencies between services i.e. the input of one is derived from the input of another.
- Imperative and Control-Based: in which a workflow is designed with services that act as primitive execution blocks and service composition is achieved using control primitives such as: sequences, conditional branching such as ”if statements” and loops such as ”for” and ”while”.

Workflows can be classed into two distinct types of models regarding how they handle data [195]:

- Task oriented - The workflow is represented as a graph. Tasks are represented as nodes and data transfers and control preconditions are represented as edges.
- Task and data transfer oriented - Where both tasks and data transfers are represented as graph nodes. This is important as this demonstrates the importance of dataflows within the workflow and ignoring this limits the ability to managed QoS in regards to completion time.

There are three distinct techniques used when scheduling workflows in Grids [29]:

- List scheduling - In this case each node of the graph receives a priority level. While there are unscheduled tasks, the highest-priority task that is ready is selected and scheduled .
- Clustering - In this class of workflow scheduling a clustering phase, occurs where all tasks are arranged into groups, aiming to reducing communication costs between tasks. In a second phase, these groups of tasks are then mapped to the available resources.

- Task duplication - The last strategy used is where tasks are duplicated, as a means to improve dependability, and the instance which finishes first sends the result to successor tasks in the workflow.

These techniques are further complicated when scheduling multiple workflows as the workflows create gaps in the schedule, such techniques are as follows [29]:

- Schedule each job independently, one after another, placing tasks into the schedule. This can include searching for gaps between tasks in the existing schedule, when placing tasks.

- Schedule the jobs in turns, interleaving parts of each job being scheduled, with the parts of other jobs in the schedule.

- Schedule multiple jobs together by merging the jobs into a single big one and schedule.

Not all jobs form workflows and form independent tasks. They may however have additional properties. Divisible Load [26, 102, 153] considers the way the work to be scheduled on the Grid can be divided and can lead to scheduling algorithms that can take advantage of the separability of the applications data. The most common form of load in Grids is *indivisible* [26] in which independent, tasks of different sizes, are submitted for execution without any possibility of dividing. Such tasks therefore have to be processed in their entirety upon a single resource. Such tasks have no precedence relations and give rise to bin-packing problems which are known to be NP-complete. Load in some special case jobs may however be divisible. This can either be *modularly divisible* [26] or *arbitrarily divisible* [26]. Modularly divisible means that it is a priori subdivided into smaller modules based on some characteristics of the current load or other system properties. Arbitrarily divisible means it can be divided up as required, as all elements in the data to be processed require an identical type of processing. These tasks may or may not have precedence relations, that have to be handled.

2.6.7 Batch vs Online Scheduling

Scheduling can either be performed as a continuous exercises as and when jobs arrive or it may be performed in batches with jobs arriving and waiting to be scheduled. Both strategies have their advantages and drawbacks. If scheduling is performed online then a near immediate response can be given, while scheduled in batches the jobs have to wait for the next round of scheduling. The benefit of batch scheduling resides in the fact that the scheduler can then make better allocations based upon total information about the jobs within that scheduling epoch, whereas online scheduling means a job may arrive soon after that cannot be served as well due to previous decisions made [200]. The batch scheduling strategy is also effected by the rate at which scheduling epochs are performed and a trade-off exists between waiting and then scheduling a larger set of jobs better and a more responsive but less efficient shorter time between scheduling epochs.

2.7 Summary

In this chapter the background to the work presented within the thesis has been given. Initially this starts with a definition of the Grid (in Section 2.2.1) and explorers its history (in Section 2.2.2). This includes a trend towards quality of service provision. This QoS provision is explored by discussing what effects QoS and how it can be provisioned for. The background then moves on to service level agreements, which is a mechanism for agreeing to a defined level of provision for a given service. In particular the Open Grid forum standard of WS-Agreement is discussed (in Section 2.4.2), followed by a discussion of the components required within an architecture to bring about negotiation (in Section 2.4.3). The concept of economics in Grids is introduced as another trend within the history of Grids. This trend aiming towards self-adaptive systems through market mechanisms. Economics is discussed initially as an overview (in Section 2.5.1 and then key concepts are introduced (in Section 2.5.2 and a discussion of the main approaches taken to implement economics in Grids (in Section 2.5.3). Mainly in terms of auctions, commodity markets, bargaining models, tender contract and catallexy. The last topic discussed in the background chapter is the scheduling that occurs in Grids (in Section 2.6 focusing upon the different problems the scheduling has focused upon.

Chapter 3

Pricing Policy

3.1 Introduction

In this chapter the broker's pricing policy that drives the process of obtaining time and cost guarantees for jobs is discussed. This policy is aimed towards providing an economic based system that distinguishes between jobs in terms of QoS which was discussed in the aims and objectives of the thesis in Section 1.4.

These aims and objectives are reiterated in Section 3.1.1, after which a review of related pricing models that focus upon cost and time of jobs in distributed computing is performed in Section 3.2.

The is then followed by the discussion of the ISQoS model in Section 3.3 and how it was designed to cope with the specific aims of the ISQoS pricing model.

The concept of simulation is then introduced (Section 3.4) and existing simulators are surveyed (Section 3.4.1). This is followed by the introduction of the ISQoS simulator in Section 3.4.2. The ISQoS simulator is a tool that was designed and built to help test the model. The simulator was initially validated via sweeping through parameter variations (see Section 3.5) in a static study. This parameter study also helped explore the model and brought about recommendations upon its usage. After the parameter study additional experimentation was performed by using discrete event simulation (see Section 3.6). Finally this chapter is finished with a comparison study between the ISQoS model and related work in Section 3.7. The chapter is then summarised in Section 3.8.

3.1.1 Aims and Objectives

The principle aims of the pricing model are listed below, they are derived from the motivating scenarios in Section 1.2, recalling the focus on high demand, time critical scenarios on a Grid with limited resources. The model that achieves these aims and objectives is tested in in this chapter in Sections 3.5 and 3.6 and also in Chapters 5 and 6 by experiment with the ISQoS Broker.

- Establish a service price for completing work on time.
- Provide a model that incentivises QoS provision in regards to both time and cost constraints i.e. bind economic and temporal factors together, so that if delays occur then less profit is made.
- Provide a mechanism to prioritise work, as it is expected that not all work can be accepted.
- To make the job submission process more user centric, avoiding resource centric approaches in the job submission process.

3.2 Related Work

In this section a survey of existing Grid pricing mechanisms that focus upon time constraints of scheduling is performed. We therefore list each pricing mechanism in turn and then summarise the models discussed at the end of the related works section.

3.2.1 First Price

First Price [54] considers pricing upon clusters, but remains highly related to Grids. The jobs maximum price is first set, by using a first-price sealed-bid auction [40], where each bidder submits one bid without knowing the others' bids. The highest bidder then wins and pays the service price indicated in their bid on successful job completion.

This service price has the potential to drop based upon the concept of slowdown. The slowdown being a comparison to what the completion time would have been if the cluster had been dedicated to the job. This therefore has significant deficits in meeting the objectives of ISQoS. Firstly user preference in terms of completion time is not captured well. It is only expressed in terms of resource cost and degree of acceptable slowdown. They use an immediacy concept to capture the users acceptable delay in multiples of the original runtime estimate. However, this assumes that the user will want the work completing as fast as possible and that the service price should diminish due to any delays caused by non-exclusive access to resources. It is therefore very system centric and does not express the user's preference for completion of the work.

This pricing model has a strong requirement for a reference system by which a notion of slowdown can be derived. This is difficult in a Grid as different providers will have different reference systems. It therefore makes the comparison between providers unrealistic.

Currency is provided to the bidders at periodic intervals into accounts that have a finite limit, as a control mechanism to ensure arbitrarily large bids are not made. In [118] the problems with users hoarding money is discussed along with predictability and issues such as starvation, which is reasonable justification for using a control mechanism that has upper limits on the account size.

3.2.2 First Reward & Risk Reward

The First Reward [99] and Risk Reward [99] pricing functions like First Price focus upon the minimum runtime and upon slowdown, so holds similar drawbacks. In [99] focus of the winning bidder is found via the use of a Vickrey auction [40], though this is not discussed further. This has the benefit over first-price sealed-bid auction as the highest bidder pays the price provided by the second highest bidder which is more likely to reveal the true price of the service. The recycling/replenishment of currency in the system, is also not discussed in detail. Given the areas that do and do not have focus in [99] the principal concern must be merely with slowdown.

In First Reward and Risk Reward once the minimum runtime is reached the job's price decays at a set rate, hence this is similar to First Price, but uses a rate factor instead to diminish the service price. The decay in the service price is determined for the i_{th} job as $\text{Price}_i = \text{MaxValue}_i \ \ (\text{delay}_i \times \text{decay}_i)$. This is system centric and sets the meaning of a job being on time as meeting its runtime estimate. There is therefore no focus upon a user's preference for completion time and as different providers could have resources of different speeds, the inter-provider estimates are unlikely to hold much meaning. The decay rate is also not user friendly in that a rate of decay is asked for instead of a final time by which the work is no longer useful to compute. In asking for a rate of decay it simply makes algorithmic details open to the end user and detracts from its usability.

The loss that is generated by the decay in the service price is also not required to be bound. This unlimited penalty is problematic in that pricing mechanisms should have properties such as budget balance and individual rationality [155, 156]. SLAs can also be seen as contracts and unlimited liability is impractical in a commercial sense [196].

The definitions are provided below [155, 156]:

Budget Balance: Mechanisms have to ensure their budgets balance. Long term deficits must be subsidised making them infeasible. In cases where penalties are without limit budget balance could never be guaranteed and is at severe risk.

Individual Rationality: The utility caused by participation in the Grid market has to increase. Hence in cases where penalties are without restriction in can severely damage this rationality. Risk Reward extends the First Reward pricing mechanism by adding the concept of present value.

Risk Reward uses a concept called *present value* as an assessment mechanism for jobs of different durations in relation to how they might delay future jobs. i.e. present value for the i_{th} job equals: price_i / (1 + (discount_rate $\times$ $\text{remaining\_processing\_time}_i$). Risk Reward in doing this considers that shorter jobs are less likely to delay more valuable/urgent task that may arrive in the future. It therefore can be used to favour smaller jobs, without altering the service price paid by the end user. This is a notable feature albeit not part of the pricing mechanism, in ISQoS as it will later be shown, it is considered to be an offer selection mechanism rather than an issue with establishing a service price.

3.2.3 First Profit, First Opportunity & First Opportunity Rate

In each of these models [148], jobs are given a utility function where the service price decays at a set rate until a fixed penalty bound is reached. In the experimentation performed in [148] decays start immediately at submission time, hence this essentially relates to slowdown with larger jobs been penalised more than smaller jobs.

Initially admission control is performed which tests if the broker's profit is likely to decrease, if this is the case then the job is rejected otherwise it is accepted. After acceptance jobs are required to be completed regardless of cost, i.e. there is no cancellation fee as in Aggregate Utility [17]. This therefore risks the broker expending large amounts of resources, rather than potentially cancelling a job.

Popovici *et al.* [148] uses resource providers that are separate from the brokering mechanism, hence multi-site Grid situations can be realised. Providers however offer predictions of the number of resources that are going to be available in the future, their prices and a probability distribution that this resource profile will actually occur in practice. A tuple of the form <start time, duration, resources, price> is hence used to represent this availability. The policy of sending a broker so much information seems misguided. A policy of having providers merely respond to offers for completing work and not expressing their resource availability to the broker, seems wiser, as it hides how the underlying service works. This is the direction the ISQoS model described in Section 3.3 takes.

In [148] the focus is on MPI applications, though this is not in terms of the model a concern and the work remains close to ISQoS. They also explicitly rule out pre-emption in their experiments due to the difficulty of implementing when gang-scheduling.

In their experimentation First Profit and First Opportunity are compared with Longest Job First, Shortest Job First, a modification of First Price and First Reward. The modifications are associated with ”reshaping” a job where a trade-off of using more processors is used in order to improve the completion time of a task. The focus on reshaping derives from gang scheduling/MPI jobs and the changing the amount of CPUs available to a job.

First Profit: It’s definition is not immediately clear; it appears to be aimed at ”maximising the per-job profit for each job independently” by sorting against profit. It appears it maximises the profit by changing the shape/location of threads in the job.

First Opportunity and First Opportunity Rate are extensions of First Profit and consider the jobs collectively, by optimising a global goodness factor.

First Opportunity: Examines the effect of running a job upon other jobs in the queue. It builds a new schedule for the entire workload by trying each possible job in turn and selecting the most profitable standalone shape. First Profit is then used to generate a schedule for the remaining jobs, as it selects the job from the queue that generates the schedule with the highest total profit.

First Opportunity Rate: is a variation of First Opportunity, it however selects the job from the queue that would produce the highest aggregate profit / the total schedule length.

3.2.4 LibraSLA

In a similar fashion to First Reward & Risk Reward LibraSLA [52] has the concept of a penalty rate. In this case the job given a deadline that is distinct from the runtime. This ensures the work is more oriented towards Grids and no longer requires a ”standard” reference speed machine, to determine slowdown. The highest acceptable delay is again potentially unlimited, so risks system viability in regards to budget balance and individual rationality [155, 156].

LibraSLA has two types of deadline hard and soft. In hard deadlines the job is stopped once the deadline is reached, while in the soft deadline case the penalty slope is used to define compensation for delays in the completion of the work.

In LibraSLA a budget is established that indicates the maximum the user is willing to pay. The user is required to pay this should the job complete on time and the service price is again calculated in the same way as FirstPrice + FirstReward i.e. Price_i = Budget_i ($\text{delay}_i \times \text{decay}_i$) as budget and max price are equivalent in this regard. This seems not always to be rational as should the load be particularly low the user may well overestimate the price of the Job.

As the budget does not differ from the service price before a soft deadline/due date, LibraSLA can be considered to lacks the economic competition that is normally associated with market based approaches. This is because payment is not linked to resource availability and informed competition between users. The budget/willingness to pay can conceivably be adjusted by the end user dependent upon market conditions, such as how many auctions they are losing, though there is not clear guided advice about the market conditions.

3.2.5 Aggregate Utility

Aggregate Utility [17] like LibraSLA considers a deadline and explicitly considers the notion of start-delay tolerance. This is advantageous in that users want jobs to complete by a given time which is not necessarily equal to the system centric notion of the job's runtime.

The service price is again set as a gradient that is determined by a set rate of decay, which does not seem user friendly. The penalty unlike others has a maximum bound that also serves as a cancellation penalty after the service provider has accepted a given job.

The pricing function is on a per task basis and jobs are simply treated as an overarching service contract. An aggregate utility function is then used to assess the benefit of all tasks completing as opposed to a proportion of them. This aggregate utility function adds a bonus for completing all of the tasks that belong to a given job on time. This aggregate utility function is described as been able to take any shape, but the form agg_utili = $\alpha \times \beta$ is used.

α is used to describe the potential benefit to the service provider for providing the service and β describes the client's sensitivity to the aggregate metric. Hence in the case of $\alpha = 1.4$ and $\beta = 1$ as marked on Figure 3.1 the client is offering a 40% bonus for completing all tasks that belong to the job.

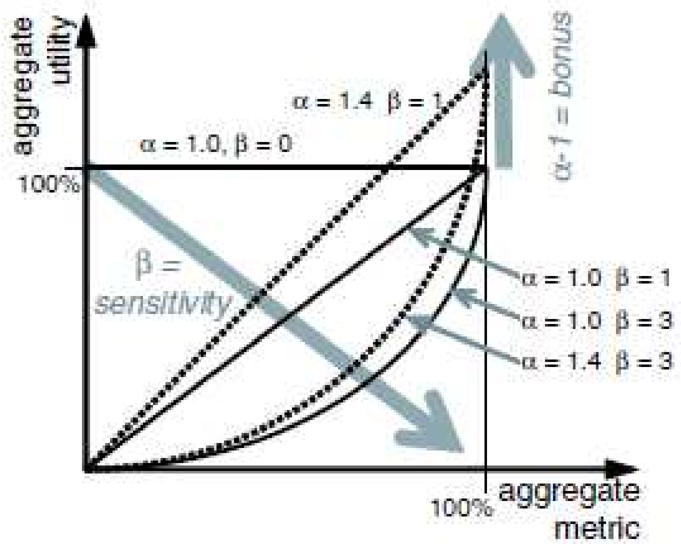


Figure 3.1: Aggregate Utility's Function For Rewarding The Completion Of A Set Of Tasks

3.3 Pricing of the Broker's Job Execution Service

In the previous section the aims of the model was established and given these requirements the models formulation is discussed next.

3.3.1 Input Parameters

One of the main roles of the broker's pricing model is to determine the *service price* the user will pay to a broker for computing jobs. The model will establish a service for computing jobs on time and on budget from underlying resources. Resource providers are going to have costs associated with their resources, so the service price will be based upon this *resource cost*. In doing this a two tied market will form, one tier will be a service market for job execution and the second a resource market. This two tier strategy is quite common in Grid based economics e.g. [16, 69, 130, 148], including work in the SORMA [130] and CATNETS [69] projects.

The model is first expected to work with fixed resource costs (as tested in Chapter 5). In such a market it could be envisaged that faster resources would be likely to command a higher price. It will then be expected to adapt to a dynamic resource pricing that reflects load (as tested in Chapter 6). Thus the model should be flexible enough to be largely independent of how the original resource cost was derived.

The user will provide a *budget* to a job which is an upper limit to how much can be paid for its completion. They will also provide their time requirements by giving a *due date* and *deadline*. The due date is the date by which the job should complete by and the deadline is the date by which the work would no longer be useful. In doing this a notion of priority can be achieved, for the job in terms of cost and completion time. The notion of a due date and deadline in Grid scheduling is fairly unique, many pricing models use gradients but do not derive this from two unique reference points i.e. a due date and deadline (see the related work in Section 3.2). The benefit of gradient based approaches is that it provides the opportunity for heuristic methods, unlike hard deadlines, which give systems little guidance on how to proceed if no feasible schedule exists for meeting the user's constraints [99].

Given due date, deadline and budget information coupled with *job requirements* it will be possible for a set of schedules to be generated, either by a single provider or by multiple providers. The various offers will generate a cost trade off curve that shows the cost to perform a given task in a given time frame. An earliest completion time may also be calculated at this time which indicates the minimum time by which a job may complete (see: Figure 3.2).

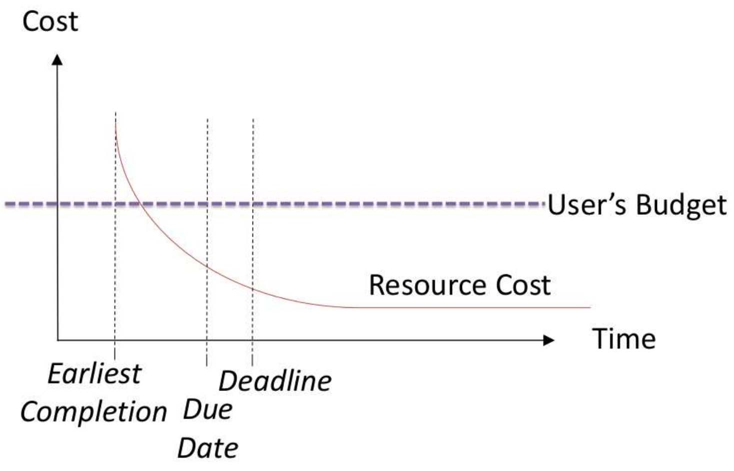


Figure 3.2: The Resource Cost in Relation to the Pricing Model

A major aim of this model is to provide guaranteed timing for a given price. This therefore means the service price is required to drop if due date is passed i.e. completion time exceeds the due date. This is because it creates an incentive for the broker to select providers that have a schedule that will make the job complete before the due date and deadline.

The broker needs an incentive to participate in the market i.e. individual rationality [155, 156]. It will therefore need to charge an additional amount over the resource cost for its usage. The user will therefore be charged a service price that will be a percentage *mark-up* of the actual cost for resources e.g. +25% of the actual cost of the resources. An alternative charging mechanism would be to have a fixed mark-up from the resource cost. This however, presents an issue in that the lengths of jobs can vary and estimates on task length are also inaccurate [98]. A fixed fee as a percentage of the overall work would be liable to fluctuate. The likelihood of this occurring in Grids means the broker may have reduced incentive to participate in the Grid marketplace.

If the service price drops below the resource cost then there is an additional requirement for an incentive to participate. This requirement arrives from the need to achieve budget balance [155, 156] where the broker must not need any supplementary funds in order for it to continue working. A *cap upon the penalty fee* is therefore introduced in order to control the maximum extent to which the broker may make a loss upon any single job.

The *mark-up* may either be decided upon by the broker or chosen by the user from a set of choices provided by the broker. The mark-up in affecting the broker's profit can therefore be seen as a mechanism to denote job priority. Mark-up offers the opportunity for tiered pricing where the user can give a priority that is attached to a numeric value of mark-up. This means a value range with low mark-up, a mid range with a medium mark-up and a premium range with high mark-up can be established, in order to prioritise work. This is useful as mark-up affects both profitability and the acceptability of a job. This is because it is a component part of the broker's ability to reschedule a job i.e. if the broker has spare budget and/or the service price is further from the resource cost then it has a greater degree of money available to create such resilience.

In summary the broker presents a service with added value, in that it will guarantee times for a given job or compensate accordingly. The due date and deadline will modify how much of the service charge is paid by the user, in accordance with the mark-up and the budget allocated to the job.

3.3.2 Overview of the Model's Sequence of Events

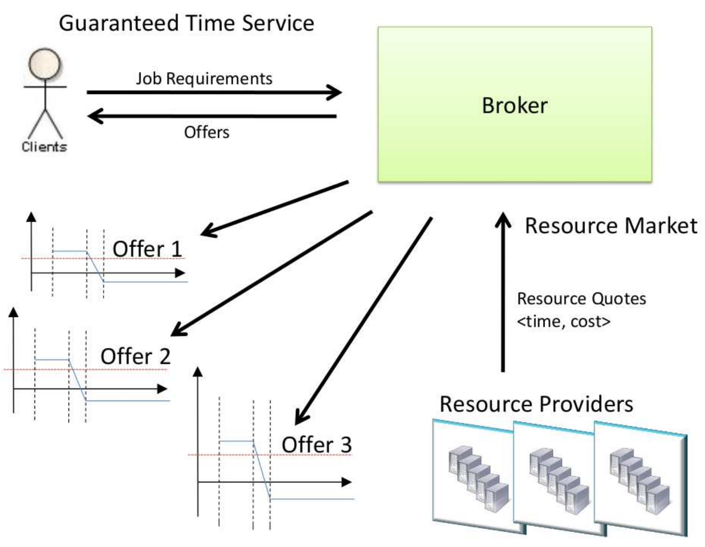


Figure 3.3: Overview of the Model's Sequence of Events

In this section the flow of events in the model is described. A diagrammatic overview is given in in Figure 3.3. The sequence of events is described below and the structure of messages passed is expressed in Extended Backus Noar Form (EBNF) . The definitions of the elements in EBNF can be found in Table 3.1. The sequence of events is therefore as follows:

1. The user sends their job requirements in the form:

   <Job Request> ::= <Task>+<Due Date><Deadline><Mark Up>
   <Budget>{Cap On Penalty Fee}

   <Task> ::= <System Requirements><Input Data Size>
   <Output Data Size><Compute Length Estimate>

2. The broker contacts the resource provider and requests quotes for each job, the resource providers essentially compete in a tender market.

   The providers return offers after the broker requests a quote they are formatted in the form:

   <offer> ::= <Jobs Request><Provider><Resource Cost><Completion Time>

3. The broker with the offer information and the user requirements may then formulate the service price for each offer, which is introduced in Figure 4.

4. The user then accepts or rejects a service offer by the broker. In the event of acceptance the broker submits the work for completion and monitors the work in case of failure or slowdown and rescheduling occurs if required.

5. The job completes, the results are sent back to end user and payment is made.

Table 3.1: EBNF Used in description of user input

Symbol	Meaning
$*$	0 or more
+	1 or more
$<\ldots>$	Required non-terminal
{...}	Optional non-terminal
(...)	Grouping
|	Or
::=	Defines the non-terminal to the left in terms of the expression to the right.

3.3.3 Calculating the Service Price

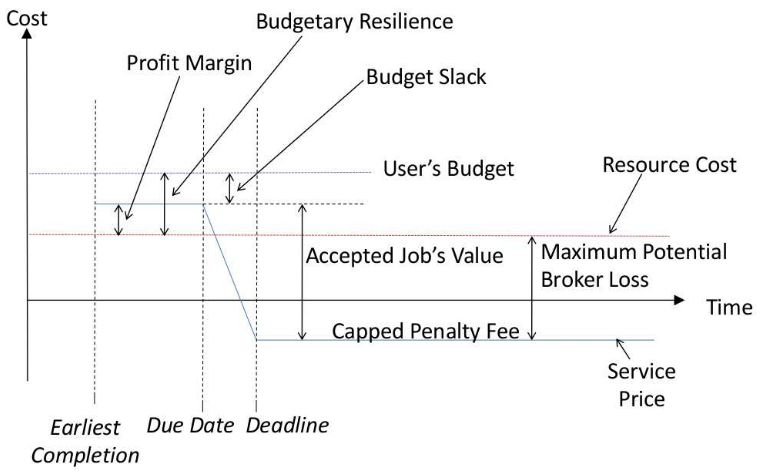


Figure 3.4: Charging of the Guaranteed Time Service

Figure 3.4 represents how the price changes in the service market. The service price is based upon the job completion time and is affected by the user's due date and deadline. Once the due date is reached the service price drops linearly until the deadline is reached, at which point any potential loss gets no greater. This is because the deadline is considered to be the point at which the job no longer has any use to the client so the job may be stopped and the service charge/penalty may then be settled.

The service price is decided upon by the broker by selecting an offer that represents a point on the resource cost curve from Figure 3.2. Taking the offer's resource cost it then applies its mark-up e.g. +20%. This provides the position of the service price between the earliest completion time and the due date. The cap on the penalty fee then provides the lowest possible point at which the z-curve pricing function may drop to, which is when the deadline is reached.

The resource cost is derived from the offers from the resource market and when the service price and resource cost meet indicates the breakeven point. If the service price is below the resource cost then a loss is incurred by the broker.

The difference between the service price and user's budget provide a measure called *budget slack*, which is important to consider when accepting a schedule.

The difference between the service price and resource cost gives the broker's *profit* margin. It may make sense in certain cases to reduce the amount of profit made in order to ensure the job completes hence avoiding loss. A further value of *budget resilience* can also be given as the difference between the resource cost and the budget, as the broker could notionally sacrifice its profit during rescheduling in order to save the job. This therefore means the profit margin can be seen as the minimum amount money available if the broker is forced to reschedule due to resource failure or slowdown.

An accepted job's worth to the provider may be derived as the difference between the service cost charged and the maximum penalty. This is because liabilities are reduced when an accepted job is completed, (especially if it has a non-zero penalty cap). In order for the broker to minimise its losses the difference between the actual cost of resources and the cap on the penalty fee must be considered. This will be important when rescheduling or deciding upon which jobs should be considered to be dropped from a schedule.

In Figure 3.5 three important measures of how vulnerable a schedule is shown. They may be derived from the difference between the selected schedule's completion time and the following three points:

- due date
- breakeven point
- deadline

These temporal slack measures will hence act as metrics on job offer quality, along with the budgetary resilience and budget slack that was previously discussed, along with more obvious measures such as completion time, service price and broker profit.

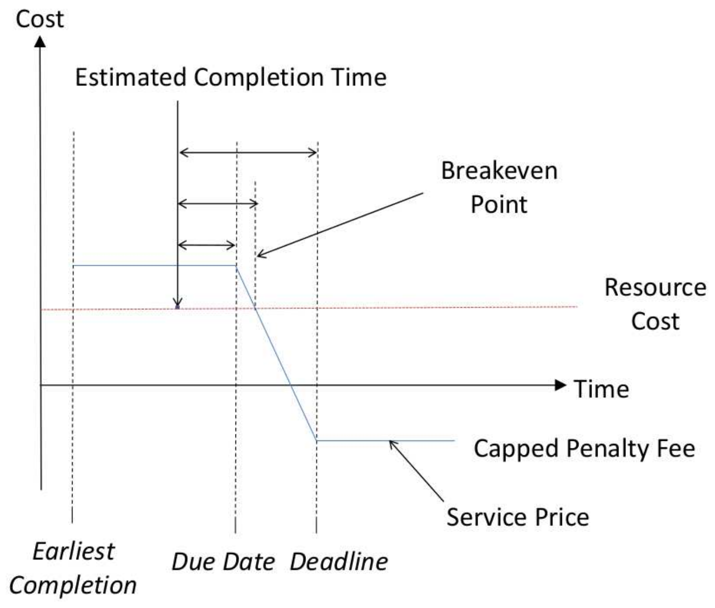


Figure 3.5: Charging of the Guaranteed Time Service - Temporal Aspects

3.3.4 Overall Effect of the Market

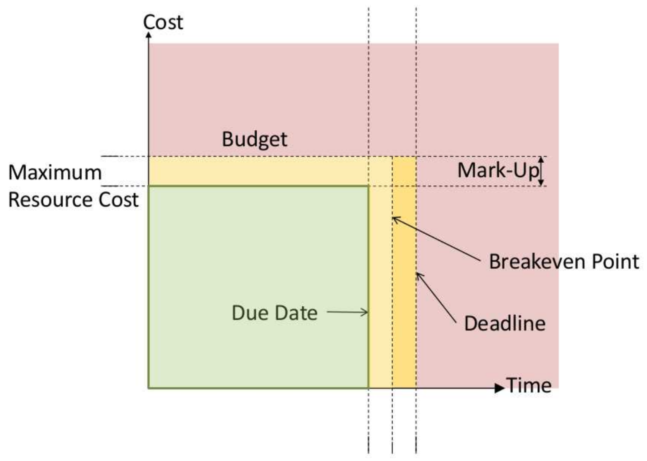


Figure 3.6: The Service Market - An overview of Temporal and Budgetary Aspects

In Figure 3.6 the effect of the temporal and budgetary constraints in the pricing model are demonstrated. Firstly the due date and deadline are represented on the x-axis as the two boundaries placed upon completion time. The budget is placed on the y-axis as the upper boundary of the service price.

Given these initial values the mark-up can be used to derive other useful aspects. The maximum resource cost possible where budget slack equals zero can be derived. This leaves the mark-up indicating the minimum amount of budgetary resilience without the broker immediately incurring lost profit. There is also a point in between the due date and deadline where the service price becomes equal to the resource cost and the broker merely breaks even. This therefore completes a border around an offer defining the acceptable service quality. The position of the breakeven point is derived from how much mark-up the broker is making, it will be shown later in Section 3.5.2 (Figures 3.14 and 3.14), the exact properties of this border and in particular that it can maintain a fixed position based upon mark-up regardless of the resource cost.

3.4 The Simulator

In order to test the model that was developed a simulator was needed. The ISQoS pricing simulator was developed to be able to test the economic mechanisms that were used in the project. In the next section existing simulators are discussed then the ISQoS simulator is introduced.

3.4.1 Simulators

Simulation is often used in Grids research to cope with limitations of large testbeds. Testbeds are often not readily available and that experiments performed on them are often completed without exclusivity to the resources. This is because Grids often continue their day to day operations, which makes results less repeatable. Simulation therefore offers a first opportunity to test theories before deploying them in practice. In our case it also makes it possible to implement the pricing model without having to deploy it to a live Grid, which would create significant interference, during the deployment as the ISQoS Grid middleware developed in Chapter 4.

In order to decide if an existing simulator should be used a selection of simulators was tested and existing literature was studied. Sulistio *et al.* [171] presents a taxonomy of simulators for computer based simulations with a focus upon parallel distributed systems. In their survey they discuss Bricks [173], GridSim [42], MicroGrid [168] and SimGrid [94]. The review is however dated and the Bricks and MicroGrid simulators have subsequently become obsolete. Simulators are often short lived, past the authors initial use of the product and do not always establish an active and broad community to support them. If a simulator is to be used during experimentation then it must be easily usable with reasonable support, otherwise the cost of using the software is outweighed by writing a simple bespoke simulator.

Quetier and Cappello [150] the authors of SimGrid describe several tools for performing experiments in Grid computing. These tools are placed in three separate categories: simulators, emulators and testbeds. The simulators they discuss are Bricks, SimGrid, GridSim, GangSim [61] and OptorSim [25]. The two emulators are also discussed MicroGrid and Grid eXplorer [150] and finally the last category is that of experimental testbeds. Quetier and Cappello [150] focus their discussion upon several key issues, which are namely:

- the original motivation behind the simulator
- the suitability in regards to which research area is of interest.
- the principles behind how each simulator works (i.e. their implementations)
- the validation of simulators and of their results.

The consideration of the ”original motivation” and suitability for a given experiment is one that is of critical importance. It was also the ultimate reasons for the writing of the ISQoS simulator. Many simulators focused on particular aspects of the overall Grid environment e.g. the network or upon auction based economics. This made many simply impractical to use as part of the experimentation.

A recent notable implementation is that of the Grid Economic Simulator (GES) [35, 125] in which scalability of various simulators is studied. In [125] the simulators: GES, SimGrid and GridSim are evaluated, thus comparing their own work to SimGrid and GridSim which have established large user bases and remained active. The testing of an implementation with a scalability study, makes it seem more plausible that the developers have performed some level of validation of the simulator, which is another criteria point for selecting a simulator.

The simulators found from the literature are briefly discussed below, with the simulators which were considered less relevant merely listed at the end for completeness.

3.4.1.1 Primary Candidates

GridSim [42]
Project homepage: http://www.buyya.com/gridsim/
Status: Active

GridSim remains an active project and has spawned several variations such as a cloud oriented version called CloudSim (http://www.cloudbus.org/cloudsim/). In assessing the simulators codebase it was decided to be not of the highest quality, with key principle classes being very verbose, but it remains a simple starting point for any simulation. It primarily links to auctions and the research of Rajkumar Buyya, which was away from the original intent of the ISQoS pricing model.

GSSim [116]
Project homepage: http://www.gssim.org/
Status: Active

GSSim was introduced in the paper [116] and built on top of GridSim, it was intended to make the simulator easier to use. At the time of building the ISQoS Simulator it was still a work in progress and the documentation was not clear upon how GSSim could be made to work. The paper by Kurowski *et al.* [117] also uses GSSim in its experiments but the product has largely been used only by its authors. In [117] they focus upon a two-level hierarchical Grid where a Grid broker makes scheduling decisions at a global level which then allocates jobs to Grid nodes where jobs are then placed in a schedule by local schedulers at site level, which was similar to the ISQoS Simulator.

Grid Economic Simulator [35, 125]
Project homepage: None
Status: Active (available only on request)

This simulator from the papers [35, 125] performed well under the scalability study that the original authors performed. The source is only however available on request hence a community has not built up around the simulator meaning support would be limited. At the time of development of the ISQoS Simulator it was undergoing significant development changes.

SimGrid [94]
Project homepage: http://gforge.inria.fr/projects/simgrid/
Status: Active

This is an active project written in C and developed by the French National Institute for Research in Computer Science and Control. This simulator during testing was difficult to setup, it was compiled successfully, yet it was not clear upon its exact usage.

OptorSim
Project homepage: http://cern.ch/edg-wp2/optimization/optorsim.html
http://sourceforge.net/projects/optorsim/
Status: Obsolete

This simulator was written in Java and was published to sourceforge in late on the 24th October 2006 and appears to have subsequently ceased to be used. The documentation was seen to be of a good standard, so it is expected that it could still be used. It was written as part of the EU DataGrid project and was aimed at testing replication algorithms, so the focus towards replication algorithms was therefore seen as a concern. The project's homepage indicated that they had been working on economic mechanisms, but that this was not complete. The documentation provided with the simulator seemed to support this, stating that features such as job budgets were to be added. Essentially users were described as price takers and providers are price setters, so users could not influence how much they were going to pay. This means with the focus of the simulator being towards replication and the lack of economic mechanisms it would be of limited. It also used large configuration files to change parameters of the experiment, which were difficult to read and hence prone to error.

CATNETS Simulator
Project homepage: http://www.catnets.uni-bayreuth.de/index.php?id=11
Status: Execution errors, EU project ended

The OptorSim simulator was extended to build the CATNETS simulator, which is based upon the market concept called catallaxy. The projects documentation provides a good assessment of existing simulators in [205]. The simulator failed under testing against examples provided within the source code with a file not found exception, the files the exception referred to were not evident in the download of the simulator provided by the Catnets project. This underlies one of the problems with using existing code, if there are insufficient high quality examples and in general documentation then such a simulator cannot live past its initial intended usage.

3.4.1.2 Other Simulators

BeoSim
Project homepage: http://www.parl.clemson.edu/ wjones/research/
Status: Not Released

This project seemed active at the time of the review but the software was not made available and other related simulation software was listed, in its stead.

HECIOS
Project homepage: http://www.parl.clemson.edu/hecios/
Status: Obsolete

The High End Computing I/O Simulator (HECIOS) is a C++ based simulator, that focuses upon MPI and IO based simulation. It is developed at the same organisation as BeoSim. It is based upon the OMNeT++ 4.0 C++ simulation library, which can be found at: http://www.omnetpp.org/models.

MicroGrid [168]
Project homepage: http://www-csag.ucsd.edu/projects/grid/microgrid.html
Status: Obsolete

This project is out of development, the last release of MicroGrid was version 2.4.6 in December/2004 and aimed to support a range of peer-to-peer and distributed applications.

Agent Grid
Project homepage: http://sourceforge.net/projects/agentgridrepast/
Status: Obsolete

Agent Grid was placed on sourceforge in 2007 and is no longer developed. The last CVS commit was on 16th February 2007. The simulator was based on the Repast Simulator which remains active. Repast may be found at: http://sourceforge.net/projects/repast/ or http://repast.sourceforge.net/.

Bricks [173]
Project homepage: http://ninf.apgrid.org/bricks/
Status: Obsolete

This project is no longer active and upon its homepage the download link is struck through with some related works listed: http://ninf.apgrid.org/bricks/related_work.shtml

GangSim [61]
Project homepage: http://people.cs.uchicago.edu/ cldumitr/GangSim/
Status: Obsolete

This is a simulator that focuses on Service level agreements, though seems to have fallen into disuse. One key part of the simulation work to be performed is on the economic model and job submission, so the focus on SLAs would not be of great use.

The Delft Grid Simulator (DGSim)
Project homepage: http://www.pds.ewi.tudelft.nl/ iosup/dgsim.php
Status: Obsolete

The homepage lists a set of papers by the original authors which used the simulator. The code was also unavailable, making use of the simulator impractical.

3.4.1.3 Summary of Simulators

In terms of decent candidates for use in the simulation based experiments, GridSim, GES and SimGrid seemed most practical.

GridSim was tested and worked reasonably well. It was particularly orientated towards auctions and economic models. Its underlying representations were poor for ISQoS needs, which made things difficult. An extension to GridSim called GSSim was seen to be useful, though at the time the simulator was needed GSSim was described as being in beta and the documentation/project wiki seem in a state of flux.

Grid Economic Simulator GES was a very promising candidate and after the authors had been contacted it was found they were performing re-factoring of major parts of the codebase and was only available via the authors so wasn't seen as useful at the time.

SimGrid was compiled and installed via configuration scripts and make, but it wasn't clear in regards to documentation on how the simulator could be used.

The final solution was therefore to quickly develop a simulator from scratch, due to the likelihood that this would be the line of least resistance. This simulator is hence introduced in the next section.

3.4.2 ISQoS Pricing Simulator

This simulator has two distinct modes of operation. The first mode is used to sweep through ranges of parameters. This means it can be used to examine the models performance for a single job across the entire range of possibilities. It also allows for verification that the simulator is correctly representing the model, as well as showing complete trends in the models output. Results from this mode are reported in Section 3.5. The second mode is used to perform discrete event simulation, which can be used to test the performance of the model in a running situation. The results of this are reported in Section 3.6.

3.4.2.1 Discrete Event Simulation

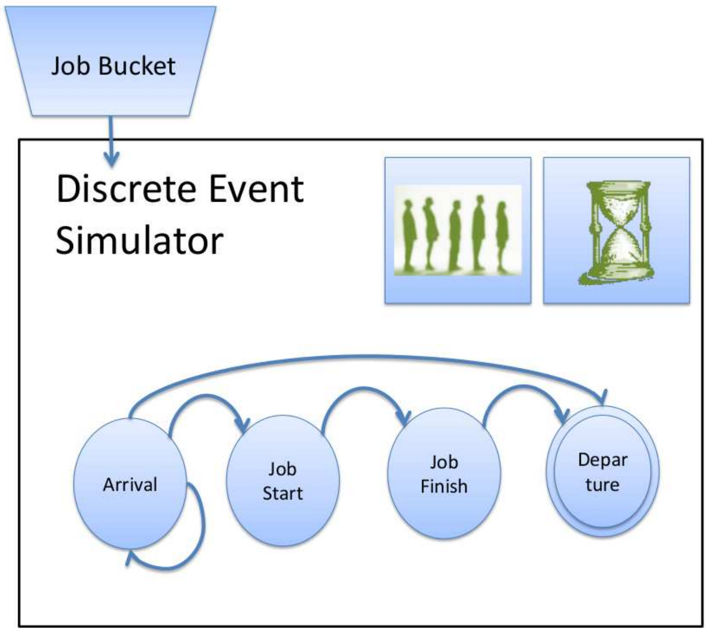


Figure 3.7: An Overview of Discrete Event Simulation

The simulator is shown in Figure 3.7 and is used as an example of discrete event simulation. Discrete event simulation relies upon the notion of independent events arriving over time. The events are then processed and results are gathered. The experimentation ends either after a given amount of simulated time or after a sufficient amount events have been processed.

In the case of the simulator presented here, a bucket of jobs is created before the simulation starts. The jobs are then taken randomly from the bucket and placed into the simulator, until the bucket is empty. The simulation starts with a job arrival event, this event takes a job from the bucket and generates a new arrival event. This new arrival event is placed into a priority queue of events to process (earliest task first). Upon a job arrival it can either be accepted or rejected. If it is accepted then a start job event is generated. This indicates the time when the job starts on the Grid. If the job is dropped without scheduling then no further processing of the event is required. The earliest/next job in the priority queue is then taken. If it is an arrival event a similar process occurs to before, otherwise if it is a start job event the end of job event is generated. The clock then moves on in a similar fashion until the end of the simulation. Finally if a job finished event is processed, the effects of billing the job are logged.

3.4.2.2 Parameter Study Mode

In the parameter study mode the aim is to generate a set of jobs with a particular range of properties. The jobs are randomly generated, but fitted to a uniform distribution. This mode can be used to determine what the outcome would be if the jobs were submitted to a Grid following the proposed economic model. This therefore allows the model to be examined and ensures the model is accurately being followed by the simulator. To assist in the verification process static code analysis[1] and JUnit testing was also used to ensure the simulator was working correctly. The various parameters and the range of possible values the simulator takes is discussed next.

The first major setting indicates how many jobs should be generated (*Jobs_to_Generate*). The other settings then indicate either the range or the specific value a parameter in the model should take. There is also the possibility to indicate if the budget, due date, deadline and penalty fee should be placed within a fixed range or bound to another appropriate value i.e. the penalty fee could be given as a percentage of the service price, the details is discussed next.

The completion time is given as an upper *Comp_Time_Upper* and a lower *Comp_Time_Lower* value with an interval *Comp_Time_Interval*. This however changes if the cost and speed of each resource as specified, the tasks then need a *Task_Work* value assigning to them to specify their size.

The due date is specified as a percentage of the completion time and is shown as a range of due dates by a minimum value *Due_Date_Min_Perc* and a maximum value *Due_Date_Max_Perc* and an interval value *Due_Date_Interval*.

The deadline is then shown as a percentage of the due date in a similar fashion *Deadline_Interval, Deadline_Max_Perc, Deadline_Min_Perc*. It takes its value as a percentage

[1]http://findbugs.sourceforge.net/

increase from the due date as practically the deadline must be a greater or equal to the due date.

The resource cost is set by the triple *Res_Cost_Interval, Res_Cost_Lower, Res_Cost_Upper* unless specific resource speed cost pairs are specified, where in this case resources are then emulated.

The budget for a job is shown as the triple *Budget_Interval,Budget_Max_Perc, Budget_Min_Perc*. It is shown as a percentage of the resource cost, unless *is_Budget_Fixed* is set to true where it then sets the value of the budget to be within the range specified, without reference to the resource cost.

The broker's markup is taken as a percentage increase on the resource cost, this markup is shown as the triple *Markup_Lower, Markup_Upper, Markup_Interval*.

The penalty fee is given as a percentage of the service price unless it is set to a fixed mode when literal values between the ranges specified are used instead. The triple used to describe the penalty fee is *Penalty_Fee_Max_Perc, Penalty_Fee_Min_Perc, Penalty_Fee_Interval*.

The final element in the configuration file is the ability to change the name of the output file *Filename_For_Output*.

The output of the simulation gives a set of jobs with a completion time, due date, deadline, breakeven point, resource cost, actual service price, max service price, budget, markup, cap on penalty fee and broker profit indicated. This however does not show the effects of a limited amount of resources and each job is as if it had been allocated sufficient resources to complete each task separately.

3.4.2.3 Discrete Event Simulation Mode

The discrete event simulation mode allows for the testing of the model, by first generating a bucket of jobs in a similar way to the parameter study. The bucket is however setup to follow distributions for each of the parameters from values found in the literature. The jobs in this bucket are selected randomly and then released one by one into a discrete event simulator and the outcome is then recorded.

An overview of the simulator is presented in Figure 3.8. The main component of the simulator is a ISQoSDiscreteEventSimulator class. Its role is to:

- maintains a copy of the simulators event queue
- create and hold the bucket of jobs following the predefined distribution
- determine the rate at which jobs arrive and are sent to the broker for processing
- decide upon the simulators stop conditions (time and or jobs submitted)

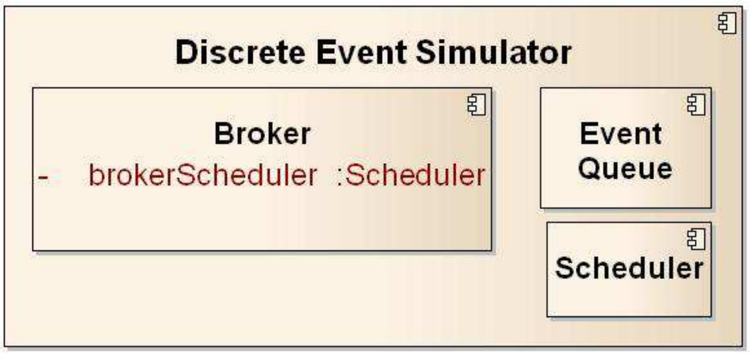


Figure 3.8: An Overview of the Simulator

The discrete event simulator then delegates the main work to the simulated broker. The broker's role is to:

- record the results of the simulation.
- controls the flow of execution,
- pass events to a broker selection mechanism called the ”broker scheduler”

The simulated broker uses an interface to perform scheduling/brokering, so it can act as either a single ”local scheduler” or have a brokering mechanism/scheduler with a set of ”local schedulers”. This is achieved as there is no difference between contacting multiple local schedulers and a single scheduler from the interfaces perspective. This abstraction enables the simulator to be used to investigate a scheduling algorithm on either a single provider or upon multiple providers.

The scheduler in use can hence be either a ”broker scheduler” that simply creates and manages a set of local schedulers or it can be a single local scheduler that performs the allocation of tasks to individual resources.

The simulator finally has a set of comparators for used for either the broker's scheduling mechanism or the local scheduler, these are for the Resources (”named Machines in the simulator”), Offers and Tasks and are listed below:

Offers (Sorting): The available offer sort orders, used in ranking offers which are similar to the broker and are discussed in more detail in Section 4.4.7.

- Cheapest: offers are sorted by the cheapest offer first.
- Earliest: offers are sorted by the earliest offer first.
- Profit: offers are sorted by the offer that makes the most profit for the broker first.

- Profit Rate: offers are sorted so the offer making the most profit per second is first. The ordering derives from the calculation: profit / (completion time - submission time)
- Profit then Latest: offers are sorted by highest profit first then a second order sort on completion time, with the latest been given preference.
- Profit then Earliest: offers are sorted by highest profit first then a second order sort on completion time, with the earliest been given preference.
- Budgetary Resilience: offers are sorted by the calculation: offers budget - resource cost, with the smallest value first.

Offers (Filtering): The available offer filtering mechanisms, used in selecting offers are similar to the broker and are given in more detail in Section 4.4.7.

- Topmost Offer: this selects the topmost offer in the sorted list
- Topmost Profitable Offer: this selects the topmost offer in the sorted list that makes a profit for the broker if such an offer exists, otherwise all offers are rejected.
- Minimum Profitability: this selects the topmost offer in the sorted list, that meets a user defined minimum level of profit, if not is found all offers are rejected.
- Minimum Profitable Rate: this selects the topmost offer in the sorted list that makes a minimum amount of profit per second.
- Near Going Rate: This calculates a rate: Profit / (Completion Time - Job Start Time). It then uses a history of accepted jobs to decide if the offer is acceptable. It returns the topmost offer that meets the current going rate with some pre-specified margin of acceptance i.e. a threshold below this current rate to which an offer could still be accepted.
- Hybrid Offer Filter: This advances upon the previous mechanism in that an offer is immediately accepted should neither the budgetary or time constraints of a job be in conflict. The current rate is then used to resolve jobs that are just past the due date or just eating into the mark-up the broker makes for itself.

Machines/Resources: The available machine sort orders, used in scheduling algorithms.

- Cost: The machines are sorted by resource cost, cheapest first.

- Cost then Speed: The machines are sorted by resource cost, cheapest first and then with a second order sort of speed, slowest first.
- Speed: This sorts the machines by the fastest first.
- Speed Cost Ratio: This sorts machines by the ratio between
- Speed then Cost: This sorts machines by the slowest machine first, then sorts by resource cost, cheapest first.
- Current Workload: This sums up the amount of work belonging to each task assigned to a given machine. It then sorts by the least loaded machine first.
- Completion Time of the Last Task: This sorts machines by the time at which their queues will become empty.

Tasks: The available Task sort orders, used in scheduling algorithms.

- Completion Time: This sorts tasks by their expected completion time, earliest first.
- Size: This sorts tasks by the amount of work that is associated with the task, smallest first.

3.5 Experimental: Parameter Variation Study

In this section the work relates to the mode of the simulator described in Section 3.4.2.2. This allows for ranges of inputs into the model to be examined, thus checking to see if the model is both correctly implemented and that outputs of the model are useful. In Section 3.5.1 both the service market and resource market, are illustrated. This means individual resource costs are specified. After this the resource cost is fixed and the results are shown only for the service market, which is shown in Section 3.5.2.

3.5.1 Resource and Service Market

In this section a study of the properties of the proposed broker pricing model is presented. Figure 3.9 is used as an illustration of the resource and service markets interacting. A set of 18 offers are shown on the graph. The resource cost as given by the provider/s is shown. The parameters given by the user of due date, deadline and budget are also shown as straight lines on the graph. The service price is shown as two separate values, the actual service price and the service price should the price not have dropped due to the due date to deadline period.

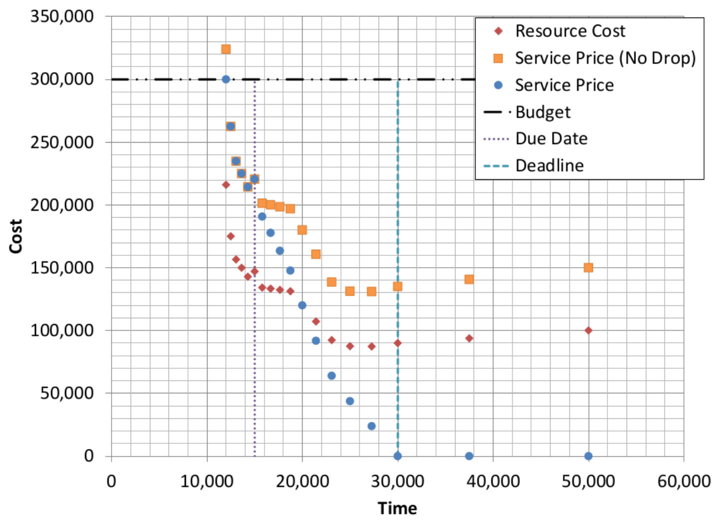


Figure 3.9: An Overview of the Resource and Service Market Interaction

The parameters of the demonstration are as follows, the due date was set to 15,000 and the deadline was set to 30,000. The mark-up was set to +50% and the budget to 300,000. The cap on the penalty fee was set to 0. The task size was fixed to 300,000. Finally a set of machines with speed and cost pairs was introduced, in order to calculate the cost of each task and its completion time. These pairs are shown in Table 3.2.

Table 3.2: The Machine Cost Pairs used in the Resource and Service Market Example

speed	cost	speed	cost
25	18	16	7
24	14	15	6
23	12	14	5
22	11	13	4
21	10	12	3.5
20	9.8	11	3.2
19	8.5	10	3
18	8	8	2.5
17	7.5	6	2

Due Date and Deadline: The service price can be seen to diminish in comparison to the maximum service price possible, between the due date to deadline period. The maximum being derived from if the due date and deadline did not exist.

Penalty Fee Cap: The maximum loss to the broker is at the deadline and notionally after it. This is limited to the resource cost as the cap on the penalty fee equal zero.

Budget: The budget can be seen on the left most point restricting the maximum service price. The candidate schedule produced remains viable; however, as the budget has been surpassed the brokers would have to absorb any loss due to rescheduling.

Viable Schedules: The points 2-5 (left to right for the service price) are all considered to be acceptable schedules with varying amounts of budgetary resilience and temporal slack. The 6th point although viable it is equal to the due date and it is dominated by 5th point (the 5th point is better in both resource cost and completion time parameters). It should therefore be removed from the candidate set that a broker would use to recommend a single schedule.

Breakeven Point: The 11th point is at the breakeven point and indicates the last viable schedule to the broker, where it will continue to make a profit from participating in the Grid.

3.5.2 Service Market

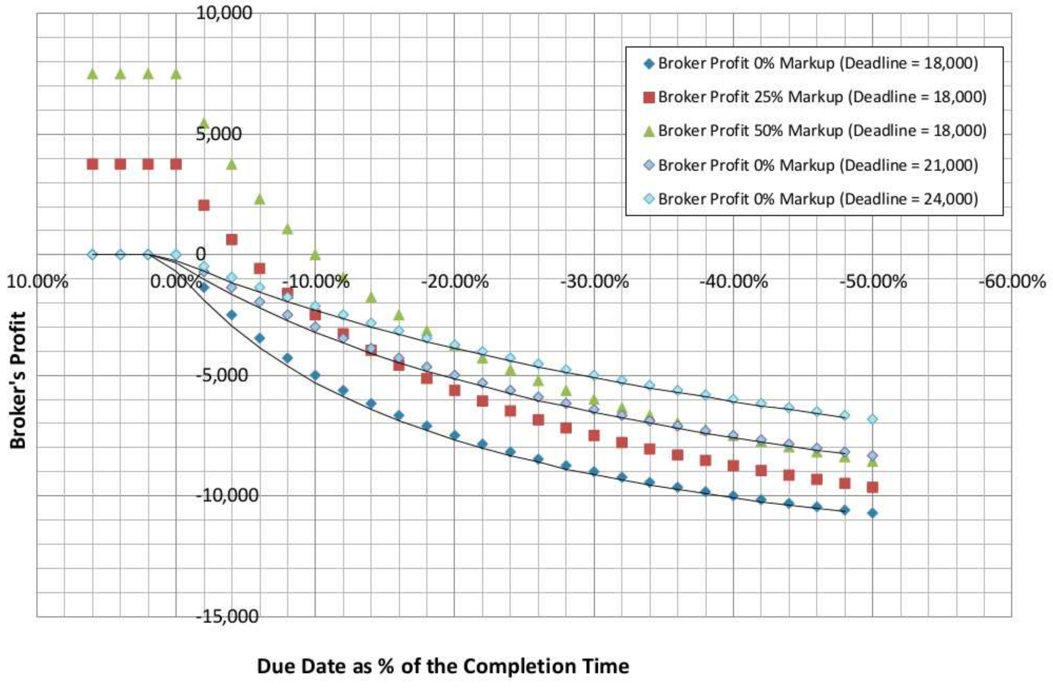


Figure 3.10: Broker's Profit - Varying Deadline and Markup (completion time = 15,000)

The study in this section will focus on the service market. In Figure 3.10 the deadline of jobs is set to 18,000, 21,000 or 24,000 (+20, 40 and 60% of the completion time). The completion time is fixed to 15,000. The due date is then altered in order to change how much slack is available i.e. where the completion time lies relative to the due date and deadline. The due date ranges as a percentage of the completion time from 50% to 120% at intervals of 2.5%.

In Figure 3.10 three variations which have 0% mark-up have been highlighted. It can be seen that those with the smaller deadline have lower profit for the same due date. This demonstrates the increased risk to a broker's profit margin to accept jobs with a smaller gap between the due date and deadline. The mark-up is shown to affect the initial profit and the point at where the broker breaks even, which will be explored in detail later.

A summary of Figure 3.10 and 3.11's parameters may be seen in Table 3.3.

Table 3.3: Parameters for Figures 3.10 and 3.11

Parameter	Fixed/Varying	Sweep Characteristics
Resource Cost	Fixed	15,000
Completion Time	Fixed	15,000
Due Date	Varying	7,500 to 18,000 in intervals of 300 (50% to 120% of completion time in intervals of 2.5%)
Deadline	Varying	18,000 , 21,000 and 24,000
Mark-up	Varying	0, 25 and 50%
Budget	Fixed	22,500 (+50% of resource cost)
Penalty Fee Cap	Fixed	0

Figure 3.11 re-represents the data in Figure 3.10. It shows how presenting the completion time as a percentage of the way through the due date to deadline period means that all values can be represented by a flat surface, where the due date and deadline only affects the position of the job on this surface. This is useful for comparative purposes for jobs in that the gradient is unaffected by the user's preference for the due date and deadline.

It should be noticed that with the smaller deadlines and hence "deadline slack" that the progression through the due date to deadline section (0% to 80%) is quicker resulting in a lower profit. This is reflected in that the red surface (deadline = 18,000) extends out further than the 21,000 and 24,000 lines. This was demonstrated in Figure 3.10 by the jobs with a smaller gap between the due date and deadline having a greater gradient. The broker's profit regardless of due date and deadline is shown to be at 33.3% of the way between the due date and deadline.

The effect of the cap upon the penalty fee is demonstrated in Figures 3.12 and 3.13. The penalty fee ranges from -50% to 100% of the maximum service charge in intervals of 10.0%. The penalty fee causes the surface to have a steeper gradient, resulting in a requirement to restrict the losses that can be imposed. This can even be seen when the pay-out is capped to zero, as the full resource cost of 15,000 would still be lost, along with the prospect of a profit of 3,750 given a mark-up of 25%.

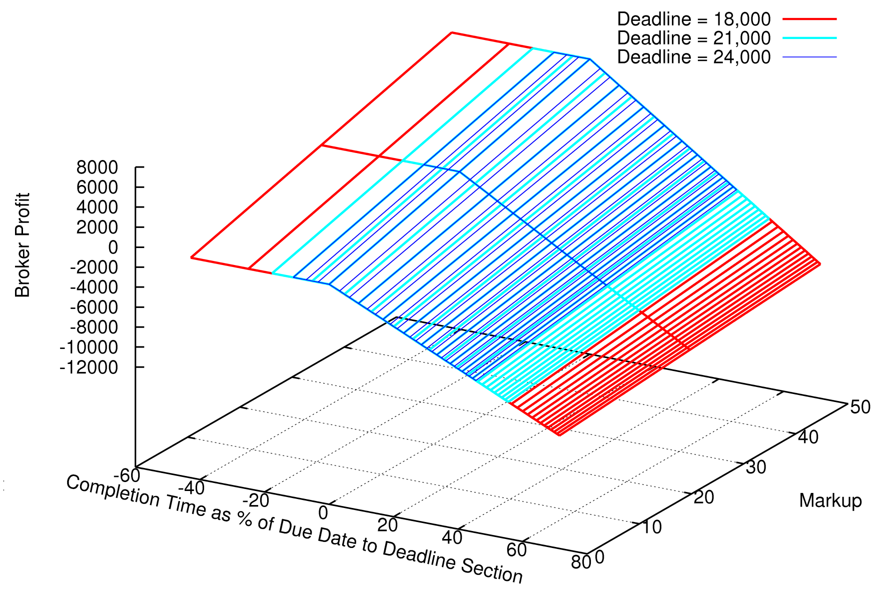


Figure 3.11: Broker's Profit Shown as a Surface - Varying Deadline and Markup (completion time = 15,000)

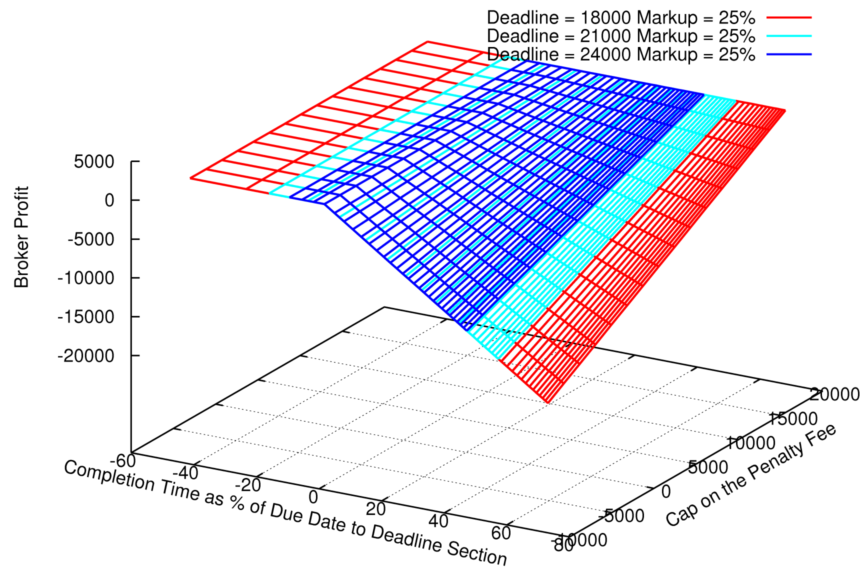


Figure 3.12: Penalty Fee Exploration Shown as a Surface - Varying Due Date, Deadline and Penalty Fee

Figure 3.12 and 3.13's parameter are summarised in Table 3.4.

Table 3.4: Parameters for Figures 3.12 and 3.13

Parameter	Fixed/Varying	Sweep Characteristics
Resource Cost	Fixed	15,000
Completion Time	Fixed	15,000
Due Date	Varying	7,500 to 18,000 in intervals of 300 (50% to 120% of completion time in intervals of 2.5%)
Deadline	Varying	18,000 , 21,000 and 24,000
Mark-up	Fixed	25%
Budget	Fixed	22,500 (+50% of resource cost)
Penalty Fee Cap	Varying	-9375 to 18,750 (-50% to 100% of the maximum service price in intervals of 10%)

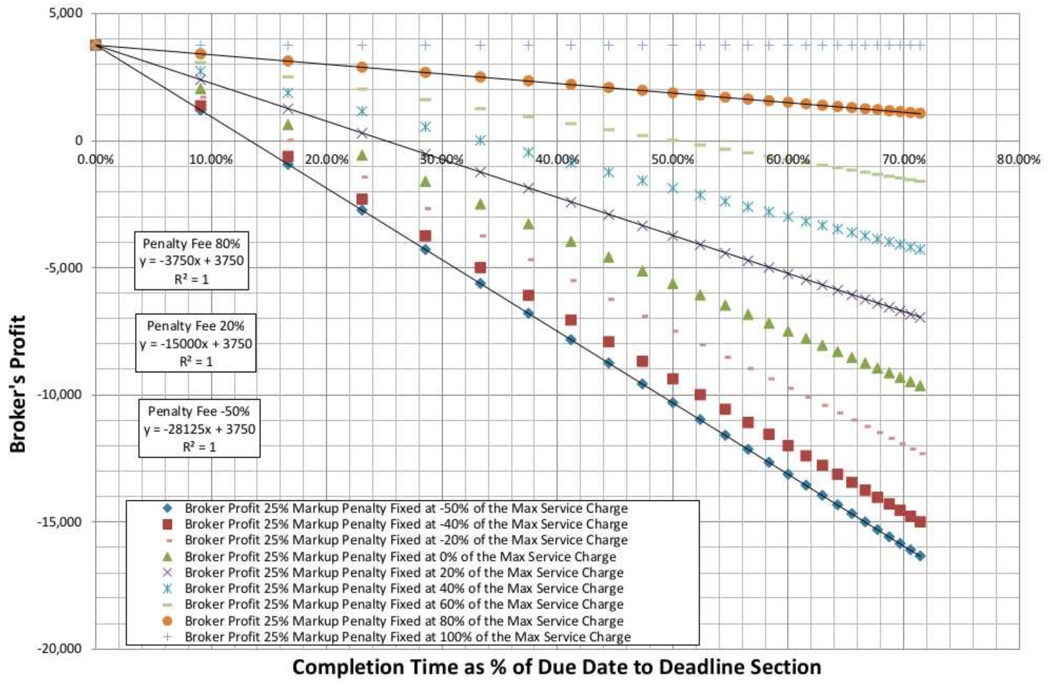


Figure 3.13: Penalty Fee Exploration at 25% Mark-up Varying Penalty Fee, (Due Date = 15,000, Deadline = 18,000

Figure 3.12 uses the same data as Figure 3.13. It remains comparable with Figure 3.11 with a due date of 25%. It however shows the changes in the penalty fee cap. The cap represents the lowest number the service price may go to. If it is negative then the broker pays the user compensation. If this occurs the broker's profit is rapidly reduced, offering further incentive to complete the job on time. The narrower deadline further impacts upon the profitability of the job and the 18,000 case loses the most amount of money for the broker.

Figure 3.13 shows the effects of the penalty fee more for a subset of the data, than Figure 3.12. The completion time is fixed at 15,000 and the deadline at 18,000. The due date is altered in the range of -50% completion time to 0%.

It can be seen that the cap on the penalty fee in relation to service price greatly affects the gradient of the line. This cap on the penalty fee is set as follows:

- Maximum Penalty = Maximum Service Price $\times$ Percentage Penalty
- Maximum Service Price = Resource Cost $\times$ Mark-up

So with a fixed percentage penalty of 100% the broker will make pay a penalty of 18,500, while at -50% it will still get 50% of the maximum service price. The values for the gradient shown in Figure 3.13, which follow the formula y = mx + c, are given as follows:

- y = Actual Profit,
- x = (Completion Time - Due Date)/ (Deadline - Due Date),
- m = (Maximum Profit - Cap on Penalty Fee)/(0% - 100%)
- m = ((Resource Cost $\times$ Mark-up - Resource Cost) - Cap on Penalty Fee)/(0-1)
- c = Maximum Profit = Maximum Service Price - Resource Cost

The gradient represents the rate of loss for a given percentage of the time between the due date and completion time. This along with the mark-up, the slack and the breakeven point will be important when deciding if a job should be accepted or not. The budget constraint represents the upper limit of the service price so long as some budget resilience remains it is not expected to take any further role in the jobs selection preference. This rate of loss is heavily affected by the cap on the penalty fee so this will have to be constrained.

In Figure 3.14 the effect of changes in resource cost and markup are highlighted. The resource cost is set to 15,000, 17,500 and 20,000 respectively, with the mark-up set to

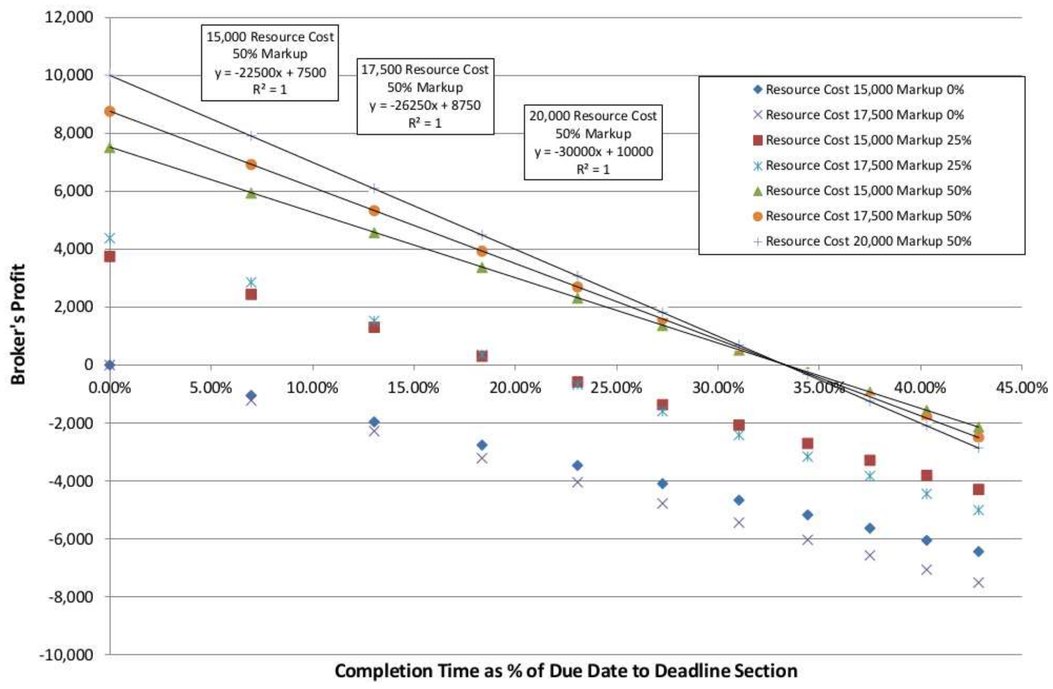


Figure 3.14: Resource Cost Exploration: Fixed Penalty Fee at 0 Varying Resource Cost, Markup and Due Date

0%, 25% and 50%. The completion time remains fixed at 15,000 and deadline is fixed at 25,000. The due date changes between 50.0% and +20% of the completion time. The penalty fee cap is fixed at 0.

The position of the "breakeven point" (a notion of the broker's "margin of error") can be seen to increase with the mark-up. At 50% mark-up the breakeven point is at 33.3% of the distance from the due date to deadline. At 25% mark-up it is 20.0%. The breakeven point is hence directly linked to mark-up and is shown on Figure 3.14 as the intersection points, one for each mark-up. The percentage of the way through the due date to deadline period can be found by:

$$x = \frac{markup}{100 + markup}$$

A worked example of this is shown at 25% mark-up:

$$0.2 = \frac{25}{100 + 25}$$

The time for the breakeven point may be found by the following formula:

$$x = \frac{Service\ Price - Resource\ Cost}{Service\ Price - Cap\ on\ Penalty\ Fee} \times (Deadline - Due\ Date) + Due\ Date$$

A worked example of this is as follows:

$$\frac{18,750-15,000}{18,750-0}\times(24,000-20,000)+20,000=20,800$$

Mark-up can therefore be seen as useful in providing relative preference for two jobs that are otherwise equal, as the broker is given more time before breakeven point and gains a higher profit, without effecting the temporal constraints of the end user. Due to its ability to change the position of the broker's breakeven point it relates to the likelihood of the broker been able to breakeven and make a profit. It is also significant that for any given resource cost the breakeven point (0 broker profit) is always fixed. This is useful in that it simplifies the model in regards to assessing the risk of the broker not breaking even.

Figure 3.14's parameters are summarised in Table 3.5.

Table 3.5: Parameters for Figure 3.14

Parameter	**Fixed/Varying**	**Sweep Characteristics**
Resource Cost	Varying	15,000, 17,500 and 20,000 (2,500 increments)
Completion Time	Fixed	15,000
Due Date	Varying	7,500 to 18,000 in intervals of 600 (50% to 120% of the completion time in intervals of 5%)
Deadline	Fixed	25,000
Mark-up	Varying	0, 25 and 50%
Budget	Fixed	22,500 (+50% of resource cost)
Penalty Fee Cap	Fixed	0

In Figure 3.15 two factors are seen to alter the point at which service prices for the same mark-up intersect. These are the mark-up itself and the cap upon the penalty fee. The value for the broker's profit at this point is given as:

$$Broker's\ Profit=\frac{Markup}{Markup+100}\times Cap\ On\ Penalty\ Fee$$

A worked example for this is:

$$2,857(4.sf)=\frac{40}{40+100}\times 10,000$$

This means that jobs with different mark-up's and different resource costs can be directly compared at the point of intersection, because the resource cost is no longer relevant as only mark-up and the cap upon the penalty fee change the profit made at this

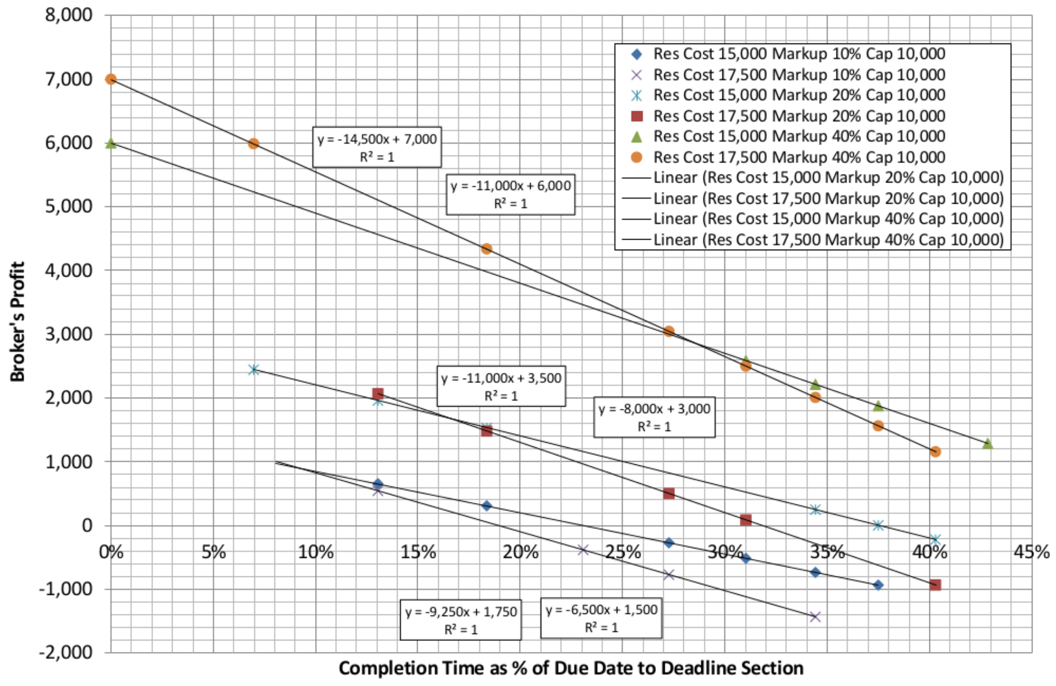


Figure 3.15: Resource Cost & Penalty Fee Cap Exploration: Varying Penalty Fee, Resource Cost, Markup and Due Date

point. In order that this intersection point is also the breakeven point for the broker, the cap on the penalty fee should be fixed to zero. This means the broker will only pay for the cost of the resources that it has used and all jobs with the same mark-up become comparable at the breakeven point regardless of the initial resource cost. The significance of the mark-up is that for an accepted job it is the minimum amount of budgetary resilience possible, while the cap on the penalty fee represents the worst possible case for payout. It therefore means that this point for any given schedule/"resource cost" it is comparable in both its profit (economic properties) and temporal (percentage of the way between due date and deadline).

As the breakeven point is a fixed point an alternative for calculating the gradient, is to therefore use this as a point or reference $m = \frac{Maximum\ Profit - Y\ At\ Reference\ Point}{0 - X\ At\ Reference\ Point}$

Figure 3.15's parameters are described in Table 3.6.

Table 3.6: Parameters for Figure 3.15

Parameter	Fixed/Varying	Sweep Characteristics
Resource Cost	Varying	15,000, 17,500 and 20,000 (2,500 increments)
Completion Time	Fixed	15,000
Due Date	Varying	7,500 to 18,000 in intervals of 600 (50% to 120% of the completion time in intervals of 5%)
Deadline	Fixed	25,000
Mark-up	Varying	10, 20 and 40%
Budget	Fixed	22,500 (+50% of resource cost)
Penalty Fee Cap	Fixed	10,000

3.6 Experimental: Discrete Event Simulation

In this section the exploration of the model is performed using discrete event simulation, by using the ISQoS pricing simulator. The aim is to advance upon the work performed in the parameter variation study (shown in Section 3.5) and to test parts of the model that requires more than just sweeping through ranges of parameters.

The next two sections discusses the experimental method and generic configuration settings that don't change during the experimentation presented in this section. The section after then discusses the results, which were largely reported in [103].

The results are split into two parts the first part focus upon the affect that a range of parameters have upon the amount of jobs accepted in relationship to slack. This is performed with a due date and deadline been equal, i.e. a hard deadline with no gradient.

The second section then demonstrates the effect of the gradient section and shifts to highlighting the behaviour of the ISQoS Hybrid Offer Filter which was introduced in Section 4.4.7.2. The experimentation largely focus upon broker profit made over the life of the simulation. This is because a broker needs to make a profit in order to maintain viability. The principle is to remain viable while ensuring as many users as possible have their requirements satisfied, which in turn generates broker profit.

3.6.1 Experimental Method

The ISQoS simulator at the start of the simulation created a bucket of jobs, who's properties were set to follow distributions as determined from trace analysis given in the literature [95–98]. These jobs were then released at a set rate to the simulated broker. The broker submitted these jobs to the local schedulers/providers, which in turn calculated the machine to task allocations needed to make an offer. The broker then receives offers from the providers. These offers are then ranked and filtered. If a winning offer existed it is then accepted and the work is scheduled. Once the initial bucket of jobs was empty the remaining scheduled work was allowed to complete and then the simulation was stopped.

Each run of the simulation was performed 10 times and where shown confidence intervals of 95.4% will be marked.

3.6.2 Experimental Configuration

The ISQoS simulator is highly configurable, but some parameters remain fixed throughout the experimentation shown in Section 3.6.3. The fixed parameters are hence listed in this section.

The experiments stop condition was to stop once 1,000 jobs had been processed from the bucket of jobs and no limit on the duration was set.

Each provider used a simple scheduler which assigned work based upon the machines workload, it took the least loaded resources first and assigned tasks in order of workload (largest task first). Eight providers were used, each with 144 machines. A range of both machine speed and cost values was chosen. The machine cost was set such that the faster the machine the more it costs, though there was no set distribution shape for this and values were set by hand.

The task workload will be placed into three categories: small, medium and large. The categories will denote the average task length. They will be set at the equivalent of 2hrs, 4hrs and 8hrs worth of work on the average machine of a provider. The average speed of the machines is 16,333 units of work per second. Thus the amount of work for a medium sized task is 240,000,000 operations. Similarly this has been done for the small (120,000,000) and large (480,000,000) categories. The workload for a task will have a normal distribution with a mean of 2.73 and a standard deviation of 6.1 following the average task runtime indicated in [98]. The durations were chosen because [96] gives the average time to run a task in a group submission/job as 14,181s and [95, 97] indicates that most runtimes are shown to between a fraction of a minute and 1,000 min (1,000min = 16.6hrs). The runtime variability between tasks belonging to the same BoT [98] is indicated to follow a Weibull distribution with a shape of 2.05 and a scale parameter of 12.25 hence this variability will also be accounted for.

The task per job will be separated into three categories: small, medium and large. A Weibull distribution will be chosen with a shape parameter set to 1.76 as per [98], the scale parameter will be adjusted in a fashion that creates the appropriate range desired.

Iosup *et al.* [98] give an average BoT size in their traces as been between 5 and 50, while the maximum BoT size can be on the order of thousands. The authors in [95] indicate that the average number of tasks in a bag in various different Grid traces investigated ranges between 2 and 70 and that most averages for the traces examined fall between 5 and 20. The ranges for the amount of tasks in a bag are therefore indicated in Table 3.7.

Bag Size	Min and Max Values	Mean
Small	2-10	4.99
Medium	11-39	21.49
Large	40-70	51.23

Table 3.7: The Minimum and Maximum Tasks per Job

The inter-arrival time (i.e. the time between arrivals of jobs) will follow the Weibull distribution and will have a shape parameter of 4.25 as per [98]. The inter-arrival time for job submissions is indicated to be between 1 and 1,000s in [95,97], with most been below 100s.

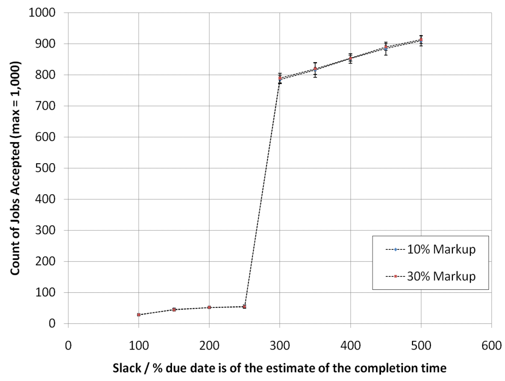
A selection of inter-arrival times have therefore been chosen to reflect this prior work. Though a wide spread of inter-arrival times has been tested a more select range which provides high levels of competition for resources will be discussed in this chapter. Inter-arrival rates will be below 334.14s with an inter-arrival time of 210.14s where it is not otherwise indicated.

3.6.3 Results & Evaluation

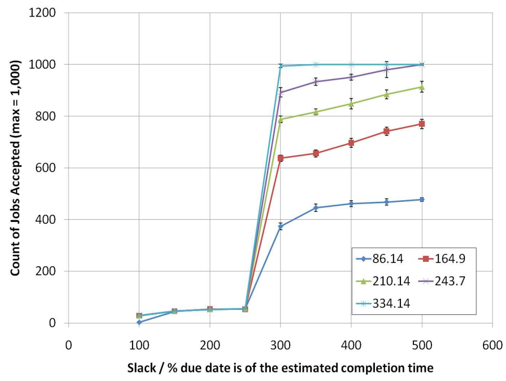
3.6.3.1 Job Acceptance and Slack

In Figure 3.16 the due date and deadline are set to be the same time, the focus will therefore be placed upon the effect of slack before this strict deadline. The effect a graduated due date to deadline period will then be demonstrated afterwards. The inter-arrival time is set at 210.14s, unless it was subject to change as part of experimentation (i.e. Figure 3.16b). This arrival rate has been determined to give a high level of competition between tasks. The markup is fixed at 10% (except in Figure 3.16b where 30% is also shown) and a medium workload per task and medium count of tasks per job were chosen. The broker was configured to sort the offers by earliest completion time first and then accept the topmost offer.

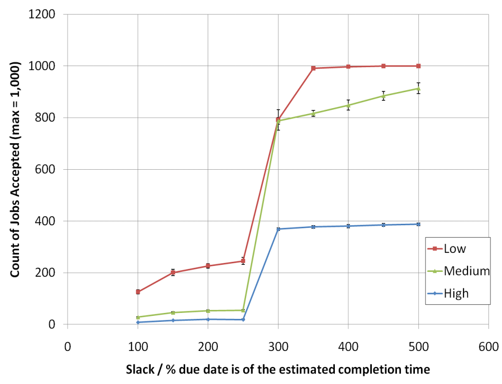
In Figure 3.16a two markups are shown, it can be seen that there is little distinguishing difference between them. The profit made which is not shown here differs in that the higher markup when compared with the lower markup makes more profit. The lack of



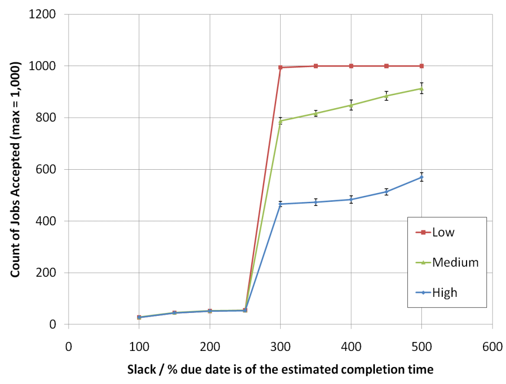
(a)



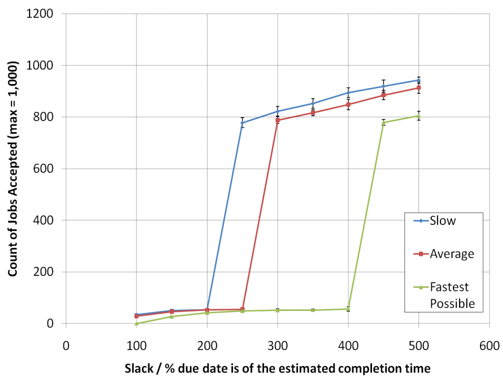
(b)



(c)



(d)



(e)

Figure 3.16: The Effects of Slack upon Job Acceptance with Varying a) Markup, b) Arrival Rate, c) Task Count, d) Task Size and e) Reference Machine Speed

difference between markups is placed down to the lack of economic pressure on the jobs. i.e. the budget constraint does not restrict the jobs, so there is limited competition between them.

A clear transition between 250% and 300% is observed. This is considered to be an aspect of competition, particularly in regard to temporal pressures. Firstly this experiment is conducted with due date and deadline that are equal, so it can therefore be said that at 100% no slack is permitted, so there would be no permitted start delay. The lack of any permitted start delay means jobs must be able to start straight away on a machine that is free of work. Given higher levels of slack then some delay can be tolerated and that this delay will be directly related to other jobs in the system and hence the competition between them. Jobs with greater acceptable start delay hence can be queued behind other jobs and can be more readily accepted in a schedule, slack therefore relates to how permissible this possibility is for a given job.

The arrival rate influences the amount of jobs accepted, but as shown in Figure 3.16b this only occurs after the transition point. It is noted that the 86.14 inter-arrival time trace is levelling out and is tending towards a maximum permissible amount of acceptable jobs, while 210.14 and 243.7 observe a much more linear relationship after this transition period. Finally the 334.14 trace has reached the maximum amount of jobs submitted after this transition. The effects shown here are because jobs have a fixed time window in which they must be completed, so high arrival rates ensure these windows overlap more and once the slack is increased more freedom of job placement permitted and fewer jobs have to be rejected.

The position of this transition point can be shown to not relate to the task heterogeneity i.e. to the task count (Figure 3.16c) or task size (Figure 3.16d).

In Figure 3.16c it is observed that before 2.5x slack the amount of tasks in a job distinguishes the traces, which is not seen in the other figures. The differences in gradients seen after the 3.0x point is also of interest as jobs with a lower amount of tasks per job are accepted much more readily, though this also relates to there simply been less work. Argument can still be made that if the slack is small then it helps if the job contains fewer tasks, especially before the transition point. This distinction occurs as there is a finite amount of machines over the average speed of the provider. The average speed was chosen as the mechanism for estimating completion time. Hence when more tasks are present more competition exists between the tasks, as there is a finite set of machines that are fast enough to complete the work in time, especially with low levels of slack.

Figure 3.16d like Figure 3.16c changes the amount of workload per job; it must therefore be asked why in Figure 3.16d before the transition shows no distinguish between the low, medium and high traces. The reason for this is placed down to the fact that the duration a job relates to the amount of work needing to be completed. In Figure 3.16d this therefore makes no difference as the size of the slack is related to the estimate of the completion time and hence to the task's size.

The start delay relative to the task length during the experimentation has been very limited. The initial slack therefore often mitigates the effect that competition and start delay causes. In Figure 3.16e we see an alternative use for slack and the reason for the sudden transition. The reference processor for estimating the job duration is seen to be responsible. Three separate reference speeds were chosen, namely that of the fastest possible machine 25,000, the average 16,333 and a slow machine 14,000. The slowest machines on a provider was 6,000 ops/sec.

Calculation Description	Calculation	Transition Point - % Due Date is of Estimated Completion Time
Fastest/Slowest	$\frac{25,000}{6,000}$	4.17
Average/Slowest	$\frac{16,000}{6,000}$	2.72
Slow/Slowest	$\frac{14,000}{6,000}$	2.33

Table 3.8: The Relative Speed Difference Between the Reference Machine and the Slowest machine

In Table 3.8 the reference machine's speed is compared to the slowest machine and in doing so indicates the location on the x axis of the transition point. This is achieved by the formula: $Transition\ Point\ As\ Percentage\ Of\ Due\ Date\ To\ Deadline\ Section = \frac{Reference\ Processor\ Speed}{Slowest\ Processor\ Speed}$

The choice of reference machine ensures tasks cannot be placed on machines slower than the reference until the slack is sufficient to compensate for the extra time taken. If the slack is always chosen to compensate for the heterogeneity in machines then job acceptance rates will be high given the temporal constraints. If the end user requests a small amount of slack then it can be assumed that they will require machines that are at least as fast as the reference machine. Figure 3.16e hence shows with a slower reference machine that jobs are accepted more readily, as more of the machines available are faster than the reference, so can complete the task within the time allotted as determined by the use of the reference machine.

3.6.3.2 The Effects of Gradient Based Deadlines

The aim of this experimentation it to show the effects of increasing the gap between the due date and deadline.

In this sections experimentation the inter-arrival time of jobs is set at 210.14s. The broker in this section sorts offers from providers as before by earliest completion time first. The filtering mechanism used is then changed. In Figure 3.17a and 3.17b the offers are then filtered so that the first profitable offer is chosen, which prevents unprofitable jobs from been accepted. This is similar to related work such as [148], given that faster processors in our experimentation cost more and that we are sorting by earliest first. This is experiment is also performed for a range of markups.

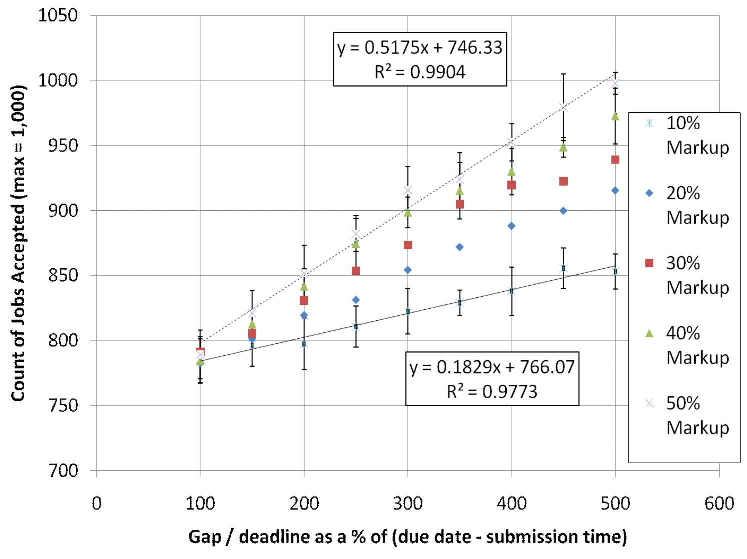
In Figure 3.17a follow a linear relationship, 10% and 50% have been marked on 3.17a to highlight this. Confidence intervals on 10, 40 and 50% markup runs have been added though for 20% and 30%'s error bars have been omitted but are broadly similar. It can be seen in Figure 3.17a when the deadline is set to be 5x (500%) that of the due date, that all jobs with 50% markup are accepted. This illustrates how higher markup encourages job acceptance and exacerbates the effect of the increased gap between due date and deadline. This occurs because the breakeven point is later in the 50% markup than it is with the 10% markup and in general the 50% markup run is more likely to make a profit on a given job.

The detrimental effects of accepting any profitable offers is shown in Figure 3.17b. The profit drops as the gap between due date and deadline increases. This is due to more jobs being accepted (Figure 3.17a) which makes it more likely that the next job to arrive will overshoot the due date. The increase in gap size effectively gives more time for job to complete, which generates an increase in competition for the finite resources available as more jobs can plausibly complete before the deadline. It can also be noted that when the gap between due date and deadline is small the rate at which the service price drops quickly, hence providing the steepness of the curve.

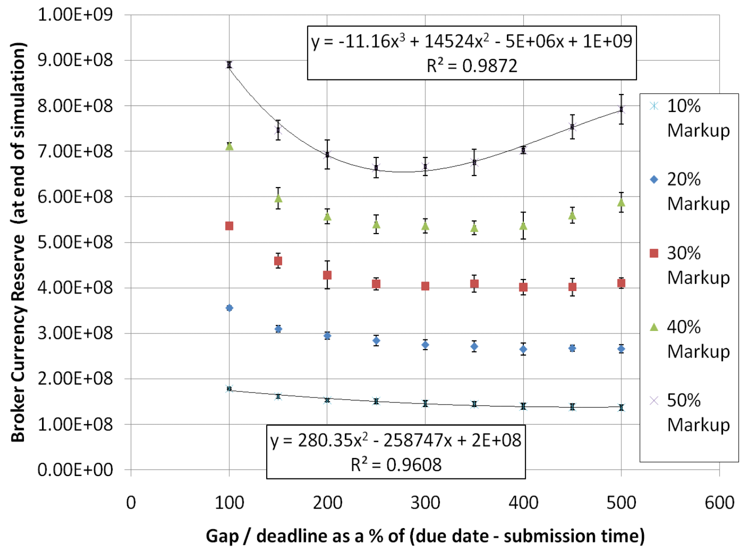
This downward trend in profit is eventually reversed as the gap increases. This is due to the broker having more time to complete the work before the breakeven point as the drop in the service price is slower and is aided by the acceptance of more jobs.

In Figure 3.18a and 3.18b the ISQoS Hybrid Offer Filter is used (see Section 4.4.7.2), with the intent of mitigating the detrimental effects on the broker's profit that was shown in Figure 3.17b. The results are shown only for a markup of 50% and the Filter takes the last 50 accepted jobs in order to establish the going rate.

In the Figure 3.18a fewer jobs can be seen to be accepted, when the threshold below the going rate was less than zero. A threshold less than zero means that jobs completing after the due date must have a higher rate of return than the average for the last 50 jobs.

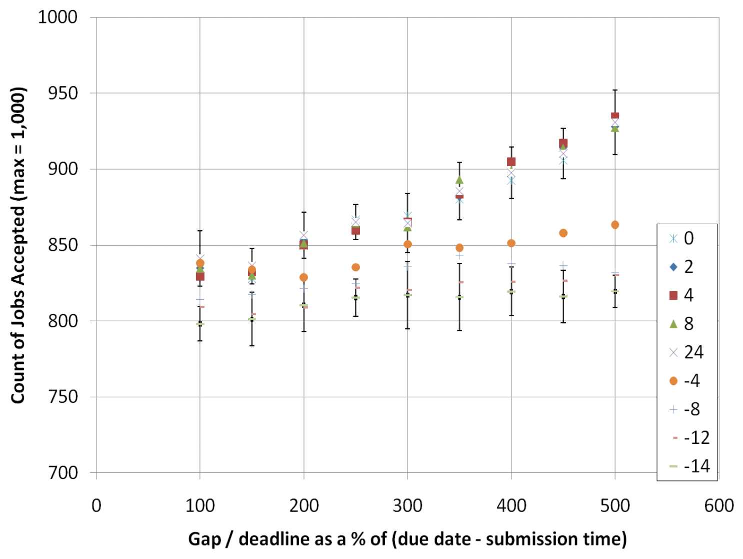


(a)

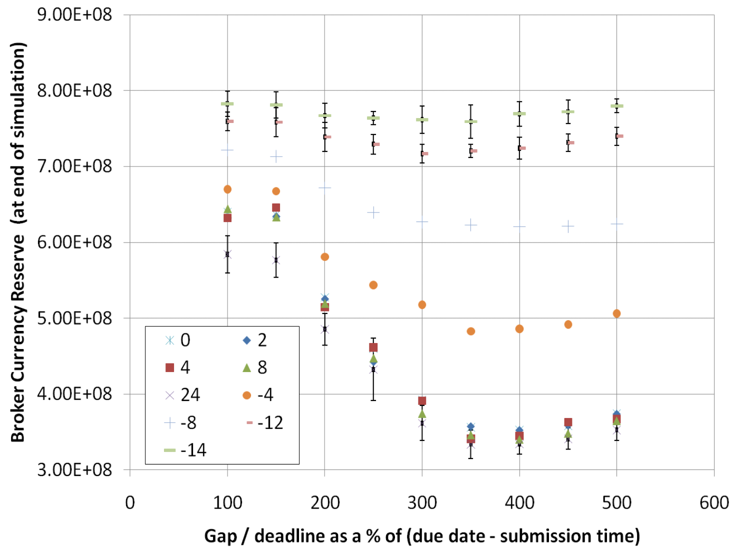


(b)

Figure 3.17: The Effect of Slack and Deadlines on Job Acceptance



(a)



(b)

Figure 3.18: The Effects of Intelligent Filtering

The broker's profit and hence the level of quality of service provided can be seen to increase in Figure 3.18b, when jobs that arrive after the completion time are required to have a higher rate of return. There remains some loss in comparison to not allowing a gradient section, but this has largely been mitigated. One way to guarantee mitigation would be to use the due date to deadline period solely as a recovery mechanism/a way to deal with completion time uncertainty. This would be completed by never accepting work that is expected to complete after the due date, though this removes the benefits of having a graduated deadline section.

3.7 Comparison Study

To summarise the discussion of similar models the Table 3.9 is presented. The criteria for judging in the summary are as given as follows with explanations of their importance. Using the chosen criteria the ISQoS variation presented in Section 3.3 demonstrates the advantages of its use.

Service Price Gradient Start Position Linked to Runtime: The user's preferences for completion time should not be based upon system centric properties such as slowdown.

Service Price Fixed to Client's Offering Price: The user is unlikely to know a great deal about the current market conditions, hence there needs guidance on how to set the correct budget for a job. ISQoS offers this in terms of a series of offers and counter offers. The user can hence see what budget is needed in order to get their job executed. The budget in ISQoS does not necessarily mean the final price the user is going to pay for the job either. The price should follow demand and should not be driven by users bidding blindly against each other. The returned offers in ISQoS can act in an advisory fashion indicating how far away a provider is from completing a job on time and on budget.

Bound Penalty Fee: This limits the providers loss, ensuring it can participate without signing a blank cheque.

Final Price Easy to Explain to End User: Simplicity to the end user is required. If a service price cannot easily be determined it diminishes the user's willingness to pay.

Table 3.9: Comparison of Related Pricing Models

Name	Runtime & Gradient Linked	Service Price & Client's Offering Price Linked	Penalty Fee Bound	Quickly Explainable Pricing
First Price [54]	Yes	Yes	Yes	Yes
First Reward [99]	Yes	Yes	No	Yes
Risk Reward [99]	Yes	Yes	No	Yes
First Profit [148]	No	Yes	Yes	Yes
First Opportunity [148]	No	Yes	Yes	Yes
First Opportunity Rate [148]	No	Yes	Yes	Yes
LibraSLA [52]	No	Yes	Optional	Yes
Aggregate Utility [17]	No	Yes	Yes	No
ISQoS	No	No	Yes	Yes

3.8 Summary

In this chapter the pricing model was introduced, as was a simulator that was written to test the viability of the model.

The pricing model allowed the service providers to set a resource cost which was then used to determine a service price. The price was setup to diminish by using a gradient based approach between a due date and deadline, which both incentivises QoS provision and binds the temporal and economic factors of jobs together. The due date and deadline were set to be independent of job completion time which avoided system centric measures such as slowdown. Jobs were also allowed to be constrained by assigning a maximum budget for job completion and by setting a mark-up, which can be used to generate prioritisation. In generating this model key measures for judging jobs were also expressed such as: due date slack, breakeven point slack and the deadline slack, as well as economic slack variations such as budget slack and budget resilience.

The simulator first of all was used in a parameter sweep mode that showed the outcome of a job with any given variation of settings.

The outcome of this was to provide the following key observations:

- Though the due date and deadline can change, when profit is assessed as a percentage through this transition period, it removes the focus on exact values of due date and deadline.
- The penalty fee needs capping and should not be too generous and that zero is reasonable.
- If the penalty fee is set to zero, hence the broker pays for resources used but pays no compensation to the end user then the broker will always breakeven at exactly the same percentage through the transition period for a given mark-up. This significantly aids comparability, as at the breakeven point becomes well defined and is independent of resource cost.
- If the penalty fee is not set to zero for a given mark-up a job will still have a point where the service price is equal to a second job with the same mark-up even if the resource cost is different, but this will not be at the breakeven point.

The simulator was then used in a discrete event mode that further tested the model, which gave the following outcomes:

- How slack is related to job acceptance counts and to the machine heterogeneity, i.e. that the amount of jobs accepted would suddenly increase once a transition point was reached which was shown by $\frac{ReferenceProcessorSpeed}{SlowestProcessorSpeed}$. This was due to the reference processor effectively specifying the minimum processor speed accepted when the deadline was closer to the completion time.
- That mark-up made jobs more preferential by moving the breakeven point, ensuring more jobs could be accepted.
- That gradient based deadlines were shown to cause a reduction in profit, as too many jobs would be accepted. A mechanism to alleviate this to use an assessment of the rate that profit was expected to accumulate for a given job and to compare this to a going rate.

Finally the ISQoS pricing model was compared in tabular form with existing models in order to highlight the differences that it has with existing models.

Chapter 4

ISQoS Broker

4.1 Introduction

This chapter outlines the implementation of the broker that was developed as part of the research. The main aims of this chapter are to: discuss brokerage and negotiation within ISQoS, highlight the architecture requirements that makes this possible and to present recommendations.

The next section 4.2 discusses the related work. This is then followed by the requirements analysis in Section 4.3 including discussion upon the architectural challenges that led to fresh development instead of extending existing work in Section 4.3.2.

The architecture is then discussed in Section 4.4 that enables the brokerage mechanism introduced in this chapter. This starts by providing an overview of the overall flow of events in the system in Section 4.4.1. This is followed by a discussion of the agreement process in the system and of the job and task representations. This is then followed by the provider's record of the Grid's state information, on which offers are based. The process of offer generation and committal is then discussed, followed by the scheduling algorithms, offer ranking and selection mechanisms and finally by how pricing occurs in ISQoS. This chapter is then concluded with a comparison study that discusses various different middleware offerings in Section 4.5.9.

4.2 Related Work

This section discusses the related work to the broker that is introduced in this chapter. A selection of brokers/schedulers and middleware is discussed in turn. They are placed into three categories and classed as either: best effort, quality of service oriented or economically oriented, in a similar fashion to the survey paper Merlo *et al.* [127], this has been chosen as a means to demonstrate progression between early best effort approaches and later efforts towards quality of service and market oriented approaches.

4.2.1 Best Effort

The best effort approaches are characterised by queue based scheduling that is centred around the resources, with the aiming of ensuring they remain busy. The listing given below includes examples of deployments where cluster/local schedulers have been adapted to work in a Grid context, rather than fresh developments aimed specifically at the Grid problem.

4.2.1.1 Condor & Condor/G

Condor [178] is a local job management system, that allows for submission to clusters at a localised level. Condor-G provides inter-domain job management services by using Condor's job submission features over a Globus controlled environment [178] hence acting as a meta-scheduler for Condor [127]. It works by using a matchmaking mechanism that uses ClassAds to describe both the available resources and the jobs requirements.

It remains an early example of meta-scheduling but has been criticised for being centralised, giving rise to a single point of failure, as well as making scheduling decisions assuming a total knowledge of the environment and assuming submission corresponds to an automatic execution on physical resources i.e. no delays occur due to staging of data [127].

4.2.1.2 Portable Batch System (PBS)

The Portable Batch System (PBS) is a batch queuing and workload management system that was originally developed by NASA [31]. PBS has a built-in FIFO scheduler whose sole objective is to maximize CPU utilization [160]. This scheduler simply loops through the queued job list and starts any job that fits in the available resources. In order to prevent large jobs from never starting, due to small jobs being more likely to be able to fit the available resources it implements a ”starving jobs” mechanism [31]. PBS is principally used for cluster scheduling but may be utilised by Globus Toolkit to schedule jobs.

4.2.1.3 Oracle Grid Engine

The Oracle Grid Engine [177] is similar to PBS in that it is a solution that aims to keep the resources busy. It hence is not oriented towards users quality of service needs. It starts by placing job is in a pending list, after which a scheduler is then periodically triggered so that the scheduling may be performed. During this scheduling it assigns the highest priority jobs in the pending list to the most appropriate resources available [177]. This gives it some notion of user orientation, though completion times and pricing are not present. It has two general classes of policy to select which jobs are most important in the pending queue, namely: ticket and urgency as well as an ability to define custom policies. The ticket mechanism gives each user an amount of tokens that can be assigned to jobs and hence acts as a primitive currency [118], while the urgency based policy however defines how long a job is permitted to wait, hence jobs waiting longer get higher priorities assigned to them [177].

4.2.1.4 Globus

The Globus Toolkit [77, 181] is often quoted as being the de-facto standard for the construction of Grid environments [76, 181]. It is principally a set of software packages that can be used to construct Grid environments and in presenting itself as a reusable tool-kit forms the basis of many other Grid middleware solutions. It including facilities such as Advance reservation, as implemented by ”General-Purpose Architecture for Reservation and Allocation” (GARA) [75]. It also has others modules for: data management, for resource allocation the module ”Grid Resource Access and Management” (GRAM) [73] and for monitoring and resource discovery the module ”Monitoring and Discovery Service” (MDS) [158, 204] and security.

4.2.2 Quality of Service

The quality of service approaches are characterised by a shift towards the user and the provisioning of Grid services. The two examples chosen here are because of their orientation towards scheduling and quality of service. OpenCCS [18, 20, 146] being an advanced scheduling platform, with extensions for strong resource resilience and risk awareness and Gridway [92, 93] offers meta-scheduling for the Grid.

4.2.2.1 OpenCCS (Paderborn Center for Parallel Computing)

OpenCCS [18,20,146] performs scheduling rather than by working in a queue based system. It comes with three scheduling strategies: First-Come-First-Serve (FCFS), shortest-job-first (SJF) and longest-job-first (LJF). It also has extensions for QoS which mainly focus upon risk awareness and management (AssessGrid [18]) and upon fault tolerance (HPC4U [90]) . OpenCCS requires shared storage and does not perform staging in or out as a result, but aims to merely perform scheduling [146]. This simplifies the scheduling process but limits some of its applicability to Grid infrastructures.

4.2.2.2 GridWay Metascheduler

Gridway [92,93] is a metascheduling solution that utilises Globus Toolkit and can therefore use scheduling solutions such as Oracle Grid Engine, PBS and Globus toolkit. It supports standards such as Distributed Resource Management Application API (DRMAA) [142] so that it can communicate with lower level schedulers such as Oracle Grid Engine [83].

A key feature of GridWay is that the scheduling algorithms can be swapped easily hence it is referred to as a meta-scheduler [60,83]. This ability to use custom scheduling algorithms is considered highly desirable for experimentation in that different algorithms may be tested within the same experimental setup. The default for GridWay however is a greedy approach using round robin [82,83] i.e. FIFO based scheduling [82].

Gridway enables QoS in terms of reliability via a performance monitor that periodically performs rescheduling in response to slowdown [92]. QoS support in GridWay also provides fault recovery mechanisms, dynamic scheduling, migration on request and opportunistic migration [110]. Extensions to GridWay have also been performed such as Scheduler in Advance Layer (SA-layer), that allows for Job rescheduling using algorithms such Bag of Tasks Rescheduling (BOT-R) and gap management for the limiting the resource fragmentation effects of advance reservations [185]. In [185] they also perform network aware rescheduling and take into account data transfers. This is highly desirable as network delays vary greatly between data-intensive and compute intensive applications. Gridway has however no economic model and it uses a single strict deadline and an earliest permitted start time. This therefore differs from the ISQoS pricing model that implements a due date and deadline pair, with no earliest start time.

4.2.3 Economic

Economic approaches are presented here to highlight the different efforts that have been made towards using economies to control the Grid and to provide self-management and quality of service for end users.

4.2.3.1 Nimrod/G

Nimrod/G is a hierarchical, decentralized, agent-based scheduler [107] for Grids, aimed at running parametric applications by using economic based scheduling [28], over a Globus based infrastructure [127].

It is however not a general solution for discovering resources and submitting general applications in a Grid environment, as it uses a restrictive parametric declarative language. It has a limited amount of primitive scheduling algorithms available [39] that are economically oriented. Criticism has also been levelled against Nimrod/G and GRACE based economics in that it does not support a comprehensive and complex market based economy [127]. Nimrod/G has however achieved some commercial interest, in the forked variation EnFuzion scheduler [28], therefore as an early example of economics it is important, but it did not manage to enable significant commercial take-up.

4.2.3.2 GridBus + Aneka

In [188] the GridBus broker [38] and Aneka [202] middleware are used to perform economically oriented negotiations for Bag of Task applications. This is quite advanced in that it can generate alternative offers in cases where the original request for work cannot be fulfilled and also uses a deadline and cost as QoS constraints, though it only has use for a single hard deadline [188]. During the submission stage it asks only one provider at once based upon cost constraints and other job requirements so does not establish a market mechanism by asking each provider to bid for work. This means if the first provider can accept the work then it is accepted even if another provider could have potentially given a better offer, which can be seen to be an oversight.

The Venugopal *et al.* [188] implementation is also not seen to be user oriented and agreements are heavily focussed upon resources oriented requirements of jobs. They detail the reward for fulfilling an offer and penalty for not, along with very rigid descriptions of reservation windows specifying how many CPUs of what speed, with a start and end time for the reservation. This lacks flexibility and an approach that does not explicitly indicate which reservations are needed and merely states when the user wants the job completing by, in a graduated manner is seen as a better approach to negotiation.

4.2.3.3 GEMSS

GEMSS [128] is a service oriented Grid aimed at supporting medical simulations. It uses scheduling and supports use of the Maui [31,100] and COSY [46] schedulers for this purpose. WSLA [105] is used for SLA formation in GEMSS for both advance reservations of resources and for the agreement between providers and clients.

The economic model driving GEMSS uses a start and single deadline for jobs, as well as a jobs budget, thus differing from the ISQoS economic model. In having only a single deadline value and no gradient it somewhat lacks the definitive economic reason for a provider to complete work on time, given no means to penalise delays. GEMSS facilitates swappable resource pricing and can perform pricing based upon either a fixed rate or dynamic load.

GEMSS uses a reverse English auction for establishing the service price between the provider and client which is used in provider selection. This price is used along with a weighted ranking mechanism for the comparison of the job offers QoS parameters. The QoS parameters in use are a jobs start time, completion time and service price. Though there does not seem a strong reason for comparing jobs using weighted ranking, other than that the economic model is not particularly complex, i.e. not incentivising timeliness in a graduated fashion.

4.2.3.4 Resource Aware Policy Administrator

Resource Aware Policy Administrator (RAPA) [85] is a Grid resource manager that aims to create SLA's for job admission and is based upon the Globus toolkit. It is similar to work presented in this chapter but has substantial limitations. It like the work here is deadline based with economics. It however has an inflexible market model. Given the proximity in nature it is discussed in detail below.

Jobs in RAPA regarding deadlines may be one of two types, either hard or soft, soft deadlines means the user can accommodate delay, which is fixed to a maximum of +20% of the user's deadline. This 20% is highly restrictive and does not reflect the users true deadline requirements. It principally derives from ensuring the middleware does not run out of money, as the profit policy used allows for notionally unlimited loss as shown by the following formula [85]:

$$profit = maximum\,reward - a\,penalty\,factor \times (completion\,time - deadline) - resource\,cost$$

Unlike in the ISQoS market model (see: Chapter 3) where the users preferences determines the gradient of the loss and it is up to the broker to decide if the risk of accepting the job is worth it. It is also limited in that resource usage is allocated by a proportional share mechanism whereas it could be hoped that various scheduling mechanisms could be made available.

The budget like in Nimrod/G is specified by the user, and represents the maximum the user is willing to pay. In ISQoS the provider's resource pricing mechanism determines the cost of the resources and the user is billed accordingly up to their budget, thus allowing the price to follow demand. In RAPA and Nimrod/G however, this is simply accepted as the amount the user pays if all goes well, thus requiring some mechanism to ensure the users provide a sensible informed value such as an auction.

4.2.3.5 GridEcon

GridEcon [8, 9] aimed to define a model for managing complex market-based service oriented Grids, introducing new stakeholders such as risk brokers, SLA monitors and price modellers [127], which provided value added services such as capacity planning and insurance contracts [9].

In the economic framework it provided features such as Bid and Asks, which is similar to the Condor ClassAd mechanism. Bids in GridEcon representing the resources asked for by the buyer and Asks representing the resources made available by the providers. An important element of matching resources to jobs is ensuring the units of trade are common such as asking for n virtual machines for *m* hours [9].

GridEcon is flexible in that it provides scope for developing either spot markets, where resources are sold and used almost immediately or futures markets, where future capacity is traded, as part of capacity planning.

It used a top down approach, defining business models around the candidate application and the high level goods to be traded. The lower levels of the Grid infrastructure had the liability to meet the demands to establish the higher level services that was traded. This approach may suffer from the semantic gap between high level application requirements and low level resource characteristics [127].

4.3 Requirements Analysis

4.3.1 Requirements

The starting point for any implementation is its requirements, these are therefore listed below:

QoS Cost and Time: To provider an architecture that can be used to support QoS in terms of job completion time and cost constraints.

Reflect Both Users and Resource Providers: The Broker should be able to act in a none system-centric fashion. It should be able to reflect the user's requirements and express them in a way that gives adaptability to the provider.

Reservation Based Allocation: This is a central requirement of planning using techniques used in operational research. In meeting this requirement it allows domain experts in scheduling to contribute to the Grid community.

Scheduling API: Provide a mechanism to swap scheduling algorithms out easily and an API that can be used without underlying expertise in Grids.

Swappable Resource Pricing: To support a wide range of experimentation the provider's resource pricing mechanisms should be replaceable.

Swappable Offer Ranking and Selection: To support a wide range of experimentation the broker's ranking and selection mechanisms should be replaceable.

SLA: Once a plan of action has been constructed, it should be able to be used as the basis of guarantees on QoS, provided by a service level agreement.

SLA Negotiation: Negotiation needs to be used as a mechanism to advise the end user of the current state of the Grid. This can be achieved through a series of offers and counter-offers made between the broker and providers. The counter-offers made by the providers, should the job submission be impossible to accept should assert how far it is from being accepted. This can therefore be used to advise end users as they re-evaluate their jobs submissions based upon the current environment.

4.3.2 Extension vs Fresh Development

As part of this research there was a requirement to implement any proposed solution as a demonstration that the requirements can be fulfilled. The development process therefore requires careful consideration between either extending existing work or developing from scratch.

The outcome was to develop from a fresh start. There were several key reasons for developing in such a way. These link strongly to the broker's requirements and what separates it from existing infrastructure. A key aspect of development however is the mitigation of risk, so consideration is placed upon how existing software can be used to support the broker's development.

The broker was expected to form agreements in regards to quality of service, with a focus upon completion time and cost. The current most widely used standard for agreement formation is WS-Agreement [141]. Its message passing structure is XML based, creating a technical requirement for the use of XML, in the message contents e.g. the job's description.

In regards to job admission the most relevant standard is Job Submission Description Language (JSDL) [139] which is an XML based language for describing jobs. The utilisation of this standard is considered to improve the software's compatibility and increase the project's overall relevance. There is however common middleware like Globus [74] that uses Resource Specification Language(RSL) [181], which in most variations uses a simple mappings between properties and a given value. This not being XML based means there is no clear path, to place the job's description into the agreement. In considering alternatives GridSAM [126] potentially offered a way to integrate into a Globus environment as it supports JSDL and has some ability to translate JSDL into RSL, but this route suffered technical problems and would only work by using adaptors rather than following common standards, which is more desirable.

In examining implementations of the WS-Agreement protocol, the most favoured was WS-Agreement for Java (WSAG4J) [91], which was a generic framework that also supported negotiation [143]. It is deployed from a Tomcat based container, thus it would also requires substantial bridging into the Globus Toolkit 4 (GT4) environment that was immediately available within the school. GT4 and WS-Agreement are also not contemporary and use different versions of WS-Addressing [21] which makes development difficult forcing it to focus on solving incompatibility issues rather than research.

Globus along with many other Grid middlewares primarily uses queue based mechanisms and is batch oriented. It hence does not fit into the projects scope for scheduling and the use of operational research techniques. There are a few environments that do support scheduling such as Maui Scheduler [31, 100] and OpenCCS [18, 20, 146].

In terms of the brokered job submission it was to going to have to be achieved by negotiation. A negotiation mechanism would need to assess job submissions to determine if they could be completed within the required constraints of the end user. This would then led to an offer accepting the work with execution details or if this could not be done then a counter-offer would need to be constructed. Offers in needing to be realistic must be based upon a candidate schedule that is derived from the current schedule. Existing middleware however, uses the scheduling solely for allocating jobs to resources. The ISQoS provider's however would need to use schedules to advise what offer can be made as part of the negotiation. This therefore makes it difficult to acquire a candidate schedule without submitting the job for execution in existing middleware solutions.

It was therefore decided given such differences to develop from scratch and where possible take software or parts of software where applicable.

4.4 Architecture

In this section we discuss the overall architecture of the ISQoS Broker, first of all an overview of the broker is presented followed by discussion of individual parts afterwards.

4.4.1 Overall Architecture

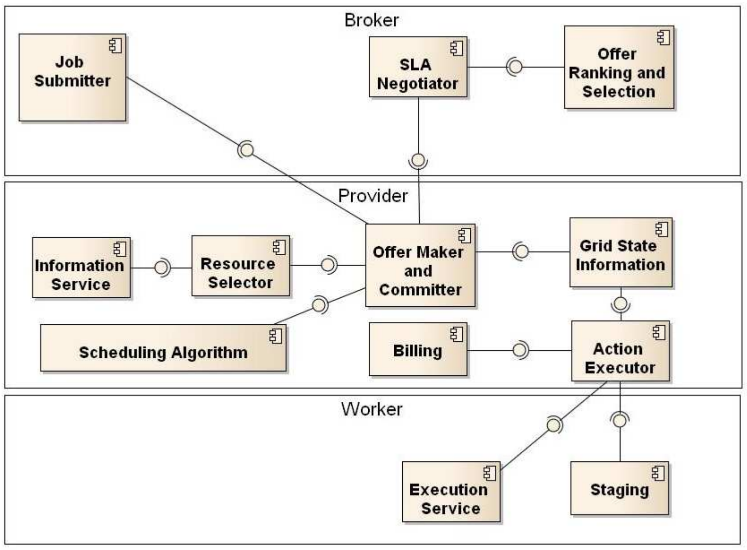


Figure 4.1: Overview of the ISQoS System

An overview of the broker's architecture is presented in Figure 4.1. It primarily consists of three tiers: A broker, a provider and workers. The broker selects the provider to use while the provider commands the workers to execute Grid jobs. The components shown in Figure 4.1 are utilised in the following fashion.

Jobs are submitted via the job submitter of the broker to the offer maker and committer component of each provider. The broker is expected to contact various different resource providers in order to establish a competitive marketplace i.e. multiple providers will compete for work in a tender market.

The information submitted to providers by the broker consists of the job's *resource requirements*, a *budget* that the user has assigned for the completion of a job, the amount of *mark-up* the broker makes for the service, which can be used as a notion of minimum available funds for rescheduling and finally its temporal requirements. These are shown as a *due date* by which the job should be completed by and a *deadline* by when it must be completed. Further details of the agreements structure can be found in Section 4.4.2.

The resource providers from this information perform the initial scheduling of the tasks within a job so they can derive an estimate for completion time and price. A job is considered to be made up of a set of tasks that are all independent of one another. This is because the broker is primarily aimed at parameter sweep / bag of task applications [41], which are the predominant workload upon the Grid [95].

The providers derive their estimates by using a scheduling algorithm (Section 4.4.6) and pricing mechanism (Section 4.4.8.1) that is plugged into the provider. The price is set by the provider's local pricing mechanism which feeds information into the scheduling algorithm. The scheduler has available to it the current Grid state information and a resource selector component that is capable of selecting the viable set of resources for computing each task within a job.

The schedules that are generated are converted into offers to complete the work, which are then submitted back to the broker. These offers include information about the time and cost for completing the job and the time for completing each task within a job.

The broker from the offers that have been returned from the tender market, ranks them and filters out the poor offers i.e. sort by earliest completion time first and filter out unprofitable offers, though various selection mechanisms may be used, which are shown in Section 4.4.7. The broker then takes the ”best offer” as determined by ranking and filtering and asks the user if it acceptable to proceed with the job at a given price and completion time.

If the offer made to the user is acceptable the broker then submits the bag of tasks that make up the job to the winning provider. The winning provider then schedules the work for a final time and then records the new state of the Grid. In the case an offer is not acceptable the user is permitted to resubmit the job with different budget and time requirements in order to acquire a different offer, that may be more acceptable.

Accepted jobs are placed into the schedule which is part of the current state information of the Grid. The state records the mappings between workers and their jobs as well as the current resource and job statuses. Workers are represented as objects which maintain a copy of the XML description of the resource and a reference to the XML parser used to interpret the description. The Worker's description parser essentially maintains a set of default questions which can be asked about the job i.e. what is the size of the memory available? or what is the CPU speed? Jobs are also represented by objects that contain XML descriptions taken from the job's associated agreement. The job description is attached to the agreement as an `xsd:any type`, which allows for XML other than JSDL documents to be used. The descriptions of the tasks are treated in a similar fashion to the resources and a basic list of default questions are provided i.e. how much memory is needed. The resource selector for a given task can then be asked to provide the list of acceptable workers to the scheduler, without it needing to have a great understanding of what a resource entails.

A job's description breakdowns down as follows: a job is a collection of tasks in the bag, that need executing. Each task has several actions, these equate to: *stage in*, *execute*, *stage-out* and *clean* the worker. Stage-out merely holds to the meaning of transferring data away from the worker. Clean removes the task's working area upon a worker node. If the clean event is the very last action to be performed for the job then the job is submitted for billing.

The action executor merely waits for the next action in the schedule to be ready for execution. At which point it sends a signal to the worker nodes to begin their work.

4.4.2 Agreement Structure

In this section we discuss the structure of the agreements that are formed in the ISQoS System. The ISQoS Grid architecture is service oriented and is based upon WS-Agreement for Java (WSAG4J) [91]. It is designed for each provider to present a job execution service that can be negotiated with. The use of WS-Agreement means requests for work are responded to by providers in the form of offers to complete the work. These offers then can be negotiated upon within a tender contract market. The negotiation and precise structure of these offers is shown in Figure 4.2.

The ISQoS agreement structure like all WS-Agreement based documents has a name assigned to the offer as well as its agreement context. The agreement context details information such as the participants belonging to the agreement. The ISQoS Agreement because it is based upon WS-Negotiation also has a negotiation id and a negotiation context.

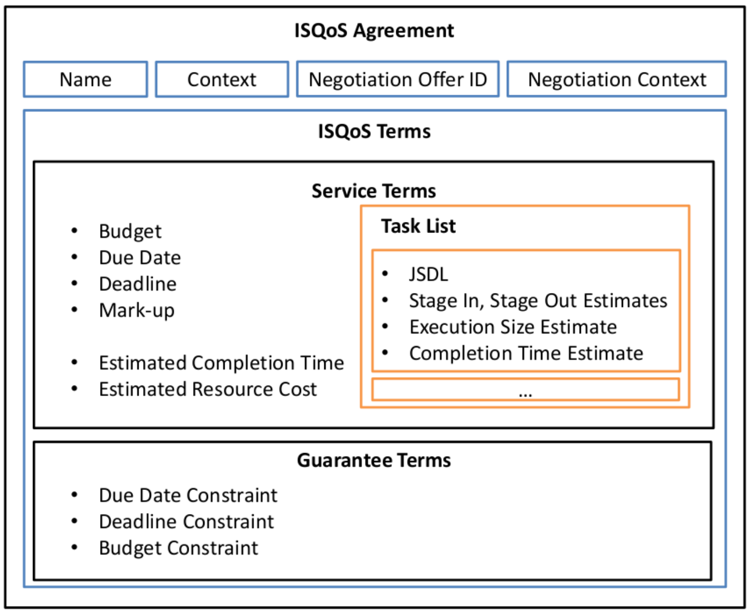


Figure 4.2: ISQoS SLA Structure

The context in terms of the agreement formation process is useful. The providers respond to offers by scheduling, after they have scheduled, they can make an offer. If however the schedule indicates the work cannot meet all the constraints the negotiation context can be set to advisory. This means further work is needed in the negotiation for both sides to agree to execute the job. This allows the ISQoS system to offer advice to the end user about the current state of the market, in terms of price and current workload of the Grid. This of course is expressed in terms of the job the user wants to execute and not a generic statement about a provider's current load characteristics.

In the service terms the jobs, budget, due date and deadline requirements are given. There is also a list of tasks in the job that the user wishes to execute, with estimates of size of execution that is required and the amount of data that is required for staging in and out. The broker also indicates to the providers what its mark-up is, i.e. its commission. This allows the providers to avoid any schedules that are likely to be rejected and informs providers of the boundaries around the budget and time constraints. These boundaries were previously demonstrated in Section 3.3.4.

The broker focuses upon standards such as JSDL [139]. The agreement document however maintains the capacity to swap out the term language used to describe the work being performed. Thus it need only be XML based so that it can form part of the agreement. This is achieved by having the part of the XML describing jobs contain an `xsd:any type` for attaching the term language describing the job. This flexibility is also present in the information provider which can use either GLUE [36] or Ganglia [81] based representations of resources, though this will be discussed later.

The final parts of the service terms are for the provider to fill in. Providers are required to give an estimated completion time for each task in the job, along with an overall completion time and service price. The values provided here hence can be used by the broker after offers have been returned from the tender market and are the basis of assessments made by the broker, that is shown in Section 4.4.7.

4.4.3 Job, Task and Resource Representations

In the previous section the transmission of data between broker and provider was discussed. In this section the storage of information about the agreements made and the Grid is discussed. In particular the representations of jobs, tasks and resources that the providers maintain is discussed and how it leads to a more generic way of handling different types of descriptions of the key actors in the Grid system.

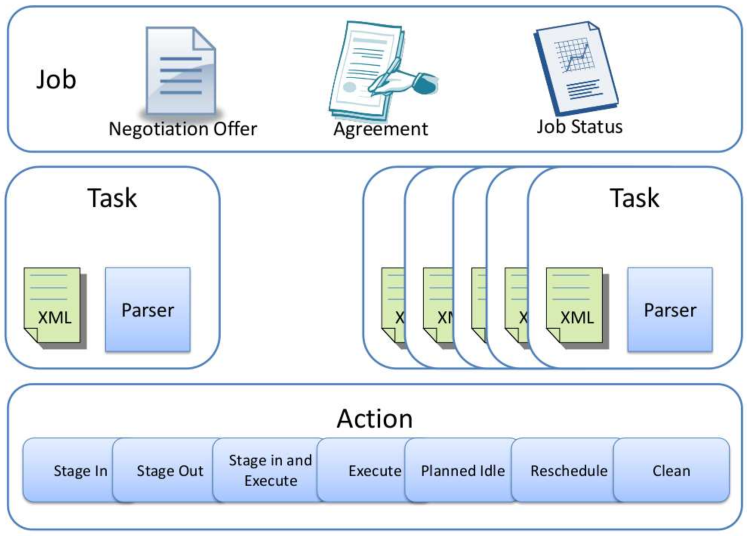


Figure 4.3: Job, Task and Action's Internal Representations within an ISQoS Provider

The topmost description of work that is agreed to in an agreement is the job. This is then composed of a set of tasks and each of these tasks is composed of a set of actions as shown by Figure 4.3.

The job covers the topmost issues in regards to the execution of work on the Grid. It maintains a copy of the agreement and the negotiation offer that caused the work to be accepted, as well as maintaining a copy of status reports for individual tasks, which are used for determining if the job has completed. A copy of the agreement and the negotiation offer are notably required to be stored. During the initial formation of the job no agreement exists. The agreement is only formed after broker and client agree, so at least one candidate schedule used to advise about the Grids status and one schedule that is the final plan of action must be generated before an agreement is associated with a job.

The second tier down is the task, jobs are made up of a set of tasks. Though the mappings are such that only a task records its association to a job so that task mappings to jobs can change between states. i.e. so that tasks can have their details altered such as start time and completion time etc.

Tasks along with a reference to the job that they belong to contain an XML description of a job and a reference to a parser that is capable of understanding such a description and answering a set of questions about such the task. The XML description of each task is a JSDL document and an associated parser is available in order to interpret the XML description. Structured in this way it becomes relatively simple to adapt to later or custom job submission standards by allowing the term language to be swapped out.

The lowest tier of the job and task representations is that of the actions that must be performed in order to complete the work. Actions are to be held in an event queue on the provider waiting to be executed. The full set of available actions are shown namely: *stage-in*, *stage-in and execute*, *execute*, *stage out* and *clean* as well as *planned idle time* and *reschedule* actions. The *planned idle time* action allows a scheduler to force a worker to do nothing, which may simply be used to keep the resource from doing other things while a data transfer is in progress. The reschedule action allows the scheduler to place into its schedule an event that forces it to consider rescheduling work. This can be useful in that a scheduler could be configured to reschedule periodically but dynamically change the rate it reschedules depending upon the current state of the Grid. The rate by which rescheduling occurs might be changed for example due to: an increased chance of resource failure or a decrease in workload or that the last rescheduling event was not very productive.

The resources within the Grid are shown in Figure 4.4. it like tasks is designed to wrap around an XML description of a resource and present a unified realisation of such a resource. It therefore has an XML parser and XML description in a similar way to the tasks. Aside from this xml stored information some basic information is taken from the XML and stored ready for use, namely: the resource name, its ip address and its speed.

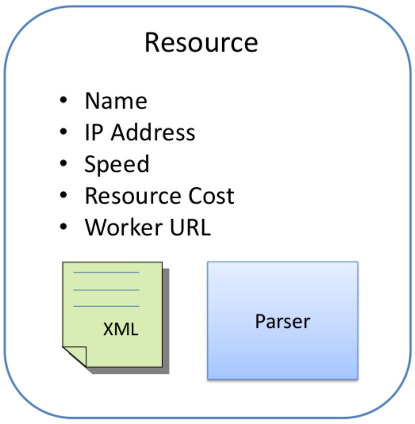


Figure 4.4: Resources

4.4.4 Grid State Representation

In the last section the description of the jobs, tasks, actions and resources was discussed. These together describe the allocation of a single agreement's work to a set of resources that are used during execution. In this section the record of the overall ”state of the Grid” is discussed. This is expected to assist in the understanding of the offer generation phase of the providers that is discussed in the next section.

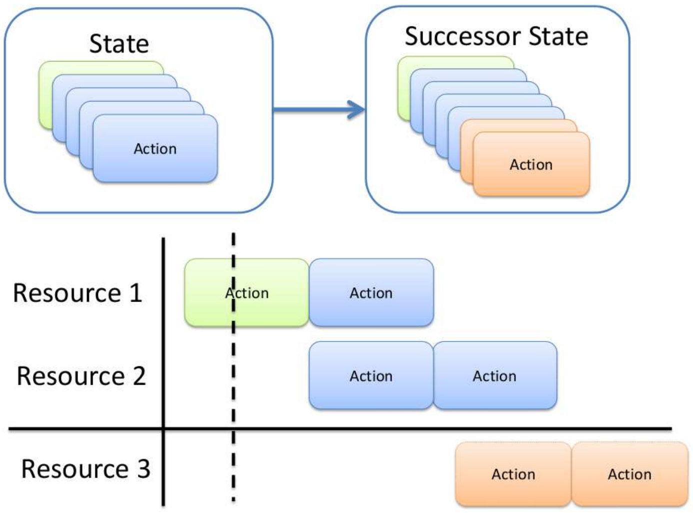


Figure 4.5: State

In ISQoS rather than using a straight forward queuing mechanism a schedule is used instead. The usage of a schedule allows for the ability to guarantee the quality of service offered in terms of time and cost, so long as reasonable estimates or at least overestimates on job size can be provided by the end user.

A schedule is basically a variation upon a queue, essentially events/actions are recorded in sequential order and are acted upon at a predetermined time. The job size being needed to find the correct to execute any given action. The main record of the schedule in ISQoS is called the State, which is shown in Figure 4.5. The main part of the State is hence an action queue which is basically a sorted linked list of actions that need to be executed. The next item to be executed in the queue can hence be recovered from this list of actions. Once an action is executed, it is flagged as being underway and is only removed once the job is completed and cleaned from the schedule.

The state illustrates ISQoS further perceptions of the nature of a schedule. The most important of these is demonstrated by the notion of a successor state. States can be shallowly cloned ensuring that derivative copies of the state can be made. The key principle this starts to demonstrate is that new candidate schedules/states are derivatives of the previous state. Essentially the newly derived state S' is directly related to the previous state, S and that it is the previous state overwritten by the changes the scheduler decided, i.e. $S' = S \oplus \{allchanges\}$. An example of this is illustrated in Figure 4.5, with the addition of two new actions.

States can be compared to one another, such comparisons includes the tests against the set of actions in a state such as: the absolute difference, set difference, intersection and union, as well as tests for is subset and is superset for determining the succession of states. Two of these tests can be used to derive a notion of change between two states, namely the absolute difference and set difference. To clarify the how these differ: set difference is merely A - B whereas the absolute difference is (A ∪ B) - (A ∩ B). This emphasises the notion of states/schedules succeeding one another.

States can provide subsets of actions in their actions list for machines, tasks, jobs and all actions in the schedule/state. This brings about specific information about parts of the overall schedule, such as the actions to execute on a given resource, or the resource where a given task is to be placed.

States can also be assessed for inclusiveness of particular machines, actions, tasks and jobs. A state also has a concept of coverage, in particular of its dependence upon a given set of resources for the schedule it represents. This reliance gives some notion of how many different resources are required to stay operational in order for the schedule to be carried out and hence its reliability. This is illustrated in Figure 4.5, in that two new actions are added to the State and the schedule that it represents gains an additional resource (i.e. resource 3) in its representation. In terms of resource discovery it is still required to make available all resources that are accessible, however if they are not component parts of the schedule they need not get recorded in the state and their current status also need not be monitored.

States in sharing actions between them require a concept of moveability of actions, firstly as actions are shallowly cloned between states if one action is changed it may affect multiple states. It is therefore required to clone certain states or part of them when rescheduling. This can be achieved for all actions belonging to a resource, to a task or job, or simply to an individual action. Once moved they merely need to be repositioned in the given states action queue.

4.4.5 Offer Generation and Committal

In this section the providers scheduling mechanisms structure is discussed and how this has an effect on the overall system. Avoiding classical architectures such as Globus [74] has allowed the focus upon scheduling and the ability to negotiate service level agreements (SLAs). This includes the capacity to perform scheduling for indicative purposes only, i.e. not for committing to work, but merely to generate a candidate schedule for negotiation purposes. The use of scheduling in the architecture also brings it closer in style to the Maui Scheduler [100] and OpenCCS [18, 20, 146] opposed to queuing based methods.

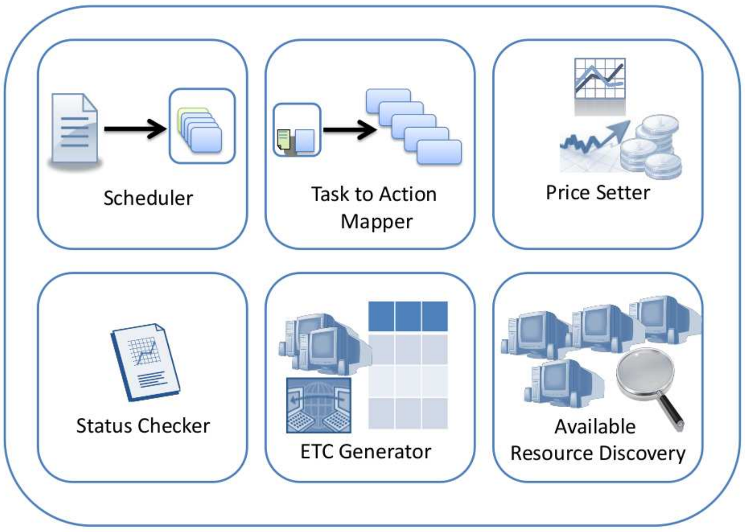


Figure 4.6: The Elements of the Scheduling System

In Figure 4.6 the overall parts of the scheduling component can be seen that drives the offer generation. All algorithms implemented in Section 4.4.6 slot into this scheduling environment. The algorithms drive the agreement making process. The aim of the algorithm is to take a negotiation offer and the current state and to generate a new candidate state or set of states. An algorithm can be asked to give a final schedule ready for committing i.e. return a single schedule only, or it can be allowed to generate multiple schedules that can be converted into several offers. As the scheduler follows the concept of single purpose, it does not carry out any further objectives, so it does not distinguish between making an offer and committing to a schedule. It is the rest of the environment that automatically generates either an offer or commits to implementing the given schedule. This presents the requirement that if middleware was made to implement an ISQoS brokering interface, their scheduling component would equally be required to be separated from the element that executes a given schedule.

The scheduling component is only intended to perform scheduling for negotiation offers. This is important in terms of timing of when the scheduling is performed, this is because a provider cannot agree to an offer and then schedule it. It must have a plan of action first. It is however required for the jobs in the schedule once the agreement has formed to have a reference to such an agreement. This therefore means once the agreement is made the representation needs updating with the correct reference. This is hence performed as part of the last actions of the create agreement action process and places a reference to the agreement in the job. The jobs structure can be seen in Figure 4.3, where both references to the negotiation offer and the agreement can be found. The flow of events in the offer maker and committer component are shown in Figure 4.6.

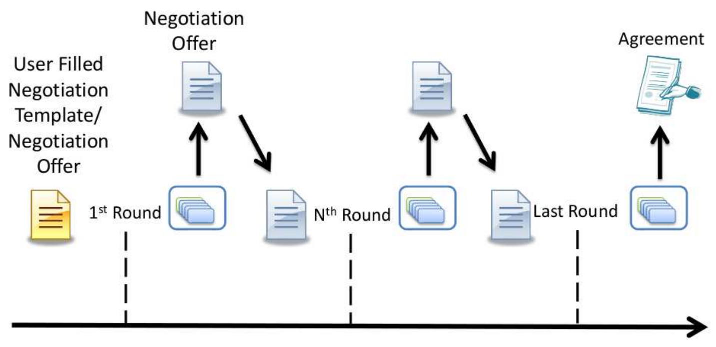


Figure 4.7: The Flow of Events in the Offer Maker and Committer Component

The broker first of all asks the provider for a negotiation template. This is then filled in and submitted to the provider. The provider then forwards these details of to the offer maker and committer component. The first round of scheduling generates a state. This is then cast into an offer that is to be sent back to the broker for assessment. After several rounds or just one round the job is able to be submitted. The user tells the broker they are happy to proceed and the broker submits the work to the winning provider. This provider then takes the agreement offer, creates a new state object, converts this to an agreement and updates the copy of the current working state/schedule.

In order to make the ISQoS more usable partial implementations have been provided for the scheduling algorithms component. The first partial implementation is for scheduling algorithms and the second is for algorithms that additionally perform rescheduling. These implementations in part deal with scheduling to generate a single final offer, requiring that algorithm authors only implement a single copy of the scheduling algorithm, that has the potential to provide many offers. These partial implementations also present a mechanism for translating offers into a set of tasks that need to be completed in order to complete the job.

The partial implementation for rescheduling algorithms additionally adds services that allows for tasks to be moved about within the schedule. This includes fitness tests to ensure inconsistent states are not generated. A mechanism that decouples tasks representations in one state from other states is also provided, so that states can be modified independently is provided. This decoupling occurs because so long as new schedules only extend the current schedule there is no need to re-record the position of old tasks within the schedule. Hence actions and tasks can belong to multiple states. If a task is going to be moved however it is required to be cloned in order to ensure the current working state is not unduly affected.

Components are made available to scheduling algorithms that provide services for the price setting of resources and the acquisition of the available set of resources. This therefore means scheduling algorithms do not need to reimplement either pricing or resource acceptability filtering services and they may merely be plugged in.

Scheduling algorithms can utilise additional helper features that have been implemented in the ISQoS scheduling API. The first of these helper features is a task to action mapper, this provides a facility for generating a set of new actions that belong to a task i.e. the *stage in*, *stage in and execute*, *execute*, *stage out* and *clean* actions. It then has the ability to remove certain elements from the generated set. This is useful as staging in with a separate execute action can be used or the stage in and execute action can be used instead. The latter merges the two actions causing the worker to start execution once staging is complete.

The second of these helper features is aimed towards expected time to compute matrix [7] based algorithms. The module provides generic capacities for recording estimates on action durations. It includes the ability to generate expected time to compute and transfer matrices for various situations, which are listed below:

- For a given action a list of estimates of expected durations on each resource within a specified list.
- For a given machine it can generate a list of estimates of task durations for a given resource.
- For a set of actions and machines, it generates the matrix of expected durations for completion.

Finally facilities in this helper module are made available for the summing of durations and costs over a set of expected time to compute collections, which is useful for ensuring that a task's actions do not violate the jobs temporal and budget constraints.

The last of these helper features is the status checker. This is useful both internally to the provider in terms of confirming job status and initiating billing, but it is also useful to scheduling algorithms that perform rescheduling for resilience purposes. It offers facilities to check the status of either an action, task or job.

In extension to this work it could be envisaged that an additional component would be added to this broker in terms of trustworthiness of providers. This would be in keeping with existing literature such as AssessGrid [18]. An offer ranking/filtering mechanism would then make use of such a component to alter the broker's preference towards dishonest/unreliable provider's offers. The aim would be make dishonesty among providers less preferential and would help ensure that the provider's kept to their promises to complete the work on time and on budget. Failures to meet the promised QoS could then be handled ensuring the broker is likely to make more profit and the users gain more utility and assurance over job completion.

4.4.6 Scheduling Algorithms

The providers use scheduling algorithms to construct schedules, for the purpose of making offers and committing to work. The providers are flexible in which algorithms may be used. In order to allow for a range of experimentation and to show this flexibility a default set of algorithms have been implemented. Algorithms from related work such as Nimrod/G where chosen for this default set, as well as common mechanisms such as round robin and earliest completion time. The Cost Roulette Wheel, Cost Rate Roulette Wheel and Cost Per Cycle Optimize algorithms are presented as new here. These are principally implemented to demonstrate the adaptability of the implementation and to demonstrate it as a platform for algorithm research. A Round Robin variation that implements rescheduling is also implemented to assist experimentation that involves rescheduling. The initial available set of algorithms are hence listed as follows:

Nimrod/G's Cost Optimize: [37, 39, 42] This is an algorithm implemented in Nimrod/G and is shown in Algorithm 1. This algorithm performs no load levelling and simply assigns work to the cheapest resource available until it is full. This has a significant drawback in that if the due date/deadline is sufficiently far in the future then there is no incentive to assign to any resource other than the cheapest. This makes the more expensive resources more likely to be idle and the free space on these resources are simply lost.

Algorithm 1 Nimrod/G's Cost Optimize

```
For Each Resource Available order by: cost non-decreasing order
  While Unassigned Tasks == true Do
    \\In ISQoS due date instead of deadline;
    If assigning next task does not pass the deadline Then
      Assign Task to Resource;
    End If
  End While
End For
```

Nimrod/G's Time Optimize: [37, 39] This acts like the earliest completion time scheduling algorithm but considers cost, ensuring the budget is not overrun, it is shown in Algorithm 2.

Algorithm 2 Nimrod/G's Time Optimize

```
For Each Resource Available order by: completion time non-decreasing order
  While Unassigned Tasks == true Do
    get 1st resource where cost per task is $\leq$ remaining budget per task
    Assign to the resource found
  End While
End For
```

Nimrod/G's Cost Time Optimize: [39,43] This algorithm is an improvement upon the cost optimize algorithm. It performs grouping on cost. It then selects the fastest machine within the cost group. This avoids some of the poor load levelling behaviour of the cost optimize scheduling algorithm, by ensuring it assigns work to all resources with the same price. This however will not cope with costs on a continuous range and a cost that is very marginally different will be placed into a different grouping. It is shown in algorithm 3.

Algorithm 3 Nimrod/G's Cost Time Optimize

```
For Each Resource Available order by: cost in non-decreasing order
    Group resources by Cost and retain groups for later use
    \\resources belonging to the same group require exactly the same cost
End For
For Each Resource Group order by: cost in non-decreasing order
    For Each (task yet to be assigned) Do
        For Each (resource in the group) Do
            Calculate completion time;
        End For
        Assign task to fastest resource;
    End For
End For
```

Cost Per Cycle Optimize: This is similar to Nimrod/G's Cost Optimization algorithm, however it sorts by cost per CPU cycle instead. This is considered to be a better way of performing the allocation, though it still however lacks load levelling features.

Cost Roulette Wheel: This allocates work to the Grid in a pseudo random fashion. It aims to load level, but disproportionately based upon the ratio of costs on resources. A random number generator specifies the position upon a roulette wheel. The inverse of cost determines the percentage of the roulette wheel given to each resource. The inverse of the cost ensures resources that are cheaper are likely to be allocated to first. It is shown in Algorithm 4.

Cost Rate Roulette Wheel: This is like the above algorithm but performs the same random allocation based upon the cost per cycle instead.

Earliest Completion Time: This creates an earliest time to compute and transfer data matrix and selects the fastest option. It is like Time Optimize algorithm in this

Algorithm 4 Cost Roulette Wheel

```
FOR EACH (Task in Job to Allocate) {
    Get set of resources that meet resource requirements;
    IF suitableWorkerSet.isEmpty() {
      Throw Cannot Schedule Exception;
    }
    getMachineSelection(suitableWorkerSet);
    FOR EACH (Action in Task) {
      assignActionToEndOfMachinesSchedule(suitableWorkerSet);
    }
}
Return Candidate Schedule;

getMachineSelection(suitableWorkerSet) {
    FOR EACH (Resource in suitableWorkerSet) {
      summedCosts = summedCosts + (1/resource.getCost());
    }
    FOR EACH (Resource in suitableWorkerSet) {
      calculateSelectionChance(summedCosts);
      \\chance = (1/ resource.getCost()) / summedCosts;
    }
    Randomly Select Resource Based upon selection Chance;
Return Selected Machine;
}
```

regard but omits budget checks. It hence is more suitable for ISQoS as it may be used for offering advice rather than simply failing to provide a schedule.

Round Robin: This performs basic round robin order selection.

Round Robin Rescheduling: This performs basic round robin order selection and pushes new jobs in as early as possible into the schedule by the moving of existing jobs. The aim of this was to diminish the acceptance preference of jobs based upon how early they arrive. The details of this algorithm are listed in Algorithm 5.

Algorithm 5 Round Robin Rescheduling

```
FOR EACH (Task) {
    Get next worker in round robin order that meets resource
    requirements, skipping the turn of workers that do not meet requirements;
    IF Worker.isEmpty() {
      Place Task's Actions; BREAK; }
    FOR EACH (ACTION in Task) {
      FIND insertion point WHERE (
        Later Actions will not go past their Due Date
        AND No action already started will be moved
        AND No action due to start will be moved );
      Place Action in earliest position possible;
      move existing actions later on;
     BREAK;
    }
}
```

4.4.7 Offer Ranking and Selection

The broker in order to make a profit must generate the appropriate level of QoS. To do this it must decide which jobs are practical to compute within the allotted time and by which provider. This brings about various selection strategies for offers to compute jobs. There are three principle mechanisms by which selection may be performed, these are namely:

- Random: This selection mechanism randomly picks a provider.
- Current load: This mechanism hooks into the information provider and then asks which provider is the least loaded.
- Sort and Filter: This is the most complex mechanism available as it gives a wide variation of options. It first of all sorts offers using a wide variety of possible comparators and then it makes a selection, while filtering out some offers.

The load based and random selection mechanisms also having variations that test for broker profitability, thus ensuring the broker cannot accept work that is expected to cause a loss in profit. The sort and filter selector has some variations of the filtering mechanism that can equally perform a test for profitability. The sort and filter selection mechanism is the most complex out of the available mechanisms due to the possibility of choosing various sorting and filtering mechanisms. The options available are therefore discussed in more detail in Section 4.4.7.1 and in Section 4.4.7.2.

4.4.7.1 Offer Sorting Mechanisms

The sorting mechanisms available are listed below, though this list is implicitly twice the size as the Sort and Filter selector may reverse the order:

- Budgetary Resilience: The difference between the budget and the estimated cost. This hence within the ISQoS pricing model ranks all offers by how much spare money is available to the broker for rescheduling should something go wrong.
- Cheapest: The offers are sorted by their total price.
- Earliest: The offers are sorted by completion time earliest first.
- Profit: The offers are sorted by the broker's profit, most profit to least.
- Profit Rate: This is the broker's profit margin / duration of the job.
- Profit Then Earliest: This first of all sorts by profit and then by completion time.
- Profit Then Latest: This first of all sorts by profit then by completion time in the reverse order.
- Provider Name: This sorts by the provider's name. This comparator is available so is permitted here even if it makes no sense.

The comparators that focus on profit and have a second order sort must be used correctly, otherwise the incorrect profit comparators could be selected. This is especially the case given the reversing of the ordering. The Profit then Earliest sort has the following properties:

Natural Order: Profit is least to most and Completion Time is earliest to latest.

Reverse Order: Profit is most to least and Completion Time is latest to earliest. *Caveat:* Offers arriving the latest are first.

This is dealt with by the Profit then Latest sort which has the following properties:

Natural Order: Profit is least to most and Completion Time is latest to earliest.

Reverse Order: Profit is most to least and Completion Time is earliest to latest.

4.4.7.2 Offer Filtering Mechanisms

The offers once sorted need to be filtered and an option selected from the list of available offers. The permitted options for filtering are:

- Minimum Profitability: A minimum acceptable profit is set. The topmost offer that meets this criteria is selected.
- Minimum Profitability Rate: A minimum rate of profit is set for the broker. The topmost offer that meets this criteria is selected.
- Hybrid Offer Filter [103]: as shown in Algorithm 6, is a filtering mechanism that aims to accept all jobs that have no constraint issues and evaluates all others based upon a comparison to a going rate that is established from the last n records of accepted offers. It is hence a hybrid between a going rate filter and a topmost profitable filter. In doing this it has several advantages, which are listed below:

Algorithm 6 ISQoS Hybrid Offer Filter

```
FOR EACH (Offer) {
    Sort the offers based upon the ranking mechanism chosen;
    IF Completion_Time <= Due_Date AND
    Service_Price <= Budget {
      Accept Offer; BREAK;
    } ELSE {
      Take the last n accepted offers and find the average rate
      at which profit accumulates and establish the going rate;
      IF Current_offer_profit_rate >=
      (going_rate − acceptable_deviation_below_going_rate) {
        Accept Offer; BREAK;
      }
    }
}
```

  - If the constraints are fully met then the job is automatically accepted. The main advantage of this is that if the arrival rate of jobs slows then unconstrained (fully profitable) jobs are always accepted. The pricing model dictates

  that these jobs are the most profitable possible anyway. This can be contrasted with a going rate calculation which may not accept a job with a lower profit even if it could fit with no problem. This is particularly advantageous if different mark-ups are in use and other factors such as differing network transport cost compared to the cost of computation.
  - If offers are constrained by either time or budget then a going rate assessment is performed. If the offer still remains profitable enough then it is accepted, hence if jobs with different mark-up's arrive they will be treated differently and higher mark-up based jobs may still make a sufficient profit to be acceptable.
  - It can be expected on an unused Grid that jobs will be accepted readily, this hence removes the immediate need for a history of records and alleviates the issues of starting with no information from which to determine the going rate.

- Near Going Rate: This establishes from a history of the last n records the current rate at which profit is accumulated by the broker. It then establishes a minimum value below this that is acceptable. If the new offer is above this threshold then it is accepted. The previous jobs are used to establish the going rate. Both the history size and a proximity below the going rate considered to be acceptable are configurable. The advantage of specifying deviation from the going rate over having a minimum profit rate is that it better follows demand and will automatically adapt if the environment changes. An example of this would be if the amount of resources available changed which would allow more space in the Grid, that in turn means more jobs could be accepted causing the service price to change when dynamic pricing is used.

- Topmost Offer: This selects the topmost offer from the sorted list of offers, with no further selection criteria.

- Topmost Profitable Offer: This selects the topmost offer from the sorted list that is profitable. If no offer is acceptable then no option is provided to the end user for execution. The offers would then solely be used to advise the end user how much they could be expected to pay to have work completed on time.

4.4.8 Pricing Mechanisms

4.4.8.1 The Available Pricing Mechanisms

In the ISQoS Grid the providers have to set their own price. This has to be set competitively and it has to reflect the current load the provider is experiencing i.e. price has to follow demand.

This demand is caused by the acceptance of jobs by the provider and is expressed in the provider's schedule. The schedule has a set of jobs which are made up of tasks which are in turn comprised of actions, which have to be assessed for the demand they represent for the provider's resources.

The actions consist of *stage in*, *execute*, *stage out* and *clean*. Stage-out only represents the transferring of files while clean-up cleans and bills the work. The stage in and execute actions may also be combined to form a single action, meaning the worker nodes will automatically start execution once the stage-in has been completed. These actions together establish the demand on the provider and its resources.

This demand the actions create when assessed may be for the provider as a whole or it may be performed so the price is different for each given resource. The provider level price setting mechanisms are therefore:

A count of actions that the provider has to perform: The amount of actions in a provider's schedule is counted and then a price is assigned accordingly. The number of actions is proportional to the number of tasks so it was not seen to be of any use to count tasks. Actions are placed in a queue of actions that the provider is expected to start, so task counting against a list of actions takes more computational effort.

The average finishing time of work on the provider's resources: This would be similar to the "count of provider actions" metric but would account for jobs of varying lengths. The completion time of the last action in the schedule is compared to the current time. This value is then mapped across to a price written in a configuration file on disk.

A static resource price: This is the most basic case where a static singular price is set for the provider as a whole.

This can also be set at a similar fashion for each individual resource/machine:

A count of actions that a particular machine has to perform: A variation upon the "count of provider actions" metric but the count is performed for each resource which is then priced separately. This variation can therefore be used when cost guides local resource selection.

The finishing time of all work on a particular machine: A variation upon the ”count of machine actions” metric but would account for jobs of varying lengths. The completion time of the last action in the schedule is compared to the current time and from this value it is then mapped across to a given price.

A static resource price: This is the most basic case where a static singular price is set for the provider as a whole forming a default. Individual named resources are then allowed to take an alternative fixed price.

A key factor that is discussed in Chapter 6 is the counting of actions that are subsequently removed during the billing process. The billing removes actions from the schedule and hence changes the price. There therefore exists variations which do not count actions that have already started. In ignoring work that has executed it ensures the count of actions and the price change more steadily over time.

4.4.8.2 The Effects of Dynamic Pricing Mechanisms

In the previous section the pricing mechanisms were discussed. The effect of establishing a resource cost dynamically in the pricing model is discussed here. To narrow this discussion the ”count of provider actions” is selected as the basis of this discussion.

When jobs are submitted to the Grid it causes resource demand. If the demand increases and additional resource allocations are made then so does the schedules size, thus the count of provider actions metric is reasonable. The schedule is essentially a queue of actions that are required to be executed on a set of machines at a given time, hence for discussion it can be simplified to being expressed as a queue. This queue length/schedule size is merely a concept of how much work is currently being performed by the provider, hence demand is proportional to the queueś length.

$$demand \propto queue\_length \tag{4.1}$$

It can be assumed as demand increases for the finite commodity of ”time on the Grid” that the price increases as well, i.e. in relation to supply and demand. The setting of the price is a supplier’s prerogative but it is reliable to assume as demand increases for the resources, that a supplier would increase the price to suppress the excess demand. The resource price is hence roughly proportional to demand, for this discussion it is considered proportional.

$$resource\_price \propto demand \tag{4.2}$$

It can therefore be seen that demand is proportional to both queue length and resource price, thus linking back to the initial assumptions that resource price should be set based upon the queue length.

$$resource\_price \propto queue\_length \tag{4.3}$$

If the budget is considered in this context its meaning may be understood better. Budget relates to the maximum resource price, as the maximum resource price is the budget reduced by the profit margin the broker is expected to make. It therefore relates to the maximum resource price available.

$$max\_resource\_price = budget - max\_broker\_profit \tag{4.4}$$

Thus it can be seen that the budget is related to the maximum queue length, i.e. the maximum amount of work the provider is allowed to have in its queue when the job is accepted.

$$max\_resource\_price \propto max\_queue\_length \tag{4.5}$$

Expressing the schedule as a queue length gives the intended impression of the maximum time ahead of submission which is allowed before the job starts. This is because if the queue length is small the price is low so the Job will not get priced out of the market. Hence previous jobs push the current job to be submitted further on in the schedule. Jobs placed in the schedule take up space on the Grid which is reflected in the start delay and the available slack. Thus the budget which is giving this maximum queue length also provides a maximum acceptable start delay/minimum slack.

This effect of maximum queue length can be seen to be true because if the price to run a single job is considered without relation to ”where specifically the job will be placed”, then resources will be more expensive given higher workloads/”the further into the future the work is placed due to previous work pushing new jobs further on in the schedule”. Thus the budget once linked in the model to time expresses the maximum permitted queue length.

This maximum queue length is also in relation to the average size of jobs, i.e. the amount of worked to be performed and the speed and quantity of the resources used to compute the jobs. If the resources are faster jobs take up less space in the queue/execute quicker. This means they get billed and removed from the schedule faster, making the queue smaller.

In regards to queue length and start delay, if the local scheduling algorithm does not load level then a resource could be unloaded and be more expensive when setting the price at provider level. The more expensive time hence could represent time closer to the current time. This earlier block of time is functionally equal to any other block of time, less the risk of resource failure, e.g. highly utilised resources fail more often, so the selection of the right pricing mechanism for the scheduling algorithm should be considered. This can also be seen to be the case if the scheduling algorithm uses cost to guide placement of jobs, as in Nimrod/G's algorithms shown in Section 4.4.6 (see Algorithms 1, 2 and 3).

4.5 Feature Comparison

In this section the ISQoS broker is compared and contrasted with similar schedulers and brokers. The Grid schedulers and brokers that are evaluated that form the current state of the art are: Condor/G [197], PBS [31], Oracle Grid Engine [177], Globus Toolkit, GridWay [60], OpenCCS [18, 20, 146], Nimrod/G [129] and GridEcon [8, 9]. An initial discussion of the evaluation criteria will be given followed by a tabulated assessment of the criteria.

4.5.1 Cost & Time Guarantees

In Grids given the finite amount of resources if time guarantees are to be implemented then it is required to have an understanding of the priority of each job to be completed as only a certain fraction of jobs can be completed within the allotted time constraints. Economics is a technique that may be used to realise this prioritisation.

The ISQoS Broker and similar brokers such as Nimrod/G have been developed with economics in mind. Nimrod/G provides the choice of four different scheduling algorithms and allows for the selection of cost and time constraints [39]. The ISQoS providers have a scheduling interface (as described in Section 4.4.5) that allows for the implementation of cost aware scheduling algorithms. In ISQoS however it is also possible for cost unaware scheduling algorithms to be used within the process while maintaining an economic approach and allowing the economic model to operate. This approach is in part preferred as it then allows ISQoS providers to return estimates as guidance, when the scheduling algorithm cannot meet the economic or time constraints.

The Nimrod/G cost algorithms are simple as previously discussed in Section 4.4.6.

For example if the cost optimize algorithm is used and the deadline constraint is not actively having an affect then it stacks all work upon the cheapest resource, without any form of load levelling. This is even the case if a 2nd equally cheapest resource is present. The Cost Time Optimization algorithm partly gets around this by grouping resources of equal cost together and then allocating to the fastest resource, thus providing better load balancing. Though this is not best when dealing with costs on a continuous scale, where using the ratio between cost and resource speed would be better. The time optimization algorithm works in a simple earliest completion time fashion while respecting the budget constraints.

Oracle Grid Engine has three classes of job ranking policy, including a ticket based mechanism that provides a proportional share based mechanism, an urgency based mechanism i.e. priority and the final option is to have custom policies [177]. Proportional share mechanisms can be seen to be a primitive economic mechanisms [118] hence Oracle Grid Engine could be considered to have some cost awareness. Though proportional share based mechanisms tend only to consider sharing resources and do not further control the flow of where jobs are allocated.

4.5.2 Advance Reservation Support

Advance Reservation is a key requirement in been able to establish any form of completion time guarantees as it provides the opportunity for guarantees to be realised.

Advance reservation is possible in several middleware solutions, but it must be noted at that it requires estimated durations for each task, which are not always available. This therefore means that reservations must last at least as long as the duration of the task for the execution to work correctly. Middleware is hence is likely to support reservations but is not likely to put them to use in environments centred upon having high machine utilisation.

4.5.3 Queuing vs Scheduling

Most Grid Infrastructure (schedulers/brokers/middleware) use queue based mechanisms as opposed to scheduling in the same fashion as operational research. Condor, PBS and Grid Engine are good examples of such a queue based approach. The requirement to act in a queue based fashion, stems from the need to keep the resources of the Grid working and no intent to provide time and cost provision. A scheduling based approach requires that job's size is known, but it then allows for guarantees in completion time to provided. The Mauri Scheduer, OpenCCS and ISQoS mechanisms differ in this regard and use a scheduling based approach. Below in Table 4.1 the differences between a queuing and scheduling based system are highlighted [18]:

Table 4.1: Queuing vs Scheduling [18]

	Queuing System	**Scheduling System**
Considered time frame	present	present and future
Runtime estimates	not required	required
Submission leads to	insert in queues	complete reschedule
Job start times known	no	yes
Reservations	difficult	yes, trivial
Deadline scheduling	optional	yes, trivial
Backfilling	optional	yes, trivial
Examples	PBS, NQS, LL, ...	Maui Scheduler, OpenCCS

4.5.4 Negotiated Job Submission

In regards to job submission it is useful to understand how quickly results are likely to be returned from each provider. This is dependent upon the current workload of the environment relative to its capacity and other constraints such as the job's priority/value and the jobs size.

In order to achieve an estimated completion time, a candidate schedule needs producing. In other none ISQoS infrastructures this mechanism simply does not exist. Jobs are simply submitted to a provider by a user whom has been informed of current resource utilisation.

The lack of this candidate schedule mechanism limits the ability to perform any sense of negotiation as it would be difficult to relate the work that is to be submitted and its completion requirements to resource availability. It also means brokers are reliant upon metrics such as current workload and queue length which are unrealistic measures to use when guarantees upon cost and time of job completion are required.

Current workload (i.e. the amount of machines/operators that are busy) and queue length measures only relate to the current situation upon the Grid, which will quickly change. They could be likened to a call centre in which a blanket statement of all operators are busy and you are x in the queue. In the call centre's case many ”jobs” are of a similar size and the expected time to been served can be guessed. In the case of the Grid jobs differ in sizes and have varying amounts of data to transfer which causes coupling between the jobs i.e. two jobs using the same network can only stage out/in together by sharing the bandwidth, thus slowing each other down.

4.5.5 Monitoring, Adaptation and Rescheduling

Rescheduling is an important tool when the dynamic nature of the Grid is considered. It can aid in recovering from errors and is also useful in moving tasks' advance reservations in order to honour more agreements or the current agreements better.

A summary of the capacity to reschedule is provided below:

- ISQoS - Periodic scheduling, with the ability to dynamically select the period at runtime and rescheduling upon job submission is also allowed.
- Oracle Grid Engine - Periodic scheduling only, jobs enter a queue before being scheduled fully [177].
- GridWay MetaScheduler - It has both periodic and event driven rescheduling i.e. when performance slowdown or remote failure is detected [92].
- OpenCCS Rescheduling is performed upon the submission of a new job.
- Condor The DAG (Directed Acyclic Graph) manager that handles workflows is capable of generating rescue DAGs when it exists with an error code. The rescue DAG is a new DAG listing the elements of the original DAG left unexecuted. To remedy the situation, the user can examine the rescue DAG correct any mistakes in submission and then resubmit it as a normal DAG.

4.5.6 Advanced Discovery, Scheduling API & Flexible Pricing

In the ISQoS provider delivers an API for developers of new scheduling algorithms, it is capable of hiding away much of the difficulty of Grids and presents the scheduling as merely the allocating of jobs to resources. For example it determines the estimated completion time for each job on a given resource and uses the same API mechanisms for both execution and data transfers. The advanced discovery mechanism given presents a list of capable resources for a given job that are available for allocation, thus removing an implementers requirements to check themselves.

The scheduling algorithm may easily be swapped out for experimental purposes see Sections 4.4.5 and 4.4.6) with alternatives and the pricing mechanisms may also similarly be swapped out (see Section 4.4.8) such that any commodity market pricing strategy can be formed by a given provider.

4.5.7 User vs System Centric Scheduling

The majority of Grid Middleware/Brokers and Schedulers are system centric [37, 40, 54, 186]. In that they aim to maximise utilisation of Grid resources or are otherwise orientated towards system oriented metrics.

ISQoS is different in its setup in that it is more balanced and considers the users objectives, especially in terms of delivery time and how much it costs. The cost however is also system centric in that the price can be made to adapt to current load and hence control utilisation, whilst also ensuring the provider benefits from providing Grid resources.

4.5.8 Open Standards

In Grid middleware in order to gain a degree of interchangeability and general compatibility so that different middleware's can work together either open standards or a large amount of adaptors, that complicate integration efforts are needed. In recent years there has been a heavy drive towards open standards. The ISQoS broker therefore implements standards such as WS-Agreement, WS-Agreement negotiation and Job Submission Description Language (JSDL) as well as web service based standards, WSRF , WS-Notification, WS-Security.

4.5.9 Comparison Study

In this section each of the brokers/middleware are compared. They are split into three tables, one for best effort approaches (Table 4.2), one for QoS oriented (Table 4.3) and one for economic approaches (Table 4.4).

In Table 4.2 the best effort approaches are discussed. These are characterised by queue based scheduling that is centred around the resources and aiming to keep them as busy as possible.

In later work there are substantial efforts to provide quality of service provision with respect to reliability. The vast variations of different middleware and their configurations gives rise to meta-schedulers such as GridWay, i.e. ones that control and submit work to local Grid infrastructures using adaptors, which in turn schedule the work for completion. The OpenCCS software provides a fully schedule based solution to deploying Grid resources and given the work of the HPC4U project allows for resilience to resource failure. These two QoS oriented middlewares are shown in Table 4.3.

There is also a focus upon the development of economic models to control Grids, given their diverse nature. Early work such as Nimrod/G focussed solely upon Auction

Table 4.2: Feature Comparison Study - Best Effort

	Best Effort			
	Condor-G	Portable Batch System (PBS)	Oracle Grid Engine	Globus
Advanced Discovery	no	no	no	yes via Globus MDS
Cost & Time Guarantees	no	no	no	no
Advance Reservation Support	no	no/yes with Maui	yes	yes
Queuing vs Scheduling	queue	queue scheduling with Maui	queue	queue
Negotiated Job Submission	no	no	no	no
Rescheduling	no	no	allows checkpointing	no
Monitoring	yes	no	some	yes
Adaptation	no	no	some - QMaster can auto-restart	no
Open Standards	no	no	some DRMAA	no
User vs System Centric	system	system	system	system
Plug-in Scheduling	no	no	yes (some flexibility)	no
Flexible Pricing	no	no	no	no

Table 4.3: Feature Comparison Study - QoS

	QoS	
	OpenCCS	GridWay
Advanced Discovery	yes	yes
Cost & Time Guarantees	time only	no
Advance Reservation Support	yes	depending on underlying infrastructure
Queuing vs Scheduling	queue	queue
Negotiated Job Submission	no	no
Rescheduling	yes with HPC4U	yes
Monitoring	yes with HPC4U	yes
Adaptation	yes with HPC4U	yes
Open Standards	no (WS-Agreement used in AssessGrid)	some DRMAA
User vs System Centric	system	system
Plug-in Scheduling	yes	yes
Flexible Pricing	no	no

based approaches. Later work such as GridEcon took a top down approach creating a set of economically oriented services which be used in the Grid. The ISQoS project takes this service orientated approach and develops quality of service around the concept of job completion time and cost. These economic approaches are shown in Table 4.4.

Table 4.4: Feature Comparison Study - Economic

Economic			
	Nimrod/G	GridEcon	ISQoS
Advanced Discovery	yes	potential	yes
Cost & Time Guarantees	yes	potential	yes
Advance Reservation Support	no	potential	yes
Queuing vs Scheduling	scheduling	scheduling	scheduling
Negotiated Job Submission	yes	potential	yes
Rescheduling	yes	potential	yes
Monitoring	no	potential	yes
Adaptation	no	potential	potential
Open Standards	no	yes	yes
User vs System Centric	system	potential	both
Plug-in Scheduling	no	yes	yes
Flexible Pricing	no auction only	yes	yes

4.6 Summary

In this chapter the broker's implementation and infrastructure was introduced. The aim was to highlight its architecture and to present recommendations resulting from its implementation. The broker was then evaluated in comparison to other available software.

The broker's implementation was driven from its requirements and in particular the requirement to negotiate cost and time requirements for job completion in a non-system centric fashion. In doing so a true compute service was developed, that supported swappable pricing and job ranking mechanisms with a range of scheduling algorithms available by an easy to use scheduling API. In doing so this provided the capacity to run a variety of economically oriented experiments, that will be shown in chapters 5 and 6.

The broker was implemented largely from scratch because of specific infrastructure requirements. These requirements are namely to:

- allow for the negotiation of job's quality of service requirements in the admission phase.
- allow offer construction, by requesting a schedule that describes how well a provider may complete the work, without asking a provider to commence execution.
- avoid legacy standards such as DRMAA and RSL that do not use XML, as they cannot be used in agreement formation, in modern standards such as WS-Agreement.
- avoid queuing based mechanisms for job admission, which allows operational research based techniques to be used for the scheduling of Grid Jobs.

The structure of the broker is discussed including how the binding of job descriptions is performed using the XML any type and how the infrastructure allows generic job and resource descriptions to be used. This includes the pairing or a parser and set of requirement tests together to create a generic way of testing if a job can be executed upon a given resource.

The scheduling API was then discussed in full, which is simple to use. The API also pushes a state based concept of new schedules being derivatives of previous schedules. Thus highlighting how ISQoS recommends a schedule should be perceived. In that changes such as adding a new job to the schedule, moving jobs in the schedule, removing jobs from the schedule and errors making resources unavailable make updates/overwrites of the previous existing schedule.

The API's discussion was followed by algorithms that have been implemented in the broker, including the implementation of Cost Roulette Wheel and Cost Rate Roulette Wheel algorithms which are new. The main aim of these was to perform load levelling while considering economic pressures, inside the ISQoS market framework.

The scheduling algorithms were followed by the offer ranking and selection mechanisms that evaluate offers made by providers. The ranking and filtering mechanism was very flexible and including the introduction of the ISQoS Hybrid Offer filter and going rate filter mechanisms, that aim to utilise the ISQoS market model.

The providers pricing mechanisms were also discussed and in particular how schedule size/queue length relates to the market price. Finally a comparison study was performed, with the intent of highlight how ISQoS is an advance upon the current state of the art.

Chapter 5

Job Admission and Profit

5.1 Introduction

In this chapter experiments are performed with the aim of demonstrating the improvements in QoS that may be obtained by submitting jobs for estimates in the ISQoS tender market and then selecting the best provider for computation. The selection mechanism for assessing offers is hence compared. A listing of all the selection mechanisms available may be found in Section 4.4.7 and the ones selected for test are discussed in the next section.

The experiments explore a difficult problem as the user agrees to pay a certain amount for each job completed, while the broker agrees to make guarantees on the jobs completion time. If the broker makes too many allocations then it is likely to have its service price drop below the cost of the resources used to execute the job, while too few and it does not achieve the maximum profit/global utility possible [147]. It will hence make a loss, which over the long term makes it unviable, due to budget balance [155]. It also requires to balance the load between the provider's it uses sufficiently such that as most jobs as possible may be accepted. A trade off situation thus occurs.

The focus is upon high load scenarios where correct selection is most required, as per the motivating scenarios for the thesis (see: Section 1.2). High load ensures far more jobs are available than can be computed on time. This means in order to ensure time constraints are met, which is directly linked to the broker's profit in the pricing model, some jobs have to be rejected. It should be noted that during low load situations both market and non-market mechanisms provide good welfare and that if utilization increases market mechanisms outperform their non-market based rivals [130].

The research questions that are answered in this section are discussed next, followed by the configuration of the experiments performed in Section 5.3. This is then followed by the results and evaluation of the experimentation and finally by the chapter summary.

5.2 Experimental Research Objectives

This chapter's research concentrates upon admission control and steering of jobs to the correct provider, within the tender contract market of the ISQoS Grid. The focus of this is to therefore to assist meeting the aim from Chapter 1 of providing a brokering architecture that can be used to support QoS in terms of time and cost constraints and in particular to ensure that the market model is constructed to allow multiple sites to compete and to manage the steering of jobs between providers. The research question this chapter aims to satisfy is therefore to determine how best can admission control be performed within the ISQoS market in order to establish QoS?

5.3 Experimental Setup

5.3.1 Experimental Method

The principle action in the experimentation reported upon in this chapter is to vary the selection and sorting mechanism used. Given the wide variety of selection and sorting mechanisms shown in Section 4.4.7 only a sample was tested in experimentation. To establish which ones were tested they are placed into three categories and listed here. These categories are namely: classical, flooding and selective.

The first classic strategies relate to current mechanisms for submitting to the Grid. They do not require any data from the offers, hence represent a situation with direct submission without negotiation. This can be achieved either randomly or by submitting based upon the current load of the provider and are indicated by orange on the graphs.

- Randomly: In the experimentation offers are first asked for and then an offer is chosen randomly. This was chosen as a baseline in order to perform comparisons from.

- Current Load: The current load selection mechanism hooks up to a Ganglia [81] information provider. An average of the `cpu_user` value across all workers for a given provider is used as a measure of load. This closely as possible represents if a CPU is busy or not as per the UK's NGS [132] load monitor tool. The user CPU usage is used so as to ignore as much as possible minor non-Grid related system activities taking place on the worker nodes.

The second set of strategies floods the Grid and tries to optimize greedily upon either time or profit and represent naive optimization strategies, which are indicated by red on the graphs.

- Earliest First: This mechanism sorts the offers by completion time and selects the topmost offer. This strategy makes no account for the broker's profit and so long as the budget and the deadline constraints are met then the job is accepted.
- Highest Profit: This mechanism sorts the offers profit and select the topmost offer. This strategy makes no account for the broker's profit and so long as the budget and the deadline constraints are met then the job is accepted.

The last set of strategies named selective aim to filter out the worst offers and ensure only jobs likely to make the broker sufficient profit are accepted. These mechanisms are Highest Profit (Profitable Only), Hybrid Offer Filter, Load Based Selection (Profitable Only), Random (Profitable Only) and a Near Going Rate mechanism. These are indicated by blue on the graphs with the ISQoS specific ones indicated in green.

- Highest Profit (Profitable Only): This extends the highest profit approach and performs checks to see if the broker will make a profit before accepting.
- Near Going Rate: This was configured to initially sort by profit and select only profitable jobs.
- Hybrid Offer Filter: This was like the going rate mechanism configured to initially sort by profit and select only profitable jobs.
- Random (Profitable Only) & Load Based Selection (Profitable Only): These extend the classical methods by allowing them to submit to the site chosen by their ranking mechanism and then checking to see if the broker will make a profit.

During the experimentation jobs were submitted by using a job submission tool in the following way. In the first stage the submitter acquires the job submission templates from each provider. It then fills the templates with the user's preferences. These preferences are for:

1. Budget: The user's maximum price they are willing to pay.
2. A due date and deadline: A preferred time and the last point in where the job is still of use.
3. Task description/s: Job Submission Description Language (JSDL) document/s describing the work to be performed.
4. File size and execution requirement: Estimates for each task within a job.

The broker/submitter tool then requests offers from providers in the tender market [40]. Each provider calculates a schedule that is suitable for the completion of the work and submits its offer back to the provider. This offer includes the estimated completion time for the job, the overall cost and completion time estimates for each individual task.

In the experimentation there are different types of selection process, which are swapped using a class loading mechanism. Some of the selection mechanisms tested however do not require the full agreement process. The offer returning process is not stopped however even when the information in the offer sent back is not used, such as in the random selection and load based selection mechanisms. The alternative is to submit an offer for direct acceptance or rejection with only yes/no feedback and although this is possible it is not done in order to keep consistency in communication patterns and in general how the experiment is performed.

The broker/mass submitter tool in its final stages applies a mark-up, performs an assessment and submits the best offer to the user for acceptance. The implemented agreement mechanism also provides the facility for multiple rounds of negotiation, this is however out of the scope of the experimentation and jobs are simply rejected. This avoids changing input values for the template in an attempt to simulate the user's preferences as this is considered to be highly subjective. It should be noted that providers will not make offers that will go past the deadline, as they know they will fail, so the offer collection phase aids the finding of a suitable provider for the work to be completed upon.

In the experimentation a Grid is established with 2 providers/clusters of machines. Each provider had 4 virtual machines (VMs) of which one also acted as a head node. Jobs were submitted at intervals, from a separate broker virtual machine instance. This prevented the broker from making one provider less competitive than the other. In total the experiment hence had 9 virtual machines.

The virtual machines ran Ubuntu 11.10 (64bit) server, with full virtualization and ran upon 4 physical hosts. The virtual environment was constructed using OpenNebula 2.0 [145] and Xen 4.0.1 [55]. Each head node had 1GB of RAM allocated and the worker nodes had 768MB and the processors each ran at a speed of 2.4GHz. The head nodes were allowed to be allocated to by the scheduler ensuring the resource space allowed was as large as possible. The size of the Grid was chosen because of the limited resources available. It was also desirable to have competition between providers so 2 providers were used.

The ISQoS Grid which was setup on the VM's used WS-Agreement for Java v1.0 for the Broker and Provider agreement process. Ganglia 3.2.0 [81] was used to provide resource information to each of the head nodes, about the availability of its resources.

Experiments were then ran with the settings discussed in the next section. There were 6 runs of each trace taken and 95% confidence intervals are hence marked on the figures shown in the next section. The first 10 accepted jobs of the traces have been ignored to counteract effects of starting with an unloaded Grid.

5.3.2 Experimental Configuration

The submitter tool sent 100 jobs with 8 tasks to the providers to requests offers. It had to either accept the job and submit to one of the two providers or reject the job. Jobs were submitted with a 30 second gap between submissions. This is shorter than the time it takes to compute a job, which means the Grid fills and resources become sufficiently scarce as per a time sensitive, high utilization scenario. Each provider was configured to use the round robin scheduling algorithm (see Section 4.4.6 for the list of implemented algorithms).

Jobs were setup to be none data intensive and the stage in/out size was 1 megabyte. This mitigates issues with considering the network configuration of the virtual cluster on the cloud testbed. The execution size estimate for a task was given a value of 3,000. This value derives from a reference processor of 3,000 MHz multiplied by an expected duration of 1 minute. This means upon the resources available, tasks are expected to last approximately 1 minute and that if a job was allocated to a single machine it would take 8 minutes to complete, the 8 tasks.

Each job's due date was set to the submission time + 8 and its deadline was set to the submission time + 12, with the knowledge that the Grid would soon be overtaxed.

Each job was given a budget of 20,000 which was chosen to be sufficiently high so as not to act as a selection pressure. A fixed mark-up for the broker of 20% was chosen, which means the broker breaks even 16.67% of the way between the due date and deadline [103], so the provider must complete work before this point to remain in profit. A static resource price was chosen that bills time for both the use of network and resource time equally at 1 unit per second. The set of available price setting mechanisms is discussed in Section 4.4.8.1.

5.4 Results & Evaluation

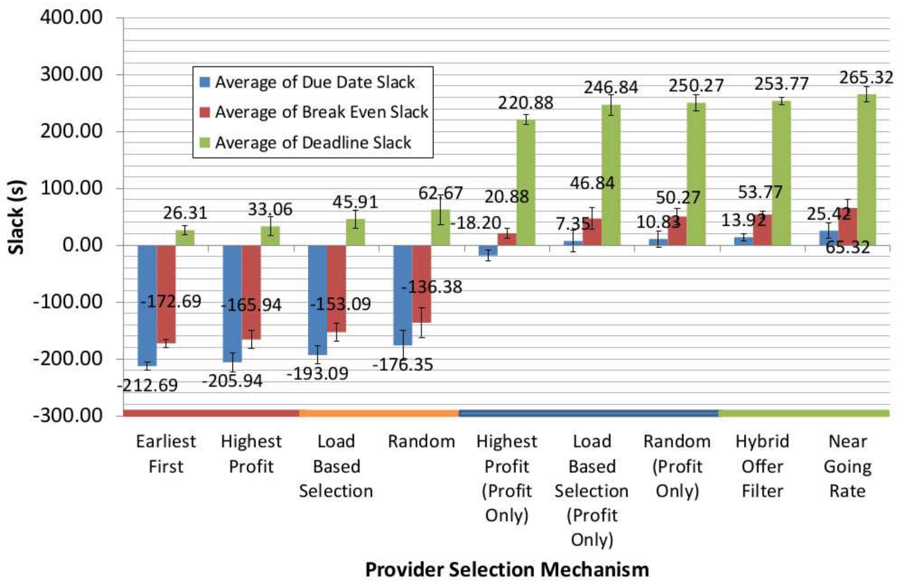


Figure 5.1: Average Slack

A key measure of QoS is the amount of available slack as it relates directly to meeting of the temporal QoS requirements and has a strong bearing upon the profit the broker is going to make. Slack is taken in this context to be the difference in time between the completion time and either the due date, breakeven point or the deadline. Hence zero due date slack is completing just in time for the broker's service price to remain unchanged. In Figure 5.1 it can be observed that there is a distinction between mechanisms where profit checking is permissible or not, in regards to the amount of slack available. Essentially if the SLA is not evaluated for the broker's profit then the spare slack is entirely consumed and time guarantees are not upheld. The Highest Profit (profitable only) selection mechanism tends to go past the due date making it less suitable. This means while filtering against profit is useful, it is not always going to offer a guarantee that the due date constraint is going to be met.

It should be noted that the relationship between slack and profit is strong, the greater the breakeven slack (the time between completion and the breakeven point) the greater the likelihood that the broker is going to make a larger profit. This may also be expressed as maintaining positive due date slack. There however remains a trade-off behaviour where it is possible to gain greater profit by fitting slightly more jobs in to a schedule and only just meeting or just passing the due date, which is more preferential to the broker and for the global utility gained.

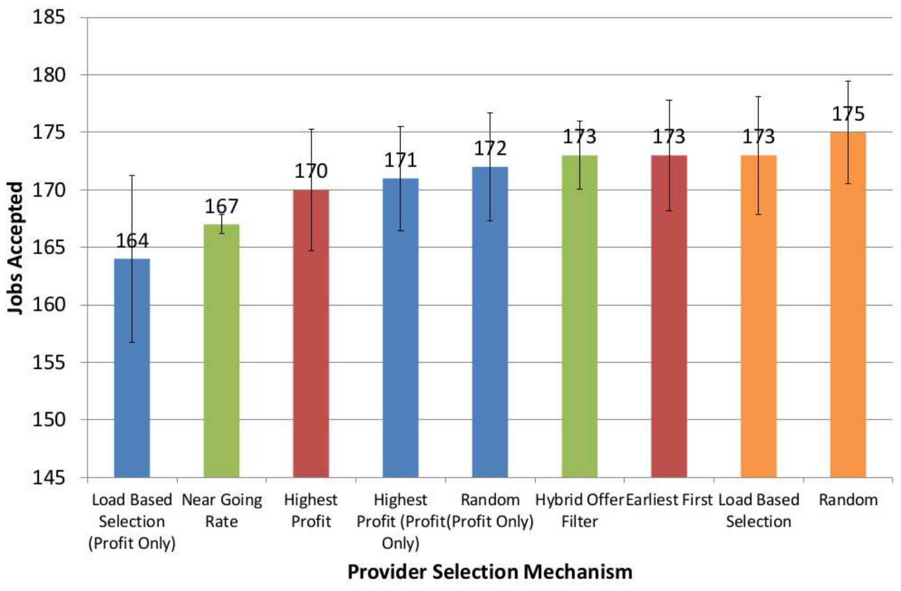


Figure 5.2: Average Number of Jobs Accepted

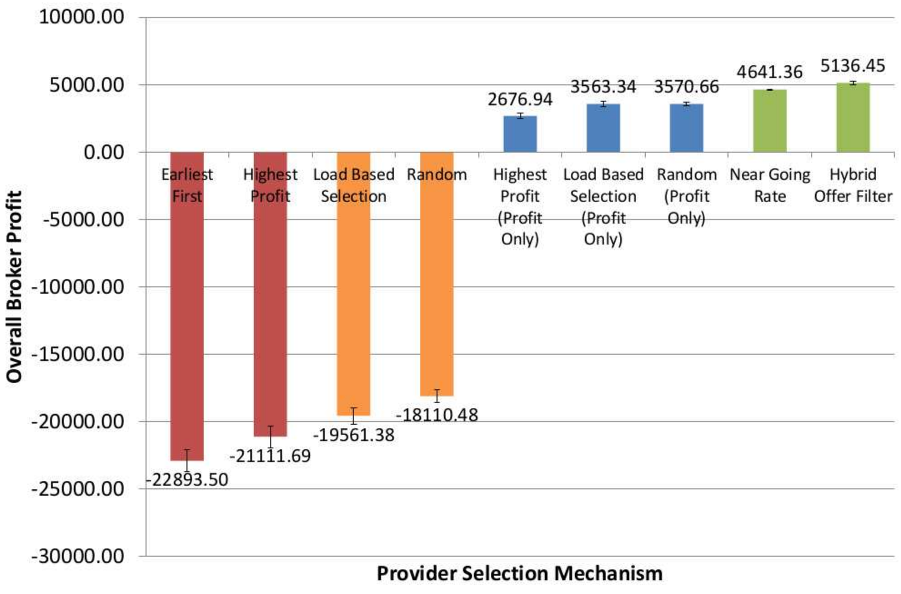


Figure 5.3: Overall Broker Profit

Adaptations of the classical submission strategies that examine the agreement and test for profit perform well, but they tend to have a wide variance in slack (Figure 5.1) and job acceptance (Figure 5.2) as compared to the Hybrid Offer Filter. This is reflected in the overall profit (Figure 5.3), with the Hybrid and Going Rate approaches winning out, some 30.0% above their nearest rivals.

In terms of job admission it can be seen that there is a requirement in order to generate the greatest QoS and profit to generate a stable and consistent selection process. This has a benefit in not only giving a consistent behaviour to end users, but it causes a more stable service price, which in Chapter 6 is shown to aid QoS provision. It should also be noted that it was previously shown that the Hybrid Offer approach works much better than the going rate mechanism in lower arrival rate situations [103].

Selecting providers by their current load is a common way to submit work to in a none economic oriented Grid. This has the disadvantage that the current load does not reflect well how long each job will take to complete. This is illustrated by an example where current load is only shown as the count of jobs in the queue, relative to how many processors are available. This disadvantage is reflected in the results by the fact that the load based and random selection mechanisms appear to be very similar. This is also the case when looking at the profit checking equivalent of both mechanisms. It is therefore suspected that the load based selection mechanism acts more as a means of random number generation. It is also believed that when a Grid is nearing full capacity (as per the experiment), this effect is exacerbated. This is because at lower load levels the amount of tasks are fewer than the CPUs available meaning this ratio would indicate the least loaded sites well enough, especially as provider selection becomes less important. At higher loads it should be noted that this ratio of jobs to CPUs simply does not reflect how long it will take the CPUs to become free.

The selective based strategies indicated on Figures 5.1, 5.2, 5.3 and 5.4 in blue and green, use details from the offer to manage their selection process. It should be noted that if the time between the first query that determines the properties of the offer and the final submission is sufficiently far apart then the offer will not represent the current state of the environment. Offers hence should expire after a sufficient amount of time has passed. In terms of guiding the selection process it is not expected to be detrimental that the first offer is indicative and that a provider can use a different schedule once the job is sent for final submission, so long as the scheduling strategy is not wholly different from that which handled the initial query. The provider is hence expected to be honest in this regard and use the same scheduling strategy.

The start delay as shown in Figure 5.4, is used here as a metric for understanding the pressures upon resources on the Grid. Strategies selecting based upon none negative

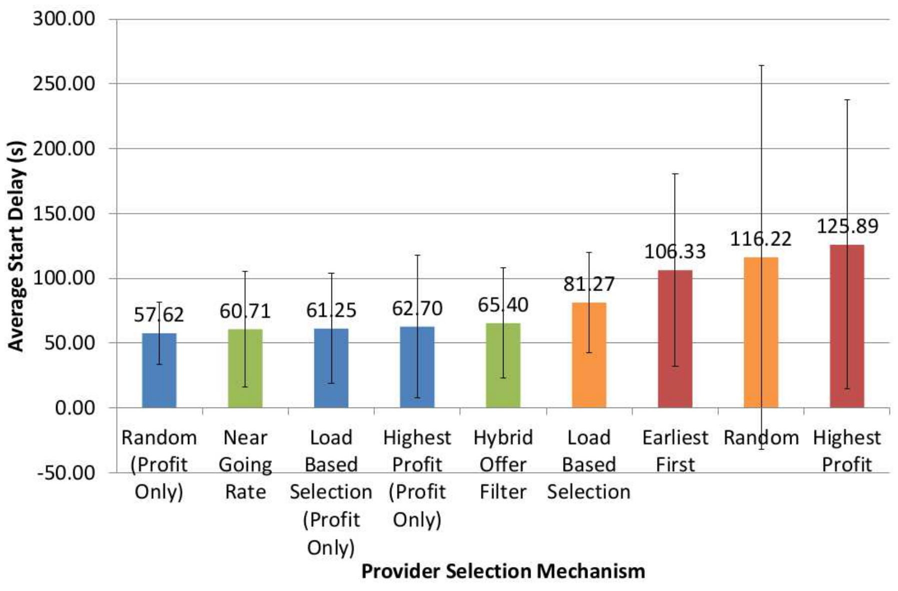


Figure 5.4: Average Start Delay

broker profit fair best and in all cases beat other strategies. The random and the highest profitable job strategies perform worst and have notable variance in their start delay. The deviation from the ordering as compared to how many jobs are accepted should also be noted as it gives some notion of the differing quality of site selection.

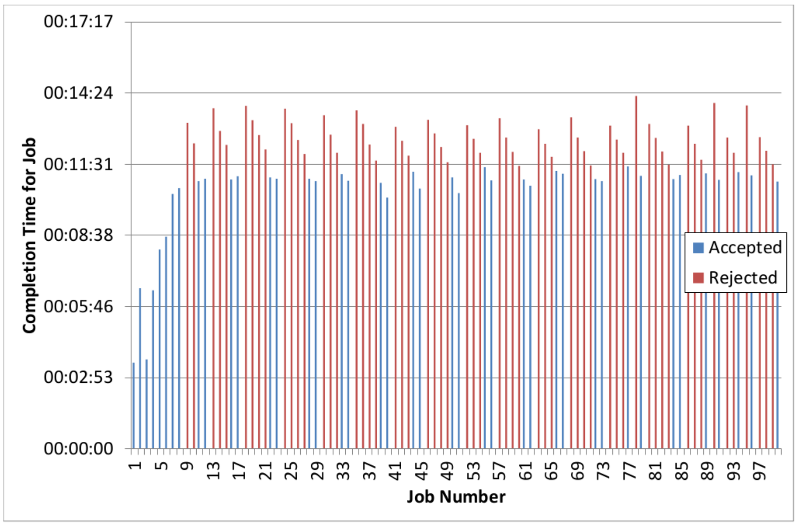


Figure 5.5: Completion Time Trace - Run 1 - Highest Profit Based Selection

Given the examination of job selection mechanisms it seems wise to examine what is actually happening, in greater detail. If a single trace is examined a sequence of loading and unloading may be observed. In Figure 5.5 a trace from the first run of the highest profit selection mechanism is shown. The completion time (completion time - time of offer) is shown against each instance of either job acceptance or rejection. It becomes clear that the trend is to have several jobs rejected until a point where the completion time drops sufficiently, because more resources are free and then a job is accepted bring the next completion times back to a higher level. The system hence is constantly going between a sequence of rejection, with ever decreasing delay and acceptance. This is symptomatic of any deadline constrained system and is important to understand in admission control.

5.5 Summary

In this chapter different offer selection strategies were tested for the brokering mechanism. Classical job submission strategies were shown to perform badly when placed within the tender market, even when providers do not accept jobs past their deadline requirement. The filtering against broker profit for jobs which is directly linked to QoS vastly improves the situation. The further correct use of the pricing model for job selection was also shown to improve matters as it then better reflected the future scheduled work, as shown by the Hybrid Offer filter. The work in this chapter was on a small scale, it is expected to translate to a larger scale. The benefit offered derive from scheduling and using offers to better steer jobs. This will therefore prove beneficial should the cost of generating candidate schedules not out way the computational cost of performing the work, which can be expected to be the case.

Finally it was illustrated how the use of deadline based systems causes undulation in job completion time as the Grid goes through a cycle of loading and unloading of work placed upon it. In considering this cyclical pattern it is likely to present itself when the providers are nearing full load, with respect to the time constants of jobs. The presence of the deadlines will therefore cause this effect regardless of scale, especially when all resource available to a provider are in use such as in a load levelling based scenario. This will also be exacerbated when users understand the runtime of jobs and set due dates and deadlines that are close to this expected runtime.

Chapter 6

Dynamic Pricing And Offer Prioritisation

6.1 Introduction

In this chapter experiments are performed with the aim of demonstrating the effects of temporal and budget constraints upon machine/task selection, with the particular regard to the creation of a mechanism by which job prioritisation may take place. This is achieved by the introduction of dynamic pricing mechanisms, which means the system can adapt to current conditions. The pricing mechanisms available are discussed in Section 4.4.8.

The focus as with Chapter 5 remains upon high load scenarios where correct selection is most required, again as per the motivating scenarios shown in Section 1.2, as during low load situations both market and non-market mechanisms provide good welfare and only at high load do market mechanisms outperform their non-market based rivals [130].

The research objectives of this chapter's experimentation is discussed next, followed by the configuration of the experiments performed in Section 6.3. This is then followed by the results and experimental evaluation. The results are shown as a progressive set of experiments in which first the transition from temporal to economic constraints (Section 6.4.1), followed by the introduction of rescheduling (Section 6.4.2). The rescheduling causes instability in the service price. This instability is then counteracted by adapting the pricing mechanism and results are presented to demonstrate this (Section 6.4.4). After this results section a discussion of how pricing instability occurs is given (Section 6.5) along with recommendations (Section 6.6) on how to counteract the issues seen. Finally a comparison of time and cost constraints is provided in Section 6.7 followed by the chapter summary in Section 6.8.

6.2 Experimental Research Objective

This chapter's research concentrates upon job prioritisation and price stability within the ISQoS Grid. The focus of this is therefore to meet the aim from Section 1.4 of providing a market model for job submission that distinguishes between jobs in terms of QoS levels.

In order to address the objective of the research the following specific research questions have been asked:

1. How do budget and time constraints interact within the market model?
2. How can jobs be prioritised by using time and budget constraints?
3. How can price be stabilised in the market, in order to drive resource allocation stability?

6.3 Experimental Setup

6.3.1 Experimental Method

The experimental method and setup of the resources used during the experimentation is first described. Followed by in the next section a description of the settings used in the experimentation.

A Grid with 2 providers/clusters of machines was established. Each provider had 4 virtual machines (VMs) of which one also acted as a head node. Jobs were submitted at intervals, from a separate broker virtual machine instance. This prevented the broker from making one provider less competitive than the other. In total the experiment hence had 9 virtual machines.

The virtual machines ran Ubuntu 11.10 (64bit) server, with full virtualization and ran upon 4 physical hosts. The virtual environment was constructed using OpenNebula 2.0 [145] and Xen 4.0.1 [55]. Each head node had 1GB of RAM allocated and the worker nodes had 768MB and the processors each ran at a speed of 2.4GHz. The head nodes were allowed to be allocated to by the scheduler ensuring the resource space allowed was as large as possible. The size of the Grid was chosen because of the limited resources available. It was also essential to have competition between providers so 2 providers were used.

The ISQoS Grid which was setup on the VMs uses WS-Agreement for Java v1.0 for the Broker and Provider agreement process. Ganglia 3.2.0 [81] was used to provide resource information to each of the head nodes, about the availability of its resources.

The experiments were run with the settings discussed in the next section. There were 6 runs of each trace taken and 95% confidence intervals are hence marked on the figures shown in the next section. The first 15 accepted jobs of the traces have been ignored to counteract effects of starting with an unloaded Grid.

6.3.2 Experimental Configuration

The experiment was run with 100 jobs with 8 tasks each being sent from the broker. The broker had to select either to accept the job and submit to one of the two providers or reject the job. Jobs were submitted with a 30 second gap between submissions. This is shorter than the time it takes to compute a job, which means the Grid fills and resources become sufficiently scarce as per a time sensitive, high utilization scenario.

Jobs were setup to be none data intensive and the stage in/out size was only 1 Megabyte. This mitigates issues with considering the network configuration of the virtual cluster on the School of Computing's testbed. The execution size estimate for a task was given a value of 3,000. This value derives from a reference processor of 3,000 MHz multiplied by an expected duration of 1 minute. This means upon the resources available the tasks are also expected to last approximately 1 minute.

This means that if a job was allocated to a single machine it would take 8 minutes to complete. The due date was hence set no lower than the submission time + 8, with the knowledge that the Grid would soon be overtaxed. The deadline was set to the due date + 4 minutes. Though the fixed size of jobs and tasks are not entirely realistic, they are chosen to generate a situation where the Grid is overtaxed and to allow the study of selection pressures of jobs. If the job size were to be changed, preference would be given to smaller jobs with the same budget hence distorting the result obtained.

Three different budgets that jobs could be given were established: 12,000 15,000 and 18,000. These values were chosen as they intersected the likely prices that would be generated at different stages of the experiment. A fixed mark-up of 20% was chosen.

During the course of the experimentation rescheduling is introduced. In Section 6.4.1 the round robin scheduling algorithm is used. A rescheduling based variation is then used in Sections 6.4.2 and 6.4.4. The algorithms implemented by the providers may be found in Section 4.4.6.

A dynamic resource pricing mechanism was chosen which bills time for both the use of network time and resource time equally. It derived its charge from the count of actions that the provider currently has in its schedule, which maps across to a set price (see Section 4.4.8.1). The provider thus tracks current demand when setting its price by this method. This method was chosen as the jobs being submitted were of equal size and there was no great need for a more complex solution. If jobs were of a different size it would be required to have a more complex solution as the tasks might not be of comparable size meaning counting actions would not truly reflect current resource demand. The scheduling algorithm in use was not price aware so a single price could be set for all resources upon a given provider. It is possible to count the price either across the provider as a whole or against a single resource. Seen as the algorithms did not account for price i.e. for load levelling purposes there was no need to set prices differently. In Section 6.4.4 the pricing mechanism is changed so it ignores work that has been started or finished.

The broker used the hybrid offer filter selection mechanism (see Algorithm 6) and ranked jobs by broker profit, the offer ranking and selection mechanism used is shown in Figure 4.4.7.

6.4 Results & Evaluation

6.4.1 Transition to Economic Constraints Dominance

This section's aim is to show how different higher budgets become prioritised over lower budgets when dynamic pricing is used. Figure 6.1 shows the gradual increasing of the allowed due date and deadline. Initially there is no preference based upon budget shown and the temporal constraints take precedence. When the due date is at 12 minutes the lowest budget jobs at 12,000 start to be penalised and by 16 minutes the lowest budget jobs are all but completely rejected. This demonstrates selection based upon budget which is highly preferential in an economic based system. This is caused because as the due date increases a greater amount of the work is allowed to be scheduled concurrently on each provider. This causes the resource costs to increase to match the demand. In doing so the lower priority/budget jobs are priced out of the market. Note that experimental noise means that when the due date equals 8 that there is a slight variation in the average amount of jobs accepted, where no job selection preference due to budget is actually

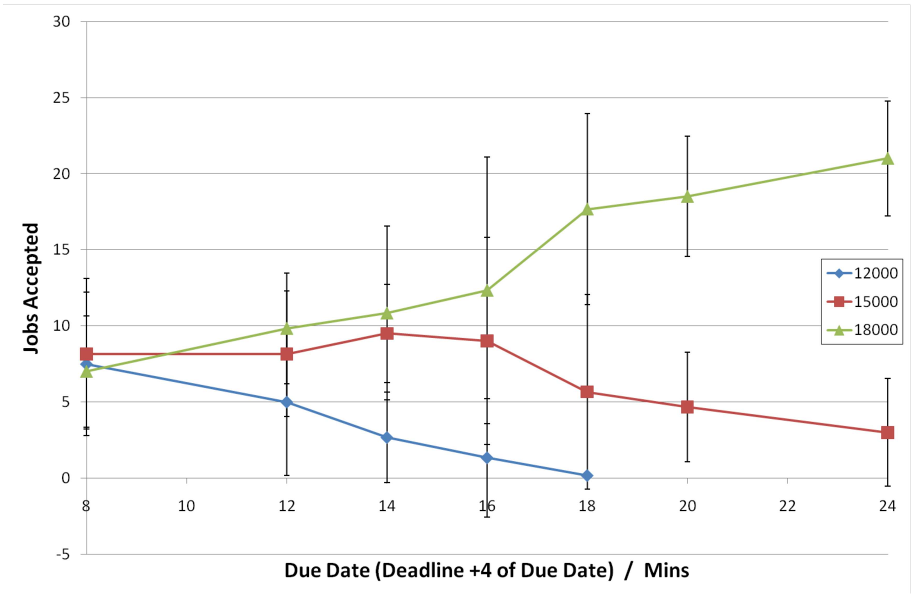


Figure 6.1: Transition to Budget Prioritisation - With Round Robin Scheduling

demonstrated.

The jobs with the highest budgets are shown to be accepted more readily in an ever increasing fashion, whilst the middle budget range jobs increases temporarily as the lowest budget jobs are no longer accepted. This prioritisation of jobs based upon the budget is seen to be a valuable property of the submission system especially as it is achieved without any need for jobs to directly compete in an auction style for resources.

In Figures 6.2, 6.3 and 6.4 constraint violations are counted. A constraint violation occurs when a job is rejected due to either the temporal or budget constraint. If the budget constrains a job then a budget violation counter is incremented. A temporal constraint violation is considered to have occurred if the time before the breakeven point is less than zero i.e. where the job is no longer making a profit. This point, where the broker breaks even, is 16.67% of the way between the due date and deadline [103]. It was useful to pick this point as the deadline was unlikely to be ever passed due to the hybrid offer filter, job admission policy in use. This policy filters out unprofitable jobs, hence any job that would complete too close to the deadline would be rejected. The number of constraint issues is incremented if either a temporal or budget constraint is violated and both constraints counter is incremented only in cases where neither constraint was satisfied.

In Figure 6.2 it can be seen that the constraint violations are initially of the temporal type only, this is because the highly restrictive temporal constraints are dominating, ensuring schedule size does not increase sufficiently, which limits the service price. As the due date gets larger the budget constraints become more dominant as the larger schedules

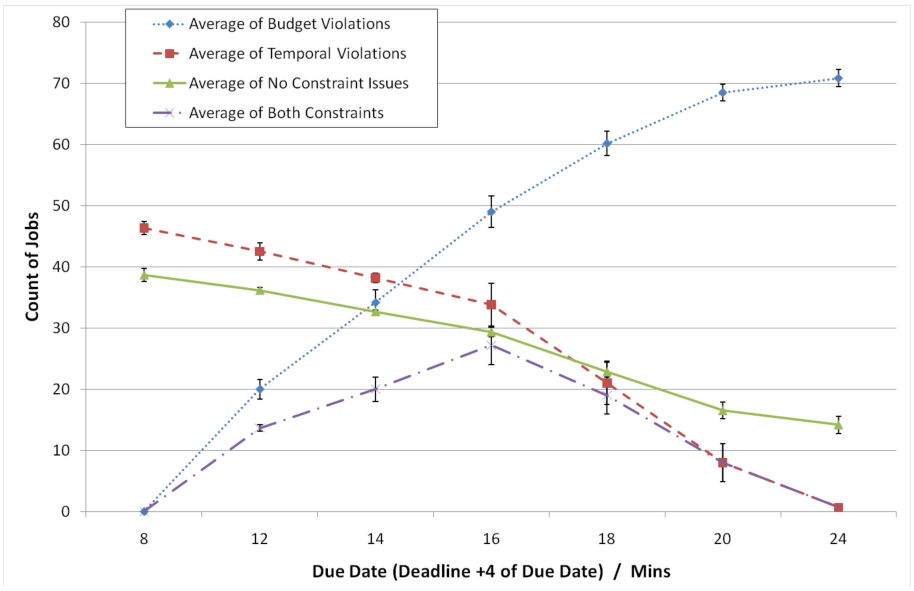


Figure 6.2: All Constraint Violations - With Round Robin Scheduling

cause the service price to rise.

In Figures 6.3 and 6.4 the selection pressures for jobs with specific budgets, namely 12,000 and 15,000 are shown, demonstrating the biasing towards the higher budget jobs.

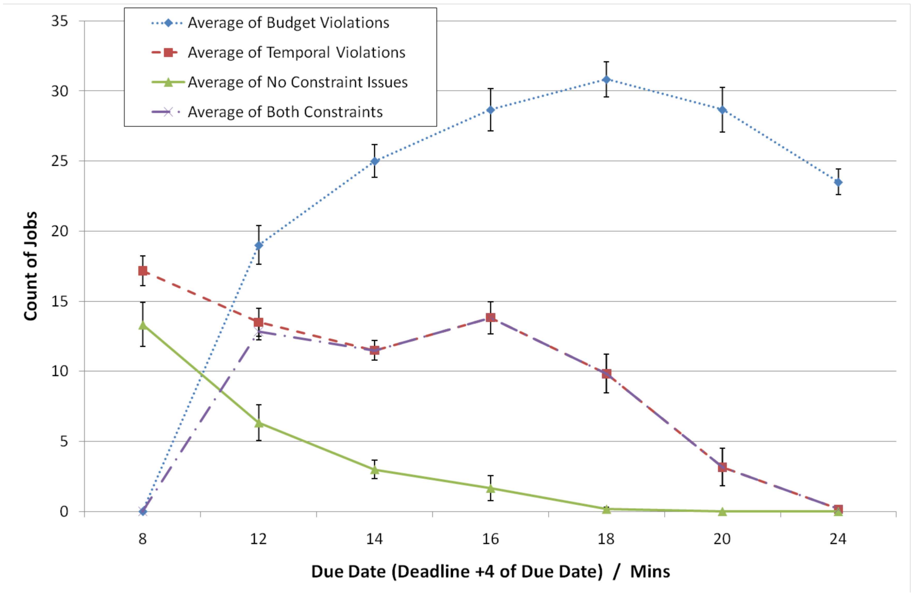


Figure 6.3: Constraint Violations with a Budget of 12,000 only - With Round Robin Scheduling

Figure 6.3 shows with a due date of 8 that jobs are either not meeting the temporal constraints or they are accepted. As the due date increases the budget constraints become dominant. Though temporal constraints are still being violated, it is also the case that the budget constraints are being violated as well, so practically the budget constraints are taking precedence, as violating both constraints has the same effect as violating only one constraint i.e. job rejection. It should be noted due to randomness that in Figure 6.3 the amount of jobs violating the budget constraint drops with a due date of 20 and 24. This is because fewer jobs had a budget of 12,000. It should also be noted that all jobs with a due date of 24 and budget of 12,000 are rejected.

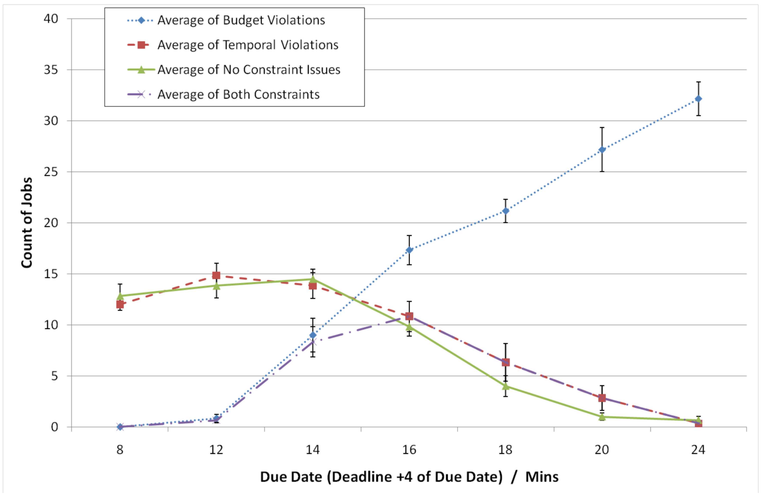


Figure 6.4: Constraint Violations with a Budget of 15,000 only - With Round Robin Scheduling

Figure 6.4 like Figure 6.3 again shows that initially jobs are either not meeting the temporal constraints or they are accepted. As the due date is increased the budget constraints again become more dominant, but it takes much longer for jobs to be predominantly rejected because of budget violations, which is an indication of the desired property of prioritisation.

To provide further insight into what increasing the budget means in terms of selection a study of the start delay associated with each billed job in the 6 runs performed is shown in Figures 6.5 and 6.6. Figure 6.5 shows values for all due dates and Figure 6.6 for due date 18, 20 and 24 only.

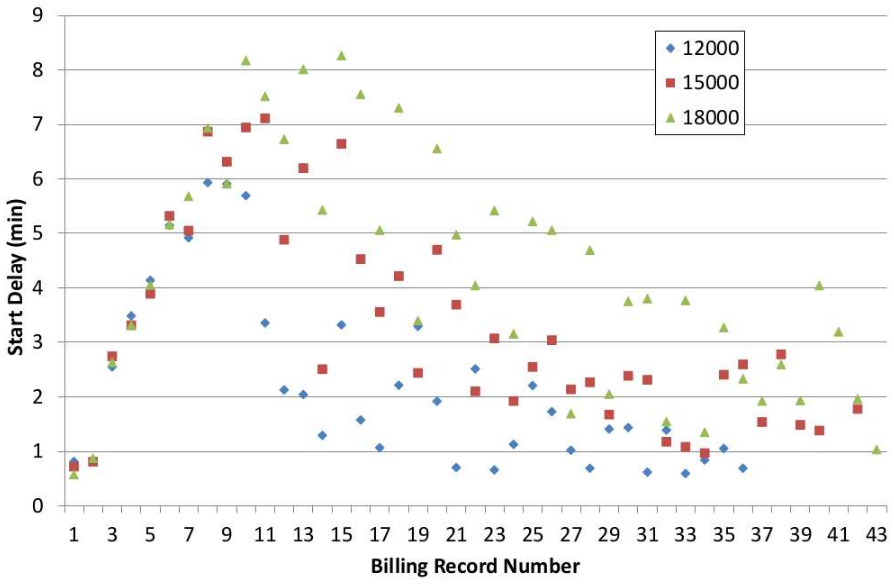


Figure 6.5: Start Delay - With Round Robin Scheduling - For All Due Dates

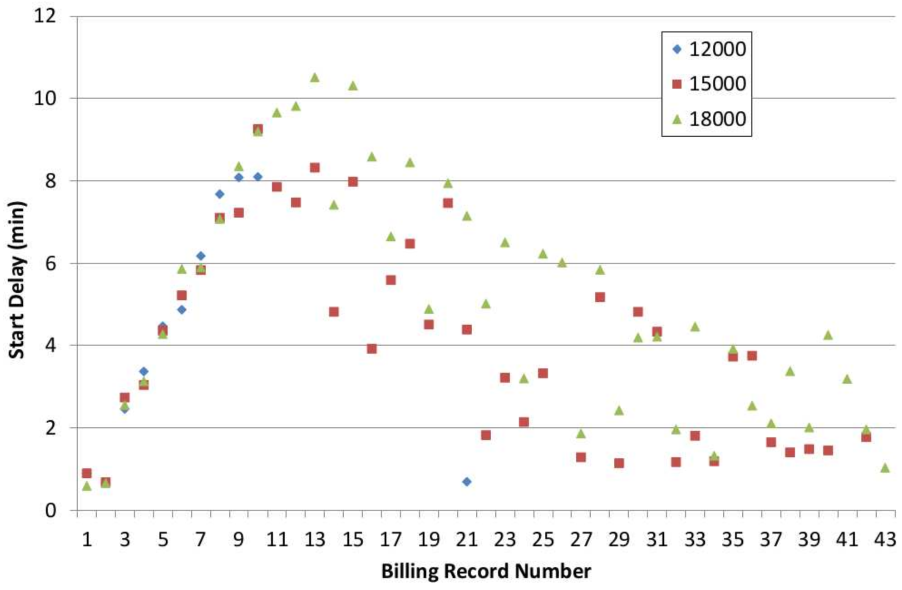


Figure 6.6: Start Delay - With Round Robin Scheduling - For Due Dates 18, 20 and 24

The higher budget jobs obtain higher average start delays than lower budget jobs. This seems initially counter to what is required, however the start delay shows how long a job has to wait for before it is executed and hence how busy the provider is at the time of billing. Therefore more jobs in the schedule means two things:

1. Jobs have to wait longer to be executed
2. Jobs have to pay more in order to be placed in the schedule, due to the higher demand

This means if a higher budget job arrives it has a greater chance of being accepted in the larger schedules where it would have otherwise been rejected, if the start delay was not accepted. This therefore brings the average start delay experienced up as compared to the lower budget jobs i.e. 12,000. The higher budget jobs would therefore experience a higher variance in the start delay that they experience.

This can be seen when examining Figures 6.5 and 6.6 side by side. Initially it can be seen in the earlier bills in Figure 6.6 that the lower budget jobs (12,000) are delayed equally as with all other jobs but are then priced out of the market by the time the Grid reaches full load. It can also be seen that the averages are higher in Figure 6.6 than Figure 6.5 due to the exclusion of the lower due dates from the figure.

Due date also effects the start delay as it simply permits greater delays to be possible as shown in Figure 6.7. The figure shows the average start delay for each billed job over 6 runs. The start delay in the earlier bills rises rapidly due to the cold start of the Grid deployed on the testbed. i.e. with no jobs initially present. Once the initial shock of accepting many jobs is over the billed jobs can be seen to be tending towards a steady consistent start delay. This steady start delay is obtained faster when the temporal constraints are made more stringent. This indicates that if production systems were to follow this model that the intent of users to obtain the fastest completion time possible and setting the temporal constraints accordingly is hence likely to cause stabilisation faster in regards to amount of delays incurred.

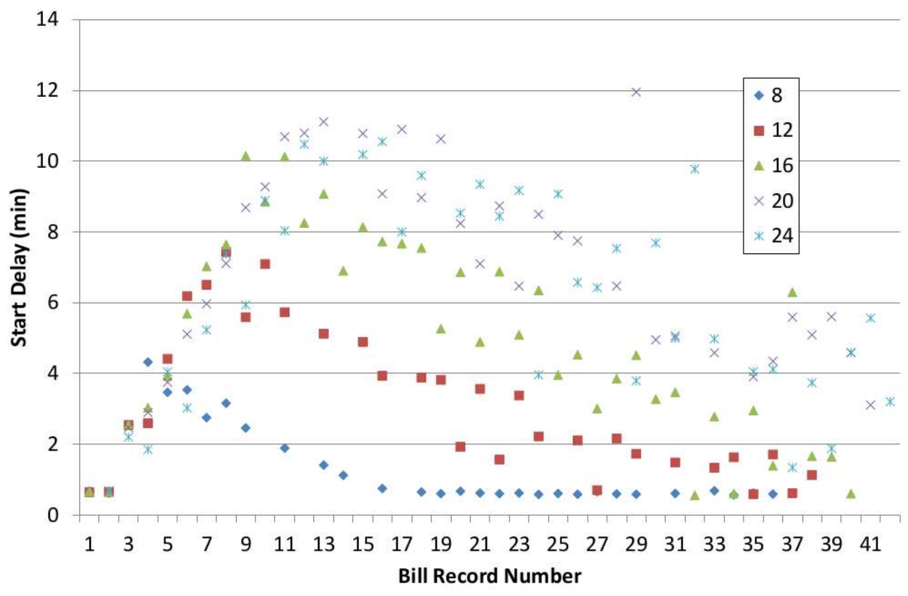


Figure 6.7: Start Delay - With Round Robin Scheduling - Shown By Due Date

6.4.2 Rescheduling

It was previously indicated in Section 4.4.6 that different scheduling algorithms may be selected at provider level. This can introduce more complex situations where rescheduling may be performed. In this case we show how this can have a profound effect upon the market.

In the case where rescheduling is performed similar results can be obtained to the none rescheduling case, there are however notable differences.

Figure 6.8 demonstrates that the budget constraint becomes more dominant than the temporal constraints much later on as compared to Figure 6.2. This is reflected in the distinction between the amount of jobs accepted of each type, where the lower budget jobs are no longer rejected entirely. This is in part caused by fewer jobs being accepted (shown later in Figure 6.28), leading to a slightly lower resource cost. On average 28/85 jobs without rescheduling and 26/85 with rescheduling, but it is also more significantly down to greater fluctuations in the service price.

To highlight the fluctuations in service price Figure 6.9 shows a trace of the service price over the 6 runs which have a due date of 20. It shows that when rescheduling occurs the service price drops significantly at various stages in the trace, which is not the case where rescheduling is not in use. The drop in price means it becomes low enough for the lower budget jobs to be accepted. This is the case even though the temporal constraints are relaxed meaning higher workloads on the server could be expected. These higher workloads would therefore give rise to an expectation that the service price was also high.

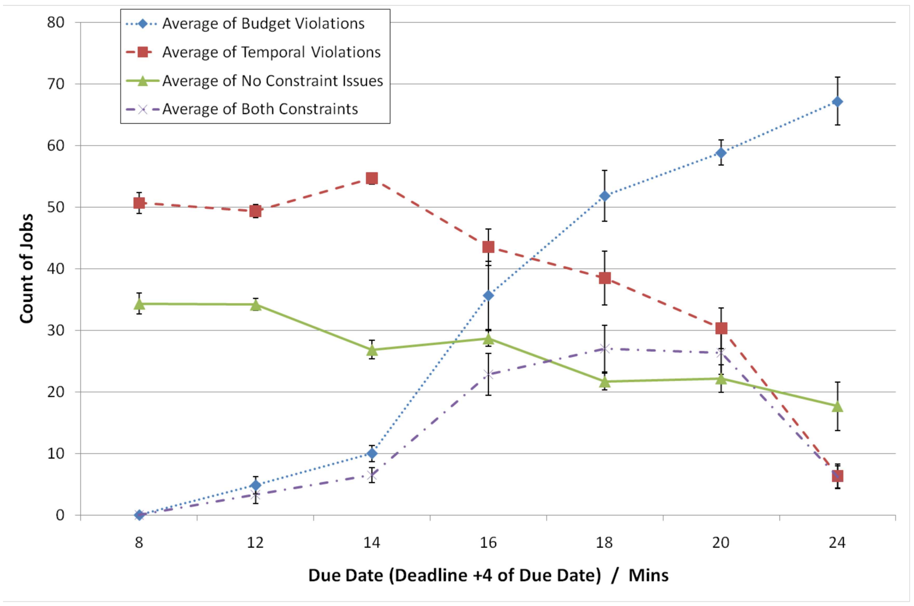


Figure 6.8: All Constraint Violations - With Rescheduling

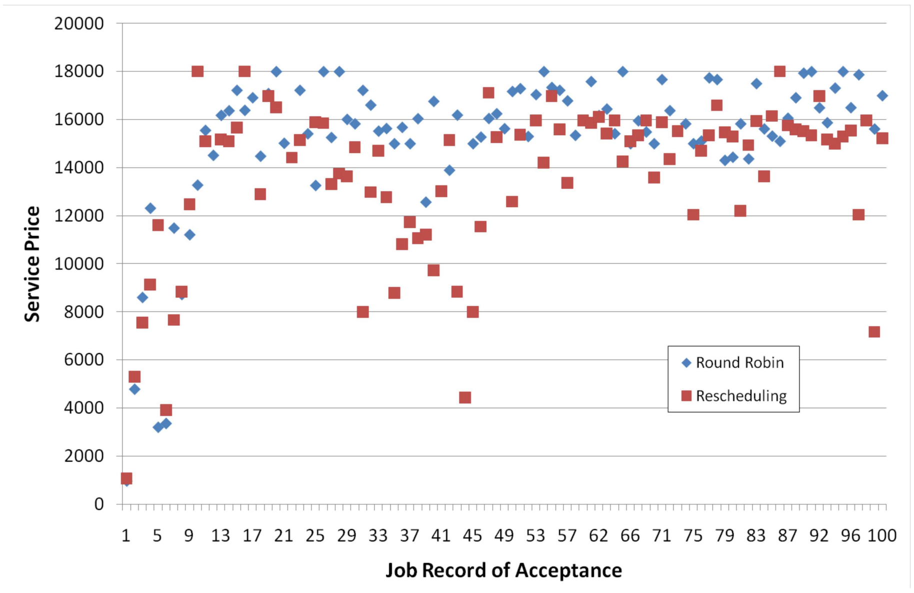


Figure 6.9: Effect upon Service Price of Rescheduling (Due Date = 20, Average of all Runs)

6.4.3 Causes of Price Instability

In order to explain what is happening and why the price is being reduced a simplified case is presented. A single processor is examined, with the arrival of four jobs that submitted in sequence. The temporal constraints is simplified and due date is considered to be equal to the deadline and the time is represented as discrete time rather than continuous. The arrival of each of the four jobs is shown, with the top row representing the processors availability and the rows after representing the jobs availability. The time increments on by one step each time and is represented by the dashed area on the figures presented next. This example is presented for the rescheduling and round robin cases to enable the examination of the differences between the two sample cases.

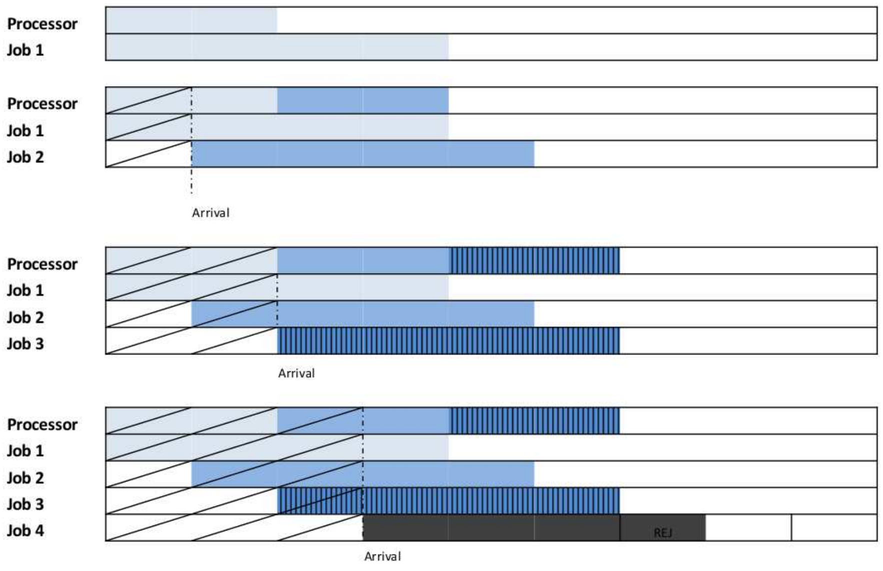


Figure 6.10: Instability Demonstration - Round Robin

In Figure 6.10 the round robin base case is presented. It can be seen that Job 1 arrives on the processor and takes up two blocks but has a deadline of four ahead of its arrival time. The processor is free so it is allocated immediately. The arrival of Job 2 one block of time later means Job 1 and Job 2's allocable space overlaps. Job 1 having already been allocated takes priority and Job 2 is placed afterwards. Similarly this happens to Job 3. Job 4 arrives and its allocatable space is all but overlapped by Jobs 2 and 3 and as they arrived first and their reservations are respected, hence Job 4 is rejected. This is unremarkable but is established so the rescheduling case may be examined as well.

In Figure 6.11 we see the rescheduling case being demonstrated. The arrival of Job

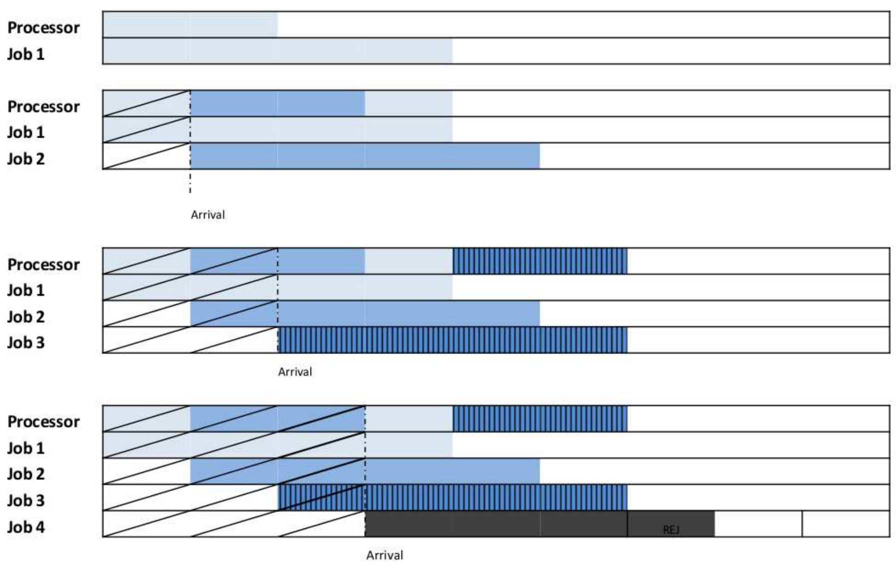


Figure 6.11: Instability Demonstration - Rescheduling

1 is handled in much the same way as before given the processor is unloaded. The next arrival one block of time later means Job 1 and Job 2's allocable space overlaps. This time in order to break the precedence of earlier jobs always coming first, Job 1 takes a lower priority. The first "block in the job"/task has already been executed. The second task may now however be postponed. Job 2 hence starts immediately in this time period. When Job 3 arrives Job 2 is prevented from moving by the end of Job 1. Job 3 is hence placed at the end of the processors available space. Finally Job 4 arrives and it is again rejected due to a lack of space.

This simple case shows how the jobs that arrived earlier can block future jobs from being moved. It also shows how the billing of each job can be moved closer together. This is important as this removes work from the schedule. Once this has been done it means the service price drops, in cases where all actions in the schedule are used to determine current load. Rescheduling lends itself towards two jobs completing on a single resource within short succession, which rapidly reduces the service price.

6.4.4 Price Stability

Given there is instability in the service price and given the explanation presented in Section 6.4.3 a mechanism to alleviate the problems is tested. Comparisons are therefore made between the round robin, rescheduling and the re-priced (rescheduling) variations.

To further emphasise what is happening and the explanation given in Section 6.4.3 the time between billing events, this is shown in Figure 6.12. It can be seen the time between

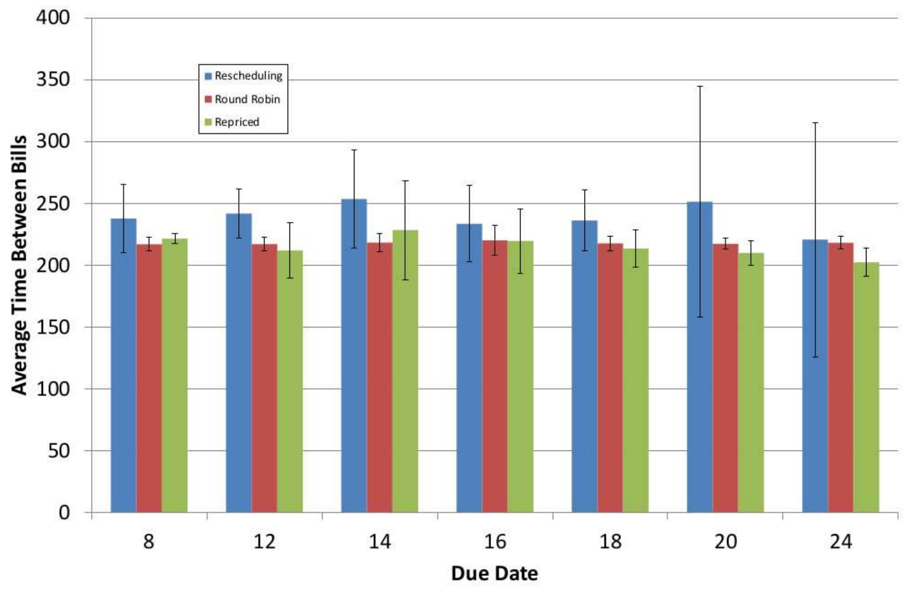


Figure 6.12: Average Time Between Billing Events for each Given Due Date)

billing events does not differ that greatly between any of the three series before due dates 20 and 24. It can then be seen that the rescheduling variation gets a much greater range of possible values and the average from one run to the next differs markedly. Essentially the average time between bills varies much more in the rescheduling case then it does in either the round robin or repriced variations.

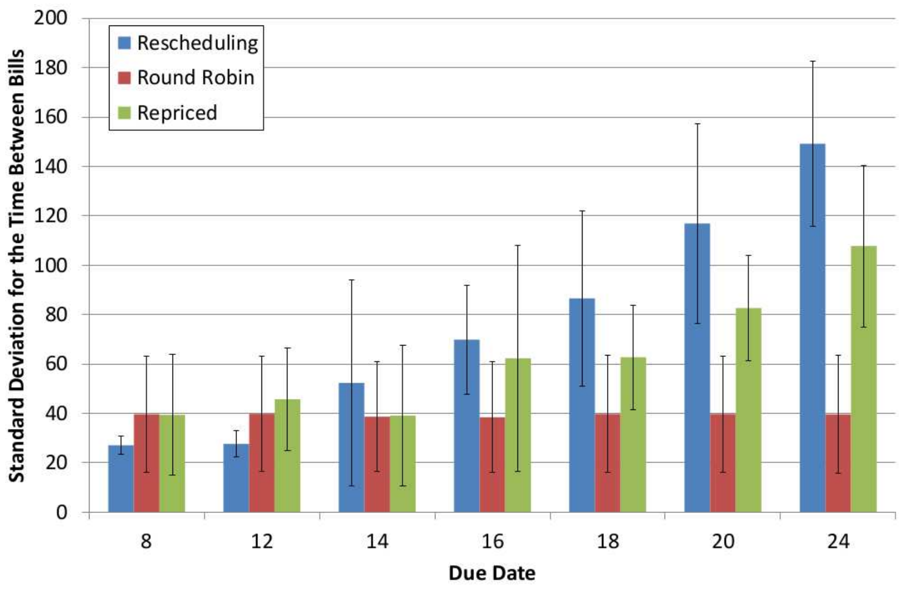


Figure 6.13: Standard Deviation for the Time between Billing Events for each Given Due Date)

To further emphasise what is happening the standard deviation for the time between bills is shown in Figure 6.13. The standard deviation for both the rescheduling case and the repriced increases but the rescheduling case does so much quicker. In considering Figures 6.13 and 6.12, it can be seen that the average time for the Round Robin and Repriced variations does not move, while the rescheduling variation does. It can also be seen that the time between bills in the rescheduling and the repriced variation begins to vary more, though the repriced variation maintains the average better. This maintenance of the average means the price is likely to be more stable.

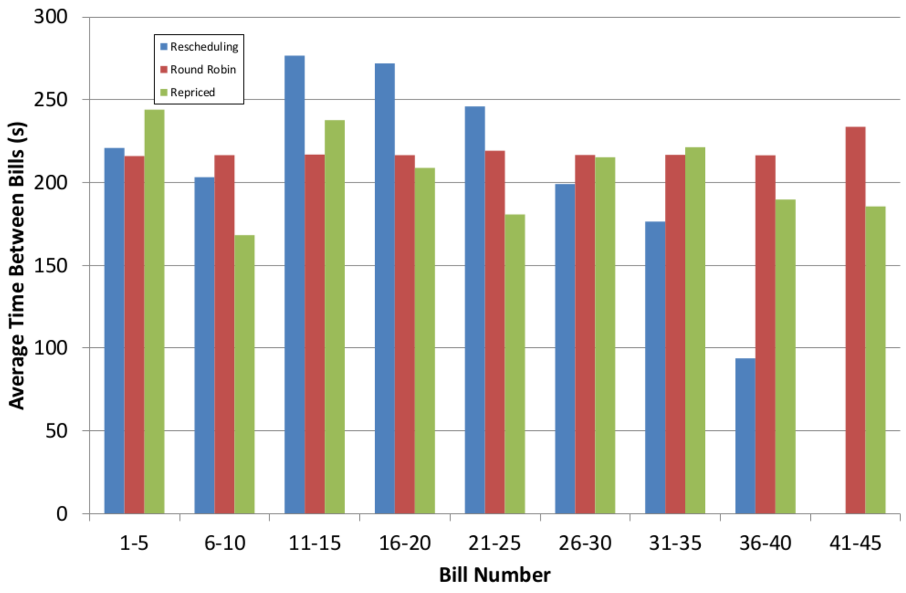


Figure 6.14: Average Time Between Billing Events for Due Date = 24)

Lastly in regards to the time between bills, the average time between bills over time is observed. Bills are grouped and the time between the bill and the previous bill on the same provider is given. The earlier and later buckets may be ignored as the trace begins and ends but are shown for completeness. The average time between bills can be seen to move over time for the rescheduling case but not the repriced or round robin cases for higher due dates such as 24 (see Figure 6.14). The round robin variation stays very stable and the repriced variation undulates around the round robins value. This is not seen in earlier due dates such as 8 and 16, as shown in Figures 6.15 and 6.16.

In testing the instability consideration also has to fall on the average service price over each of the runs (see Figure 6.17). The service price for rescheduling and round robin are about equal/and within error margins until the later due dates of 20 and 24, which mirrors what is happening with the time between bills. It can be deduced that the only thing that distinguishes round robin and the rescheduling variation is related to how the rescheduling occurs i.e. something happens due to rescheduling, or the ineffectiveness

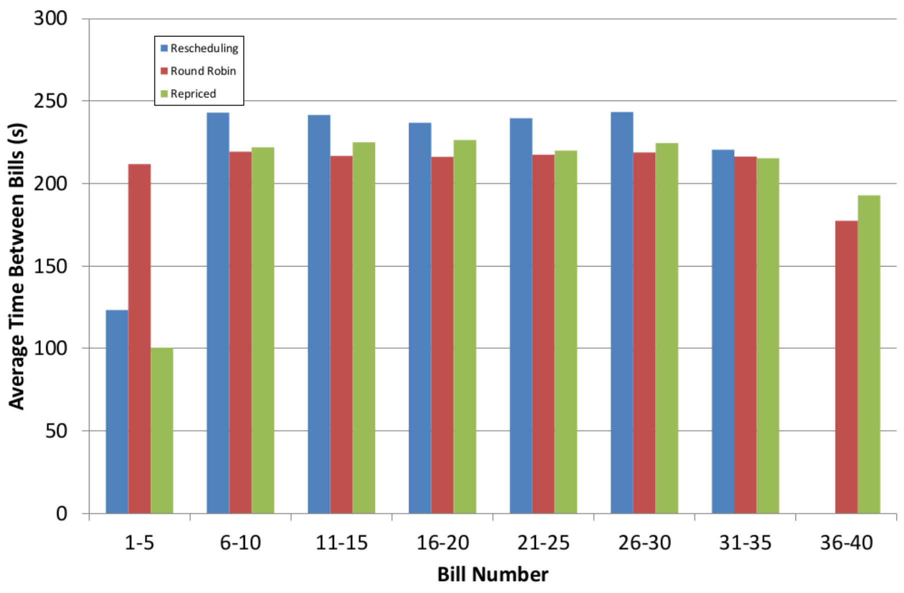


Figure 6.15: Average Time Between Billing Events for Due Date = 8)

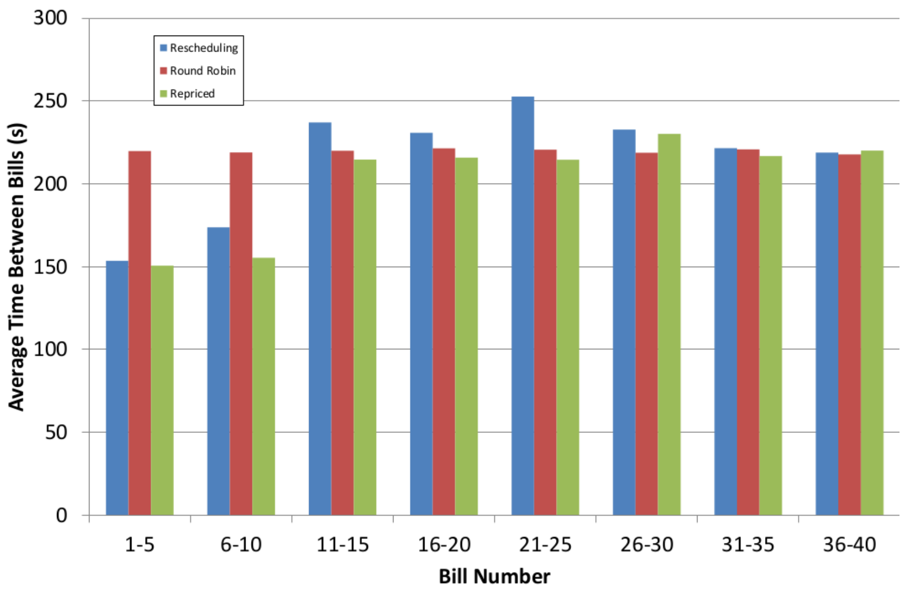


Figure 6.16: Average Time Between Billing Events for Due Date = 16)

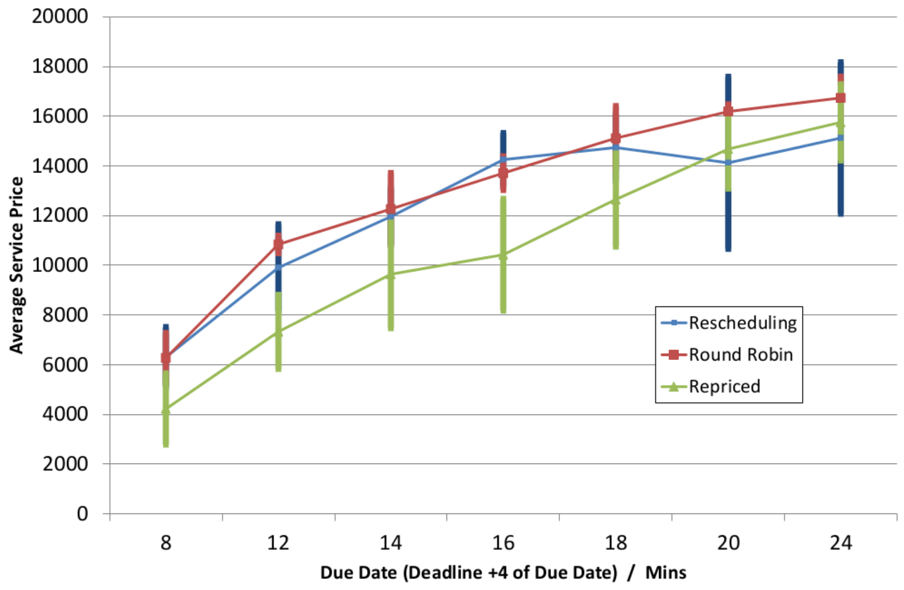


Figure 6.17: Average Service Price (For Each Due Date)

of the rescheduling algorithm. The rescheduling variation cannot therefore be efficiently performing the rescheduling with respect to the pricing model, causing the service price to go down.

It can also be seen in Figure 6.17 that the re-priced rescheduling variation does not initially perform as well in terms of service price. This is not surprising as actions that have been executed do not contribute to the service price, hence it is at a disadvantage. It can be observed that as the due date increases, this effect is diminished. This is due to larger schedules, which ensure the work that has been executed and is yet to be billed makes up a smaller proportion of the schedule.

This disadvantage should however exist throughout and it could be expected that its average service price should never approach that of the rescheduling variation. It however surpasses the service price of the plain rescheduling variation, which should always be in a better position to obtain a higher service price.

It should also be noted that there is a possibility as the maximum budget possible is 18,000 that the service price is tending towards a maximum in an asymptotic fashion i.e. it will not reach this maximum value, but will approach very close. The re-priced variation could be less susceptible during its approach to this maximum.

The variation in service price shown in Figure 6.17 is further explored in Figures 6.18 and 6.19, as this gives a notion of price stability, which is an important factor in good allocations within a Grid market [199].

It should be noted how the re-priced service price does not vary as much as the

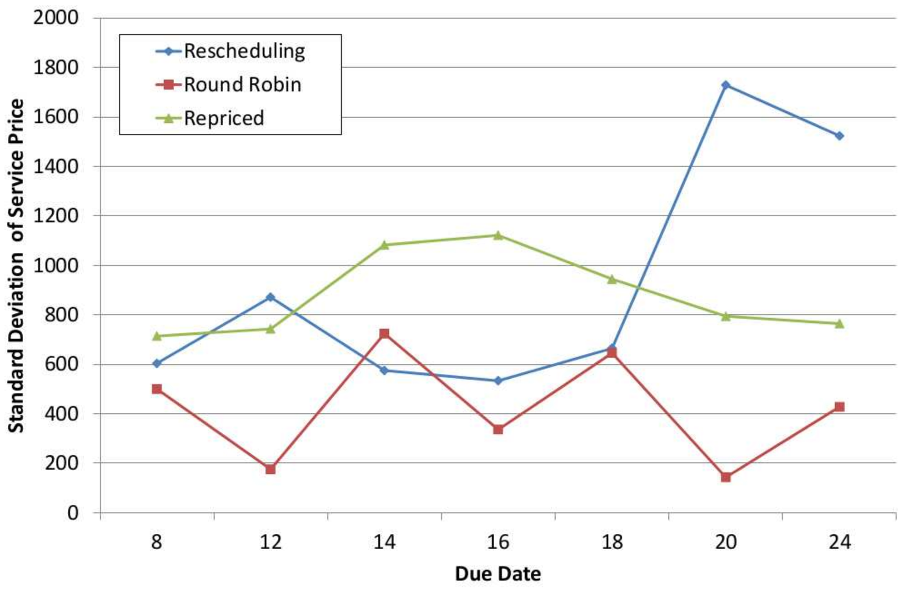


Figure 6.18: Standard Deviation of the Average Service Price (as shown in Figure 6.17)

rescheduling variation when the due date is larger i.e. 20 or 24. The rescheduling variation starts small and then gets disproportionally larger at due dates 20 and 24, whereas the re-priced variation starts slightly larger and stays roughly the same size.

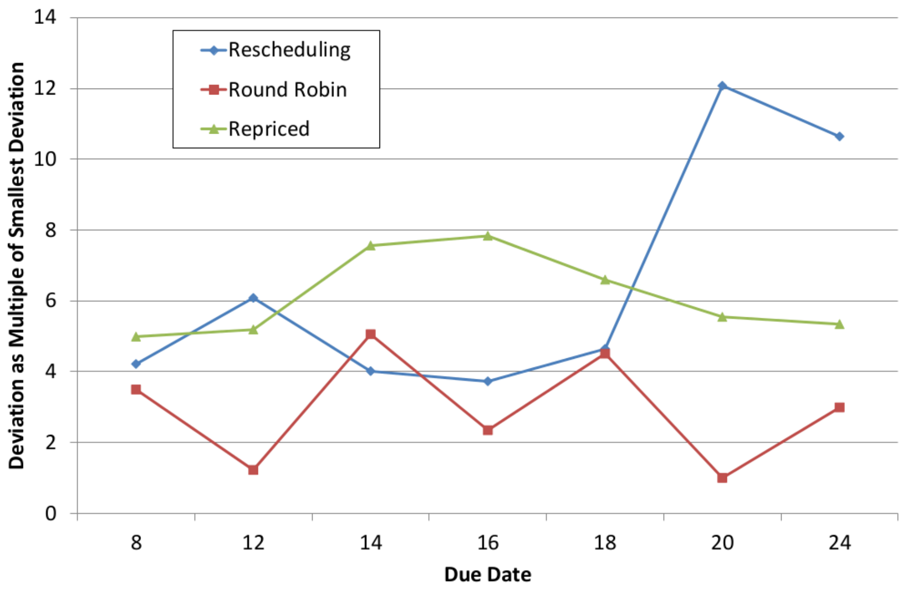


Figure 6.19: Standard Deviation as Multiple of Smallest Deviation Value

In order to highlight the change in standard deviation and hence how under control the pricing mechanism is Figure 6.19 is introduced. The standard deviation is altered to be a multiple of the lowest standard deviation found (Round Robin at Due Date 20). The summary of values is shown in Table 6.1.

Table 6.1: Summary of Standard Deviation of Service Price

	Rescheduling	***Round Robin***	***Re-priced***
Lowest Standard Deviation	3.72	**1**	4.99
Highest Standard Deviation	**12.07**	5.06	7.83
High − Low	8.35	4.06	**2.85**
(High − Low) − (Re-priced's High − Low)	**5.5**	1.21	**0**

It can be seen that the lowest standard deviation is provided by the round robin scheduling algorithm. The highest standard deviation and hence the one that is most out of control is the rescheduling variation. This is only seen at due dates 20 and 24, where more opportunity for rescheduling is available. The re-priced variation does not suffer any ill effects at the higher due date values.

Re-priced has the lowest difference between the highest and lowest standard deviations for any given due date i.e. regardless of the due date the standard deviation/spread of possible services prices is the least.

In comparison the round robin has a similar spread at 1.21 times the re-priced variations spread. The rescheduling variation suffers heavily though and has a standard deviation 5.5 times as large as the re-priced variation. The rescheduling variant before if ignoring due dates 20 and 24 would be much improved and have the following values (Table 6.2):

Table 6.2: Summary of Standard Deviation of Service Price Ignoring Due Date 20 and 24

	Rescheduling
Lowest Standard Deviation	3.72
Highest Standard Deviation	6.08
High − Low	2.36
(High − Low) − Re-Priced's Delta	-0.49

Hence it can be seen that it performed very similarly to re-priced and round robin until this point.

To highlight the fluctuations in service price in a similar fashion to Figure 6.9 and to show the repriced variation does not show the same variation in prices Figure 6.20 is introduced. It examines the service price over time in the same fashion as Figure 6.9. It shown a trace of the service price over 6 runs where the due date equals 24. It shows that when rescheduling occurs the service price drops significantly at various stages in the trace, which is not the case when repricing is used even though rescheduling is still used in addition to the changed resource pricing strategy. The re-priced variation obtains

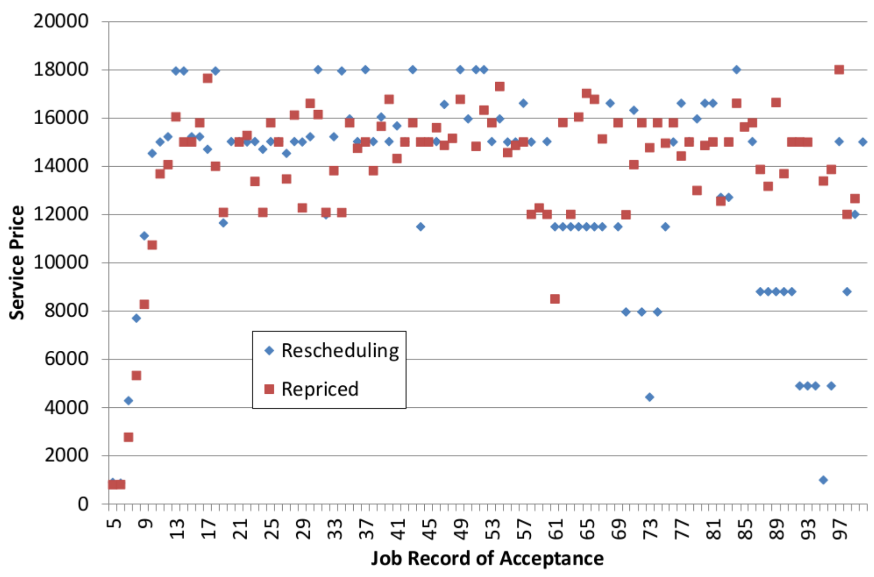


Figure 6.20: Service Price over Time (Due Date = 24)

a similar service price to the rescheduling variant however more importantly, it does not suffer from random dips in the service price and only one outlier moves away from the 12,000 to 18,000 band of service prices.

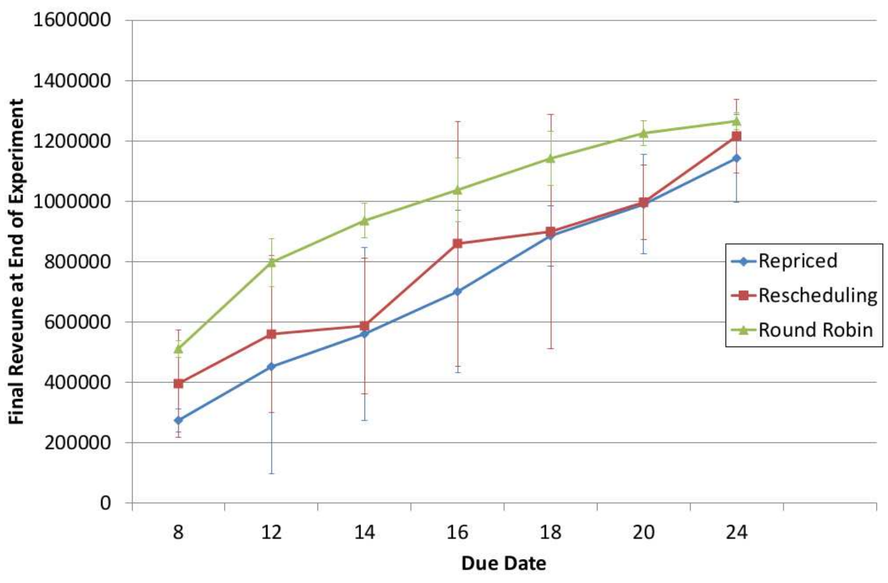


Figure 6.21: Final Overall Revenue

In considering the price stability the final revenue generated by the broker at the end of the experiment is also of great interest. It relates both to the average service price and to the job acceptance count. This is shown in Figure 6.21. We can see that the re-priced variation returns roughly the same amount of revenue by the end of each run in the experiment as the rescheduling variation.

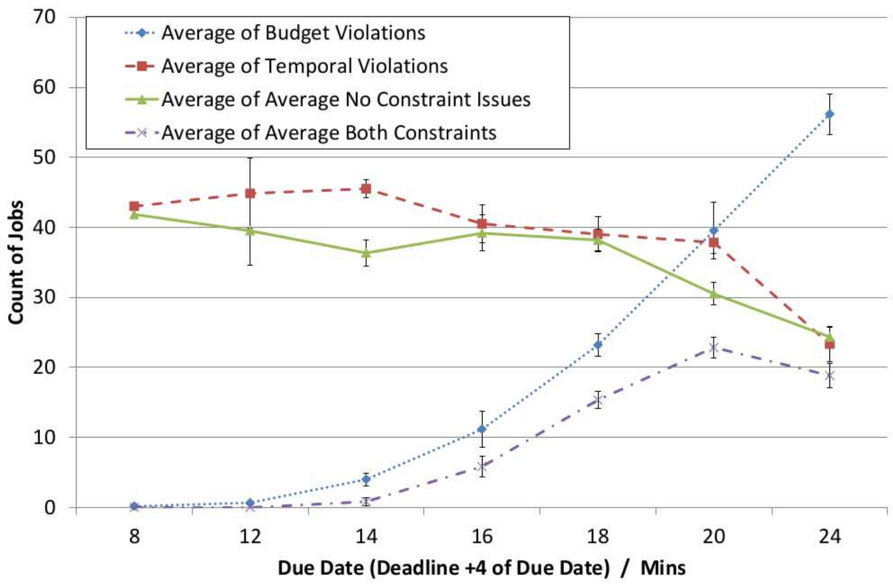


Figure 6.22: Constraint Violations - Re-Priced (All Budgets)

The change in service price is likely to affect job submissions in regards to the preferential treatment of higher budgets in comparison to lower ones. Hence the next aspect to consider is the constraint violations, in a similar fashion to Figures 6.2, 6.3, 6.4 and 6.8.

Hence Figure 6.22 is presented and placed in comparison to Figures 6.2 and Figure 6.8. It is clear the budget constraints are delayed in taking effect. This can however be compensated for by increasing the service price associated with a given count of jobs i.e. the mapping between price and the current count of actions in the schedule. The curve for the budget constraint violation also appears concave rather than convex as shown in Figure 6.2, Figure 6.8, which is of some interest. This therefore gives some indication of what happens if the job acceptance count remains high (see Figure 6.28) when the budgetary constraints are not violated regularly.

Considering that the constraint violations differ the acceptance preferences between different budgets is required to be understood. The preference mechanism remains strong in the re-priced variation, as shown in Figure 6.23, though the lowest budget jobs are not removed entirely, which is the case with Round Robin (see Figure 6.1) and Rescheduling (see Figure 6.24).

The most notable comparison when observing Figure 6.1, 6.23 and 6.24 is looking at the position in which the separation based upon available budget occurs. Essentially the higher the pressure upon the budget constraint the earlier this separation appears. The

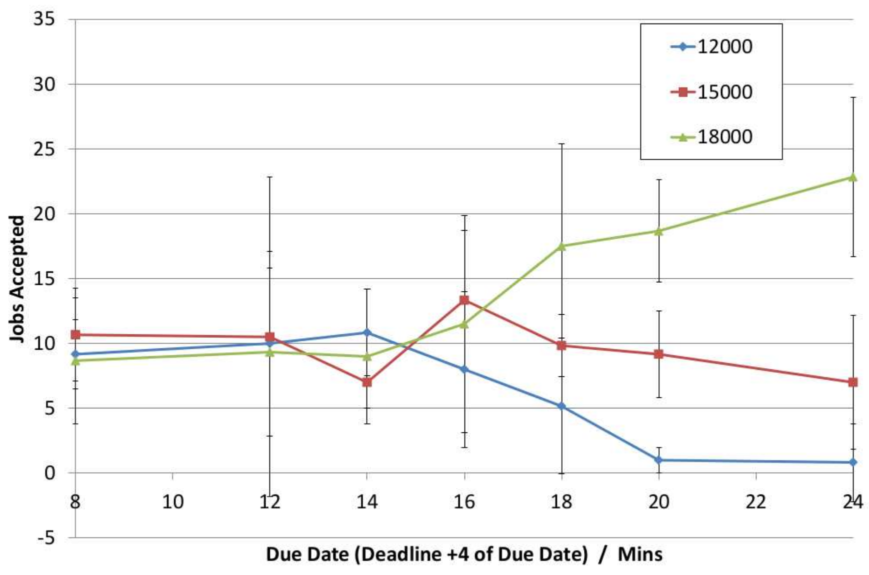


Figure 6.23: Job Acceptance and Budget Selection Preference for re-pricing

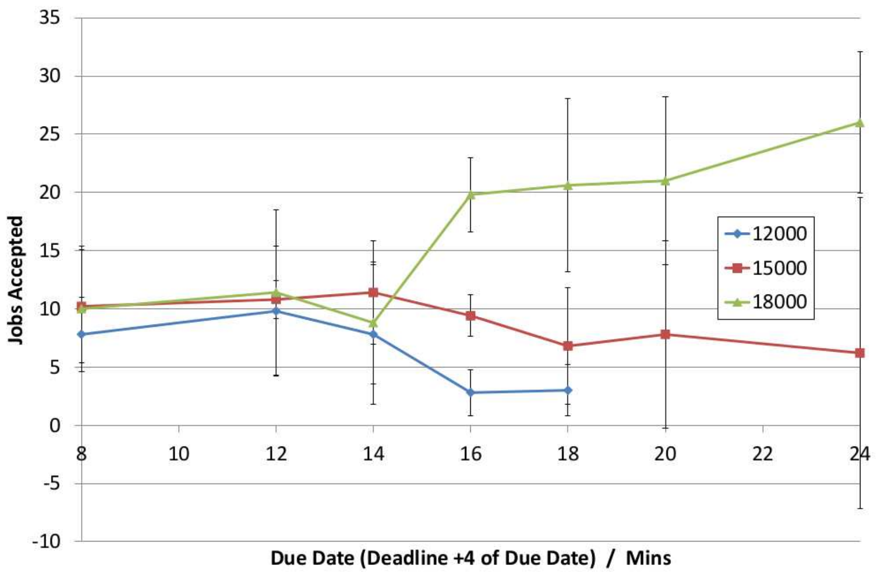


Figure 6.24: Job Acceptance and Budget Selection Preference for Rescheduling

Round Robin algorithm has the greatest selection pressure followed by the Rescheduling and then Repriced variations.

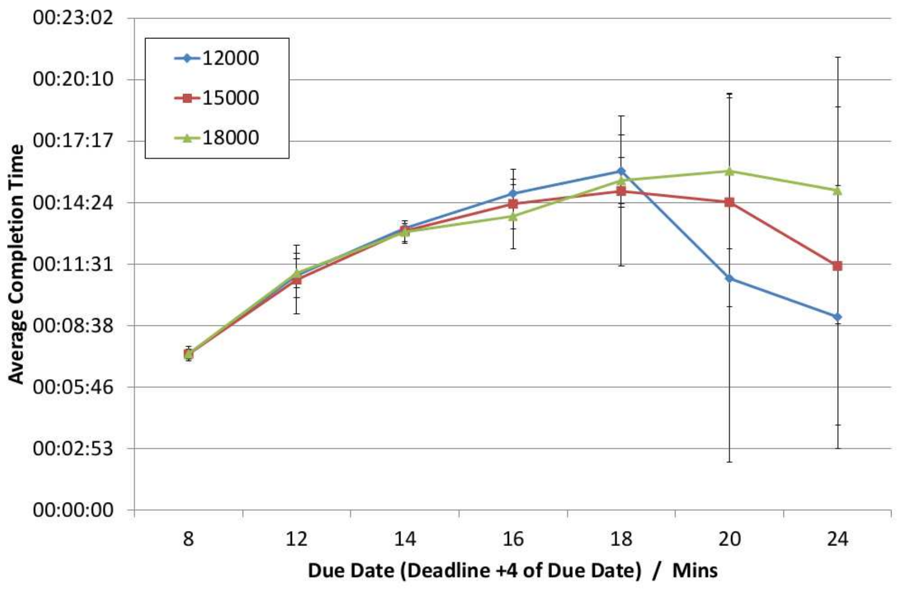


Figure 6.25: Rescheduling - Job Completion Time

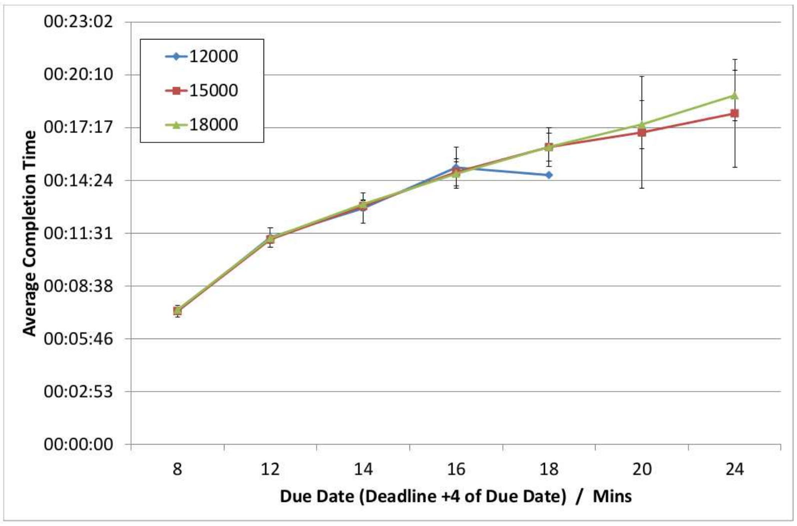


Figure 6.26: Round Robin - Job Completion Time

The completion time for each job is also considered to be deeply relevant. The average completion time seems to support the idea that before a due date of 18 that the round robin and rescheduling are just as efficient as each other i.e. the schedules size/queue lengths

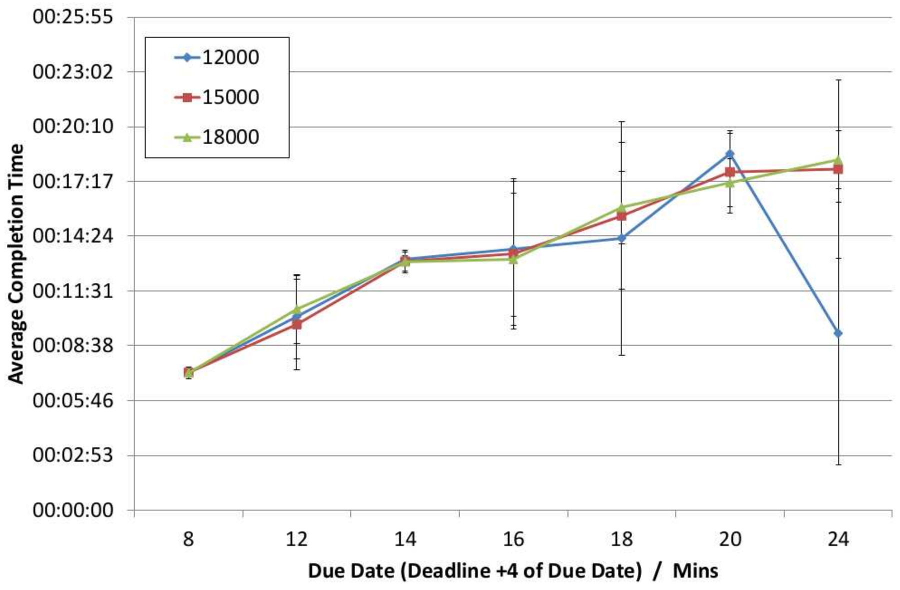


Figure 6.27: Rescheduling Re-priced - Job Completion Time

have not really developed enough to cause a lot of rescheduling to occur. All three graphs seem to show times with a due date of 16 to have a completion time of around 14 minutes and 24 seconds.

In terms of rescheduling when ignoring completed work (re-priced) it is about as efficient as the other rescheduling mechanisms when comparing completion time. Rescheduling as seen in Figure 6.25 seems to show separation based upon budget (a fanning out of different completion times) which is not seen in Figures 6.26 and 6.27 where only the lower budget jobs seems to be greatly affected.

Finally in this section we consider the count of jobs/offers accepted. This count is higher for the re-priced variation in comparison to either the rescheduling or round robin variations (see Figure 6.28).

We can see in terms of overall acceptance that all three variations have very similar results, the average for re-priced remains slightly higher and round robin is the next most favourable. Rescheduling variation is finally the least favourable.

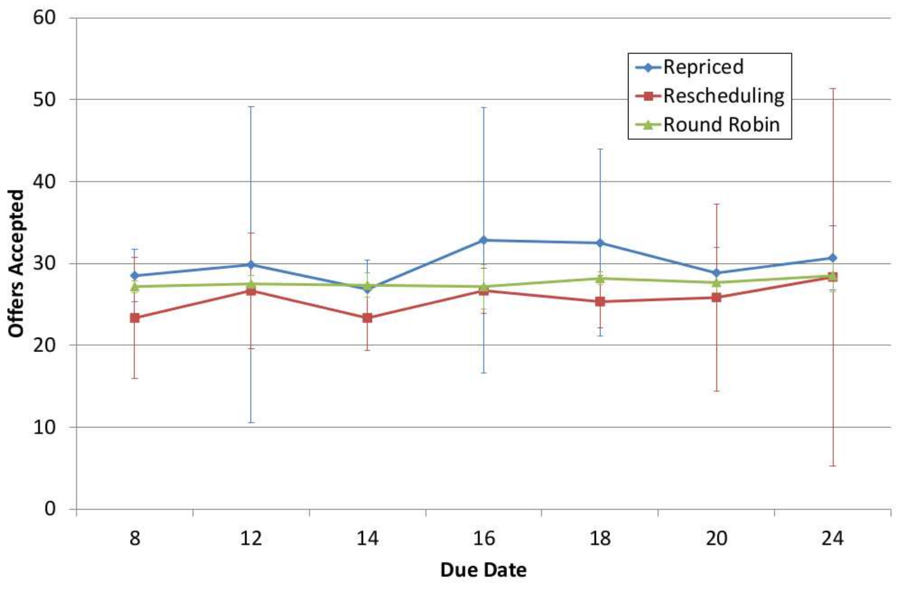


Figure 6.28: Job Acceptance

6.5 Discussion upon Price Stability

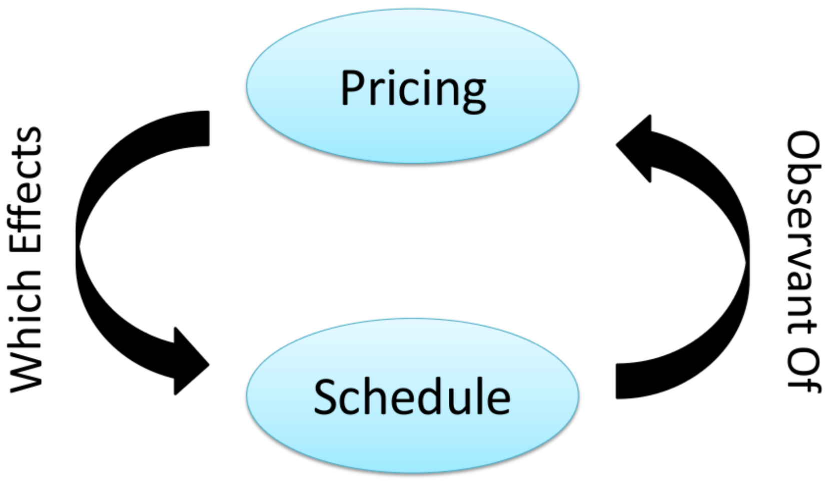


Figure 6.29: Price and Schedule Dependence

In this section we discuss the stability of prices in relation to dynamic pricing models. Dynamic pricing is the adaptation of the price of resources in accordance with demand. This gives rise to a situation shown in Figure 6.29 where the following is the case:

- The price is set by observing the current schedule for demand information
- The schedule has work added to it which is effected by the price of resources, causing a circular relationship.

- The schedule has work removed from it which may affect the resource price, though this depends upon the pricing strategy used.

6.5.1 Maintaining Selection Pressure

If this relationship is to be maintained, in order to get prioritisation effects then the price must feature as a selecting pressure in the schedule. If it does not then the price effects the schedule relationship is diminished or removed while the price will remain observant of the schedule.

This relationship may diminish if the budget is set sufficiently high as to no longer be a selection pressure. Noting that the budget being to high is in relation to the mark-up and amount of work that is part of the job i.e. more work or higher mark-ups means the budget must also be higher given the same resource price. In considering the user's preference towards saving money, it is likely that they are going to render as little money available for the job as possible, so they are incentivised to maintain the price effects the schedule relationship.

6.5.2 Effectors of Price Stability

In terms of price stability heterogeneity is an important aspect this can either relate to the jobs, aspects of the schedule or the resources used. In terms of jobs the following aspects cause effects:

Size of the Job (Amount of Work) If the price is set by how far ahead of time the last work completes is then the following is the case. The greater the size of the job, in terms of how much is to be computed then the greater the resource price. If billing occurs with larger jobs then the comparison is performed against the current time so the larger jobs no longer affect the price. This is beneficial as the price is gradually becomes less dependent upon a single job. This mechanism prevents the price from dropping rapidly but when new jobs arrive the price can still spike, although admittedly in doing so it follows demand.

Size of the Job (Task Count) This directly effects how many actions are placed in the queue. If pricing is based upon an action count then during billing actions are removed from the schedule and the price drops. Hence when jobs have many tasks the price drops further, making this form of pricing subject to heterogeneity in the job. The use of action counts in this case also does not respect the relative size of work, so again causes difficulties.

Billing Frequency/Rate Work is Cleared from the Schedule The frequency of billing is of importance, in cases where the removal of work from the schedule causes changes in the price. This can be seen in cases such as where prices are based upon action/job/task counts, or amount of work in the schedule in terms of cycles, if completed work that is not billed contributes towards the current price.

Billing Frequency If billing is performed at the end of each job more work is cleared from the schedule then if it was billed at the completion of every task. In terms of stability it therefore may be beneficial to bill on a per task basis, even though this breaks the initial premise of billing for a bag of tasks.

Job Acceptance Frequency which is in part upon the arrival rate of jobs, i.e. if no jobs are submitted none may be accepted. It is the principle actor in terms of increasing the current resource price and the greater the amount of work accepted at any one time then the more the price will rise.

It is also based upon parameters such as the remaining time available for the job, i.e. due date and deadline and the budget requirements and size of the job. Hence the overlapping of jobs in regards to their available space should be considered such as in Section 6.4.3.

The job acceptance frequency is important as the more work in a given schedule then the greater the price. The acceptance rate should also relate strongly to the ability to clear work from the schedule i.e. over the long term accepting more work than can be cleared is unsustainable as wait times increase indefinitely. Therefore in terms of resources the following aspect also causes effects:

Machine Availability and Speed This relates directly to the ability of the system as a whole to both accept work, i.e. the permitable queue size is bigger and also the ability to clear work away from the schedule is faster.

Network Speed this relates to how quickly data may be transferred and hence the ability to clear and bill work from the schedule. The faster this can happen the greater the possibility for price fluctuations. In been able to clear work away faster this changes how fast work may be accepted.

The Schedule's Quality This is variation upon the comment of machine availability, speed and network speed. It however relates to the schedule generated. If the schedule utilises the resources more efficiently then it is more likely to be able to clear work from the schedule and cause billing events to occur, which in turn changes the price.

6.6 Counteracting Price Instability

In Sections 6.4.1, 6.4.2 and 6.4.4 it was seen that before the budget constraints became dominant that selection pressures did not favour jobs based upon the budget available. If a Grid become overworked then the economic selection pressures are required. This means the economic factors should be made to be dominant by changing the resource/service price.

It was also seen in Section 6.4.2 how price stability affected the ability of the mechanism to maintain this selection pressure. This was then further discussed in Section 6.4.3 and assessed further in Section 6.4.4. The various ways that might be used to obtain a more stable price are therefore discussed and the possible options are hence listed below:

Step Change From a Previous Price The mechanism by which the price is selected may be changed to make an incremental step change from the previous price. This would hence avoid some of the fluctuations and allow for rescheduling. The moderated changes however would have to reflect the completion of spikes in load. Spikes in load can occur for example just before an important conference [95]. The price smoothing mechanism would be required to remain responsive and should not for example artificially maintain a high price at the end of a peak in demand. This is because the system as a whole could either have unrealized profit or utility [118, 130].

Do not Rely upon Completed Work for Pricing The is by far the simplest solution, by simply ensuring that pricing does not rely upon already completed work for determining the resource cost. This means any billing event will not affect the price, by removing jobs from the schedule that are used as part of the measure of current load. An example of such a measure would be to take the difference between the current time and the average completion time of all jobs for the provider.

Maintain an Even Billing Distribution An alternative though needlessly complicated option would be to ensure the billing events were purposefully held apart. In holding these events further apart it would reduce the number of occurrences, where a lot of work is removed from the schedule at the same time. This would require a scheduling algorithm that was geared specifically towards the pricing mechanism. If individual tasks were not allowed to interweave, then this would help prevent the near simultaneous execution of billing actions. However, if a single very large job completes then regardless of dispersion of billing events the price would fall and potentially harm the selection process.

6.7 Time and Cost Constraints Compared

Following the experimentation in this section it is possible to compare time and cost constraints and show how they are related within the pricing model that the experiment tests.

The budget constrains across all jobs evenly based upon the current system load, regardless of if they can be allocated in a position that is far into the future. This is in difference to due date and deadline which will not constrain based on load if the current load is high and the due date/deadline is sufficiently far in the future. Essentially the closer the temporal parameters are to the current time then the lower the available slack in scheduling and hence the more likely it is that they respond to the current system load. The budget as a selection pressure also adapts based upon the mark-up, current system load and the expected resource usage as indicated by the schedule covering work in the future. Thus if more work is required to be performed then it is more expensive and hence more likely to act as a selection pressure.

The due date and deadline acts as a selection pressure based upon the current load and amount of work to be performed, but does not directly take a mark-up as a modifier. The mark-up however changes the breakeven point which is between the due date and deadline, thus the temporal pressures can be adjusted as well. This effect is not however seen with other [17,52,54,99,148] related gradient decent based models and is due to the removal of the direct relationship between the available budget and the price paid, causing the introduction of a mark-up for the broker, e.g. in LibraSLA [52] the budget is the price paid unless the soft deadline is passed.

Temporal constraints within the ISQoS system and other related models are also likely to cause a sequence of loading and unloading as shown in the previous chapter in Figure 5.4. This is caused by the pressure of such a time constraint becoming more effective under heavy load and then waning after it has been effective in preventing jobs from entering the market and can be considered symptomatic of deadline based approaches. The budget constraint is however more complex in this regard and can be used in this situation to preferentially select jobs with different budgets, job sizes and mark-ups.

The budget in the experimentation presented acts like a spot price (see Section 2.5.3.2 for a definition of futures and spot pricing). The price is determined from the immediate load profile. If the due date is far in advance a futures price might be more applicable, as it no longer reflects the current load. This means the price in the future could be different to a more current price and be based upon the estimated future demand!

The ISQoS negotiation mechanism acts as a tender contract market. An auction and batch based system seems incompatible with the idea of rescheduling to meet time constraints. Though auctions do still have attractive properties in terms of competition, which is something ISQoS finds more difficult. This discussion is summarised in Table 6.3.

Table 6.3: Cost and Time Constraints Compared

Concept	**Due Date/Deadline**	**Budget**
Reflects current load	Yes: unless constraints are far in the future.	Yes: unless budget is very high. Requires price to follow demand.
Influencing factors	None: effects gradient in due date to deadline period, though the breakeven point moves.	Mark-up: Makes the job more favourable to the broker, but consumes the budget quicker.
Constraining upon reflects the job's expected resource usage	Partially: Only when due date and deadline are close to the current time relative to the amount of work required to be performed. Slack however reflects usage.	Yes: Cost increases with more time upon resource that is requested.
Creates availability window (Temporally)	Yes: By Definition.	Yes: If price increases due to load then the budget creates a notion of maximum acceptable load. The work already present removes space earlier on in the schedule.
Creates availability window (Fiscally)	Yes: A breakeven point is formed, between the due date and deadline.	Yes: By Definition.

6.8 Summary

This chapter has focussed upon job prioritisation (see Section 6.4.1) and demonstrated how economic constraints can establish this, whilst also showing an initial period where temporal constraints dominate. Given that users will be aiming to set the budget assigned to jobs as low as possible it can be reasoned that the budget constraints are likely to always have the desired job prioritising effect.

Rescheduling was then introduced in Section 6.4.2 and showed how it can lead to price instability, by causing the schedule to have several jobs cleared from it, due to completion, within close succession. The reasons for this were then discussed in Section 6.4.3. This instability was then tackled in Section 6.4.4, leading to the recommendations made in Section 6.6). The primary recommendation was to break the link between the resource cost and work that would be removed during billing. An additional key recommendation was to ensure that prices are made as an incremental step change, that is not too large from the previous price.

Chapter 7

Conclusion

7.1 Summary

The work presented in this thesis has developed an economic model for the execution of Grid jobs with particular focus upon completion time and upon the cost of execution. This model has then been implemented in order to realise the models usage and to establish a platform by which Grid scheduling research from an economic perspective may be carried out. The developed platform focuses upon utilising scheduling to guide negotiations, between a broker and a tender market for job submissions. The main purpose of the negotiation being to ensure that quality of service is met. This is achieved through steering jobs to the correct provider and by using a market that ensures trade-offs between price and completion time can be realised, while keeping the user informed of the current state of the Grid.

Chapter 2: begins this with introducing the Grid and demonstrating current research trends towards quality of service provision in Grids. Quality of service provision is then discussed in greater detail, focusing first upon the difficulties in Grids for provisioning QoS and then on to mechanisms used to provide QoS. The discussion then moves onto Service level agreements, which is a mechanism that can be used to define the levels of quality of service required and modern standards such as WS-Agreement are discussed in detail. Economics in Grids is then introduced which is a mechanism for quantifying QoS in monetary terms as well as proving compensation given failings in QoS. It also offers opportunities to commercialise and expand the use of the Grid. This is finally followed by a discussion of scheduling in Grids that allows for the co-ordination of actions on the Grid as well as a means to guide the negotiation process.

Chapter 3: is aimed at introducing the market model for the Grid, that drives the processes within the proposed negotiation system for QoS. The aims of the model are first discussed along with existing literature that has a similar deadline constrained nature. The model is then introduced defining a clear graduated boundary for the quality of service constraints of both completion time and cost. Discrete event simulation is introduced as a means of testing the model. In order to carry out the simulation of the model a simulator was developed. This simulator is shown to be tested with a sweeping study of possible parameters. After testing the simulator is utilised to perform discrete event simulation on the market. The model is finally compared in a study with existing literature.

Chapter 4: introduces the broker that implements the proposed economic model. This chapter starts with a review of the literature with a selection of brokers and middleware that are available. This review is performed showing the trend towards quality of service provision in Grids, to which the proposed negotiation system is aimed. This review is then followed by a requirements analysis phase, based upon the aims of the research. The overall architecture is introduced with the focus being placed upon the agreement structure. Discussion progresses on to the internal structure of the providers which, allows for the negotiation to occur, followed by the scheduling algorithms that drive both the negotiation and resource allocation processes. This is followed by the pricing mechanisms and offer ranking and filtering mechanisms that drive the decision making process within the broker and provider. Finally Chapter 4 concludes with a feature comparison study that compares the work presented in the thesis with existing work from the literature.

Chapter 5: focuses upon admission control within Grids, ensuring that only the work that can be performed within the constraints is accepted. Various high level approaches to job admission are tested from current existing methods such as examining current workload, to naive strategies such s flooding the Grid with jobs, to more selective strategies including ones that take advantage of the proposed economic model. The admission of jobs is also shown to be cyclical in nature due to the presence of the deadline constraints, causing a sequence of loading and unloading of the Grid to occur.

Chapter 6: focuses upon the effects of the budget and temporal constraints within a dynamic pricing model. Firstly it demonstrates a transition between the temporal constraints being dominant to the economic constraints becoming the overriding factor as the temporal constraints are relaxed. The market then being driven by economic constraints demonstrates a preference towards jobs of a higher priority based upon the mark-up assigned to the job. An often used technique of rescheduling is introduced, which is shown to have effects upon the market, leading to price instability. The root causes for this are discussed and then countered by changing the pricing mechanism within the market. Finally the constraints of time and cost are compared and the overlap in their nature is discussed.

7.2 Research Contributions

The core contributions that have been demonstrated in this thesis are the following:

- An architecture has been presented that allows for the negotiation of quality of service within the Grid and in particular on the completion time and cost. This is established as a tender market by which service providers may bid for jobs to perform.
- An economic model has been established that is the central part of the negotiation process. It binds the price to the end user with the quality of service provided by the service provider. It achieves this by establishing clear boundaries around the expected quality of service, in terms of both completion time and cost. It utilises SLAs to both incentivise the provision of quality of service and to mediate levels of provision by informing the users of the current state of the Grid.
- A simulation study was conducted to test the proposed pricing mechanism. A simulator was developed to enable repeatable experiments that test the viability of the brokering mechanism, ensuring that it maintained budget balance and remains incentive compatible [155].
- Finally recommendations on how to establish an economic market for Grids is provided. Covering multiple areas such as:

  Job selection strategies: such as the ISQoS hybrid offer filter, that utilises the economic model presented.

Negotiation as advice: how to use the negotiation process to advise users of the current state of the Grid and how it can be more efficient than current load based approaches.

economic model utilisation: such as the relationship between slack and job acceptance in the Grid and how to ensure the breakeven point is highly describable and predictable, regardless of resource cost.

The similarities and differences of the budget and temporal constraints: showing how they can act in a very similar fashion upon the Grid and discussion upon how the budget can limit the maximum amount of work permitted on the Grid at anyone time.

Pricing mechanisms: how they can be utilised to introduce stability in the pricing and can be made to ignore events that might cause significant fluctuations in price.

7.3 Future Work

The future work can be extended in several directions the two obvious main avenues are towards Cloud computing and further depth in the economics used to control the compute market.

7.3.1 Cloud Computing Deployment

There are several avenues for future work, one of the key areas is upon the broker which could be advanced along the follow lines:

Cloud computing has become an important paradigm and has overshadowed Grids. The ISQoS system could be adapted to work in a Cloud infrastructure. The main part where changes would be necessary would be the information provider. If it was able to create new VM instances and then declare their presence to the provider's scheduler that it belongs then, it would work in a cloud context by being able to dynamically scale, while avoiding the need for the instant provisioning of VMs. There is always a delay in the provisioning of new VMs so the emphasis would be on not trying to instantly provision and let scheduling remove much of the need to do so. Due to the large scale of computing resources involved in Grids or Clouds it would more naturally be able to balance the type of VM instances needed. The scheduler would then not need to know about VM starting and finishing and the commissioning and decommissioning of VM instances would be masked by what information was presented by the information provider to the scheduler.

The work could therefore be likened to Platform as a Service (PaaS), in that it provides an execution platform by which it is possible to pass an executable and its description for execution without further knowledge. Alternatively if the VMs are setup for specific software and only job descriptions and data are provided, a Software as a Service (SaaS) model could be realised.

Tasks that remain running at the end of advance reservations, are currently required to be terminated and the user must accept the time on the resources they bought is up. This system can be likened to a parking meter system i.e. if a user is over the limit then an instant penalty is received. The data losses incurred are however undesirable, so the state of the process execution could be saved ready for renegotiation. The provider/broker could then ask the user how much they are willing to pay to complete the work, while asking for a fresh estimate on completion time. The alternative to this is to either reschedule and charge accordingly or let the jobs overrun by a given amount and to take this into account when scheduling.

7.3.2 Economics

The research associated with the work may also be extended by following up in the areas of:

Working further on price stability: Mechanisms could be generated to better follow the current demand for resources. This would ensure the price was set closer to the optimum point. They would have to comprehend how much prices should change, the nature of the work being performed, in relation to the resources available.

Adding more rescheduling and adaptive capacity: Currently rescheduling may be performed either at intervals (which may be changed as desired) or at events such as a new job arriving. It does not however reschedule at the time when an error occurs, such as machine failure. This additional adaptive capacity could be tested and shown to be an improvement over the existing system. The trade-off between status signals would have to be compared with the cost of not having such an adaptive mechanism.

Extending work into scheduling algorithms specifically for the ISQoS system: The capacity to reschedule exists but this is underutilized by the algorithms that have been implemented within the ISQoS System. Further scheduling algorithms could be developed to that periodically reschedule, as opposed to the round robin rescheduling algorithm currently implemented that reschedules only at submission time.

Ensuring the mark-up becomes more associated with risk to the broker: New Pricing policies could be developed that would focus on areas such as risk management. The broker in such a scenario would be able to modify the percentage mark-up it takes, hence ensuring the broker selects the provider most likely to get the job done on time. The mark-up could therefore be used to penalise the providers that are not performing well. The experimentation would have to derive a mechanism that would separate unreliable providers from reliable ones.

Bibliography

[1] Adami, D., Giordano, S., Pagano, M.: Dynamic network resources allocation in grids through a grid network resource broker. In: Davoli, F., Meyer, N., Pugliese, R., Zappatore, S. (eds.) Grid Enabled Remote Instrumentation, pp. 115–130. Springer US (2009)

[2] Adami, D., Giordano, S., Repeti, M., Coppola, M., Laforenzu, D., Tonellotlo, N.: Design and implementation of a grid network-aware resource broker. pp. 41–6. ACTA Press, Anaheim, CA, USA (2006)

[3] Adamson, W., Kornievskaia, O.: A practical distributed authorization system for gara. In: Infrastructure Security, pp. 314–324. Cancun, Mexico (2002)

[4] Agbaria, A., Friedman, R.: Model-based performance evaluation of distributed checkpointing protocols. Performance Evaluation 65(5), 345–365 (2008)

[5] Al-Ali, R., Rana, O., Walker, D.: G-qosm a framework for quality of service management. In: Cox, S. (ed.) UK e-Science Programme All Hands Meeting 2003. Cardiff University Wales, Nottingham, UK (2003)

[6] Al-Ali, R., Sohail, S., Rana, O., Hafid, A., Von Laszewski, G., Amin, K., Jha, S., Walker, D.: Network qos provision for distributed grid applications. International Journal of Simulation: Systems, Science & Technology 5(5), 13–28 (2004)

[7] Ali, S., Siegel, H.J., Maheswaran, M., Hensgen, D.: Task execution time modeling for heterogeneous computing systems. In: 9th Heterogeneous Computing Workshop (HCW 2000). pp. 185–199 (2000)

[8] Altmann, J., Courcoubetis, C., Darlington, J., Cohen, J.: Gridecon the economic-enhanced next-generation internet. In: Grid Economics and Business Models, pp. 188–193 (2007)

[9] Altmann, J., Neumann, D., Fahringer, T., Courcoubetis, C., Stamoulis, G., Dramitinos, M., Rayna, T., Risch, M., Bannink, C.: Gridecon: A market place for computing resources. In: Grid Economics and Business Models, vol. 5206, pp. 185–196. Springer Berlin / Heidelberg (2008)

[10] Andreetto, P., Andreozzi, S., Ghiselli, A., Marzolla, M., Venturi, V., Zangrando, L.: Standards-based job management in grid systems. Journal of Grid Computing 8(1), 19–45 (2010)

[11] Anglano, C., Brevik, J., Canonico, M., Nurmi, D., Wolski, R.: Fault-aware scheduling for bag-of-tasks applications on desktop grids. In: Grid Computing, 7th IEEE/ACM International Conference on. pp. 56–63 (2006)

[12] Anglano, C., Canonico, M.: Scheduling algorithms for multiple bag-of-task applications on desktop grids: A knowledge-free approach. In: Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on. pp. 1–8 (2008)

[13] Anglano, C., Canonico, M.: Fault-tolerant scheduling for bag-of-tasks grid applications. In: Advances in Grid Computing - EGC 2005, pp. 630–639. No. Lecture Notes in Computer Science (2005)

[14] Appleton, O., Cameron, D., Cernak, J., Db, P., Ellert, M., Frgt, T., Grnager, M., Johansson, D., Jnemo, J., Kleist, J., Koan, M., Konstantinov, A., Knya, B., Mrton, I., Mohn, B., Mller, S., Mller, H., Nagy, Z., Nilsen, J., Ould Saada, F., Pajchel, K., Qiang, W., Read, A., Rosendahl, P., Rczei, G., Savko, M., Skou Andersen, M., Smirnova, O., Stefn, P., Szalai, F., Taga, A., Toor, S., Wnnen, A., Zhou, X.: The next-generation arc middleware. Annals of Telecommunications pp. 1–6 (2010)

[15] Ardaiz, O., Freitag, F., Navarro, L., Eymann, T., Reinicke, M.: Catnet: Catallactic mechanisms for service control and resource allocation in large-scale application-layer networks. In: Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on. pp. 442–442 (2002)

[16] Ardaiz, O., Artigas, P., Eymann, T., Freitag, F., Navarro, L., Reinicke, M.: The catallaxy approach for decentralized economic-based allocation in grid resource and service markets. Applied Intelligence 25(2), 131–145 (2006)

[17] AuYoung, A., Grit, L., Wiener, J., Wilkes, J.: Service contracts and aggregate utility functions. In: 15th IEEE International Symposium on High Performance Distributed Computing (HPDC-15). IEEE, New York (2005)

[18] Battre, D., Hovestadt, M., Kao, O., Keller, A., Voss, K.: Planning-based scheduling for sla-awareness and grid integration. In: Bartk, R. (ed.) PlanSIG 2007 The 26th workshop of the UK Planning and Scheduling Special Interest Group. vol. 1, p. 8. Prague, Czech Republic (2007)

[19] Battre, D., Hovestadt, M., Kao, O., Keller, A., Voss, K.: Implementation of virtual execution environments for improving sla-compliant job migration in grids. In: Parallel Processing - Workshops, 2008. ICPP-W '08. International Conference on. pp. 47–52 (2008)

[20] Battre, D., Hovestadt, M., Kao, O., Keller, A., Voss, K.: Virtual execution environments for ensuring sla-compliant job migration in grids. In: IEEE International Conference on Services Computing SCC '08. vol. 2, pp. 571–572 (2008)

[21] Battre, D., Kao, O., Voss, K.: Implementing ws-agreement in a globus toolkit 4.0 environment. In: Talia, D., Yahyapour, R., Ziegler, W. (eds.) Grid Middleware and Services, pp. 409–418. Springer US (2008)

[22] Beaumont, O., Carter, L., Ferrante, J., Legrand, A., Marchal, L., Robert, Y.: Centralized versus distributed schedulers for bag-of-tasks applications. Parallel and Distributed Systems, IEEE Transactions on 19(5), 698–709 (2008)

[23] Becker, M., Borrisov, N., Deora, V., Rana, O.F., Neumann, D.: Using k-pricing for penalty calculation in grid market. In: Hawaii International Conference on System Sciences, Proceedings of the 41st Annual. pp. 97–97 (2008)

[24] BEinGRID: Beingrid, business experiments in grid (2010), `http://www.beingrid.eu/`

[25] Bell, W.H., Cameron, D.G., Millar, A.P., Capozza, L., Stockinger, K., Zini, F.: Optorsim: A grid simulator for studying dynamic data replication strategies. International Journal of High Performance Computing Applications 17(4), 403–416 (2003)

[26] Bharadwaj, V., Ghose, D., Robertazzi, T.G.: Divisible load theory: A new paradigm for load scheduling in distributed systems. Cluster Computing 6(1), 7–17 (2003)

[27] Bhatti, S.N., Sorensen, S.A., Clark, P., Crowcroft, J.: Network qos for grid systems. International Journal of High Performance Computing Applications 17(3), 219–236 (2003)

[28] Biette, M., Voss, K., Padgett, J., Gourlay, I., Djemame, K., Fally, B., Ponsard, C., Mouton, S., Stmke, J., Ttard, F.: Preliminary exploitation plan version 1.0 (2006)

[29] Bittencourt, L., Madeira, E.: Towards the scheduling of multiple workflows on computational grids. Journal of Grid Computing (2009)

[30] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: An Architecture for Differentiated Service. RFC Editor (1998)

[31] Bode, B., Halstead, D., Kendall, R., Lei, Z., Hall, W., Jackson, D.: The portable batch scheduler and the maui scheduler on linux clusters. In: Proceedings Of The 4th Annual Linux Showcase And Conference. pp. 217–224. Usenix Association, Atlanta (2000)

[32] Bolze, R., Cappello, F., Caron, E., Dayd, M., Desprez, F., Jeannot, E., Jgou, Y., Lanteri, S., Leduc, J., Melab, N., Mornet, G., Namyst, R., Primet, P., Quetier, B., Richard, O., Talbi, E.G., Touche, I.: Grid'5000: A large scale and highly reconfigurable experimental grid testbed. International Journal of High Performance Computing Applications 20(4), 481–494 (2006)

[33] Bouguerra, M., Gautier, T., Trystram, D., Vincent, J.: A flexible checkpoint-restart model in distributed systems. In: PMAA 2009, the 8th International Conference on Parallel Processing and Applied Mathematics. IEEE, Wroclaw, Poland (2009)

[34] Broberg, J., Venugopal, S., Buyya, R.: Market-oriented grids and utility computing: The state-of-the-art and future directions. Journal of Grid Computing 6(3), 255–276 (2008)

[35] Bubak, M., van Albada, G., Dongarra, J., Sloot, P., Vanmechelen, K., Depoorter, W., Broeckhove, J.: A simulation framework for studying economic resource management in grids. In: Computational Science ICCS 2008, vol. 5101, pp. 226–235. Springer Berlin / Heidelberg (2008)

[36] Burke, S., Andreozzi, S., Field, L.: Experiences with the glue information schema in the lcg/egee production grid. Journal of Physics: Conference Series 119(6, 062019) (2008)

[37] Buyya, R., Abramson, D., Venugopal, S.: The grid economy. Proceedings of the IEEE 93(3), 698–714 (2005)

[38] Buyya, R., Venugopal, S.: The gridbus toolkit for service oriented grid and utility computing: an overview and status report. In: Grid Economics and Business Models, 2004. GECON 2004. 1st IEEE International Workshop on. pp. 19–66 (2004)

[39] Buyya, R.: Economic-based Distributed Resource Management and Scheduling for Grid Computing. Ph.D. thesis, Monash University, Melbourne, Australia (2002)

[40] Buyya, R., Abramson, D., Giddy, J., Stockinger, H.: Economic models for resource management and scheduling in grid computing. Concurrency and Computation: Practice and Experience 14(13-15), 1507–1542 (2002)

[41] Buyya, R., Giddy, J., Abramson, D.: An evaluation of economy-based resource trading and scheduling on computational power grids for parameter sweep applications. In: The Second Workshop on Active Middleware Services (AMS 2000), In conjunction with Ninth IEEE International Symposium on High Performance Distributed Computing (HPDC 2000). Kluwer Academic Press, Pittsburgh, USA (2000)

[42] Buyya, R., Murshed, M.: Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. Concurrency and Computation: Practice and Experience 14(13-15), 1175–1220 (2002)

[43] Buyya, R., Murshed, M., Abramson, D., Venugopal, S.: Scheduling parameter sweep applications on global grids: a deadline and budget constrained cost-time optimization algorithm. Software: Practice and Experience 35(5), 491–512 (2005)

[44] Caminero, A., Carrin, C., Caminero, B.: Designing an entity to provide network qos in a grid system. In: 1st Iberian Grid Infrastructure Conference (IberGrid). Santiago de Compostela, Spain (2007)

[45] Caminero, A., Rana, O., Caminero, B., Carrin, C.: Performance evaluation of an autonomic network-aware metascheduler for grids. Concurrency and Computation: Practice and Experience 21(13), 1692–1708 (2009)

[46] Cao, J., Zimmermann, F.: Queue scheduling and advance reservations with cosy. In: Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International. p. 63 (2004)

[47] Cardenas, C., Gagnaire, M.: Evaluation of flow-aware networking (fan) architectures under gridftp traffic. Future Generation Computer Systems 25(8), 895–903 (2009)

[48] Cardenas, C., Gagnaire, M., Lopez, V., Aracil, J.: Admission control in flow-aware networking (fan) architectures under gridftp traffic. Optical Switching and Networking 6(1), 20–28 (2009)

[49] Casanova, H., Legrand, A., Zagorodnov, D., Berman, F.: Heuristics for scheduling parameter sweep applications in grid environments. In: Heterogeneous Computing Workshop, 2000. (HCW 2000) Proceedings. 9th. pp. 349–363 (2000)

[50] Castillo, C., Rouskas, G.N., Harfoush, K.: On the design of online scheduling algorithms for advance reservations and qos in grids. In: Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International. pp. 1–10 (2007)

[51] Castillo, C., Rouskas, G.N., Harfoush, K.: Online algorithms for advance resource reservations. Journal of Parallel and Distributed Computing In Press, Corrected Proof (2011)

[52] Chee Shin, Y., Buyya, R.: Service level agreement based allocation of cluster resources: Handling penalty to enhance utility. In: Cluster Computing, 2005. IEEE International. pp. 1–10 (2005)

[53] Chronz, P., Wieder, P.: Integrating ws-agreement with a framework for service-oriented infrastructures. In: Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on. pp. 225–232 (2010)

[54] Chun, B.N., Culler, D.E.: User-centric performance analysis of market-based cluster batch schedulers. In: Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on. pp. 30–30 (2002)

[55] Citrix Systems: Home of the xen hypervisor (2012), `http://www.xen.org/`

[56] Cohen, J., Harder, U., Martinez Ortuno, F., Richardson, C., Darlington, J.: Node-level architecture design and simulation of the magog grid middleware. Conferences in Research and Practice in Information Technology Series 99, 57–67 (2009)

[57] De Roure, D., Baker, M.A., Jennings, N.R., Shadbolt, N.R.: The evolution of the grid. In: Berman, F., Fox, G., Hey, T. (eds.) Grid Computing: Making the Global Infrastructure a Reality, pp. 65–100. Wiley, Chichester (2003)

[58] Deelman, E., Gannon, D., Shields, M., Taylor, I.: Workflows and e-science: An overview of workflow system features and capabilities. Future Generation Computer Systems 25(5), 528–540 (2009)

[59] DeFanti, T.A., Foster, I., Papka, M.E., Stevens, R., Kuhfuss, T.: Overview of the i-way: Wide-area visual supercomputing. International Journal of High Performance Computing Applications 10(2-3), 123–131 (1996)

[60] Distributed Systems Architecture Group: Documentation (2009), `http://www.gridway.org/doku.php?id=documentation`

[61] Dumitrescu, C.L., Foster, I.: Gangsim: a simulator for grid scheduling studies. In: Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on. vol. 2, pp. 1151–1158 Vol. 2 (2005)

[62] Eerola, P., Ekelof, T., Ellert, M., Hansen, J.R., Konstantinov, A., Konya, B., Nielsen, J.L., Ould-Saada, F., Smirnova, O., Waananen, A.: The nordugrid architecture and tools (2003)

[63] EGEE: glite (2009), `http://glite.web.cern.ch/glite/`

[64] EGI: European grid initiative - towards a sustainable production grid infrastructure (2010), `http://www.egi.eu/`

[65] Ejarque, J., de Palol, M., Goiri, i., Juli, F., Guitart, J., Badia, R.M., Torres, J.: Exploiting semantics and virtualization for sla-driven resource allocation in service providers. Concurrency and Computation: Practice and Experience 22(5), 541–572 (2010), `http://dx.doi.org/10.1002/cpe.1468`

[66] Elmroth, E., Tordsson, J.: An interoperable, standards-based grid resource broker and job submission service. In: e-Science and Grid Computing, 2005. First International Conference on. pp. 9 pp.–220 (2005)

[67] Elmroth, E., Tordsson, J.: A standards-based grid resource brokering service supporting advance reservations, coallocation, and cross-grid interoperability. Concurrency and Computation: Practice and Experience 21(18), 2298–2335 (2009)

[68] EMI: European middleware initiative (2010), `https://www.eu-emi.eu/`

[69] Eymann, T., Reinicke, M., Streitberger, W., Rana, O., Joita, L., Neumann, D., Schnizler, B., Veit, D., Ardaiz, O., Chacin, P., Chao, I., Freitag, F., Navarro, L., Catalano, M., Gallegati, M., Giulioni, G., Schiaffino, R.C., Zini, F.: Catallaxy-based grid markets. Multiagent Grid Syst. 1(4), 297–307 (2005)

[70] Eymann, T., Streitberger, W., Hudert, S.: Catnets - open market approaches for self-organizing grid resource allocation. In: Proceedings of the 4th international conference on Grid economics and business models. Springer-Verlag, Rennes, France (2007)

[71] Farooq, U., Majumdar, S., Parsons, E.W.: A framework to achieve guaranteed qos for applications and high system performance in multi-institutional grid computing. In: Parallel Processing, 2006. ICPP 2006. International Conference on. pp. 373–380 (2006)

[72] Farooq, U., Majumdar, S., Parsons, E.W.: Achieving efficiency, quality of service and robustness in multi-organizational grids. Journal of Systems and Software 82(1), 23–38 (2009)

[73] Foster, I.: A globus primer or, everything you wanted to know about globus, but were afraid to ask describing globus toolkit version 4 (2005)

[74] Foster, I.: Globus toolkit version 4: Software for service-oriented systems. Journal of Computer Science and Technology 21(4), 513–520 (2006)

[75] Foster, I., Fidler, M., Roy, A., Sander, V., Winkler, L.: End-to-end quality of service for high-end applications. Computer Communications 27(14), 1375–1388 (2004)

[76] Foster, I.: What is the grid? a three point checklist. Grid Today 1(6), 22–25 (2002)

[77] Foster, I., Kesselman, C.: Globus: a metacomputing infrastructure toolkit. International Journal of High Performance Computing Applications 11(2), 115–128 (1997)

[78] Foster, I., Kesselman, C., Nick, J.M., Tuecke, S.: The physiology of the grid. In: Berman, F., Fox, G., Hey, T. (eds.) Grid Computing: Making the Global Infrastructure a Reality, pp. 217–249. Wiley, Chichester (2003)

[79] Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: Enabling scalable virtual organizations. In: Berman, F., Fox, G., Hey, T. (eds.) Grid Computing: Making the Global Infrastructure a Reality, pp. 169–197. Wiley, Chichester (2003)

[80] Fox, G., Ko, S.H., Pierce, M., Balsoy, O., Kim, J., Lee, S., Kim, K., Oh, S., Rao, X., Varank, M., Bulut, H., Gunduz, G., Qiu, X., Pallickara, S., Uyar, A., Youn, C.: Grid services for earthquake science. Concurrency and Computation: Practice and Experience 14(6-7), 371–393 (2002)

[81] Ganglia Project: Ganglia monitoring system (2012), `http://ganglia.sourceforge.net/`

[82] Garg, S.K., Venugopal, S., Buyya, R.: A meta-scheduler with auction based resource allocation for global grids. In: Parallel and Distributed Systems, 2008. ICPADS '08. 14th IEEE International Conference on. pp. 187–194 (2008)

[83] Gaweda, I., Wilk, C.: Grid brokers and metaschedulers market overview (2006)

[84] Grosu, D., Das, A.: Auctioning resources in grids: model and protocols. Concurrency and Computation: Practice and Experience 18(15), 1909–1927 (2006)

[85] Han, Y., Youn, C.H.: A new grid resource management mechanism with resource-aware policy administrator for sla-constrained applications. Future Generation Computer Systems 25(7), 768–778 (2009)

[86] Haque, A., Alhashmi, S.M., Parthiban, R.: A survey of economic models in grid computing. Future Generation Computer Systems 27(8), 1056–1069 (2011)

[87] Harchol-Balter, M.: Performance Modeling and Design of Computer Systems: Queueing Theory in Action. Cambridge University Press (2013)

[88] Hardin, G.: The tragedy of the commons. Science 162(3859), 1243–1248 (1968)

[89] Hong-Linh, T., Samborski, R., Fahringer, T.: Towards a framework for monitoring and analyzing qos metrics of grid services. In: e-Science and Grid Computing, 2006. e-Science '06. Second IEEE International Conference on. pp. 65–65 (2006)

[90] Hovestadt, M.: Fault tolerance mechanisms for sla-aware resource management. In: Parallel and Distributed Systems, 2005. Proceedings. 11th International Conference on. vol. 2, pp. 458–462 (2005)

[91] Hudert, S., Ludwig, H., Wirtz, G.: Negotiating slas-an approach for a generic negotiation framework for ws-agreement. Journal of Grid Computing 7(2), 225–246 (2009)

[92] Huedo, E., Montero, R.S., Llorente, I.A.: A modular meta-scheduling architecture for interfacing with pre-ws and ws grid resource management services. Future Generation Computer Systems 23(2), 252–261 (2007), futur. Gener. Comp. Syst.

[93] Huedo, E., Montero, R.S., Llorente, I.M.: A framework for adaptive execution in grids. Software: Practice and Experience 34(7), 631–651 (2004)

[94] INRIA: Welcome to the simgrid project! (2010), `http://simgrid.gforge.inria.fr/`

[95] Iosup, A., Epema, D.: Grid computing workloads. Internet Computing, IEEE 15(2), 19–26 (2011)

[96] Iosup, A., Jan, M., Sonmez, O., Epema, D.: The characteristics and performance of groups of jobs in grids. In: Euro-Par 2007, Parallel Processing, pp. 382–393 (2007)

[97] Iosup, A., Li, H., Jan, M., Anoep, S., Dumitrescu, C., Wolters, L., Epema, D.H.J.: The grid workloads archive. Future Generation Computer Systems 24(7), 672–686 (2008)

[98] Iosup, A., Sonmez, O., Anoep, S., Epema, D.: The performance of bags-of-tasks in large-scale distributed systems. In: Proceedings of the 17th international symposium on High performance distributed computing. ACM, Boston, MA, USA (2008)

[99] Irwin, D.E., Grit, L.E., Chase, J.S.: Balancing risk and reward in a market-based task service. In: High performance Distributed Computing, 2004. Proceedings. 13th IEEE International Symposium on. pp. 160–169 (2004)

[100] Jackson, D., Snell, Q., Clement, M.: Core algorithms of the maui scheduler. In: Job Scheduling Strategies for Parallel Processing, pp. 87–102 (2001)

[101] Jarvis, S.A., Thomas, N., van Moorsel, A.: Open issues in grid performability. International Journal of Simulation and Process Modelling (IJSPM) 5(5), 3–12 (2004)

[102] Jeeho, S., Robertazzi, T.G., Luryi, S.: Optimizing computing costs using divisible load analysis. Parallel and Distributed Systems, IEEE Transactions on 9(3), 225–234 (1998)

[103] Kavanagh, R., Djemame, K.: A grid broker pricing mechanism for temporal and budget guarantees. In: Thomas, N. (ed.) 8th European Performance Engineering Workshop (EPEW'2011), LNCS. vol. 6977. Springer, Borrowdale, The Lake District, UK (2011)

[104] Kearney, K.T., Torelli, F., Kotsokalis, C.: Sla*: An abstract syntax for service level agreements. In: Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on. pp. 217–224 (2010)

[105] Keller, A., Ludwig, H.: The wsla framework: Specifying and monitoring service level agreements for web services. Journal of Network and Systems Management 11(1), 57–81 (2003)

[106] Kenyon, C., Cheliotis, G.: Architecture requirements for commercializing grid resources. In: High Performance Distributed Computing, 2002. HPDC-11 2002. Proceedings. 11th IEEE International Symposium on. pp. 215–224 (2002)

[107] Kertsz, A., Kacsuk, P.: A taxonomy of grid resource brokers. In: Distributed and Parallel Systems, pp. 201–210 (2007)

[108] Kesselman, C., Foster, I.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers (1998)

[109] Kleinrock, L.: Queueing Systems, Volume 1, Theory, vol. 1. Wiley, John & Sons, Incorporated (1975)

[110] KnowARC: D3.1-1 interoperability minimal service survey (2007), `http://www.knowarc.eu/documents/Knowarc_D3.1-1_07.pdf`

[111] Kokkinos, P., Varvarigos, E.A.: A framework for providing hard delay guarantees and user fairness in grid computing. Future Generation Computer Systems 25(6), 674–686 (2009)

[112] Korpela, E., Werthimer, D., Anderson, D., Cobb, J., Leboisky, M.: Seti@home-massively distributed computing for seti. Computing in Science & Engineering 3(1), 78–83 (2001)

[113] Kshemkalyani, A.: Fast and message-efficient global snapshot algorithms for large-scale distributed systems. Parallel and Distributed Systems, IEEE Transactions on PP(99), 1–1 (2010)

[114] Kubert, R., Hai-Lang, T., Axel, T.: A soap performance comparison of different wsrf implementations. In: Proceedings of the International Conference on Management of Emergent Digital EcoSystems. ACM, France (2009)

[115] Kumar, S., Dutta, K., Mookerjee, V.: Maximizing business value by optimal assignment of jobs to resources in grid computing. European Journal of Operational Research 194(3), 856–872 (2009)

[116] Kurowski, K., Nabrzyski, J., Oleksiak, A., Weglarz, J.: Grid scheduling simulations with gssim. In: Parallel and Distributed Systems, 2007 International Conference on. vol. 2, pp. 1–8 (2007)

[117] Kurowski, K., Nabrzyski, J., Oleksiak, A., Wglarz, J.: A multicriteria approach to two-level hierarchy scheduling in grids. Journal of Scheduling 11(5), 371–379 (2008)

[118] Lai, K.: Markets are dead, long live markets. SIGecom Exch. 5(4), 1–10 (2005)

[119] Lai, K., Rasmusson, L., Adar, E., Zhang, L., Huberman, B.A.: Tycoon: An implementation of a distributed, market-based resource allocation system. Multiagent Grid Syst. 1(3), 169–182 (2005)

[120] Lamanna, D.D., Skene, J., Emmerich, W.: Slang: a language for defining service level agreements. In: Distributed Computing Systems, 2003. FTDCS 2003. Proceedings. The Ninth IEEE Workshop on Future Trends of. pp. 100–106 (2003)

[121] Lee, Y.C., Zomaya, A.Y.: A grid scheduling algorithm for bag-of-tasks applications using multiple queues with duplication. In: Computer and Information Science, 2006 and 2006 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse. ICIS-COMSAR 2006. 5th IEEE/ACIS International Conference on. pp. 5–10 (2006)

[122] Lee, Y.C., Zomaya, A.Y.: Practical scheduling of bag-of-tasks applications on grids with dynamic resilience. Computers, IEEE Transactions on 56(6), 815–825 (2007)

[123] Lehner, W., Meyer, N., Streit, A., Stewart, C., Gruber, R., Keller, V., Thimard, M., Waldrich, O., Wieder, P., Ziegler, W., Manneback, P.: Integration of grid cost model into iss/viola meta-scheduler environment. In: Euro-Par 2006: Parallel Processing, vol. 4375, pp. 215–224. Springer Berlin / Heidelberg (2007)

[124] Liu, C., Baskiyar, S.: A general distributed scalable grid scheduler for independent tasks. Journal of Parallel and Distributed Computing 69(3), 307–314 (2009)

[125] Luque, E., Margalef, T., Bentez, D., Depoorter, W., De Moor, N., Vanmechelen, K., Broeckhove, J.: Scalability of grid simulators: An evaluation. In: Euro-Par 2008 Parallel Processing, vol. 5168, pp. 544–553. Springer Berlin / Heidelberg (2008)

[126] McGough, A.S., Lee, W., Das, S.: A standards based approach to enabling legacy applications on the grid. Future Generation Computer Systems 24(7), 731–743 (2008)

[127] Merlo, A., Clematis, A., Corana, A., Gianuzzi, V.: Quality of service on grid: architectural and methodological issues. Concurrency and Computation: Practice and Experience 23(8), 745–766 (2011)

[128] Middleton, S., Surridge, M., Benkner, S., Engelbrecht, G.: Quality of service negotiation for commercial medical grid services. Journal of Grid Computing 5(4), 429–447 (2007)

[129] Monash eScience and Grid Engineering Laboratory: The nimrod toolkit (2009), `http://messagelab.monash.edu.au/Nimrod`

[130] Neumann, D., Ster, J., Weinhardt, C., Nimis, J.: A framework for commercial grids - economic and technical challenges. Journal of Grid Computing 6(3), 325–347 (2008)

[131] Neumann, D., Stoesser, J., Anandasivam, A., Borissov, N.: Sorma building an open grid market for grid resource allocation. In: Grid Economics and Business Models, pp. 194–200 (2007)

[132] NGS: National grid service (2009), `http://www.ngs.ac.uk`

[133] Nimis, J., Anandasivam, A., Borissov, N., Smith, G., Neumann, D., Wirstrm, N., Rosenberg, E., Villa, M.: Sorma business cases for an open grid market: Concept and implementation. In: Altmann, J., Neumann, D., Fahringer, T. (eds.) Grid Economics and Business Models, vol. 5206, pp. 173–184. Springer Berlin / Heidelberg (2008)

[134] Nou, R., Kounev, S., Julia, F., Torres, J.: Autonomic qos control in enterprise grid environments using online simulation. Journal of Systems and Software 82(3), 486–502 (2009), j. Syst. Softw.

[135] Nudd, G.R., Jarvis, S.A.: Performance-based middleware for grid computing. Concurrency and Computation: Practice and Experience 17(2-4), 215–234 (2005)

[136] OASIS: Oasis web services resource framework (wsrf) tc (2010), `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf`

[137] Oey, M.A., Timmer, R.J., Mobach, D.G.A., Overeinder, B.J., Brazier, F.M.T.: Ws-agreement based resource negotiation in agentscape. In: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems. ACM, Honolulu, Hawaii (2007)

[138] Oliveros, E., Muoz, H., Cantelar, D., Taylor, S.: Brein, towards an intelligent grid for business. In: Altmann, J., Neumann, D., Fahringer, T. (eds.) Grid Economics and Business Models, vol. 5206, pp. 163–172. Springer Berlin Heidelberg (2008), `http://dx.doi.org/10.1007/978-3-540-85485-2_13`

[139] Open Grid Forum: Job submission description language (jsdl) specification, version 1.0 (2005)

[140] Open Grid Forum: The open grid services architecture, version 1.5 (2006)

[141] Open Grid Forum: Web services agreement specification (ws-agreement) (2007)

[142] Open Grid Forum: Distributed resource management application api specification 1.0 (2008)

[143] Open Grid Forum: Web services agreement negotiation specification (ws-agreement negotiation) (2010)

[144] Open Grid Forum: Ws-agreement negotiation version 1.0 (2011)

[145] OpenNebula Project: Opennebula homepage (2012), `http://opennebula.org/`

[146] Paderborn Center For Parallel Computing: Openccs user manual version 0.9.1 (2012), `https://www.openccs.eu/core/attachment/wiki/WikiStart/user.pdf`

[147] Palmer, J., Mitrani, I., Mazzucco, M., McKee, P., Fisher, M.: Optimizing revenue - service provisioning systems with qos contracts. pp. 187 – 191. Spain (2007), admission policies;M/M/N/K queue;Per units;QoS requirements;Revenue maximization;Service provisioning systems;

[148] Popovici, F.I., Wilkes, J.: Profitable services in an uncertain world. In: Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference. pp. 36–36 (2005)

[149] Priol, T., Snelling, D.: Next generation grids: European grid research 2005 - 2010 (2003), `ftp://ftp.cordis.europa.eu/pub/ist/docs/ngg_eg_final.pdf`

[150] Quetier, B., Cappello, F.: A survey of grid research tools: simulators, emulators and real life platforms. In: IMACS'2005 - the 17th IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation. Paris, France (2005)

[151] Rahman, M., Ranjan, R., Buyya, R., Benatallah, B.: A taxonomy and survey on autonomic management of applications in grid computing environments. Concurrency and Computation: Practice and Experience 23(16), 1990–2019 (2011)

[152] Rak, M., Liccardo, L., Aversa, R.: A sla-based interface for security management in cloud and grid integrations. In: Information Assurance and Security (IAS), 2011 7th International Conference on. pp. 378–383 (2011)

[153] Robertazzi, T.G.: Ten reasons to use divisible load theory. Computer 36(5), 63–68 (2003)

[154] Romberg, M.: The unicore grid infrastructure. Special Issue on Grid Computing of Scientifc Programming Journal 10, 149–157 (2002)

[155] Schnizler, B., Neumann, D., Veit, D., Weinhardt, C.: A multiattribute combinatorial exchange for trading grid resources. In: Proceedings of the 12th Research Symposium on Emerging Electronic Markets (RSEEM) (2005)

[156] Schnizler, B., Neumann, D., Veit, D., Weinhardt, C.: Trading grid services - a multi-attribute combinatorial approach. European Journal of Operational Research 187(3), 943–961 (2008)

[157] Schopf, J.M.: Ten actions when grid scheduling. In: Nabrzyski, J., Schopf, J.M., Weglarz, J. (eds.) Grid resource management: state of the art and future trends, pp. 15–23. Kluwer Academic Publishers (2004)

[158] Schopf, J.M., Raicu, I., Pearlman, L., Miller, N., Kesselman, C., Foster, I., DArcy, M.: Monitoring and discovery in a web services framework: Functionality and performance of globus toolkit mds4 (2006)

[159] Schwiegelshohn, U., Badia, R.M., Bubak, M., Danelutto, M., Dustdar, S., Gagliardi, F., Geiger, A., Hluchy, L., Kranzlmller, D., Laure, E., Priol, T., Reinefeld, A., Resch, M., Reuter, A., Rienhoff, O., Rter, T., Sloot, P., Talia, D., Ullmann, K., Yahyapour, R., von Voigt, G.: Perspectives on grid computing. Future Generation Computer Systems 26(8), 1104–1115 (2010)

[160] Seneviratne, S., Levy, D.C.: Cost profile prediction for grid computing. Concurrency and Computation: Practice and Experience 22(1), 107–142 (2009)

[161] Shakhlevich, N., Djemame, K.: Quality of service provision for grid applications via intelligent scheduling (2009)

[162] Silva, D.P.D., Cirne, W., Brasileiro, F.V., Grande, C.: Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids. In: Euro-Par 2003, Applications on Computational Grids. pp. 169–180 (2003)

[163] da Silva, F.A.B., Senger, H.: Improving scalability of bag-of-tasks applications running on master-slave platforms. Parallel Computing 35(2), 57–71 (2009)

[164] SLA@SOI Consortium: Sla@soi homepage (2011), `http://sla-at-soi.eu/`

[165] Smirnova, O., Cameron, D., Dobe, P., Ellert, M., Fragat, T., Gronager, M., Johansson, D., Jonemo, J., Kleist, J., Kocan, M., Konstantinov, A., Konya, B., Marton, I., Moller, S., Mohn, B., Nagy, Z., Nilsen, J.K., Saada, F.O., Qiang, W., Read, A., Rosendahl, P., Roczei, G., Savko, M., Andersen, M.S., Stefan, P., Szalai, F., Taga, A., Toor, S.Z., Waananen, A.: Recent arc developments: Through modularity to interoperability. Journal of Physics: Conference Series 219(6, 062027) (2010)

[166] Smirnova, O., Eerola, P., Ekelf, T., Ellert, M., Hansen, J., Konstantinov, A., Knya, B., Nielsen, J., Ould-Saada, F., Wnnen, A.: The nordugrid architecture and middleware for scientific applications. In: Computational Science ICCS 2003, pp. 665–665 (2003)

[167] Smith, W., Foster, I., Taylor, V.: Scheduling with advanced reservations. In: 14th International Parallel and Distributed Processing Symposium (IPDPS). pp. 127–132 (2000)

[168] Song, H.J., Liu, X., Jakobsen, D., Bhagwan, R., Zhang, X., Taura, K., Chien, A.: The microgrid: A scientific tool for modeling computational grids. Sci. Program. 8(3), 127–141 (2000)

[169] Spooner, D.P., Jarvis, S.A., Cao, J., Saini, S., Nudd, G.R.: Local grid scheduling techniques using performance prediction. Computers and Digital Techniques, IEE Proceedings - 150(2), 87–96 (2003)

[170] Stober, J., Neumann, D.: Greedex - a scalable clearing mechanism for utility computing. Electronic Commerce Research 8(4), 235–253 (2008)

[171] Sulistio, A., Yeo, C.S., Buyya, R.: A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools. Software: Practice and Experience 34(7), 653–673 (2004)

[172] Sun, Y.L., Perrott, R., Harmer, T.J., Cunningham, C., Wright, P., Kennedy, J., Edmonds, A., Bayon, V., Maza, J., Berginc, G., Hadalin, P.: Sla-aware resource management grids and service-oriented architectures for service level agreements. pp. 35–44. Springer US (2010), `http://dx.doi.org/10.1007/978-1-4419-7320-7_4`

[173] Takefusa, A., Matsuoka, S., Nakada, H., Aida, K., Nagashima, U.: Overview of a performance evaluation system for global computing scheduling algorithms. In: High Performance Distributed Computing, 1999. Proceedings. The Eighth International Symposium on. pp. 97–104 (1999)

[174] Talia, D., Yahyapour, R., Ziegler, W., Wieder, P., Seidel, J., Waldrich, O.: Using sla for resource management and scheduling - a survey. In: Grid Middleware and Services, pp. 335–347. Springer US (2008)

[175] Tan, W., Missier, P., Foster, I., Madduri, R., De Roure, D., Goble, C.: A comparison of using taverna and bpel in building scientific workflows: the case of cagrid. Concurrency and Computation: Practice and Experience (2009)

[176] Taylor, S., Surridge, M., Laria, G., Ritrovato, P., Schubert, L.: Business collaborations in grids: The brein architectural principals and vo model. In: Altmann, J., Buyya, R., Rana, O. (eds.) Grid Economics and Business Models, vol. 5745, pp. 171–181. Springer Berlin Heidelberg (2009), `http://dx.doi.org/10.1007/978-3-642-03864-8_14`

[177] Templeton, D.: Beginner's guide to sun grid engine 6.2 installation and configuration white paper (2009)

[178] Thain, D., Tannenbaum, T., Livny, M.: Condor and the grid. In: Berman, F., Fox, G., Hey, T. (eds.) Grid Computing: Making the Global Infrastructure a Reality, pp. 299–335 (2003)

[179] Thain, D., Tannenbaum, T., Livny, M.: Distributed computing in practice: the condor experience. Concurrency and Computation: Practice and Experience 17(2-4), 323–356 (2005)

[180] Thaler, R.H.: Anomalies: The winner's curse. The Journal of Economic Perspectives 2(1), 191–202 (1988)

[181] The Globus Alliance: Globus homepage (2009), `http://www.globus.org/`

[182] The Globus Alliance: The ws-resource framework (2012), `http://www.globus.org/wsrf/`

[183] Thomas, N., Bradley, J.T., Knottenbelt, W.J.: Stochastic analysis of scheduling strategies in a grid-based resource model. Software, IEE Proceedings - 151(5), 232–239 (2004)

[184] Toms, L., Caminero, A.C., Carrion, C., Caminero, B.: Network-aware meta-scheduling in advance with autonomous self-tuning system. Future Generation Computer Systems 27(5), 486–497 (2011)

[185] Toms, L., Caminero, B., Carrin, C., Caminero, A.: On the improvement of grid resource utilization: Preventive and reactive rescheduling approaches. Journal of Grid Computing pp. 1–25 (2012)

[186] Vanmechelen, K., Broeckhove, J.: A comparative analysis of single-unit vickrey auctions and commodity markets for realizing grid economies with dynamic pricing. In: Veit, D., Altmann, J. (eds.) Grid Economics and Business Models, vol. 4685, pp. 98–111. Springer Berlin / Heidelberg (2007)

[187] Vanmechelen, K., Depoorter, W., Broeckhove, J.: Combining futures and spot markets: A hybrid market approach to economic grid resource management. Journal of Grid Computing 9(1), 81–94 (2011)

[188] Venugopal, S., Xingchen, C., Buyya, R.: A negotiation mechanism for advance resource reservations using the alternate offers protocol. In: Quality of Service, 2008. IWQoS 2008. 16th International Workshop on. pp. 40–49 (2008)

[189] Venugopal, S., Buyya, R.: A deadline and budget constrained scheduling algorithm for escience applications on data grids. In: Distributed and Parallel Computing, pp. 60–72 (2005)

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

[191] VIOLA: Vertically integrated optical testbed for large applications in dfn (2007), `http://www.viola-testbed.de/`

[192] Waldrich, O., Wieder, P., Ziegler, W.: A meta-scheduling service for co-allocating arbitrary types of resources. In: Parallel Processing and Applied Mathematics, pp. 782–791 (2006)

[193] Weng, C., Lu, X.: Heuristic scheduling for bag-of-tasks applications in combination with qos in the computational grid. Future Generation Computer Systems 21(2), 271–280 (2005)

[194] Wieczorek, M., Prodan, R., Fahringer, T.: Scheduling of scientific workflows in the askalon grid environment. Sigmod Record 34(3), 56–62 (2005)

[195] Wieczorek, M., Hoheisel, A., Prodan, R.: Towards a general model of the multi-criteria workflow scheduling on the grid. Future Generation Computer Systems 25(3), 237–256 (2009)

[196] Wilkes, J.: Utility functions, prices, and negotiation. In: Buyya, R., Bubendorfer, K. (eds.) Market Oriented Grid and Utility Computing, pp. 67–88. No. Wiley Series on Parallel and Distributed Computing, John Wiley & Sons, Inc (2008)

[197] Wisconsin-Madison, U.O.: Condor manual version 7.2.4 (2009), `http://www.cs.wisc.edu/condor/manual/`

[198] Wolski, R., Brevik, J., Plank, J.S., Bryan, T.: Grid resource allocation and control using computational economies. In: Berman, F., Fox, G., Hey, T. (eds.) Grid Computing: Making the Global Infrastructure a Reality, pp. 747–771. Wiley, Chichester (2003)

[199] Wolski, R., Plank, J.S., Brevik, J., Bryan, T.: Analyzing market-based resource allocation strategies for the computational grid. International Journal of High Performance Computing Applications 15(3), 258–281 (2001)

[200] Xhafa, F., Abraham, A.: Computational models and heuristic methods for grid scheduling problems. Future Generation Computer Systems In Press, Accepted Manuscript (2009)

[201] Xiaohui, W., Zhaohui, D., Shutao, Y., Chang, H., Huizhen, L.: Csf4: A wsrf compliant meta-scheduler. In: World Congress in Computer Science Computer Engineering and Applied Computing. pp. 61–67. Las Vegas, USA. (2006)

[202] Xingchen, C., Nadiminti, K., Chao, J., Venugopal, S., Buyya, R.: Aneka: Next-generation enterprise grid platform for e-science and e-business applications. In: e-Science and Grid Computing, IEEE International Conference on. pp. 151–159 (2007)

[203] Yu, J., Buyya, R., Ramamohanarao, K.: Workflow scheduling algorithms for grid computing. In: Xhafa, F., Abraham, A. (eds.) Metaheuristics for Scheduling in Distributed Computing Environments, vol. 146, pp. 173–214. Springer Berlin / Heidelberg (2008)

[204] Zhang, X., Freschl, J.L., Schopf, J.M.: Scalability analysis of three monitoring and information systems: Mds2, r-gma, and hawkeye. Journal of Parallel and Distributed Computing 67(8), 883–902 (2007)

[205] Zini, F., Giulioni, G., Reinicke, M., Streitberger, W.: Ist-fp6-003769 catnets d2.1 analysis of simulation environment (2005), `http://www.catnets.uni-bayreuth.de/fileadmin/publications/d2_1.pdf`