Technical Report

UCAM-CL-TR-622
ISSN 1476-2986

Number 622

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

Cooperation and deviation in market-based resource allocation

Jörg H. Lepler

March 2005

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

This technical report is based on a dissertation submitted November 2004 by the author for the degree of Doctor of Philosophy to the University of Cambridge, St John's College.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/TechReports/

ISSN 1476-2986

Abstract

This thesis investigates how business transactions are enhanced through competing strategies for economically motivated cooperation. To this end, it focuses on the setting of a distributed, bilateral allocation protocol for electronic services and resources. Cooperative efforts like these are often threatened by transaction parties who aim to exploit their competitors by deviating from so-called cooperative goals. We analyse this conflict between cooperation and deviation by presenting the case of two novel market systems which use economic incentives to solve the complications that arise from cooperation.

The first of the two systems is a pricing model which is designed to address the problematic resource market situation, where supply exceeds demand and perfect competition can make prices collapse to level zero. This pricing model uses supply functions to determine the optimal Nash-Equilibrium price. Moreover, in this model the providers' market estimations are updated with information about each of their own transactions. Here, we implement the protocol in a discrete event simulation, to show that the equilibrium prices are above competitive levels, and to demonstrate that deviations from the pricing model are not profitable.

The second of the two systems is a reputation aggregation model, which seeks the subgroup of raters that (1) contains the largest degree of overall agreement and (2) derives the resulting reputation scores from their comments. In order to seek agreement, this model assumes that not all raters in the system are equally able to foster an agreement. Based on the variances of the raters' comments, the system derives a notion of the reputation for each rater, which is in turn fed back into the model's recursive scoring algorithm. We demonstrate the convergence of this algorithm, and show the effectiveness of the model's ability to discriminate between poor and strong raters. Then with a series of threat models, we show how resilient this model is in terms of finding agreement, despite large collectives of malicious raters. Finally, in a practical example, we apply the model to the academic peer review process in order to show its versatility at establishing a ranking of rated objects.

Contents

Contents **3**

List of Figures **8**

List of Tables **11**

1 Introduction **15**
1.1 Motivation and Problem Statement 15
1.2 Resource Pricing 17
1.3 Reputation Aggregation 18
1.4 Analytical Methodology: Simulations of Synthetic Threat Models 20
1.5 Simulation Technology 20
1.6 Protocol Environment 21
1.7 Thesis Layout 21
1.8 Claims 22
1.9 Collaboration in this Thesis 23

2 Background and Related Work **25**
2.1 Introduction to a Microeconomic Perspective of Cooperation 25
2.1.1 Incentives for Cooperation 25
2.2 Resource Pricing 29
2.2.1 Pricing of QoS Allocations 30
2.2.2 Pricing, Bundling and Markets 30
2.2.3 Resource Economy Systems 31
2.2.4 Ogino's Protocol for Competitive Bandwidth Allocations 31
2.3 Reputation Systems 31
2.3.1 Transfer of Endorsement 31
2.3.2 Collaborative Filtering 32
2.3.3 General Reputation Aggregation Systems 33
2.3.4 Peer-to-Peer Systems 34
2.3.5 Agent Misbehaviour in Networks 35
2.3.6 Trust Systems 36
2.4 Conclusions 36

3 The Supply Function Pricing Model **39**
3.1 Introduction 40
3.2 The Allocation Protocol 42

3.3 Economic Framework . . . 44
3.4 Clients . . . 45
3.5 Strategy of Resource Providers . . . 46
3.6 Updating . . . 48
3.6.1 Initial Request Probability . . . 48
3.6.2 Received Quotes . . . 48
3.6.3 Accepted Offers . . . 49
3.7 Conclusions . . . 50

4 Effectiveness and Resilience of the Pricing Model 51
4.1 Competition With Symmetric Pricing Functions . . . 52
4.1.1 Trading over Example Periods . . . 52
4.1.2 Scale of Competition . . . 52
4.2 Competition with Undercutting . . . 55
4.2.1 Negative Effects of Increasing the Weight on the Undercutting Sensitivity . . . 57
4.2.2 The Weight Level Under Different Market Configurations . . . 58
4.3 Conclusions . . . 58

5 Consensus Seeking Recommendation Aggregation 61
5.1 Motivation for the Recommendation Aggregation Service . . . 63
5.2 Reputation System Design . . . 64
5.2.1 Reputation System Philosophy . . . 64
5.2.2 Rating Commentators . . . 66
5.3 Economic Incentives . . . 66
5.3.1 Truth Revelation Incentives . . . 66
5.3.2 Financing the Operations of the Reputation System . . . 67
5.4 The Rating Mechanism . . . 69
5.4.1 Definitions . . . 69
5.4.2 The Iterative Loop . . . 70
5.4.3 Convergence . . . 72
5.4.4 Confidence Value . . . 73
5.5 Conclusions . . . 73

6 Analysis of the Reputation Model 75
6.1 First Example . . . 75
6.2 Two Opposing Groups . . . 77
6.2.1 One Group Removed From The Two Opposing Ones . . . 80
6.2.2 Alternate Convergence Point . . . 80
6.3 Many Diverging Rater Groups . . . 81
6.4 Algorithm Convergence . . . 82
6.4.1 Conditions for Algorithm Convergence . . . 82
6.4.2 Rate of Convergence . . . 84
6.4.3 Convergence Threshold . . . 86
6.4.4 Multiple Convergence Points . . . 87
6.5 Conclusions . . . 88

7 Applications of the Reputation Model **91**
7.1 Statistical Analysis . . . 92
7.2 Different Rating Accuracy . . . 92
7.2.1 Few Poor Raters . . . 92
7.2.2 A Large Majority of Poor Raters . . . 93
7.3 Filtering Deviating Commentators . . . 94
7.3.1 Malicious Rater Collective Manipulating All Scores . . . 94
7.3.2 Malicious Rater Collective Manipulating One Provider Score . . . 97
7.4 Deliberately Inconsistent Provider Performance . . . 99
7.5 Choosing the Appropriate Model Selectivity Weight . . . 100
7.6 Client-Specific Rating Customisation . . . 101
7.7 Aging Comments . . . 101
7.8 Limitations of the Rating Model . . . 102
7.9 Social Choice . . . 102
7.10 Further Application Areas of the Reputation Model . . . 103
7.10.1 Stock Market Predictions . . . 103
7.10.2 School Grading . . . 103
7.10.3 User Movie Ratings . . . 104
7.11 Conclusions . . . 104

8 Reputations Applied to the Academic Peer Review Process **107**
8.1 Theory of Reputations for Peer Reviews . . . 107
8.1.1 Peer Review Process Parameters . . . 108
8.1.2 Contributions of the Recommendation System . . . 111
8.1.3 Discussion of Possible Objections . . . 112
8.2 Example Peer Review at the PET Workshop . . . 113
8.2.1 Review Data . . . 113
8.2.2 Reputation Model Analysis . . . 114
8.2.3 Discussion of the Results . . . 115
8.3 Conclusions . . . 119

9 Conclusions **125**
9.1 Pricing Model . . . 125
9.2 Reputation Aggregation Model . . . 126
9.3 Cooperation and Deviation . . . 128

A Protocol Simulation **131**
A.1 Approach . . . 131
A.1.1 Application or Simulation? . . . 131
A.1.2 Discrete Event Simulation . . . 131
A.2 Core Simulator Implementation . . . 132
A.2.1 Discrete Event Simulation Packages . . . 132
A.2.2 DES Implementation Approach . . . 134
A.2.3 Communicating Processors . . . 134
A.2.4 Coroutines . . . 134
A.2.5 Random Number Generator . . . 136
A.3 Thread Switching . . . 136

A.3.1 General Principle . . . 136
A.3.2 Synchronisation Through Locks . . . 137
A.3.3 Explicit Thread Scheduling . . . 137
A.3.4 Implementation . . . 138
A.3.5 Java Threads . . . 138
A.4 Event Queue . . . 139
A.4.1 Performance Critical Functions . . . 139
A.4.2 Priority Queue Theory . . . 139
A.4.3 Event Horizon . . . 140
A.4.4 Qheap . . . 140
A.4.5 Events . . . 141
A.5 Reservation Protocol Implementation . . . 141
A.5.1 Simulation Startup . . . 141
A.5.2 Simulation Message Communication . . . 142
A.6 Simulation of Clients . . . 142
A.7 Simulation of Servers . . . 143
A.7.1 The Processing Loop . . . 143
A.7.2 Processing an Event . . . 144
A.7.3 Message Handling . . . 144
A.7.4 Resource Management . . . 145
A.8 Performance Evaluation . . . 146
A.8.1 The Micro-Benchmark . . . 146
A.8.2 Event Queue . . . 147
A.8.3 Thread Switching . . . 148

B The Scenario and Language of the Allocation Protocol 151
B.1 Scenario Overview . . . 152
B.2 Client Applications . . . 153
B.2.1 Allocation Strategies . . . 154
B.2.2 Platform Independent Applications . . . 155
B.3 Resource Providers . . . 155
B.4 Negotiating Agents and Brokers . . . 156
B.5 Open Services Platform - Web Services Technology . . . 157
B.5.1 UDDI — Universal Description, Discovery, and Integration . . . 158
B.5.2 WSDL — Web Services Description Language . . . 159
B.5.3 SOAP — Simple Object Access Protocol . . . 159
B.6 Facilitating the Allocations With Web Services . . . 160
B.7 The Allocation and Negotiation Language . . . 161
B.8 Complex Allocations . . . 163
B.9 Security . . . 164
B.9.1 Web Services Security . . . 164
B.9.2 Allocation Protocol Security . . . 166
B.9.3 Trusting the Client Applications . . . 166

Bibliography 167

List of Figures

2.1 A game with a dominant strategy equilibrium. 26
2.2 A game with a Nash-Equilibrium. 27
2.3 The prisoner's dilemma in a payoff matrix. 28

3.1 The Communication Protocol 43
3.2 Relationship between average price and demand 49

4.1 Price quotes of three providers and 15 clients per provider over one example period. 53
4.2 Price quotes of three providers and 15 clients per provider over one example period. 53
4.3 Scaling Market Size: Equilibrium price as a function of number of clients per provider for 3, 5, 10 and 20 providers. 54
4.4 Price quotes with one seller undercutting (selling out at t=33800) the two competitors. 56
4.5 Prices with undercutter (selling out at t=34000) and increased weight on accepted bookings. 56
4.6 Profitability of Undercutting (red) as a function of weight on successful bookings. 57
4.7 Weights required on successful bookings to prevent undercutting as a function of number of providers and clients. 59

5.1 Structure of the Reputation System. 65

6.1 Scores for the Example Scenario Dependent on the Selectivity Weight. . . . 76
6.2 Influence Shares for the Example Scenario Dependent on the Selectivity. . 76
6.3 Scores with Two Opposing Groups of Raters. 79
6.4 Influence Shares with Two Opposing Groups of Raters. 79
6.5 Iteration Updates by the Rating Algorithm of the Scores in the Two-Group Scenario for $w = 0.85$. 79
6.6 Corresponding Iteration Updates of the Influence Shares. 79
6.7 Scores with only the larger of the two opposing groups of raters present. . . 80
6.8 Scores for the same two group comments, but algorithm Initialisation Vector on the smaller group. 80
6.9 Scores with four opposing groups of raters. 83
6.10 Scores for provider p_A depending on the Initialisation Vector (IV) supplied by comments of each rater $r_{E,..,P}$. 83

6.11 Reputation Model Scores for the Divergent Rating Scenario Examining the Convergence Rate. . . . 85
6.12 Influence Shares for the Divergent Rating Scenario Examining the Convergence Rate. . . . 85
6.13 Iteration Cycles Required to Converge for the Divergent Rating Scenario, Depending on the Selectivity Weight. . . . 86
6.14 The Largest Update Step Sizes of Scores per Iteration for Selected Selectivity Weight Values. . . . 86
6.15 Slope of Convergence of Scores for Divergent Example with $w = 1.77$, Converging to Voting Block A. . . . 88
6.16 Slope of Convergence of Scores for Divergent Example with $w = 1.78$, Converging to Voting Block B. . . . 88
6.17 Slope of Influence Shares During Convergence for the Divergent Example with $w = 1.77$, Converging to Voting Block A. . . . 89
6.18 Slope of Influence Shares During Convergence for the Divergent Example with $w = 1.78$, Converging to Voting Block B. . . . 89

7.1 Scores for a Mix of 90 Raters with $\sigma = 0.25$ and Ten Raters at $\sigma = 1.0$. . . . 93
7.2 Influence Shares for a Mix of 90 Raters with $\sigma = 0.25$ and Ten Raters at $\sigma = 1.0$. . . . 93
7.3 Scores for a Mix of Ten Raters with $\sigma = 0.25$ and 90 Raters at $\sigma = 1.0$. . . . 94
7.4 Influence Shares for a Mix of Ten Raters with $\sigma = 0.25$ and 90 Raters at $\sigma = 1.0$. . . . 94
7.5 Scores for a Scenario with 40 Raters Deviating with a Rating Offset (-2.5) Applied to All Providers. . . . 95
7.6 Influence Shares for a Scenario with 40 Raters Deviating with a Rating Offset (-2.5) Applied to All Providers. . . . 95
7.7 Scores for same scenario with the Initialisation Vector set to the deviators' comments. . . . 96
7.8 Influence Shares with the Initialisation Vector favouring the deviators' and their convergence point. . . . 96
7.9 Scores for 40 Raters Deviating with a Distinct Rating Offset (-2.5) on Provider p_A Only. . . . 97
7.10 Influence Shares for 40 Raters Deviating with a Distinct Rating Offset (-2.5) on Provider p_A Only. . . . 97
7.11 Scores when algorithm initialised in favour of second convergence point, belonging to the deviators. . . . 98
7.12 Influence Shares with Initialisation Vector favouring deviators' convergence point. . . . 98
7.13 Scores for both convergence points where provider p_A is deliberately delivering inconsistent performance. . . . 100
7.14 Influence distribution with initialisation vector set on comments of minority group and revealing the second convergence point from $w \geqq 1.35$. . . . 100

8.1 Reputation system scores and their confidence (papers p0-p20). . . . 120
8.2 Reputation system scores and their confidence (papers p21-p48). . . . 121
8.3 The reputations of the reviewers. . . . 122

8.4 The distribution of the maximal distances of ranking change due to the application of our reputation system. 123

A.1 The structure of a discrete event simulator. 133
A.2 Simulation of communicating processors. 135
A.3 Switching execution threads. 137
A.4 Context switching implementation. 138
A.5 The clients' resource request procedure. 143
A.6 The servers' processing loop. 144
A.7 Processing a Request. 145
A.8 Event queue processing benchmark. 148
A.9 Thread switching benchmark. 149

B.1 Example Application Scenario 152

List of Tables

6.1 Example scenario showing comments submitted by raters r, about providers p, the resulting model scores s and the influence share in, both depending on the chosen selectivity weight w. 76
6.2 Comments Submitted by Two Opposing Groups of Raters 78
6.3 Comments Submitted by Four Opposing Groups of Raters 82

8.1 Peer review comments (raters r0-r12) and reviewers' influence for the maximal trust sharpening factor. 116
8.2 Peer review comments (reviewers r13-r18), the outcome and the reputation system's scores for the maximal trust sharpening factor. 117

Chapter 1

Introduction

This thesis investigates how business transactions in the environment of electronic services can be enhanced through different kinds of cooperation. In this chapter we first motivate why we work on the general problem (section 1.1), then we introduce the pricing model (section 1.2) and reputation aggregation model (section 1.3). Following, we discuss the methodology (section 1.4) and simulation technology (section 1.5) that we apply to analyse the models. Finally, we present the layout of the thesis (section 1.7) and conclude by stating the claims we make with regards to its contributions (section 1.8).

1.1 Motivation and Problem Statement

Just as much as competition is a fundamental concept of market economics, so is cooperation a fundamental concept that is necessary to develop effective markets. We can observe cooperation in economic markets at many levels and in very different forms and usually to the benefit of all participants. The purpose of cooperation in a market economy is to achieve better resource allocations and to allow for specialisation of the players, both of which lead to gains in market efficiency and rewards for all involved players. The main difficulty for cooperative arrangements is that these can be exploited by an individual party who furthers his own welfare at the expense of the cheated cooperation partners. Even worse, some parties could form their own cooperative subgroup, resembling a cartel that beats the rest of the market.

These "games", that are being played amongst businesses, are well known to economists and are a well accepted and researched topic.[1] When we build computer-based systems for open market allocation of resources and services, we need to take these business behaviours into account, since they are a fundamental aspect of what powers the efficiency of market economics.

It has often been stated in the popular press that electronic markets, "Information Markets", follow different rules from the traditional "brick-and-mortar" economies. This is true in as much as the "Information Revolution" changes the way goods are produced,

[1] See for example chapters 28 and 29 in Varian [92] as a starting point.

distributed and consumed - incidentally this is analogous to the scope of change during the industrial revolution. At a fundamental level though, Information Markets will follow the same theoretical behaviours, because the incentives for behaviours remain the same, and as Hal Varian has put it in his introduction to information economics: "After all, economics is primarily about people not goods." (Chapter 34 in [92]). However, the parameters for these business games have changed with the advent of electronically automated business functions.

Among the parameters that are specific to information economics is: (1) The almost non-existent marginal cost (the cost to produce another copy of the product); Less exclusive to information economics, but very characteristic for it are (2) network effects (the more people use a certain product, the more valuable it becomes, e.g. e-mail); (3) Complimentary products (the output of one product is able to work as input to the next, e.g. Office documents); and (4) Product lock-in (one loses access to some data when changing a software product). The latter three properties rely on information technology's need for standard software and the inertia of legacy data.

There are some more information economy specific parameters, all of the following will play a role in chapter 3, where we develop a pricing model for a resource reservation protocol: (5) Low transaction overhead; (6) Automated product comparison incurs practically zero-level search cost; if one solves aspects (2-4) through open standards, we also obtain (7) highly substitutable products; and (8) Practically no switching cost. The latter three aspects lead to a very intense, theoretically perfect level of competition.

For the second part of this thesis we develop another economically motivated model, this time for a reputation system. This model, which we present in chapter 5 pays particular attention to a further set of information economic aspects: (9) Parties remain relatively anonymous, if most of the transaction is performed through virtual contact; if aspects (6-8) are present, we are facing (10) large numbers of transient business relationships. The last two aspects make it difficult for business parties to develop meaningful reputation knowledge and ease the way for manipulation, when one aggregates reputation information.

The two economically incentivated systems that we develop and analyse in this thesis - pricing and reputation - are linked through the central transaction of this thesis, namely: the market-based allocation of resources. Both systems are implemented in the same simulation that captures a market of resource providers and clients who allocate the providers' services. To facilitate this market we develop a protocol that allows for competition between providers as well as clients. Both systems use economic models to alleviate the novel Information Market problems and both of these economic models use cooperation to achieve their aim. While in both cases, the cooperation is induced through economic incentives, they differ greatly in the level where within the models these incentives are placed. In the case of the pricing system, the incentives are indirect, not visible, and need to be discovered through careful economic analysis. The incentives in the reputation system are explicit and very visible, since reputation is a public aspect of economics. Conversely, the threats for each of the systems are working the opposite way. The pricing model is at direct economic threat from a participant who tries to exploit the economic benefits directly by cutting out the competition. The threats for the reputation system

are indirect, such as a participant who attempts to achieve undue reputation standings, to then, in a secondary step, benefit from this economically in the market place.

In the following sections we introduce the general functionality of the two market systems that we will develop, the pricing system and the reputation system.

1.2 Resource Pricing

The pricing model (see chapter 3 for its formal description) was developed for the allocation protocol that we mentioned above, where the clients simultaneously negotiate with all suitable resource providers, and the providers independently decide on the price they offer to each client. This model assumes that resources are any kind of non-storable service, such as computer processing time or network communication capacity.

Since all these negotiations are independent and possibly concurrent, providers have to compete for clients when supply exceeds the demand, and conversely if demand exceeds supply the clients have to compete for provider services. Our model's contribution focuses on the case when supply exceeds demand. We assigned this focus, because first of all, the pricing situation is straightforward for the providers when demand exceeds supply, since in this case they simply have to calculate the market clearing price for the aggregated demand and supply. Secondly, a situation where some of the demand is denied services is undesirable, because economically it is a loss that calls for new market entrants who will serve this unsatisfied demand, and technically it is a loss for business operations if a client intends to perform some operations that then are being blocked. However if supply exceeds demand, and we consider competition to be perfect, namely search and transaction cost to be zero, then traditional economic theory suggests that prices will drop to zero. While some observations of international communication bandwidth markets suggest that these assumptions hold up in the real world, such a result is undesirable, because it means that the relative level of supply will drop, by means of providers going out of business, until sufficient periods of supply shortages allow for sufficient profits.

When looking at a situation with excess resource supply, one solution to avoid prices falling to zero, is for all providers to equally withhold some of their supply. However, if only one or a minority of providers withhold some supply, the competition will sell even more and this withholding strategy would turn out to be economically irrational. Any attempt by the providers to establish this equal withholding cooperation by means of explicit communication would be considered cartel behaviour and therefore illegal. Since all providers benefit equally from this withholding strategy, it is conceivable that each of them individually decides to choose this strategy. Ideally, we reach a situation whereby all providers choose the withholding strategy and are all content with this choice. Content in this context means, that in retrospect when they know the supply decisions of their competitors, they recognise that a different choice would not have made them any better off. In microeconomic terms, such a solution forms a Nash-Equilibrium (NE). In order to model such a solution correctly, the providers need to have a model that contains a function for the supply of each of the providers, dependent on the demand, and be able to continuously update the model's assumptions about the market situation's parameters.

We will develop a model for our allocation protocol under the described constraints, where we use supply-functions to calculate the market clearing price at the corresponding Nash-Equilibrium. Since supply-function theory was originally developed for centralised market exchanges, we had to extend the model to the distributed bilateral negotiation case. In order to enable the providers to adapt to their competitors strategic decisions, the model continuously updates its assumptions with the actual realised numbers of client requests and the successful sales. In this way, the pricing model avoids any form of explicit cooperation with the competitors that could be considered illegal.

A general question is whether this supply-function model is technically able to achieve the equilibrium price levels that were intended, and if so, under what market conditions? Further, an important question is whether providers can beat their competitors if they know about the competitions' strategies, namely to calculate prices through this supply-function model? Would it be possible for one or more providers to do better by employing a "deviant" strategy of their own, for example by undercutting the competitors? We will have to show that the model can be adjusted, such that the choice for providers to cooperate with the supply-function model dominates the choice to deviate with an undercutting strategy. To this end, we will analyse in chapter 4 various threat conditions and how high we have to set the parameter that determines the model's responsiveness to deviators in order to diminish the threat's profitability.

1.3 Reputation Aggregation

Reputations are the aggregation of past transaction experiences. In real life reputations are used to predict the future behaviour of a potential transaction partner. Reputations are particularly important where one party has to commit at the beginning of a series of transactions to a partner. In this case, we use reputations to allow ourselves to enter transactions that carry a higher risk. For example, to buy fruit from a market stall does not require much reputation, since the produce can be inspected very well before committing to the purchase. This is different if one is about to choose the doctor for a medical procedure, or if one were to order the fruit unseen over the telephone or the Internet.

For some business domains it is perfectly sufficient for each transaction party to build their own memory of experiences and reputation implications, since a market stall does not survive for long without return customers, when they have sold foul lemons. However, if most of transactions in a business domain are one-off relationships, then it is necessary to aggregate the experiences of many parties to build meaningful reputations. An example is signing up for a medical procedure like heart surgery, or a purchase from an Internet trader, where a multitude of suppliers and their easy discovery allows for transient business relationships. One would probably not enter such transactions without any indication that our transaction partner has done well in past transactions.

The reputation model we build in chapter 5 will focus on the aggregation function of a reputation system that collects transaction experiences. The aggregation function derives a common ordering for the preference of the rated parties. Obtaining such an common

ordering is useful when one seeks to make recommendations about the performance of a party. A further useful effect of publishing the aggregated scores is to mask the individual comments a rater has provided. This is helpful when one finds it necessary to protect the rater from the response of the rated party, while still leaving the incentives for the rated parties to aim for a good rating in place.

Raters will inevitably give different comments for the same rating aspect, and it is an interesting question for the rating aggregation function how to treat these variances. The design choice for this rating aggregation policy depends on the assumptions we make about the sources for these variances, and we might find some of these possible sources to be unacceptable or avoidable. In an ideal case, these variances reflect the different levels of service that the provider actually managed to deliver to the different clients. However, in some cases the service is identical for all the raters, and then it still is possible that some raters perceive the rating level differently to others, possibly because some are not as able to assess performance as other are. Or, some raters might be biased in general, and in the worst case a rater could have external reasons to attempt to influence the outcome of the aggregated scores in a favourable way. To solve this problem, we assume that not all raters are "equally" able to provide comments that generate agreement with the other raters' preferences. Therefore, in the aggregation function that we develop, we assign raters who contribute more than others to a consensus solution, a higher influence on the final outcome of the recommendations.

The goal of our aggregation function is to maximise the level of agreement within the comments at hand. In order to achieve this goal, the model penalises deviation (actually: variance) from the consensus. This approach has its analogies in finance, where risk management weighs down titles with a high deviation.

Questions that arise from the design approach of our reputation aggregation model fall into two categories, (1) technical and (2) pragmatic. The technical category is addressed in chapter 6 and discusses questions such as the convergence of the iterative aggregation algorithm. There, we also verify if the model output actually follows the stipulated theory, and analyse the algorithmic properties, such as different possible convergence points. We address the pragmatic questions in chapters 7 and 8, with questions regarding the recommendations that the reputation model produces in response to particular scenarios of rater behaviour. In chapter 7 we develop threat scenarios with statistically modelled raters, where a subset of the raters applies a "deviant" behaviour. These deviant behaviours address questions such as: (1) What can our aggregation model achieve, if some raters give less accurate comments? (2) What if some raters are biased and attempt to influence the scoring outcome? (3) Or what can be done if a service provider delivers differing qualities of service to different clients who then rate this provider? In chapter 8 we apply our reputation model to the academic peer review process, placing reviewers in the spot of the raters, the submitted papers being the rated entities. There, we discuss how to arrange this process, such that it is effective for numeric review comments, and further on compare an actual conference review's outcome with the recommendations our model produces.

1.4 Analytical Methodology: Simulations of Synthetic Threat Models

The theme of this thesis about cooperation and deviation is carried in our analysis. Both of the systems we build, pricing and reputation, promote cooperation among independently operating parties. In the analysis we need to "stress test" our models for the potential breakdown of this cooperation. The cooperation may break down if some of the parties deviate from the cooperation in order to advance their own interests.

Throughout the majority of our analysis in this thesis we use simulations of synthetic threat scenarios to analyse the viability of the pricing and reputation model. Synthetic threat scenarios are best suited to present the models with challenging deviation situations. A common scepticism against synthetic scenarios is that they might not be sufficiently realistic and critics then demand the use of real data to verify the practical utility of this model. And indeed, it would be interesting to run the models presented in this thesis on real-world data, since it would allow for an evaluation of the practical benefits when employing such a model. However, in the analysis done in this thesis, we focus on the model's abilities to respond to a lack of cooperation down to outright deviation. Real-world data might contain such scenarios, but would be unlikely to present a similar challenge, unless some of the real-world participants actually have an agenda similar to the deviators in the synthetic scenarios. Furthermore, the models we present are entirely novel, and then to use demand profiles, or supply strategies, that are derived from present real-world data, would lead to misrepresentations, since all market participants will adapt their demand/strategy to the presence of the new model, as well as the behaviour of the other participants.

Alternatively, it could be possible to use formal analysis to reason about the stipulated properties of the two systems. However, the complexity of both systems makes it nearly impossible to reason at a formal theoretical level about the more interesting scenarios that we will analyse.

1.5 Simulation Technology

In order to examine the resource allocation protocol, the pricing system, the reputation system and the threat models, we implement a discrete event simulator to bind all these modules together. It was necessary to build a time-based simulation to be able to simulate the allocation protocol and the pricing model with its updating mechanism. A discrete event simulation is most suitable to capture the asynchronous behaviour of our concurrently interacting parties.

Appendix A describes the discrete event simulator that we implement in Java to suit the requirements of our simulation. We decided against using existing packages, because they either did not allow for dynamic allocations, or did not provide the most desirable programming model, or else simply performed poorly. We simulate the interacting parties through a model of "communicating processors", which makes time and state transparent

for the simulated parties, they can be implemented as one would do in a real application. Using the simulator base we also build a framework for servers (service providers) and clients that implement the allocation protocol. In order to increase the efficiency of our simulation, we optimised the thread switching by using low-level thread control. The other performance critical function in a discrete event simulator is the event queue, where we implement a Qheap structure that has near optimal scalability, and involves low overhead. Finally, we demonstrate the performance of this simulator in terms of its core speed and scalability.

1.6 Protocol Environment

We describe, in appendix B, the protocol environment for market-based resource allocations at different layers of abstraction and discuss the implications of the available technologies for our allocation protocol. There, we give examples of the different types of clients and their applications, who might seek resources and services. We continue with a description of applicable types of resources and then move on to the technical interface between clients and providers. We describe how clients could discover the providers and the negotiation language they could use to implement the steps of our allocation protocol of chapter 3. We further elaborate on the strategies clients and providers are allowed to apply in their allocations, the scope for complex allocations with price bundles, and security issues around such a protocol in an open environment.

1.7 Thesis Layout

Chapter 2 introduces the background to this thesis, and, for readers who are not familiar with game theory, it starts out with a basic introduction to microeconomics and how its principles are applied in this thesis. It continues with a discussion of the relevant literature for market-based pricing systems and the reputation system. Chapter 3 describes the resource allocation protocol and the formal model of the pricing system we introduced previously in section 1.2. Chapter 4 then analyses the effectiveness and resilience of the pricing system. In chapter 5, we present the formal model of our consensus seeking reputation aggregation system, and following evaluate its technical properties in chapter 6. Then, in chapter 7, we put the reputation model's applicability to the test with a suite of challenging threat models. To show the versatility and practical applicability of the reputation model, we formulate in chapter 8 how it can be applied to the academic peer review process and analyse an actual conference review with our reputation model. We conclude in chapter 9 with a summary of this thesis's contributions to the discussion of cooperation and deviation in market-based resource allocation.

In appendix A we present the discrete event simulator we implemented to perform the analysis of the models, and in appendix B we illustrate the larger technical environment that enables an open market-based allocation protocol.

1.8 Claims

Here we summarise the contributions of this thesis, broken down by the research domains:

Pricing System

- We develop a resource pricing system that successfully solves the problem of imploding resource price during times of supply exceeding demand.
- This pricing model is innovative because of its application of supply functions to a distributed, bilateral market protocol.
- We implement this pricing system in a discrete event simulation and show that it is able to raise prices above competitive levels.
- We evaluate the resilience of the pricing system against undercutting and demonstrate the effectiveness of its response measures.

Reputation Aggregation

- We develop a reputation aggregation system that is able to discern between strong and weak raters, based on their contribution to a common consensus.
- This reputation system contains a novel aggregation algorithm that is inspired by search engines' transfer of endorsement and risk analysis's view of statistical variations.
- We evaluate the technical properties of the iterative aggregation algorithm and demonstrate that it scales well, even to low numbers of raters.
- With a simulation of threat models we show how the reputation system is able to recognise large proportions of weak raters, and that it is able to adjust the scores accordingly.
- As a case example we apply this reputation system to the academic peer review process. We use an actual workshop's review comments as input to the system to evaluate its scoring/ranking method.

Synthetic Threat Model Simulations

- We develop a suite of threat models that challenge the limits of the pricing and reputation systems. These synthetic threat models resemble an insightful evaluation, since they are designed to maximise the threat.
- We implement an efficient discrete event simulator that outperforms several similar tools.

Cooperation and Deviation

- We show that both the pricing and reputation systems promote cooperation that yields benefits for all the contributors.
- Through the suite of threat models we show that both systems are able to place dis-

incentives on strategies that deviate in an exploitative fashion from the cooperation strategy that the systems seek to promote.

Economic Factors in Computer Systems

It is our intention that this dissertation contributes to the computer science community's understanding about the type of decisions that economically rational agents would make. In open market systems, the resource allocation decisions fundamentally depend on economic factors. If one intends to optimise the resource utilisation, such as it is usually done in computer science research, one needs to take into account a model of the economic behaviour of the market's agents. In computer science, one of the most popular questions start with the words: "but what if agent A does behaviour B...", and additionally implies"... for whatever irrational reason". Traditional system design then amends the design such that it is impossible for A to do B. However, for many scenarios, we do not need to adhere to such stringent standards. In this dissertation's scenarios, we allow agents to engage in any behaviour, but then limit the damage this can do to the utility of our system, and make sure that any unwelcome behaviour is not rewarded.

1.9 Collaboration in this Thesis

Since the main thrust of this thesis is to argue the case for cooperation, it should come as no surprise that parts of its inception are products of collaboration. This collaboration is limited to chapters 3 and 4 where the author was working with Karsten Neuhoff, a PhD student at the Department of Applied Economics at Cambridge University. While it was the author's part to draw up the problem scenario, it was Karsten Neuhoff's idea to use Supply-Functions to address the problem. We jointly developed the model that adjusts the Supply-Functions to a decentralised market of resources and drew up the threat models for the analysis. It was the authors sole responsibility to implement all of the discrete event simulation that hosts the model and to develop all the data analysis. This joint work, as it is contained in chapters 3 and 4, was presented and published at a conference [59]. Subsequently it was also published in a Journal [60].

Chapter 2

Background and Related Work

In this chapter, we introduce the background material and the related research for the systems and models we develop in the following chapters. First, however, we start out in section 2.1 with an introduction to Microeconomics and Game Theory. While this basic text book style introduction is mainly intended for computer scientists, it also points out how and where in our thesis we make use of the corresponding microeconomic reasoning. We then continue in section 2.2 with a review of the related research for resource pricing, which sets the background for chapters 3 and 4. Finally, in section 2.3, we cover various research on reputation systems that can be contrasted with our work on this topic in chapters 5-8.

2.1 Introduction to a Microeconomic Perspective of Cooperation

We start with a brief introduction into Microeconomics and Game Theory to illustrate the rationale we will apply in the development and analysis of our thesis. The examples are based on Hal Varian's book on Microeconomics [92], and readers who are familiar with game theory and microeconomics might want to skip this section.

2.1.1 Incentives for Cooperation

Cooperation is a prevalent element in the business world and usually benefits all participants. Cooperation starts at the point where two players agree on a contract (or one transaction) and leads up to more intricate forms with market places where competing players come together at the same venue to attract a larger number of clients. All players participate in the cooperation to increase their own individual benefit. Since the cooperation achieves better resource allocations and allows for specialised players, the whole market becomes more efficient and thereby is able to reward all the cooperating players. Cooperation can also take up forms that do not benefit all players, or the overall market, as for example where a group of competitors forms a cartel. In any developed economy,

cooperation is prevalent in all its mechanisms and has been researched well.[1] Here we focus on the theoretical foundation for the economic mechanisms that lead to cooperation.

Game Theory

Game theory is the subarea of microeconomics that captures strategic interaction and the incentives for players to make a choice in one or another way. By comparing the individual payoffs in a matrix for the individual players and their combination of choices one can compare the benefits of applying one or another decision strategy. A strategy in this context can be one choice a player makes, which is called a *pure strategy*, or can be a more complex construct in a repeated game that involves a series of decisions that depend on the behaviour of the other players, which is called a *mixed strategy*. Throughout this thesis, but particularly intensively in chapters 3 and 4, we will rely on microeconomics to build founded models for the scenarios we discuss.

Rational Agents

Economics assumes that the players make rational decisions and choose the strategy that yields for them the highest payoff. To clarify what economic rationality is, consider for example the payoff matrix shown in figure 2.1 for a one-off game, where player A has the choice between playing top or bottom and player B has the choice between going left or right. Depending on both players' choice each player receives his payoff, player A the first tuple entry and player B, the second one.

		Player B	
		Left	Right
Player A	Top	1, 2	0, 1
	Bottom	2, 1	1, 0

Figure 2.1: A game with a dominant strategy equilibrium.

In this example player A obtains a higher payoff from going bottom, regardless what player B does and player B reaches his higher payoff by choosing left, independent of player A's decision. This is where economic rationality comes in and dictates that each of the players will make the choice that yields the higher payoff for themselves. Any different choice would be considered irrational according to the strategic possibilities of this game. Since the players were able to reach this conclusion independent of the other's choice, we call such a clear-cut decision the *dominant strategy*. The combination of the two strategies,

[1] In order to keep this thesis concise, we decided against including a more detailed exposition of the various forms of cooperation in economics. If the reader would like to know more about the theoretical background of this area, we recommend Hal Varian's book on Microeconomics [92].

bottom/left, yield what is called the *dominant strategy equilibrium*. Note that in real life there may well be reasons for player A or B to make a different choice; however, in the case of this example we chose not to include these reasons in the model and these therefore remain external to the strategic considerations to be made. When developing our models throughout this thesis, we will assume that the players make rational decisions.

Nash-Equilibrium

Dominant strategy equilibria make for an easy strategic choice, since we require that player A's choice has to be optimal for all choices of player B. However, in many cases such an equilibrium may not exist. A less demanding equilibrium is the *Nash-Equilibrium*, where we only require that A's choice is optimal for the optimal choice of B. A Nash-Equilibrium exists for a pair of strategies if A's choice is optimal, given B's choice, and B's choice is optimal given A's choice. Consider for example the matrix in figure 2.2, which has two Nash-Equilibria: Top/Left and Bottom/Right. In this thesis we will use the Nash-Equilibrium to motivate strategies in the pricing model of chapter 3, where the players have an incentive to implicitly cooperate.

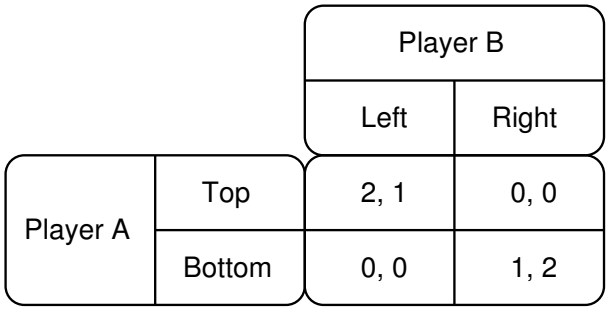

Figure 2.2: A game with a Nash-Equilibrium.

Collusion

However, there are also some fundamental problems with the Nash-Equilibrium. Consider for example the case of the well-known "Prisoner's Dilemma" - see figure 2.3 for the payoff matrix. It is a Nash-Equilibrium for both players to choose to confess, since given each other's choice it would be beneficial for the other one to confess. But this is not the optimal outcome either, both players could be better off, if both of them would choose to deny. Microeconomics describes such an undesirable situation as *pareto-inefficient*.[2] To reach this more beneficial strategic outcome, where both players choose to deny, the players would need to alter the rules of the game and coordinate in some way so that they reach an agreement that binds both of them to their strategic choice. This could be achieved by adding a penalty to the payoff matrix that punishes the player who does not uphold the agreement.

[2] A given solution is called pareto-efficient if no player can be made better off without making another player worse off.

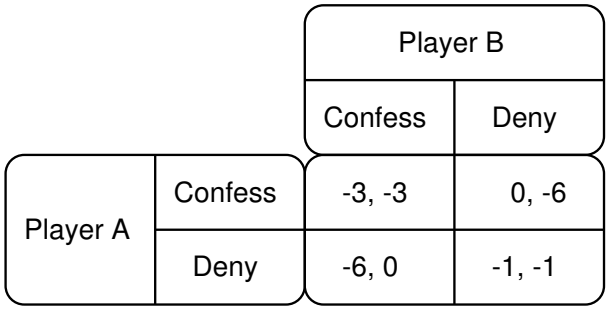

Figure 2.3: The prisoner's dilemma in a payoff matrix.

Unsurprisingly, the real business world provides sufficient examples, where businesses would benefit from such explicit coordination by means of setting up an agreement. Consider for example the case of two airlines who compete for the same flight route. If they enter a price war, they are likely to achieve less revenue than if they both would agree to set the prices at the level that they would choose if they were monopolists on this route. Monopolistic prices are determined such that they maximise the total industry profits, and neither airline could do better than splitting this monopoly profit without making the other one worse off. Such an explicitly coordinated practise of price-setting is called *collusion* and the participating firms form a *cartel*. Since such price fixing practise counteracts the benefits of competition, it tends to draw attention of law enforcement authorities in most developed economies.

Contrary to collusion, the implicit cooperation that we obtain from a Nash-Equilibrium is not illegal. As we have seen, it also is less effective in terms of profit maximisation than collusion, and therefore does not stifle competition. Some markets where competition is more fragile receive regulatory attention to safeguard the behaviour of the firms (i.e. telecommunications and electricity). The price model we will develop in chapter 3 does not assume any collusion or punishment strategies. However, in our analysis of the theoretical models, in chapters 4, 6 and 7, we will build threat models that consider "defection" as a potential strategy and evaluate if such a strategy can break down our model equilibrium.

Dynamics in Sequential Games

If games are played repeatedly, many times over, the players can use mixed strategies. In a mixed strategy, a player makes a choice with a certain probability, and since both players can choose the probabilities they assign for this decision, we can obtain a range of new payoff matrices. With mixed strategies it is always possible to find a pareto efficient outcome.

With repeated games, we also can come up with a satisfactory solution to the prisoner's dilemma. The most successful strategy in a repeated prisoner's dilemmas appears to be a very simple one that is called "tit-for-tat". With this strategy, the player initially cooperates and chooses to "deny". Subsequently, on every round you mirror the behaviour of the opponent player's behaviour of the previous round. So, if the other player has chosen to "confess", then we will punish him in the next round by doing the same. And if the

other player changes his mind to cooperate again, this strategy is forgiving and we will cooperate in the next round again.

Another variant of playing games are sequential games. In a sequential game, player A makes his decision first, player B can see the result and then make his own choice. Consider for example the case where a customer brings his car for repair to a garage and for the time of the repair gets a rental car. Mid way through the repair the mechanics discover that an additional part needs replacing. The garage phones up the customer and informs him about this need and quotes a price for this additional replacement. At this point the garage can choose between charging exactly the cost of the replacement, or this plus an additional amount of extortion. The client knows that the garage is quoting an inflated price, but now he has to make a choice. He can decline the extra repair and pay the garage for the original task only. In that case he would have to go into the effort of finding a new garage and then hire again a rental car for the days of this second replacement. Alternatively, he can simply pay the extra bit of extortion that will amount to less in money and hassle than the rental car and arranging with an alternate garage. The rational choice in this game would be to agree to the additional replacement with the original garage and to keep in mind not to return to this garage for future services. This is also the solution to this kind of problem - reputations - are an incentive for a garage not to make the choice of charging the extortion in the first place.

In the case of the garage, it may be sufficient if every customer acquires his own experiences and shares his knowledge only locally, such as the neighbours, since the garage and the customer are likely to be confined to a local area that depends on repeat customers. This is not the case for many other situations where business relationships are more transient. In such situations, it is helpful to aggregate reputation information through services, such as a better business bureau. However, when one seeks to collect the experiences that individual customers report from their business interactions, one has to be aware of customers who might have other incentives and strategies than to altruistically supply a correct account of their observations. Therefore, we seek to develop in chapter 5 reputation aggregation model that is resilient towards individual customers' rating agendas.

2.2 Resource Pricing

In this section, we survey the range of research on pricing of resources, which has been often recognised in connection with QoS admission systems. Further, a large body of work on various forms of E-Commerce transactions exists.

The classic work on resource pricing is *Spawn* [94], which explores issues of using currency to achieve fairness, priority, distribution and scalability. Contrary to most of the other research, this work uses prototypical applications instead of simulations for the evaluations. However, while Spawn is using currency mainly as a control variable and show its use as such, it does not make proper use of market forces and mechanisms.

2.2.1 Pricing of QoS Allocations

Market-based methods have been widely used for the control and management of distributed systems as a means of providing a fair allocation of priorities to clients. Previous research on QoS-network allocations has considered the effects of dynamics in bandwidth allocations. The simple bundling of network and server resources can be found in [28, 57].

Danielson [20] provides a taxonomy of admission rules for different auction schemes and used auctions for admission control and pricing for RSVP [10] connections. Further work on QoS-network allocations, effects of dynamics of bandwidth allocations and primitive bundling of network and server resources can be found in [50, 4].

Several statistical models for advance reservations for admission of networked calls are developed in [96, 29, 39]. These differ in their approach to probability distributions and assumptions on call duration.

Reiniger [77] describes a market of QoS network bandwidth containing consumers, retailers and wholesalers using cost-benefit (utility) functions to derive pricing on MPEG video streams. However, his focus is on clients adapting their demand depending on the current load-based resource price, and therefore expediting traffic without congestion.

2.2.2 Pricing, Bundling and Markets

Looking at Internet pricing, Fishburn, et al. [30] and Sairamesh, et al. [80] provide models for charging network users and enabling differential services based on the willingness to pay as well as on passing network congestion feedback on to the user of the network. The research differs from ours, as it does not model competition between resource providers; or, as in the case of Kelly [51], works on the basis of congestible connections; or, as in the case of Kuwabara [69], assumes cooperative price adjustments, which would be inhibited by competition authorities.

To compare centralised and decentralised market designs, Kirchsteiger, Niederle and Potters [53] studied public versus private exchange markets on the Internet. They concluded that many of the public exchange markets failed to obtain trading volume. Traders individually prefer to hide information from their competitors, although they might, according to auction theory, collectively profit from a public exchange market, which could facilitate collusion.

Much of the research on pricing for competitive markets has focused on sales of electronic information products using autonomous agents for negotiation [42, 52]. The area of market mechanism research also applies to negotiations and agent bidding strategies, but does not address limited capacities and time dependency.

Various uses of auctions have been explored in the context of electronic commerce systems [55, 99, 67, 3]. In addition, the hard problem of bundling auctions has been investigated [81, 73].

2.2.3 Resource Economy Systems

Various systems have been developed to centralise buying and selling CPU cycles in online peer-to-peer markets. POPCORN [76] evaluates a number of centralised auction methods for matching buyers and sellers, in contrast to our decentralised bilateral model. Challenger [14] evaluates load-balancing effects when CPU cycles are offered in a peer-to-peer system without remuneration; therefore, there are no distortions due to the exercise of market power. A practical attempt at implementing such a market system in a peer-to-peer content distribution system has been implemented, with the Mojonation [44] project. However, it did not succeed in real life, because it ran out of funding.

2.2.4 Ogino's Protocol for Competitive Bandwidth Allocations

Ogino [71] presents a protocol for mutual selection that allows for simultaneous requests and provides an example of utility functions that are adopted by users and providers to drive this selection. Our allocation protocol that we present in chapter 3 (section 3.2) is based on the one presented by Ogino. Our protocol differs inasmuch as it is not limited to bandwidth allocation, and it also is able to discriminate between users. In Ogino's protocol, users with a high valuation may not be given preference.

In the analysis, Ogino focused on bilateral interaction and developed a negotiation protocol between clients and network bandwidth providers. Users randomly make requests to providers. Providers have a fixed price schedule for each service level and only decide whether to provide bandwidth or whether to wait for more lucrative requests. Ogino shows that, if clients are price sensitive, providers who are given a lower price schedule increase their market share and profits. In contrast, we allow providers to charge prices according to market conditions; providers continuously assess demand and competitors' behaviour to determine the optimal price for their bids. The equilibrium prices stay above competitive levels.

2.3 Reputation Systems

Reputation systems have been developed for various different target applications and follow different aggregation concepts. Here we describe the relevant research, grouped by context.

2.3.1 Transfer of Endorsement

The idea to introduce reputations for raters into our reputation aggregation algorithm was inspired by page ranking calculations in search engines, such as Google, and its transitive concept of transfer of endorsement [72, 12].

Transfer of endorsement has been picked up previously by other researchers for their

development of reputation systems, most notably Chen and Singh [15]. Their kernel for the rating aggregation system involves building reputations for raters. Their system is versatile and is able to handle uni as well as bi-directional ratings and clusters ratings into groups. These groups have different levels of endorsement for the other groups' ratings and thereby the concept of the transfer of endorsement gets established, through a recursive computation that iterates over the importance each group obtains from the other groups' endorsements. The importance provides the weight of a group for their endorsements and for the compilation of resulting object scores. In order to make use of the rater reputations, the user's reputation is decomposed in a hierarchy of knowledge domains and the user's rating comments are assigned to the leaf categories, which contain objective, subjective and even text based comments. They analyse the commenting data of Ebay, Amazon and Epinions by applying their statistical reputation model and find several desirable properties. They are able to categorise raters in "Good" and "Bad" ones, and show that the good raters are more consistent with their comments, as well as being better able to finely differentiate items. Further they are able to establish a correlation between the rating experience of a rater and his resulting reputation as a rater (in the form of his reputation rank versus the other raters). In comparison with our model, which we present in chapter 5, Chen and Singh's work is using a very similar approach yet is applicable to a broader range of rating situations, since ours focuses on numeric comments only. Therefore, their results are able to draw conclusions on more general effects, such as people's view on privacy versus reliability of comments. However, due to its focus on working with an objective scale, our model and analysis (in chapters 6-8) is better able to establish a consensus on the score and ranking of a rated object. Further, it is able to visualise any clusters of agreement, where these exist, and otherwise has a parameter for balancing inclusiveness versus discrimination of rating influences.

2.3.2 Collaborative Filtering

Collaborative filtering systems are recommender systems that use similarity-based approaches to provide a statistical function which aggregates other users' recommendations on the basis of a specific user's preferences. With collaborative filtering methods it is possible to investigate similar questions as we have set forth in our analysis of such discriminatory comments aggregation.

Serjantov and Anderson [83] present a broad survey of the applicability of Social Choice theory to voting, recommender systems, collaborative filtering and other mechanisms that address the problem of how to deal with adverse minorities in open systems. They introduce voting theory, the axioms of Social Choice theory (Arrow's Theorem [7]) and discuss winner selection under these constraints and other market mechanisms. With this methodology, they criticise Dellarocas's Collaborative Filtering methods [22] and suggest a better approach. Further they evaluate research on reputation system metrics and real reputation systems (i.e. eBay) as well as research on Peer-to-Peer reputation aggregation. These observations lead them to a number of observations and conclusions about desirable properties of Peer-to-Peer preference aggregation systems. This research is a useful resource for the methodology to apply when making critical design decisions and what effects to take into account therein.

Dellarocas [22] presents a cluster filtering algorithm, which has the main aim to remove the effects of raters who appear to submit "unfair" comments. This set of mechanisms eliminates ratings that stem from "conspiracy" scenarios, such as "ballot stuffing", "bad-mouthing", seller discrimination and "flooding" with unfair ratings, such as in the case of a user who creates many fake identities in order to "rig" the rating score of a rated object. While our reputation system could be used to achieve similar effects, the strength of our system comes to play when raters supply comments on different objects, which this statistical approach would not specifically exploit. Further, a criticism voiced by Serjantov and Anderson [83] on Dellarocas's cluster filtering, is that from a viewpoint of social choice research, it is undesirable to simply ignore "outliers", since this violates the basic idea that "everyone's preferences count" and social choice research frowns upon such interpersonal comparisons in general. By contrast, the reputation aggregation function that we will develop in chapter 5, allows for gradual adjustment between dictatorial and egalitarian aggregation of each raters's comments. Additionally, in our system, the question whether a rater is considered to be an outlier, is a gradual process, proportional to the distance from the overall consensus.

Pennock, et al. [74] analyse the theoretical foundations of collaborative filtering (CF) systems under the aspect of social choice theory. They define four axiomatic properties for a CF function that Social Choice theorists have found to be desirable. In their analysis of different collaborative filtering functions they find that the first property, "universal domain", is universally accepted by all CF functions, the second, "unanimity", is common and accepted by most CF functions, the third, "independence of irrelevant alternatives", is obeyed by similarity-based CF functions, however the fourth, "scale invariance" can be obeyed by some similarity-based CF functions, if these then violate the third property in some cases. The authors propose a nearest neighbour CF function and prove that only this one can satisfy all four properties. This research approach is interesting, not only because Social Choice theory provides useful methodology to formulate desirable properties for the analysis of CF functions and recommender systems, but also aids the development of new functions and systems that satisfy the desired properties.

2.3.3 General Reputation Aggregation Systems

From a practical perspective it is important to mention eBay, since its reputation system is vital to the success of this trading platform. This centralised reputation system allows buyers and sellers to supply comments about each other after a completed transaction. Ebay's reputation system has several obvious weaknesses, for example it invites free-riding and consequently a comment is provided only half of the time, and then the comments are nearly always positive, because of the threat of retaliation with a bad rating supplied by the other transaction party. The properties have been well discussed (e.g. [79]) and Resnick finds that despite all the shortcomings, scores are predictive of future performance, and that good ratings did not let sellers boost their prices [78]. In comparison with the reputation system we present in this thesis, eBay's reputation aggregation function is very simplistic, since it only calculates average scores, regardless of the conditions in which the comments were made. Amazon employs a similar reputation system for the private resale of used books and articles.

Mui, et al. [66] decompose reputation in a context and relation dependent typology. Doing so allows them to describe different notions of reputation in terms of strategies applied by the agents. With an experimental simulation of these strategies they are able to quantitatively compare notions such as "Encounter-derived individual reputation", "Observed individual reputation", "Group-derived reputation" and "Propagated reputation". This is an interesting approach to making an quantitative comparison, where normally qualitative assessment is applied. Given the parameters for their reputation aware prisoner's dilemma game, they are able to determine which kind of reputation notion supports a particular agent strategy.

Following, we list a number of publications about reputation systems that are related to our work, though their research effort addresses different aspects from ours: XenoTrust [27, 26] is a repository platform that allows XenoServer clients to deposit trust and reputation information. Client applications then can access this information in a flexible fashion that allows the clients to specify their own query rules. This research is mainly concerned with architectural questions of deploying such a flexible platform. Cosley et al. [17] build a recommender system for ResearchIndex, a huge online digital library of computer science research papers. This database is ideal to evaluate recommenders that combine information and collaborative filtering techniques. Dellarocas and Resnick summarise the current research that was presented at a symposium on reputation systems [23] and attach a roadmap with the different directions for opportunities for future research on reputation systems. Wang et al. [95] develop an game theoretic model for agents to make strategic decisions taking reputation into account. Braynov and Sandholm [11] attempt to construct a mechanism that incites agents to reveal their true trustworthiness in terms of upholding contractual agreements. Dingledine et al. [24] describe Free Haven's approach for using reputations in order to enhance the trustworthiness of anonymous publishing on a pseudonymous distributed publishing system.

2.3.4 Peer-to-Peer Systems

Peer-to-Peer systems have been an obvious target platform for reputation systems, due to their inherent problems with freeloading and cheating agents. Much of these efforts address Peer-to-Peer specific problems that arise from peers being anonymous and transient, as well as distributed.

Of this body of research, most relevant to ours is Kamvar's EigenTrust algorithm [49]. The interesting point in this algorithm is that trust values are stored locally, but then are synchronised among a set of peers who had prior interaction (and thereby trust) through an iterative exchange process that yields global agreement on the trust values. This EigenTrust algorithm relies on transfer of endorsement to achieve rapid convergence, which is vital for the scalability of the distributed algorithm. In their analysis they show how the reputation aggregation algorithm is able to cut down the vast majority of downloads of an inauthentic file, that originates from a malicious collective that resembles up to 40% of the population. If the malicious peers do not form a collective, but apply individual strategies, then even a set of 70% malicious peers can be neutralised by EigenTrust. However, if no cost is attached to creating new identities, the system is at threat from a Sybil

attack. Compared to our reputation system that we present in chapter 5, both systems use a similar trust propagation concept, transfer of endorsement, for the aggregation of scores. Given that EigenTrust has been designed to suit a more specific application than our reputation system, it is interesting to note that when analysing both with various different threat models (see chapters 6-8), both are able to identify similar sized proportions of malicious peers.

Aberer and Despotovic present a reputation aggregation system [1], that is concerned with the problem that global trust values might originate from untrustworthy peers and therefore they multiply the complaints made against one peer by the number of complaints this peer has made. This approach creates other problems and is only able to deal with low proportion of cheating peers (up to 25%).

Gupta, et al. developed a reputation computation agent [41] that controls credits and debits for uploads and downloads in a Gnutella-like Peer-to-Peer system. Further, Damiani, et al. propose a polling algorithm [19] that allows peers to test the credibility of other peers. In so doing, a peer polls a number of other peers for their vote on a peer and then uses cluster algorithms to aggregate these votes. While they have not evaluated the maximal size of a malicious collective that this system could withstand, it is our guess that it would be significantly lower than EigenTrust's or anything else that is similar to the reputation system presented in this thesis.

2.3.5 Agent Misbehaviour in Networks

Buchegger develops in her PhD dissertation [13] a routing protocol for mobile ad-hoc networks called CONFIDANT, which copes with misbehaviour of routing nodes by means of a reputation system. The reputation system isolates misbehaving nodes and to this end feeds the reputation system with first-hand and second-hand information. In order to eliminate false information among the second-hand reputation information, CONFIDANT uses a modified Bayesian estimation and classification procedure. When nodes exchange first-hand reputation information, this information is only accepted as second-hand reputation information if it is compatible with the current reputation rating. The protocol is sophisticated enough to allow for node redemption and yet prevent sudden change in agent behaviour that attempts to exploit good reputation acquired over time. The protocol simulation demonstrates the performance and effectiveness of the reputation system in being able to ensure an operational network even with up to 50% of the network population misbehaving. This research is interesting from the perspective of our research since it uses a different approach to address the problem of untrusted reputation information. The main difference between both approaches is that CONFIDANT is calculating the reputation aggregation from the viewpoint of an individual node, where our approach is the one of a service that does not use first-hand information. Being able to use first-hand information strengthens the filtering of misinformation, but it always makes the resulting recommendations partial to the collector of the first-hand information. Our service-suitable approach therefore can claim to be impartial about the collection of the reputation data.

2.3.6 Trust Systems

Trust modelling is borne out of the traditional concept for resource access control, where a system of contracts enforces compliant client behaviour. In such a system, both parties state their requirements and constraints and then apply usage accounting to monitor if the party is complying with the contract. Compliant parties are being "trusted" and the trust systems are concerned with issues such as propagation and revocation of trust. Shand describes in his PhD thesis [84] such a system of contracts and trust for resource allocations (as well as all the relevant research).

Ideally, trust/contract systems and reputation systems should complement each other. A system of contracts with a basic notion about authentication and settling of transaction terms is a prerequisite for market-based resource allocations. The allocation system that we present in chapter 3, including its syntactical details in appendix B, involves settling of contractual terms, even though this functionality could be serviced through a separate contract settling protocol. Further, both the pricing as well as the reputation system assume that parties are being securely authenticated. However, the purpose of trust and reputations differ conceptionally. Trust concerns itself with establishing compliant behaviour, and ultimately aims at a the boolean question whether trust is satisfied. In contrast, reputations capture all the shades of grey within the full range of possible compliant behaviours. One implication arising from this difference is that, since trust is transitive, trust is more difficult to transfer and aggregate, and therefore reputation systems are more suited for such operations.

2.4 Conclusions

In this chapter we surveyed the research context for the pricing of allocation systems and reputation systems. For the context of our reputation aggregation model, which we will introduce in chapter 5, we were able to describe and criticise several research efforts that relate to, but differ from our model. The target application makes for the biggest difference between the architectures of these reputation systems, and ours will take a different approach in terms of its definition of the notions of reputation, comments and agreement in the model. Some of these prior reputation systems have demonstrated high levels of effectiveness, and we will aim to reach similar levels of success with our model, in the threat scenario analysis, in chapter 7.

The prior work that relates to our pricing model is more dispersed, since all of these papers address different questions from ours, namely the implications of perfect competition on online resource selling. Some of this research resembles system building efforts that use price as a unified allocation signal, although they do not use it in an market economic sense. Other research efforts use economic theory, especially game theory, in an intricate way, but then, in order to solve too large of a problem, make market modelling assumptions that are unlikely to be accepted by actual markets. In general, it appears to be difficult to unify economic and computer scientific modelling approaches with a level of rigour that does justice to both disciplines equally. In order to demonstrate a contribu-

tion to the one or the other discipline, it becomes all too easy to cut corners in the other discipline, i.e. building an auction trading system with certain economic properties, when actually the markets for the targeted goods are unlikely to entertain at all an auction for these goods. It is our aim to improve on the interdisciplinary aspect of economics in computing systems with the pricing model presented in the next chapter.

Chapter 3

The Supply Function Pricing Model

In this chapter we develop a pricing model for the dynamic allocation of electronic services. In its introduction (section 3.1) we motivate this allocation situation with the example of network communication resources, but this model is actually able to address any kind of non-storable service, such as computer processing time.

This pricing model assumes that practically all the time there is more supply of resources than demand. In such a scenario, if one considers the competition to be perfect, traditional economic theory suggests that prices will drop to zero.[1] Such a result would be very undesirable, because it would imply that over time providers would continue to go out of business until sufficient time periods of supply shortfall yield sufficient profits. Depending on the variance of the market demand and the economic loss from unsatisfied demand (which may incur external effects), such a result may be economically disastrous for some markets. In the long term, such a situation drives competitors out of the market and leads to market concentration. And even in the short term, in a market where suppliers are unable to recover their costs, we obtain a market that lacks further investment and innovation. To prevent such a result, the providers need collectively to withhold some of their supply, such that prices do not fall to zero for too much of the time. If, due to a lack of cooperation, only a minority of providers would withhold supply, these providers would stand to lose out in the competition, and the elevation of market prices would be insignificant.

In order to be able collectively to withhold supply, the providers need to establish some form of cooperation. A naive approach could be to form a cartel with explicit exchange of signals to form and enforce this cooperation. However, anything of this kind would be illegal in most places. Moreover, if a market gets too concentrated, it can even be brought under government regulation and investigators would then even look out for hidden cooperation signals, such as price patterns that are used for dynamic price strategy games.

[1] In the case of the long-distance telephony market one can observe such a situation practically. Prices for phone calls all the way around the globe to rural parts of Australia or Mongolia are now the same as for a local call down the street. This is in part due to the "fiber-glut", that was funded by the Internet stock market bubble of the late '90-ies, and has lead to an abundance of excess communication capacity.

One possible solution to facilitate this cooperation would be to form a centralised market exchange, which still obeys legal and regulatory limits, such as it is done in electricity markets. Practically, however, one can neglect centralised markets as an option for such electronically accessible services, since the centralised exchange would require all participants to agree on standardised contracts. Contrary to bilateral negotiation, standardised contracts would not allow sufficient flexibility for both parties to exploit a large variety of innovative resource utilisation optimisations, which makes for a more efficient economy.

Out of this consideration, we develop in section 3.2 a bilateral allocation protocol as a base for the pricing model. In this allocation protocol, the clients send their requests for resources to all possible providers, collect the replies and settle the transaction with the provider whose offer they choose. In appendix B we present a realistic environment for clients and providers, including potential applications, resources, protocol platform technology and an allocation language. In appendix A we further describe the technicalities and performance aspects of the discrete event simulation that we have built to implement the protocol and pricing model.

In order to solve the need for cooperation in our bilateral pricing model and under these conditions without collusion, we use economic game theory. In so doing, we seek to form an economic pricing model that allows for a Nash-Equilibrium (NE). This NE is reachable if all providers choose a strategy whereby they withhold a minor part of their supply. Therefore, in section 3.3 we develop a supply-function based pricing model, where each provider simulates the market situation and updates the simulation continuously with information collected from (1) their own bilateral trades and (2) the requests for price quotes they receive from clients. Since such a pricing model does not even involve observation of competitors' prices, it would be considered legal.

3.1 Introduction

Future QoS networking will not only provide services with assured quality and charges for these services, but will also allow clients to choose 'just in time' between different network providers. Users benefit from renting variable amounts of a dedicated resource to satisfy their varying demand requirements. For example software by the company 'Invisible Hand' enables web hosting applications to allocate output bandwidth through flexible reservation and dynamic pricing from their web hosting provider.[2] If, in future a hosting application is able to obtain offers for services from more than one provider[3], we will observe continuous competition between providers for consumers, like the web hosting applications. In periods of high demand, providers can increase prices until demand matches supply. In times of low demand, providers' aggregate capacity exceeds demand and providers will under-cut each others' offers until prices drop to marginal costs, which are zero. Such a business model apparently results in long periods of low revenues, which

[2]This software is employed by the web hosting provider Streaming Hand, serving about 40 commercial web sites.

[3]In this example, this is the case if a customer employs two web hosting companies such as StreamingHand, and balances the services between both, to his economic benefit.

do not allow cost recovery, followed by times of insufficient supply and high prices. If these prices are not passed on to final consumers, the prices will not allow for efficient rationing, and shortages can result in undesirable unavailability of networks.

We ask whether it would be possible for resource providers to cover a larger share of their fixed costs even in times of low demand. Is there a mechanism that would result in positive prices if aggregate supply exceeded demand? To answer this question, we anticipate bilateral trading and provide a general protocol which facilitates the corresponding negotiations. We model the prices offered by resource providers, given their market power, and run a simulation to verify whether we found equilibrium. We do not allow for collusive agreements, which are maintained by threatening low prices to parties who deviate from the agreement.

Klemperer and Mayer [54] proposed to model market power in commodity markets with supply functions. In the supply function model, market participants submit price-quantity functions to a central auctioneer who then determines the market-clearing price. In a supply-function equilibrium, no market participant would prefer to deviate and submit another function. This approach is traditionally applied to centralised auctions, and would have been appropriate to model initial B2B sites which wanted to provide such auction places. However, "only few public exchanges are remaining ... and action in B2B turned elsewhere" (Economist 17.5.2001), because companies seem to prefer bilateral trade, which allows for price discrimination and which facilitates customer-specific product offerings.

We therefore assume that computing resources will be allocated in bilateral negotiations, and we extend our modelling approach from a central auctioneer to decentralised trading. Decentralised trading allows for continuous interaction and therefore lacks the commitment device previously set by the auctioneer. Therefore, providers can sell capacity at random times, and neither buyers nor competitors are ever certain whether the provider might not sell additional or cheaper capacity at a later stage. Allaz and Vila [2] showed that repeated selling opportunities eliminate market power in a Cournot market and result in a competitive outcome.

Our model differs in that demand is stochastic, both in total number of interested users and in their individual marginal value; supply is capacity constrained; and bilateral trade restricts information flows. A further difference from most other markets is that negotiations can be automated and therefore the costs of searching for the best offer are eliminated. We extend a communication protocol by Ogino [71] for automated resource reservation and implement it in a simulation. The protocol builds on existing support infrastructure and standards under development, including a directory of services, service descriptions and service interactions. It defines the message exchanges between providers and clients wanting to reserve resource units. The protocol is simulated with several providers offering resources and clients seeking access to these. Clients make requests to all providers, specifying the time of their usage and desired parameters, such as performance or the Quality of Service level. In order to allow clients to compare offers of different providers, it is not necessary for all providers employ the same negotiation protocol, as long as it also contains the message parts of our protocol. The protocol is simple enough to be an inclusive model for bilateral negotiation. On receiving the request, a provider calculates whether his capacity suffices to satisfy the request and decides on the price at

which to offer the resource.

Contracts and prices are confidential. Therefore, resource providers only know the total number of quotes requested from them and the number of times a user has booked their resource. They use this information to continuously update their beliefs about demand. Based on these beliefs, the resource providers estimate prices which would evolve in a supply function equilibrium, and would offer this price as a quote to consumers. In the absence of any publicly available price information, we will refer to this price as estimated market-clearing price.

We use the simulation to assess whether it is profitable for a resource provider to deviate from this strategy and to offer a price below the estimated market-clearing price. The resource provider can capture a larger market share by offering slightly lower prices. Fellow resource providers make fewer sales, correct their demand expectation downward and subsequently make cheaper price quotes. Therefore, the deviating provider receives less revenue per reservation, although he obtains more reservations. The net effect must be simulated by comparing the revenue achieved from selling more slots at a lower price with the revenue obtained when no provider deviates. In the analysis of the next chapter, we ask under what circumstances the increased market share compensates for lower prices. We only assess simple deviation strategies and show that, for some combinations of the number of users, number of resource providers and the capacity per resource provider, deviation is not profitable. For the remaining parameter combinations, deviation can be made unprofitable. Resource providers anticipate the possibility of deviation and therefore put more weight on their observation of their own, successful, reservations when updating the supply function for quoting the next price.

The analysis suggests that the supply-function equilibrium can be transferred to a scenario with bilateral trade and no search costs, by using all available information to continuously observe the market.

3.2 The Allocation Protocol

Here we describe the message sequence at a transactional level for resource allocations between client applications and resource providers. In appendix B we give more details about the scenario of this protocol, potential client applications, the kind of resources on offer and how it can be implemented with the use of Web services technology and an allocation description language. This protocol is a modification of an existing protocol that works well for competition between resource providers, and is good for load balancing (see section 2.2). But here we have an open environment with potential contention between clients. Therefore, this protocol is extended to enable client competition for contended resources. Figure 3.1 shows the message sequence of the protocol.

1. The client application sends a service-request message to the list of possible resource providers including the attributes of the resource usage (type of service (e.g. Linux V6.2), estimated time for the contract, QoS-level, etc.). Also, with the request, the client sends his maximum valuation for the contract.

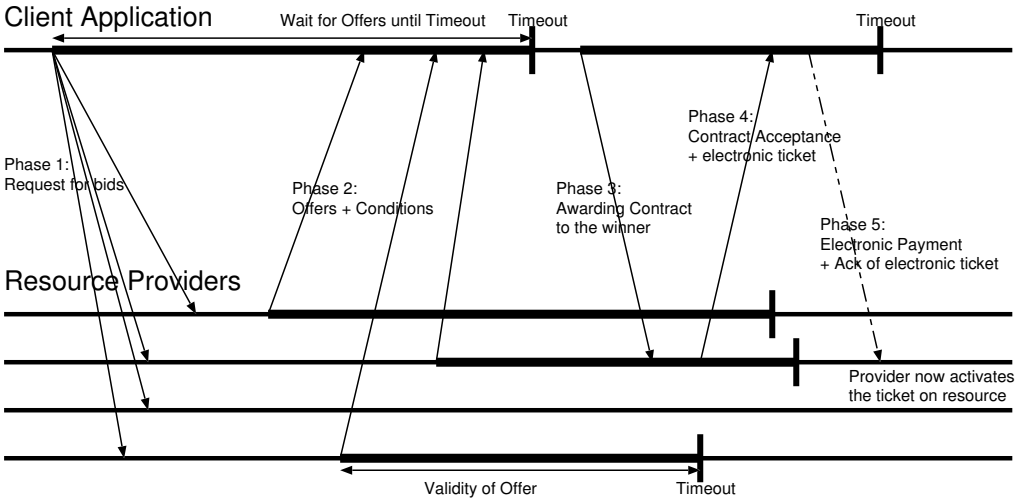

Figure 3.1: The Communication Protocol

2. The resource provider calculates his cost for performing the requested service. The provider can ask any price he wants, but will try to keep his price competitively low, since the user will select the cheapest satisfying offer. Along with the calculated price, the provider sends a bid message to the user including information on additional options, i.e. service-level, duration and volume.

3. The client collects the bids he receives and calculates correspondingly how valuable each bid is for him. Sometimes a cheap offer may not be performing well enough or be too far away. The client selects the resource provider who is maximising this utility and sends an award message to the winning resource provider.

4. The provider generates an electronic ticket and sends it to the client, since he will need it to gain access to the resource.

5. The client sends the electronic payment to the resource provider, acknowledging the receipt of the electronic ticket. The resource provider then activates the issued ticket and sends it to the actual resource. In case of a dispute, the client can verify the receipt of this last acknowledgement by checking whether the resource provider has accepted the electronic payment.

In an effort to discriminate between clients, the broker determines the price for a resource using the current usage of the resource and the applications' willingness to pay (their maximal valuations). In an ideal situation an auction is expected to yield a fair allocation. However, for immediate allocations, the broker cannot run a real auction, rather he has to determine the price empirically, based on the last awarded contracts and the state of system resources. But for advance reservations, in cases where the waiting time between allocation decisions is long enough to collect a larger number of requests, the resource providers could have the opportunity to launch an auction.

Bilateral negotiation protocols have a practical advantage over centralised designs. A centralised market mechanism, like an auction, would require the market participants to agree on essential service parameters, where bilateral negotiation can be individualised.

Further, as pointed out in the previous chapter (section 2.2), economic comparison of public versus private exchanges [53] indicates the preference of sellers for private exchanges.

3.3 Economic Framework

Investment in computer resources depends on the expected market price for such resources. In a long-term equilibrium, the average price equals total cost of providing the service. Higher prices result in capacity expansion or entry of additional providers to offer the service, whilst lower prices would result in reduced investment. In the short term, prices can differ from total cost, as telecommunication bandwidth providers discovered in 2002 when low revenues prevented recovery of fixed costs. Investment costs in the network are irreversible and therefore sunk when short term sales are made. The objective in short term sales decisions is therefore to maximise revenue, independent of initial investment cost. We simulate prices which would evolve in short-term markets. Collusion could enforce higher prices by punishing deviations in subsequent periods, but is illegal and not part of the model.

In a perfectly competitive world which is characterised by a large number of resource providers and supply exceeding demand, providers make their offer at marginal cost which, for electronic services, is close to $\emptyset$. If one of hundreds of resource providers withheld capacity, it would have no effect on the market price and therefore would not increase profits on his remaining capacity. If demand exceeds total capacity, the price will rise until demand and capacity once again match each other. The resulting allocation is efficient because consumers who are willing to pay the most obtain the resources, and resources are only unused if no further demand exists, even at price zero.

However, if the number of resource providers is small, they can exercise market power and profitably maintain prices above competitive levels by restricting output below competitive levels. Typically, Cournot-Nash models are used to model market power. They define an equilibrium by a set of output choices of all generators such that, given other providers' output, each generator chooses its own profit-maximising output. In the continuous bilateral negotiations model, the assumption that providers take the output of other providers as fixed is no longer appropriate. Therefore, we model the strategy space of providers based on a supply function model introduced by Klemperer and Mayer [54] and subsequently applied to the electricity market by Green and Newbery [38]. Supply function equilibria are based on a setting of producers bidding a supply function into a wholesale market, which specifies how much output the producer is willing to provide at any given price. The auctioneer aggregates all supply functions and matches demand with supply, thereby determining the market-clearing price as well as the output of all providers. Klemperer showed that market equilibria exist if providers specify their supply function to maximise profits, whilst taking other providers' supply functions as fixed. These traditional supply function equilibria require that providers commit to the supply functions they have submitted. Decentralised reservation mechanisms do not have such a commitment device. Providers continuously receive requests for resources and can deviate from a suggested price at any time. Instead of the commitment achieved through

a centralised auction place in a traditional supply function equilibrium, we introduce continuous updating about market demand. If a provider reduces prices below the price suggested by the supply function, other providers receive fewer contracts. When updating their beliefs the other providers assume lower overall demand and respond by reducing their price. Total price level falls and profits of all providers are reduced.

We define a supply function equilibrium in a bilateral trade setting by two criteria. First, given competitors' supply functions and assuming that competitors bid according to their supply function, the provider's supply function is profit-maximising. Second, given the supply functions and the mechanisms which providers use to update their beliefs about demand, it is most profitable for a provider to bid according to his supply function. Therefore, we must analyse whether it suffices to update beliefs about total market demand in order to prevent deviations. The analysis of the next chapter will show that, for small numbers of providers and higher demand, faithful updating suffices to make deviations unprofitable. For other parameter choices, providers must be suspicious and put excessive weight on the number of reservations they count, in order to ensure deviations from the supply function equilibrium are unprofitable.

We first present the model and assumptions about the demand and, in a second step, apply the supply function equilibrium to the model. In a third step, the updating process is presented.

3.4 Clients

We assume a point market of N symmetric clients.[4] Demand is stochastic, with two random components for the request probability and a maximum price the client is willing to pay. The reservation time on a resource is divided into uniform size time slots. One time slot is associated with a fixed size share of the total resource capacity a provider can supply.

Clients request computational resources with probability R. In equilibrium, prices does not change over the bidding period; therefore, the time when clients ask for quotes for resources is independent of the strategies of resource providers and, without loss of generality, can be assumed to be uniformly distributed on an interval $[0, T]$.[5] The reservation price of any client requesting computational resources is uniformly distributed between 0 and P. Therefore, demand expected at the beginning of the bidding period at price p is $N \cdot R \cdot (1 - \frac{p}{P})$ with $p \in [0, P]$.

[4] Such a perspective is appropriate for computational applications. Effects of transmission + processing latencies and network locality turn out to be negligible with the reservation granularity simulated in our model.

[5] Any distribution of requests on a closed interval can be brought in such a shape by a strictly monotonic transformation.

3.5 Strategy of Resource Providers

All M symmetric resource providers develop a strategy $q(p)$ which determines the capacity q they provide at a given market price p. Each resource provider subsequently calculates the estimated market clearing price, balancing demand and supply, based on his own strategy, the strategy he believes other resource providers follow and his beliefs about demand parameter r.

A resource provider chooses his strategy so as to maximise expected profits. Profits equal the product of price and capacity sold. Capacity sold equals total demand at price p, minus the capacity provided by the $M-1$ other resource providers at that price. The short term profit function contains no cost term, because we assume that all costs are fixed in the short term, irrespective of utilisation level:

$$\max_p E_r \left[\left(N\, r\, (1 - \frac{p}{P}) - (M-1)\, q(p) \right) p \right]. \tag{3.1}$$

When determining their supply function, resource providers assume that demand function and supply function $q(p)$ of other resource providers are given. Demand decreases with price and supply functions increase in price; therefore, a one-to-one mapping exists between price p and the residual capacity to be provided. We differentiate equation (3.1) with respect to p in order to determine the optimal price-quantity pair for a given demand parameter r:

$$N\, r\, (1 - \frac{2p}{P}) - (M-1)\, \frac{d\, q(p)}{d\, p}\, p - M\, q(p) + q(p) = 0 \tag{3.2}$$

Assuming all M suppliers are symmetric in equilibrium, each will produce the same share of total demand. This gives us the market clearing condition, where the index e in p_e denotes that this condition represents equilibrium situations:

$$M\, q(p_e) = N\, r\, (1 - \frac{p_e}{P}). \tag{3.3}$$

Now we apply the market clearing condition (3.3) to the differentiated profit function from equation (3.2). Since the market clearing condition relies on an equilibrium situation, we cannot apply it to the the profit function (3.1) directly, because the first difference of the demand function $N\, r\, (1 - \frac{p}{P})$ is different from the first difference of the supply function of individual providers $q(p)$. Therefore we apply this substitution of the market clearing condition (3.3) to the differentiated profit function, which represents an equilibrium in the form that it is represented by equation (3.2):

$$\left(-N \frac{r}{P} - (M-1)\, \frac{d\, q(p_e)}{d\, p_e} \right) p_e + q(p_e) = 0. \tag{3.4}$$

The general solution for $q(p)$ to satisfy equation (3.4) is:

$$q(p_e) = A\, p_e^{\frac{1}{M-1}} - N\, \frac{r}{P\,(M-2)}\, p_e. \tag{3.5}$$

For analytic simplicity, we assume the number of providers is $M \geq 3$. To determine the free parameter A in this supply function, we follow Klemperer. He showed that the supply schedule $q(p)$ must be, for all feasible prices, between the competitive schedule as a lower bound, and between the Cournot oligopoly schedule as an upper bound. Providers can choose any schedule within this range. Subsequent deviations will not be profitable. Therefore, we assume that the providers choose the most profitable schedule. This is the schedule which crosses the Cournot schedule for individual providers' output at the capacity K of each provider. Such a Cournot schedule for each of the n providers is given by maximising the profit function of any provider who takes the output q not, as previously, the supply function $q(p)$, of other providers as given:

$$\max_p E_r \left[\left(N\, r\, (1 - \frac{p}{P}) - (M-1)\, q \right) p \right]. \tag{3.6}$$

The first order condition with respect to p is $N\, r - (M-1)\, q - \frac{2\, N\, r}{P}\, p = 0$.

We combine the first order condition with the market clearing condition (3.3) and solve for the case when the output of each resource provider equals capacity $q = K$ and obtain the following equilibrium values:

$$p = \frac{P}{M+1}, \qquad q = K, \qquad r = K\, \frac{M+1}{N}. \tag{3.7}$$

Supply function and Cournot output are the same at this point; therefore, we can substitute p, q and r from equation (3.7) into supply function (3.5) to determine A:

$$A = K\, \frac{M-1}{M-2} \left(\frac{M+1}{P} \right)^{\frac{1}{M-1}}. \tag{3.8}$$

With a given A, the supply function (3.5) is complete, and market participants can use it to solve for the equilibrium price $p \in [0; P]$, where demand equals supply $M\, q(p) = N\, r(1 - \frac{p}{P})$, based on their estimation of r:

$$M\, A\, p^{\frac{1}{M-1}} - \frac{2\, N\, r}{P\,(M-2)}\, p - N\, r = 0. \tag{3.9}$$

We could not obtain a general analytic solution for p in equation (3.9). The left-hand side is negative for $p = 0$, steadily increasing for $p > 0$, and is positive for p towards P. Therefore, there exists a unique solution for $p > 0$, which we solve numerically.

3.6 Updating

Providers use three types of information to update beliefs about r at any time $t \in [0, T]$ in the bidding period. Two types are identical for all providers, namely the externally-known probability R and the number of requests for quotes sent out by clients. Each provider, furthermore, privately counts how many of his offers have been accepted by time t.

All three information types are weighted with the inverse of their variance to form the updated r. To simplify the calculation, the calculation of variances is based on R, not on r. To deter deviation, additional weight can be put on the number of successfully booked resources. This is implemented by scaling down the variance of resources booked.

3.6.1 Initial Request Probability

The expectation of r deduced from R is:

$$E(r|R) = R. \tag{3.10}$$

The variance of r deduced from R is given by the binomial distribution of N independent clients requesting a quote with probability R. The variance of the number of clients requesting bids equals $N\,R\,(1-R)$, creating a variance for $\sigma^2(N\,r) = \sigma^2\Big(\sum_i R_i\Big) = \sigma^2(R)\,N$ or:

$$\sigma^2(r) = \frac{\sigma^2(R)}{N} = R\,\frac{1-R}{N}. \tag{3.11}$$

3.6.2 Received Quotes

The second approach to estimating r is based on the number of requests for quotes received until time t. As the probability that clients request a quote is constant over the bidding period, a linear extrapolation is unbiased.

$$E\Big(r|\#Bid_{obsv}, t\Big) = \frac{\#Bid_{obsv}}{N}\,\frac{T}{t}. \tag{3.12}$$

The variance of r is due to events in the interval $[t, T]$. Clients will ask for quotes with reduced probability $R\,\frac{T-t}{T}$. Therefore, the variance due to the binomial distribution is $\sigma^2(r'|t) = \frac{1}{N}\,R\frac{T-t}{T}\,(1 - R\,\frac{T-t}{T})$.[6]

A second contribution to the difference between extrapolated r and final r is caused by $\#Bid_{obsv}$ being itself a random variable. The expected variance of $\#Bid_{obsv}$ based on

[6]For simplicity, we assume that N is great enough that the correlation between realised events is $[0, t]$ and potential events in $[t, T]$ can be ignored.

R is given by $\sigma^2\Big(\#Bid_{obsv}|t\Big) = R\,\frac{t}{T}\,\frac{1-R\,\frac{t}{T}}{N}$. The variance of the extrapolated r is $\left(\frac{T-t}{t}\right)^2$ times this value. Adding both of the independent effects gives:

$$\sigma^2\Big(r|t,\#Bid_{obsv}\Big) = R\,\frac{T-t}{T}\,\frac{1-R\,\frac{T-t}{T}}{N} + R\,\frac{t}{T}\,\frac{1-R\,\frac{t}{T}}{N}\left(\frac{T-t}{t}\right)^2. \tag{3.13}$$

3.6.3 Accepted Offers

A third source of information to update beliefs about demand parameter r is the number of offers which has been accepted by clients until time t. The linear demand function allows us to apply the theorem of intersecting lines (Figure 3.2). The proportion of offers accepted $\#Bid_{acpt}$ to the total number of quotes requested is equal to the proportion of δp to P, where δp is the difference between P and the average offer price $p_{av} = \frac{\sum_{bid} p}{\#Bid_{obsv}}$ charged during the bidding period. All resource providers believe that fellow resource providers have offered the same price and have sold the same number of slots at time t.

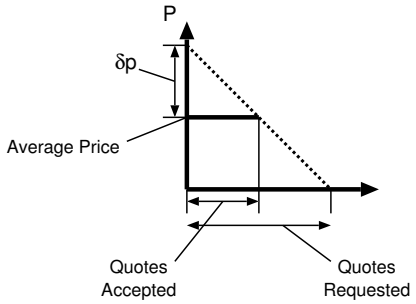

Figure 3.2: Relationship between average price and demand

The r following from this calculation is:

$$E(r|\#Bid_{acpt},p_{av}) = \frac{M\cdot\#Bid_{acpt}\cdot P}{P-p_{av}}\,\frac{T}{t\,N}. \tag{3.14}$$

The variance of r is mainly caused by the volatile number of bids accepted by clients $\#Bid_{acpt}$. According to the binomial distribution, the variance of the number of accepted bids is: $\sigma^2(\#Bid_{acpt}) = N\left(\frac{P-p_{av}}{P}\right)\frac{t}{MT}\left(1-\left(\frac{P-p_{av}}{P}\right)\frac{t}{M\cdot T}\right)$. This gives for the variance of r:[7]

$$\sigma_r^2(\#Bid_{acpt}) \sim \frac{\frac{P}{P-p_{av}}\,M\,\frac{T}{t}-1}{N}. \tag{3.15}$$

The updating of the beliefs that providers maintain to calculate their supply-function market prices concludes the pricing model.

[7]If deviation were profitable, providers would put additional weight on the number of observed offers, which we represent in the simulation by reducing σ_r by a constant factor $w : \sigma_r \to \sigma_r/w$.

3.7 Conclusions

In this chapter we were able to develop a pricing model for allocations in the e-services market, based on a protocol that arranges for competition among providers as well as clients, and under the assumption of supply exceeding demand. Neither do we require collusion based on punishment strategies, nor do we use prices in other periods with higher resource demand to maintain prices above competitive levels. The model calculates an equilibrium for the supply-functions and in so doing generates prices above zero level. The equilibrium assumes that all providers choose the same strategy and relies on continuous updates of the model beliefs to adjust this equilibrium to the current market demand.

The interesting question at this point is whether the providers are able to gain sufficient information from the trading to actually reap the supply-function model's benefits? Further, what happens if not all providers choose to apply this supply-function strategy? If the supply-function strategy is supposed to lead to a Nash-Equilibrium, then is it really impossible for a provider to adopt a different strategy and profit from doing so? Is deviation able to break the cooperation? We will examine these questions in the next chapter.

Chapter 4

Effectiveness and Resilience of the Pricing Model

Having developed in the previous chapter a pricing model, which relies on all the providers to cooperate by choosing the same strategy in order to find a Nash-Equilibrium, we need to analyse whether the information the providers are able to collect is sufficient to adjust to the market situation, and if this kind of cooperation does allow for prices above zero-level. In this chapter, in section 4.1, we show the results of a simulation of the pricing model in the bilateral negotiation protocol and demonstrate that the equilibrium prices are able to stay above perfectly competitive levels.

Beyond achieving the equilibrium prices, the main challenge for this Nash-Equilibrium-based pricing model comes from providers who do not join in this cooperation that relies on all of them choosing the same strategy. Such providers could simply continue to follow a perfect competition style pricing scheme, or even more effectively, assume that all the other providers will follow the same Nash-Equilibrium's pricing model and then deviate from the cooperation by undercutting the projected prices marginally. The strongest challenge of this kind is, if there is only one provider attempting to beat the pricing model and profits from doing so. However, if the supply-function model is working correctly in accordance with its theory, then the deviator should not stand to gain from his behaviour. If the deviator can make more profit with his strategy than with the supply function model, then we do not have a Nash-Equilibrium. Our supply-function model allows us to increase a weight on how strongly our model responds to any kind of deviation, this is the *Weight on the Undercutting Sensitivity*. Adjusting this weight to the potential threat level allows us to balance the model's vigilance with its efficiency in times without deviators present. In section 4.2 we will show, that the deviator makes less profits with his strategy, than if he were to follow the supply-function equilibrium strategy, even if he is able to outsell his peers and earns more revenue than them at this moment.

In section 4.2.2, we analyse a range of market conditions, by comparing the necessary level for the weight on the undercutting sensitivity, to achieve the required Nash-Equilibrium criteria. With such a map of market conditions, we assess whether the pricing model is able to thwart a deviator's challenge in all situations, or if there are limits to the market conditions it is able to work with.

4.1 Competition With Symmetric Pricing Functions

We start our analysis by evaluating the prices that the supply function model yields under various market conditions, if all the providers apply the same strategy and adhere to the model developed in the previous chapter. Since an important element of our model is the updating of the market assumptions (see section 3.6), we show a series of trading periods over time and show how the model responds to random variations of demand timing and volume. Further, we plot the relative gain in elevated prices depending on different combinations of market supply and demand situations.

M servers offer the resource and have ten identical slots to satisfy client requests. N clients have a random valuation between $\emptyset$ and 300 for each of these slots. We abstract from intertemporal dependencies and only analyse the allocation for one time slot. Users ask for price quotes at randomly chosen points during the reservation period. Scaling to any other period or to a non-uniform distribution of expected bid probability would not influence the results. Users ask for price quotes at randomly chosen points during the reservation period. We assume any client is interested in the resource with probability $R = 0.5$.

4.1.1 Trading over Example Periods

Figure 4.1 shows the prices quoted by three providers A, B and C over ten independent bidding periods. Figure 4.2 is a closer look at figure 4.1, showing the development of prices over the last bidding period. Clients make requests at random times, prompting the provider to reply immediately with a quote based on the conditions at the current time. Providers continuously update their assumptions about the realised demand r (as described in Section 3.6); therefore, quoted prices fluctuate between high and low areas within a bidding period, as can be seen in figures 4.1 and 4.2. The clients collect all replies and select the cheapest offer.

Each of the three servers makes a sale about every third time. These round robin sales are due to the updating of the number of accepted offers by this server, as explained in section 3.6.3. A server keeps lowering its prices if its offers are not accepted. However, a close observation of figure 4.2 reveals that, in the period of $[33100 - 33300]s$, provider C wins all three sales. This is possible because C lost the first two sales (by chance, as all start equally), but then has the lowest offer with his number of accepted offers $\#Bid_{acpt} = \emptyset$ (equation 3.14). Subsequent offers are lower, because his average offer price $p_{av} = \frac{\sum_{bid} p}{\#Bid_{obsv}}$ is lower than his competitors' p_{av}.

4.1.2 Scale of Competition

Now we will evaluate how prices are influenced by the number of competitors. Figure 4.3 displays the average price quoted by providers for different numbers of competing providers dependent on demand, as given by the number of clients per provider. The market-clearing price under perfect competition would be $\emptyset$ for fewer than 20 clients per

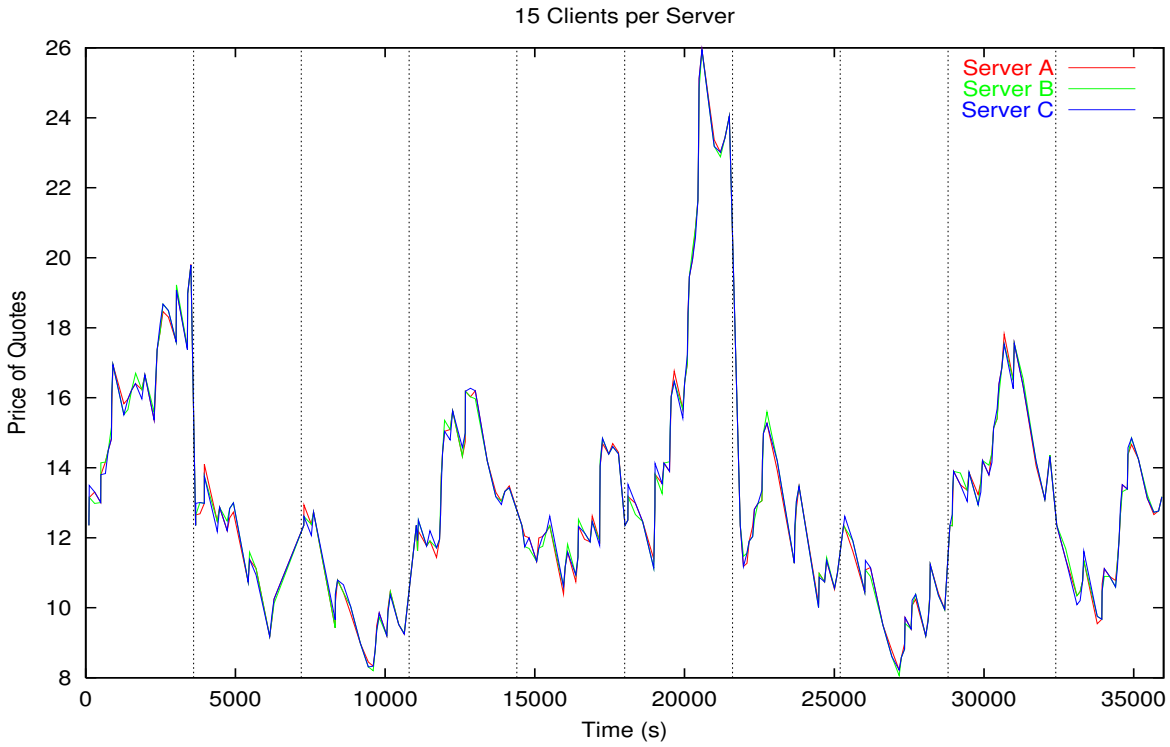

Figure 4.1: Price quotes of three providers and 15 clients per provider over one example period.

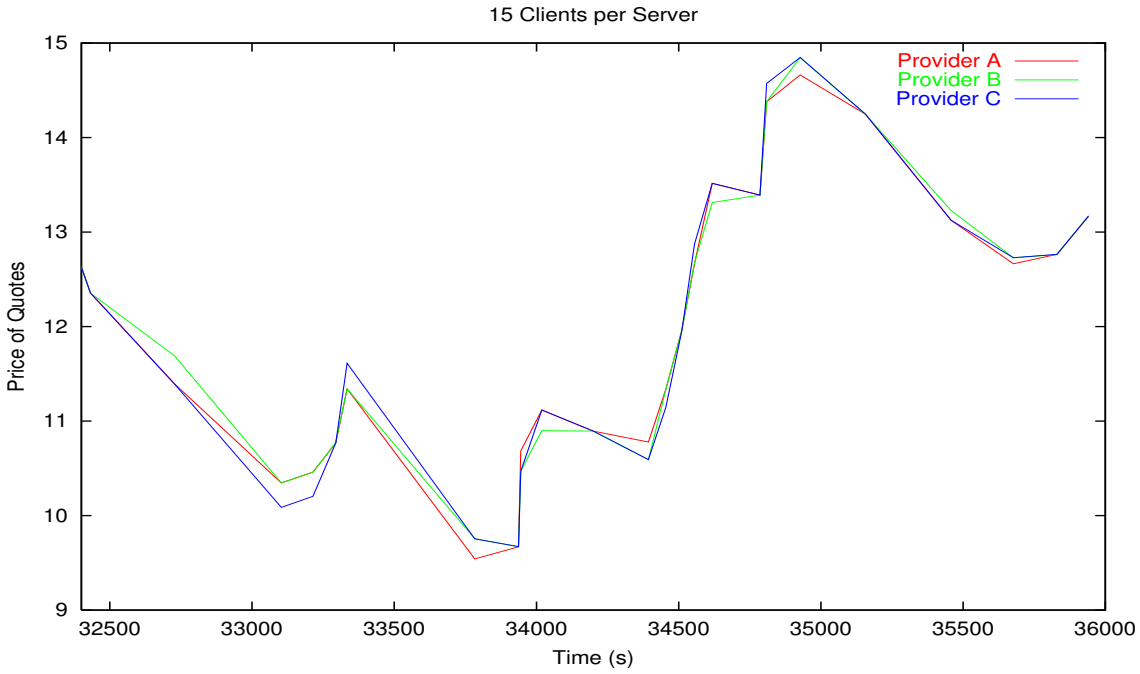

Figure 4.2: Price quotes of three providers and 15 clients per provider over one example period.

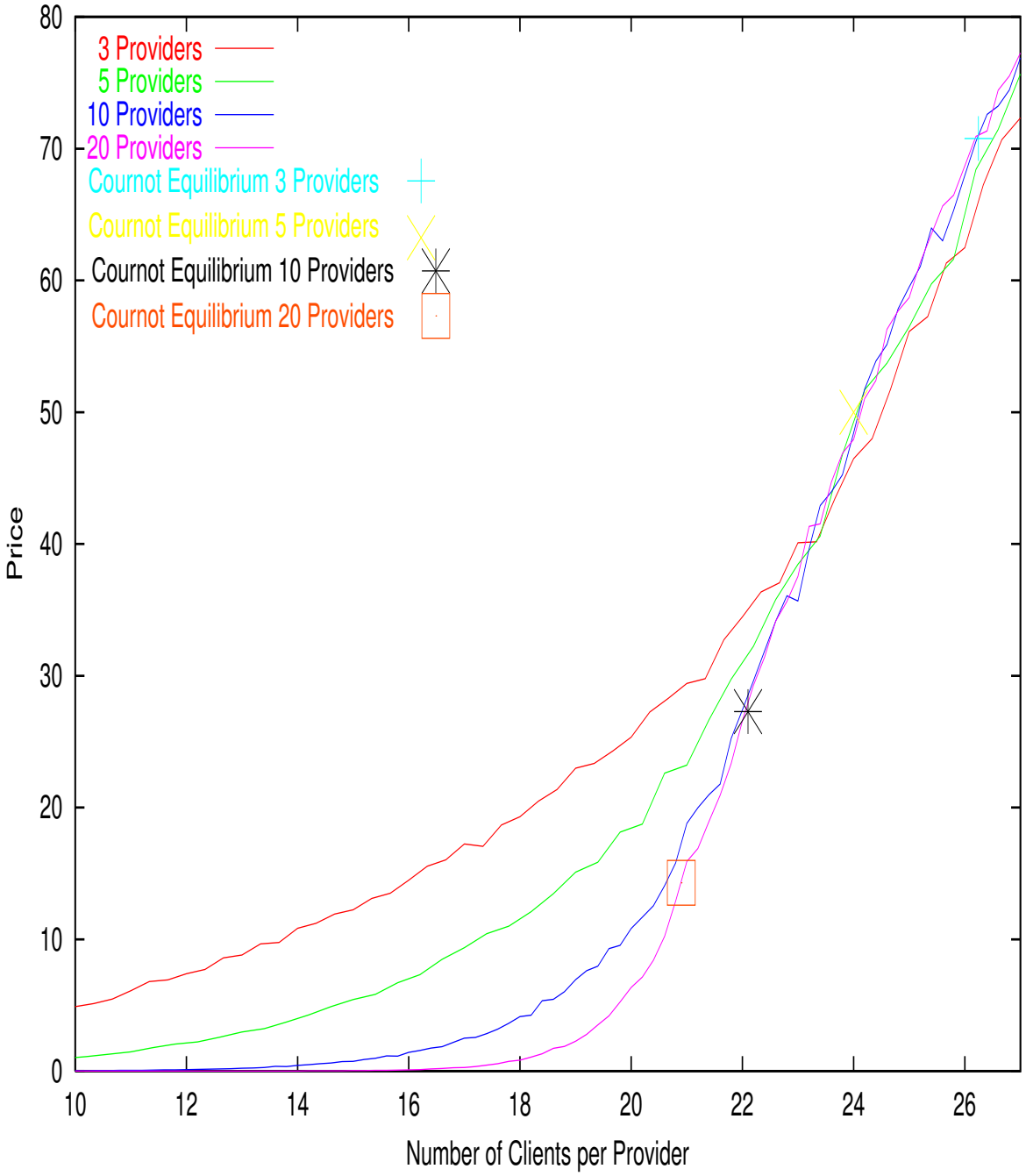

Figure 4.3: Scaling Market Size: Equilibrium price as a function of number of clients per provider for 3, 5, 10 and 20 providers.

server. The fewer servers competing, the more market power they exercise, raising the price. Equation (3.9) is in equilibrium for higher p, if M is bigger. If the number of clients per server is above 25, then the Cournot output choice, which gives the upper limit of exercise of market power (section 3.5), is above the available capacity. Providers offer all capacity and return to the competitive price.

When the Cournot output choice is below the available capacity of each provider (in figure 4.3, in the regions of less than 22 clients per server), results depend on the market structure. Providers can still maintain prices above the competitive level, but for the same number of clients per server, prices are lower for a larger number of servers due to increasing competition.

4.2 Competition with Undercutting

We test the robustness of this set of bidding strategies against deviation of a resource provider. The bidding strategy is not robust if deviation is profitable. The deviating resource provider calculates which price other providers will quote by keeping the counter for successful bids at $\emptyset$ ($\#Bid_{acpt} := \emptyset$). Then he slightly reduces the price to undercut all offers.

Is this undercutting strategy profitable? Figure 4.4 shows the same bidding period as in Figure 4.2, with resource seller A applying the undercutting strategy. A is selling all its resource capacity before any other provider makes the first sale.

The prices drop with the undercutting because all sellers assume that sales are evenly distributed among them. If a server is not selling his predicted share of the demand, this server drops the price according to equation (3.15). This is the danger with the undercutting strategy: if the undercutting seller makes less profit than by not undercutting, the supply function is an equilibrium strategy.[1]

In the example, price only fell slowly; therefore, undercutting was profitable. To prevent deviations, resource providers put additional weight on the observation of accepted bids, by scaling σ_r^2 in equation (3.15): $\sigma_r^2 \rightarrow (\sigma_r / w_r)^2$. Figure 4.5 shows the price development for the same hour as in Figure 4.4, this time with A undercutting and all servers applying the weight $w_{v3} = 5$. Now, prices drop very quickly to low levels. The undercutting seller makes, on average, about 20% less profit than if not deviating.[2]

[1] There seems to be a second threat to the stability of the bidding strategies. If a server undercuts on the last offer at the end of the bidding period, he will not feel the negative impact of lower prices on subsequent bids. However, as clients bid at random times, servers never know how many more requests they will receive. Future work will focus on evaluating such deviation strategies.

[2] We chose a simple undercutting strategy and can therefore not guarantee that a complex strategy of undercutting might not be more profitable and require a higher factor of sensitivity w to prevent deviations.

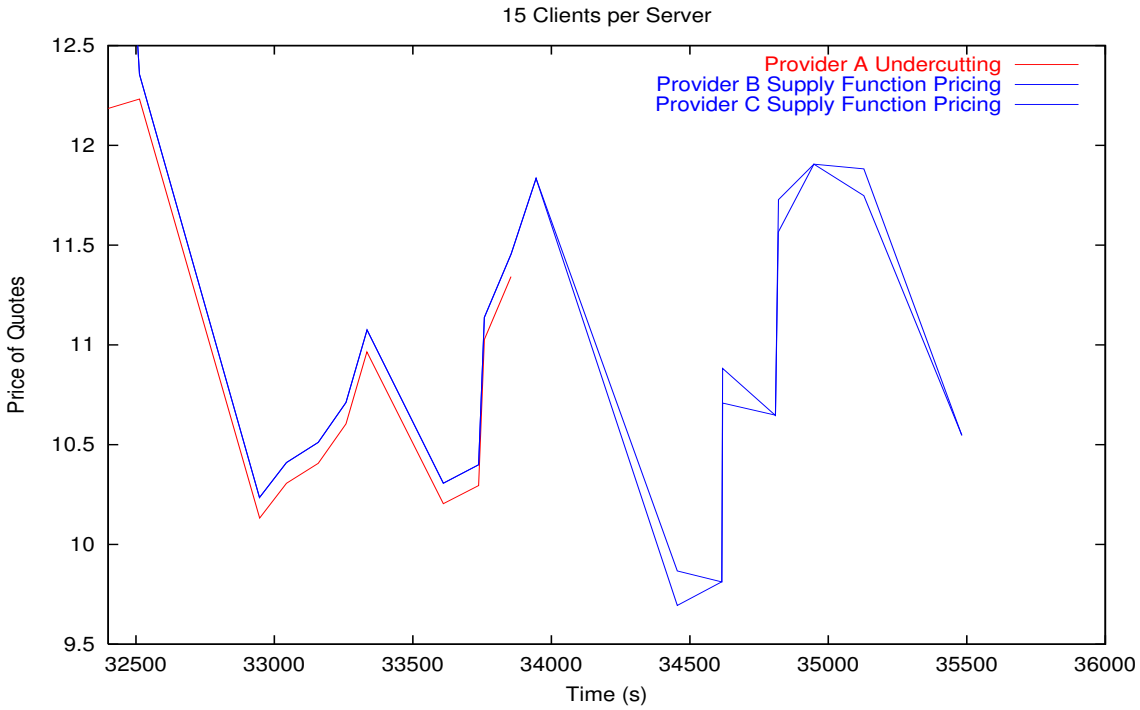

Figure 4.4: Price quotes with one seller undercutting (selling out at t=33800) the two competitors.

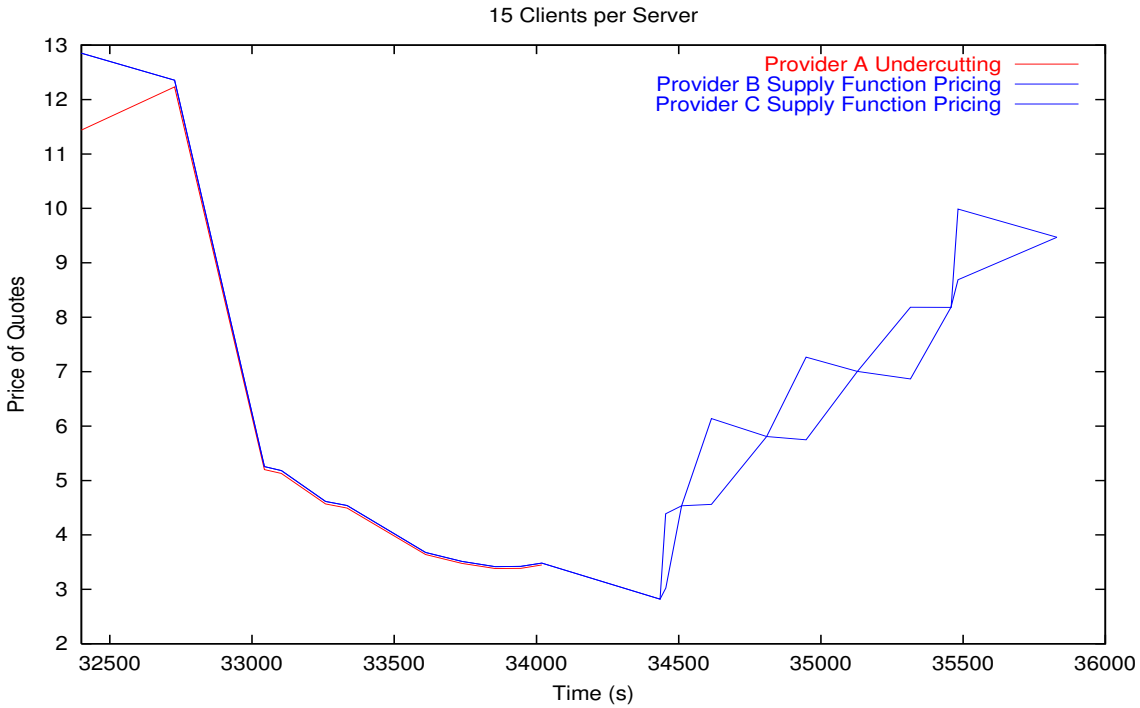

Figure 4.5: Prices with undercutter (selling out at t=34000) and increased weight on accepted bookings.

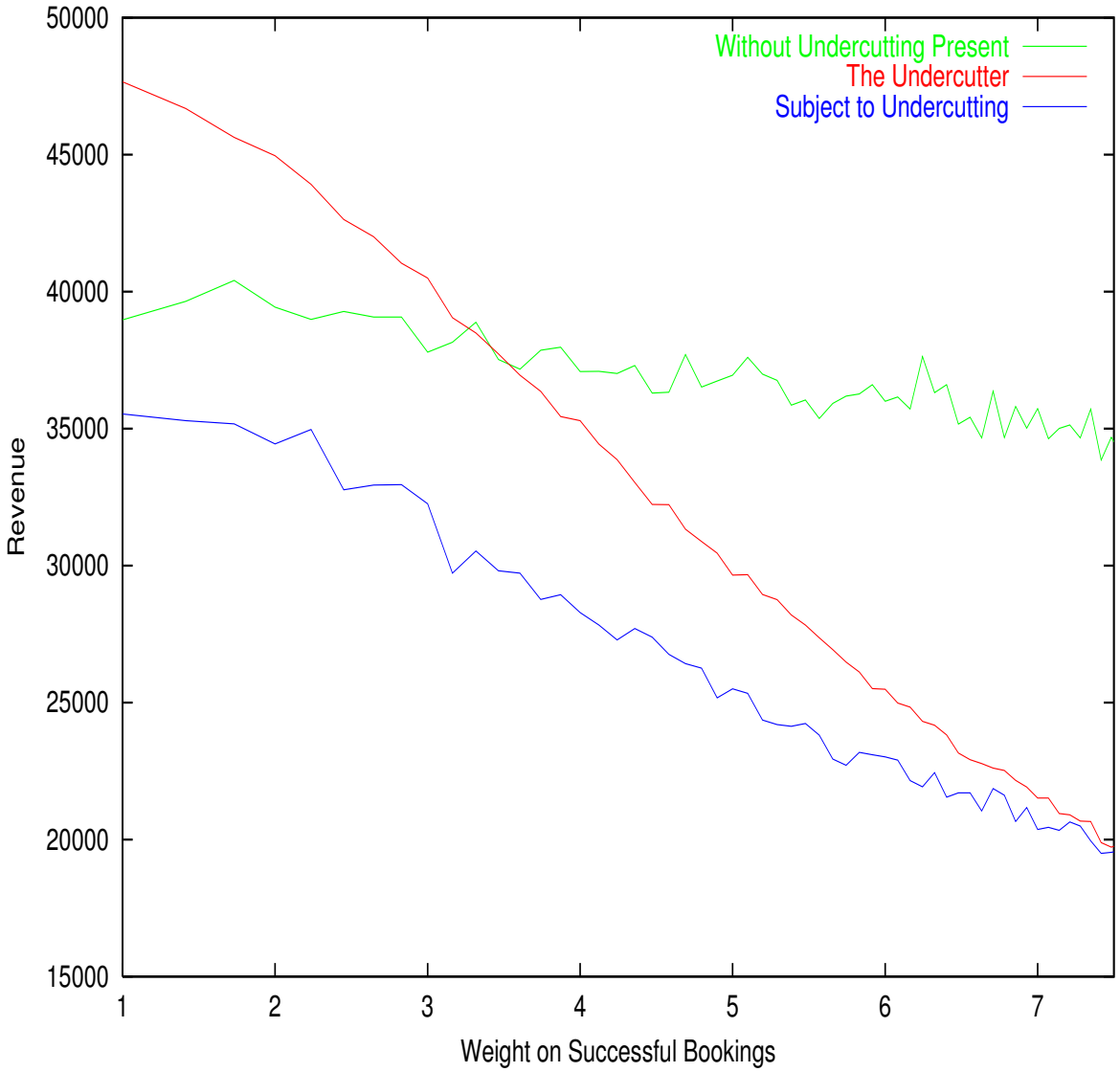

Figure 4.6: Profitability of Undercutting (red) as a function of weight on successful bookings.

4.2.1 Negative Effects of Increasing the Weight on the Undercutting Sensitivity

The disadvantage of increasing the weight w is that it reduces the profit for all sellers, even if there is no undercutting seller present. For very high values of $w > 7$, this loss due to increased weight amounts to about 10%. However, it suffices for providers to decrease the profits of the undercutter such that they are below the profits he would make if he were following the supply function strategy. The example in Figure 4.6 shows that undercutting becomes unprofitable for any $w > 3.3$. The lost profits due to the increased weight amounts in this case to about $3 - 5\%$. One would not expect an unbiased random effect to have a net impact on the average price. However, as only the lowest bid is

accepted, the selection bias reduces the average price. Furthermore, expected revenue is concave in the bid price; therefore, the revenue at the average price is higher than the average of revenues at different prices.

4.2.2 The Weight Level Under Different Market Configurations

In the above examples, we assessed one specific parameterisation of proportions of clients and sellers. Now we investigate the appropriate weight to react to undercutting for a larger range of competition scenarios.

Figure 4.7 maps the weight w at the breaking point when undercutting becomes unprofitable as a function of number of clients and resource providers. The number of clients per server is an indicator of total demand. The weight w must be high in cases of low demand. Conversely, deviating is unprofitable with high demand because lowered prices reduce revenue more than can be gained by a small increase in capacity utilisation.

By analysing the market situation, providers are able to choose appropriate parameters for our decentralised supply-function model and ensure that staying with the model is economically the best choice. We are therefore able to maintain prices above perfect competition.

Figure 4.7 illustrates a further phenomenon. In the case of low demand, a scenario with more servers requires greater weight w than in a scenario with fewer servers. On the other hand, in the case of higher demand - just short of exceeding resources - fewer servers require higher steering weight w than more servers. This difference is due to two factors.

First, in the case of higher demand, the prices with fewer servers are higher, because the resource providers shorten the supply, as described in section 4.1. Higher prices and a shortened supply imply that undercutting becomes more profitable. Therefore, the other resource sellers need to react more strongly in order to prevent undercutting.

Second, in the case of low demand, the limits of preventing undercutting from being successful are reached. The more resource sellers are participating in the market, the longer it takes for the participants to detect undercutting because they are not getting their share of the market.

4.3 Conclusions

In this chapter, first of all, we were able to establish that, with the use of the supply-function model, providers are able to maintain prices above marginal variable costs, and these allow for recovery of fixed costs in times of excess capacity. The model also behaves very much in accordance with the expectations, when in correlation with the relative demand being higher, it is able to raise prices higher. Less intuitive, but also expected, is that the pricing model is able to raise higher prices for fewer numbers of competitors, even if the relative demand is the same.

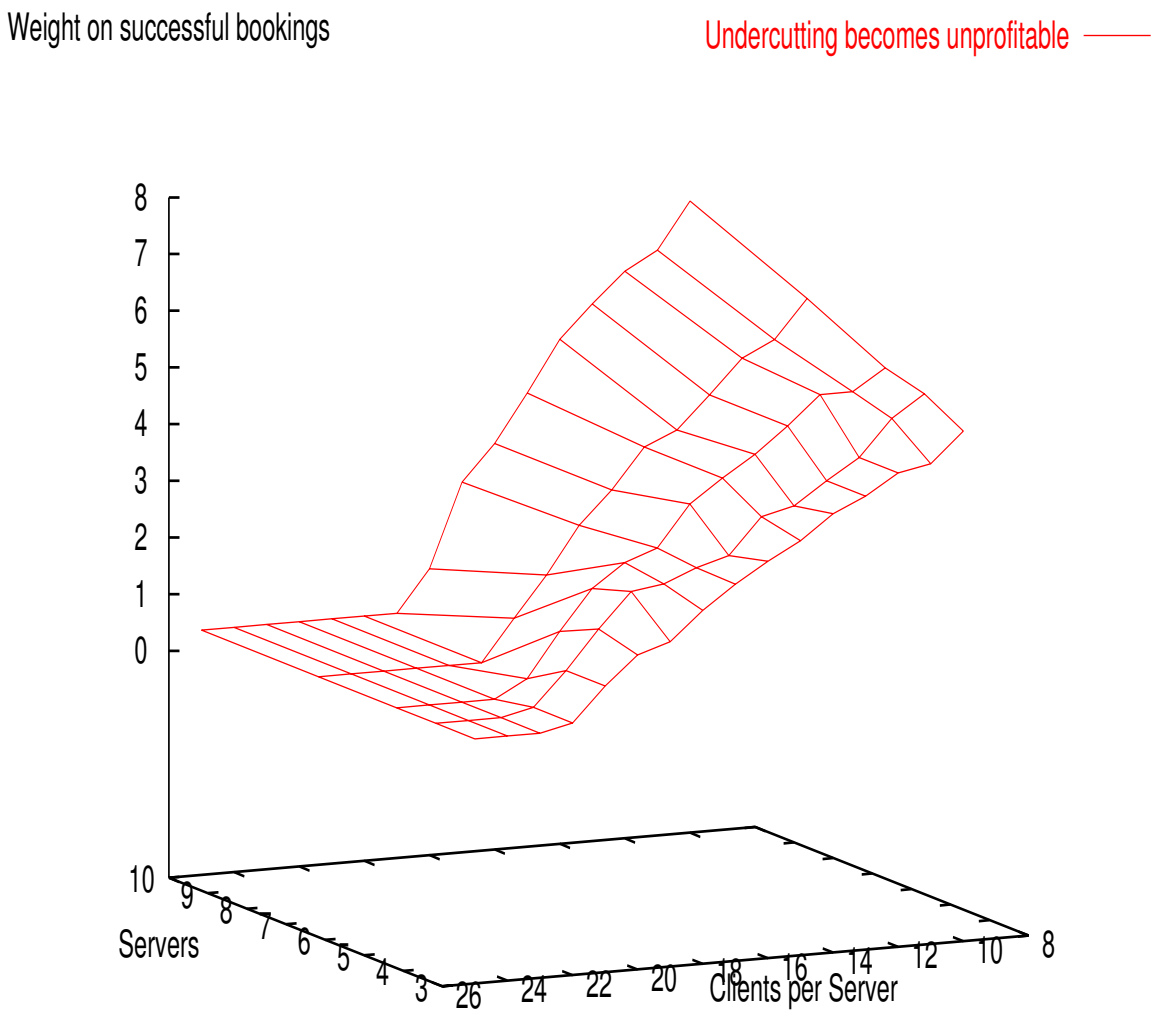

Figure 4.7: Weights required on successful bookings to prevent undercutting as a function of number of providers and clients.

Secondly, we demonstrated that the model is able to make the "deviation" strategy unprofitable. This means that due to the pricing model's response, the provider who applied the "deviant" strategy would have made more profits, if he had followed the supply-function equilibrium strategy instead. The model is able to achieve this result even if only one deviator is present, whereas any more deviators would only bring down the prices faster, and partition the deviators' collective profits further. By setting the undercutting sensitivity to a sufficiently high level, the model was able to reduce the deviators profits in any of the analysed market configurations. The deviator, when applying his strategy, can be allowed to make more profits than his peers, as long as he would be better off to join the supply-function strategy, since in this case we have the necessary conditions for a Nash-Equilibrium. Thus, while the model may not enforce cooperation from market participants, it does however provide an incentive to join the cooperation.

Further, it turned out that the necessary level of the undercutting sensitivity, to achieve the required Nash-Equilibrium criteria, resembles an useful indicator that shows the effectiveness of this pricing model in the present market conditions. The higher we have to set the undercutting sensitivity, the more difficult it is to achieve the pricing model's goals in the present market conditions. Using this indicator, in section 4.2.2 we compared a range of market conditions in terms of how difficult it is for the supply-function model to thwart the deviator's attempts to exploit the cooperation in this market model, and we could explain in detail why the model works well for fewer numbers of providers, as well as conditions with demand being closer to supply.

In the next chapter we will turn to a different type of cooperation and deviation from the one we analysed in the past two chapters. While the supply-function pricing model relies on an implicit kind of cooperation, by contrast, the recommendation aggregation model, which we will present in the next chapter, is explicitly concerned with finding consensus. The cooperation in the supply-function pricing model is implicit because it only gives an incentive to cooperate, and the providers then are free to choose their strategy. The recommendation aggregation function in the next chapter will prioritise some recommendations over others and explicitly redistribute the influence of the raters to meet its objectives.

Chapter 5

Consensus Seeking Recommendation Aggregation

In this chapter, we present a recommendation/reputation model that is trying to find the set of comments that maximises the consensus among the raters.

Reputations are important for businesses that try to increase the reliability of their contracts and supply chains, since these businesses rely on reputations to predict whether potential business transaction partners will behave as expected. The most credible source of reputation information is past experience. Other possible sources of reputation signals are formal certifications or less direct signals, such as impressive offices that indicate previous success. These indirect reputation signals are less relevant in the anonymity of the e-services world. Moreover, formal certifications often fall short, since they tend to state specific qualities that have to be met, and do not allow for much differentiation in the assessment. This leaves past experience as the most useful reputation signal for e-services.

In markets with a low turnover of transaction parties, it is appropriate to assume that each party gathers their own experience of reputation information about the possible set of contractors. In a more transient setting however, this would be a dangerous assumption, because it would potentially allow faulty parties a long lifetime until they have conned all other parties. The market we are working with, namely the on-line e-services market, relies on a very high turnover with many new players and fast-changing products. If such a market cannot do better than to rely on market participants to make their own experiences, then this market is likely to lose a lot of efficiency, since the participants would not exploit the innovative high turnover that one could potentially achieve, and instead stay with business partners about whom they have sufficient historic performance data. This situation, as we further describe it in section 5.1, therefore mandates cooperation between the market's participants in the form of a reputation system that collects and aggregates the experiences from these participants as an information service.

While the technicalities of implementing a reputation system might contain some interesting points for research, here we focus on the aggregation function that such a reputation system employs. The challenge in the design of such an aggregation function lies in the

conflicting interests of the commenting parties. If one rater's ordering of preferences differs substantially from another's, one can quickly find a situation whereby the conflicts between the orderings cannot be resolved to one meaningful common ordering. Arrow [7] formalised this inherent conflict within Social Welfare Functions, and with his impossibility theorem proved general limitations for any such aggregation functions. Yet another common problem for comment aggregation functions is that every rater naturally has an individual rating bias. Worse, a rater may even have a deviant agenda, whereby he unduly attempts to alter the outcome of a recommendation in his favour.

The individuality of rating biases can be solved through normalisation, and in our approach we will assume that the rating scales are objectively observable by all participants. This allows us to treat comments as a numeric value and aggregate it as such. While this assumption excludes irrational and less conscious reputation aspects and draws criticism for ignoring individual peculiarities, it is a meaningful approach when the ultimate goal is to achieve consensus in a larger community. In a nutshell, if the raters cannot accept and work with a common objective scale, then why should they accept the resulting common ordering of ratings and the performance scores of the rated entities? While requiring this precondition reduces the range of aspects the ratings can be applied to, it does strengthen the resulting output.

With regard to the questions that arise in the light of the Arrow impossibility theorem, we took a different approach into what denotes a useful aggregation method. Later on, in section 7.9 we will discuss the relationship between our approach and Social Choice Theory. However, the goal of our aggregation function is not to maximise its acceptability from an angle of social welfare, but to maximise the level of agreement within the comments at hand. Therefore, we take our approach in a more "dictatorial"[1] direction. We assume that not all raters are equally able to provide comments that are in agreement with the other raters' preferences, and moreover we assume that some raters might have an uncooperative agenda whereby they deviate with comments that attempt to rig the outcome. Therefore, in the aggregation function that we develop, we assign raters, who contribute more than others to a consensus solution, a higher influence on the final outcome of the recommendations.

An additional benefit of our reputation system's approach is that this kind of an aggregation method allows us to integrate into its system design incentives for raters to behave cooperatively. One common purpose of reputation systems is to pass incentive signals to the rated entities, pressuring them into improving their performance. Since our approach of the aggregation function also differentiates between the abilities of the raters, we are able to generate incentive signals geared to the raters as well. By publishing in an appropriate way the rater's reputation as a rater, the reputation system gives an incentive to the raters to be cooperative raters and to obtain a high reputation as raters, since the rated service provider will be able to see this rater's reputation and presumably deliver good service to an influential rater.

In section 5.1 of this chapter we start by explaining why reputation systems are necessary for the e-services market and raise some questions about the challenges for such a

[1] A *dictatorial* Social Welfare Function exists when one voter's set of preferences singly determines the aggregated outcome.

system. We continue in section 5.2 with an exposition of the philosophy of our approach to the design of the rating system. In section 5.3 we show how our design approach lends itself well to integrate with economic incentives for cooperation, to enhance the pragmatic value of the reputation system and to secure its funding. Finally, in section 5.4 we present the iterative calculations of the aggregation function, including criteria for its convergence points and a confidence value for the results.

5.1 Motivation for the Recommendation Aggregation Service

Clients choose a service provider on the basis of (1) price versus promised performance aspects such as, for example, a provider's contractual terms in an SLA and (2) the client's confidence in how well this provider will deliver on the negotiated performance. Aggregating the confidence of many clients resembles the reputation of a provider. The importance of reputation in the clients' decisions often is underestimated in the designs of electronic market places. This has been the case in, for example, Giovanetti and Ristuccia's [34] analysis on the band-x backbone bandwidth market, where the researchers found that clients did not rely much on the reported performance numbers, but more so on the reputations of large, well-known providers. This result is a bit surprising, since the main purpose of establishing the band-x trading platform was to eliminate switching cost and therefore to remove any possible lock-in.

Reputations are very important to the clients, because it is one of the main factors that ensures that they are able to get the business accomplished. One of the many threats to thriving e-service business are malicious servers, for example when a client has handed over his application to a Hosting Server, and the Server could modify the code intentionally. Spotting such deviant providers on the basis of their reputation is problematic in the anonymity of e-services and the absence of reputation systems. As establishing a widely accepted reputation system entails many pitfalls, researchers sought for alternate technical assurances on enforcing quality in service delivery. For example, researchers developed verification algorithms containing checksums on the computations, ensuring that the hosting server cannot modify the client's code undetected. Generally though, the more complex a hosting application becomes, the less likely one will be able to ensure correct verification all times. Another threat to e-services is independent of such solutions, as a service provider can fail entirely for reasons of poor maintenance or performance planning. For such cases frameworks were suggested in which contracts are enforced by policing all terms and conditions of an SLA at a detailed level, and possibly also by applying contract penalties for failed delivery. However, even in this case, businesses are not interested in running operations by collecting contract penalties where they could engage with suppliers who they expect would deliver well. Consequently, even if many such technical safeguards are available, we expect clients to seek a provider with an accepted "good" reputation, because such a provider would be more likely to fulfil the contract satisfactorily. Specifically, Bakos and Dellarocas [8] showed that online reputation mechanisms can (in many cases) outperform litigation in terms of maximising

the resulting social welfare.

Introducing widely accepted reputation systems for e-services requires addressing a collection of pitfalls that are inherent to them. Most of these pitfalls are not of a technical nature. For technical policies such as "we want to keep commentators anonymous", we are able to devise technical solutions. The questions that are difficult to address are in the nature of the design philosophy.

For once, one needs a definition of the scales by which to measure the reputation of a provider - a definition which includes the understanding of the pragmatic meaning of these scales. The reputation model we are presenting makes only one assumption about the choice of the reputation valuation function, namely that the reputation system operator has chosen an appropriate function. One can define in many practical ways such a valuation function that maps a notion of reputations to formal numerical metrics. However, it is accepted knowledge among economists that every individual has a different utility function, and therefore every user of a reputation system will have different requirements towards this valuation function (see Arrow [7] and Sen [82] on elections). Thus, in order to achieve wide acceptance, one needs to satisfy as many users as possible. This may turn out to be difficult to achieve, if not impossible, depending on the extent to which different users' requirements conflict with one another.

The question, then, arises: Can we trust the rating submissions from all raters? Should we give more weight to those raters who are seemingly more trustworthy than those who might be less well-informed? To address these questions, our model has the ability to redistribute the influence it assigns to a rater on the aggregated scores, and to do so in favour of certain better informed raters. In addition, we are able to produce different views of the reputation results, depending on which rater, or set of raters, we ex ante trust more.

A separate threat model category exists on the side of the providers: Do they treat all their clients equally? Can we recognise provider discrimination? Can we distinguish discrimination behaviour from biased client behaviour? We address these different threat models by simulating them in challenging rating scenarios.

Finally, do clients have some incentive to submit ratings at all? Do they have a tangible incentive to state their experiences truthfully? We create these incentives by listing the clients as contributors to the reputation system and adapt policies that discourage undesired behaviour.

5.2 Reputation System Design

5.2.1 Reputation System Philosophy

The first significant characteristic feature of this reputation system is that it distinguishes between raters and rated entities, and allows only for uni-directional ratings. Our design requires this distinction because our market model assumes a regular business-style sce-

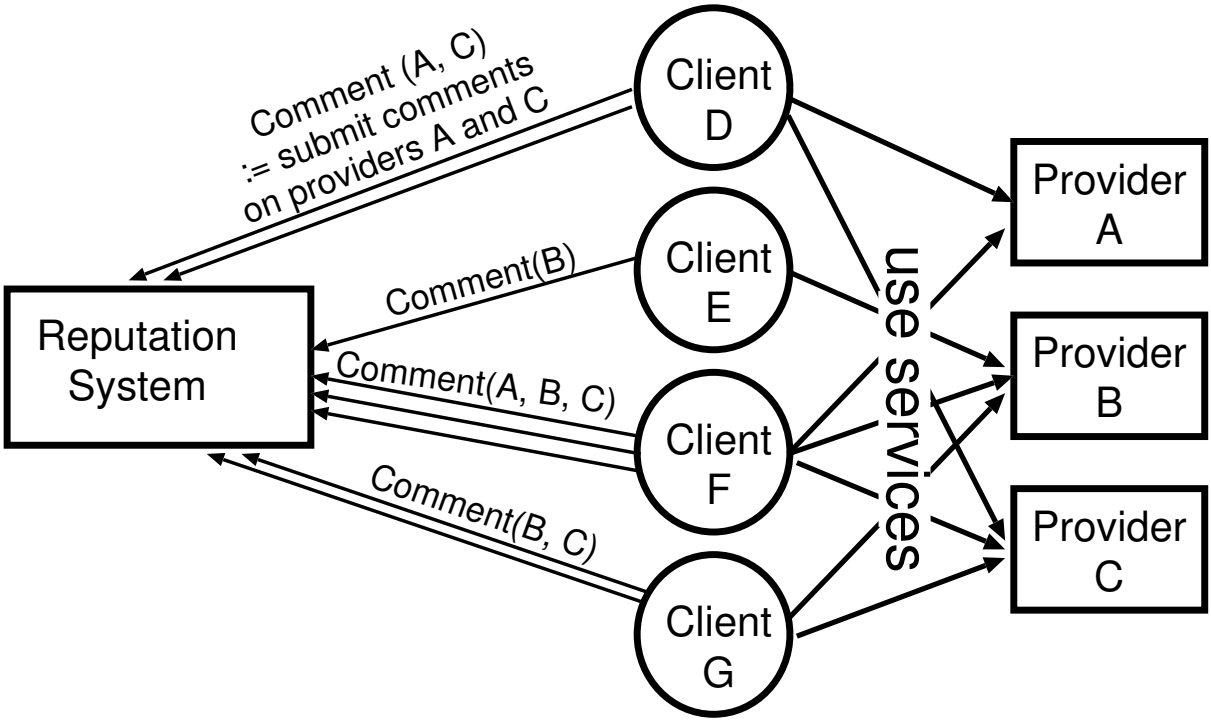

Figure 5.1: Structure of the Reputation System.

nario with separate clients and providers. Figure 5.1 shows the relationship between clients, providers and the rating system. In this market model, clients are usually *not* providers for the same service, as in peer-to-peer networks or eBay-style trading transactions. This distinction has significant drawbacks, because it is easier to assess the credibility of raters when judging their rating behaviour alongside their performance. For example, in peer-to-peer systems, I can distrust both the ratings submitted by a peer who has a poor provision performance, and also all the peers who rated this peer highly. In a client-server market model, we lack such a high degree of interaction and rating connectivity. Nevertheless, we want to use such a model to calibrate raters' comments by the quality of comments they are submitting.

The second characteristic feature of our reputation model is that we require all users of the system to agree on the scale and assigned meaning of the reputation variable, which together characterise the recommended ordering of preferred providers. This means that the reputation variable has to be an objectively measurable, numeric value $\in \mathbb{R}$ which can be observed by all users in the same way. This $\mathbb{R}$-reputation variable may be a technically measurable metric, such as the fact that the service has a response latency of a few seconds. But the metric could also be a percentage variable which notes providers' performances in the form of "90% of all transactions with this service were successful". With transformation functions like this, one can adapt many real world metrics, including binary ones, to a numeric $\mathbb{R}$-variable. However, anyone has to agree and be able to observe this variable in the same way.

The main reason for demanding objectively measurable reputation parameters is that our rating model is not able to adjust for to generally biased raters, i.e. a rater who judges

all entities in the same way as other raters, but throughout would assign one full grade lower than the other raters. In fact, such behaviour would be one of the conditions the model recognises as a "poor rating", and will reduce the influence such a rater will have on the final scores.

On a more general level though, as pointed out by Pennock [74], a recommender system needs to be invariant to scales. This means that the choice of a particular scale should not make a difference on the recommendations that yield from the recommender system. Economists in general believe that the absolute magnitude of one user's scale cannot be compared to another user's (see Arrow [7] and Sen [82]). This is due to users' having a certain bias as mentioned previously, and implies that user's utilities cannot be added up because these are invariant under linear transformations. Therefore it is necessary to require agreement among all the users on the scale of the reputation values. This still allows for clients to have a certain variance in terms of the technical application of this metric.

In the case of measuring a technical parameter, one could wonder why a reputation system is necessary at all to gain agreement on assessing a providers performance. Could we not simply install some performance probing and report system? However, why should we trust whoever is undertaking these measurements? But more generally, having one central, constraint benchmarking point leads to providers trying to figure out how to look good with respect to this benchmark. It is much harder to cut corners when one is trying to satisfy a large collection of raters, who might pay attention to different aspects of a transaction in order to be satisfied, and therefore resemble a large collection of benchmarking entities.

5.2.2 Rating Commentators

The reputation system calculates the performance scores of the services by averaging the comments submitted by the rating-clients. After collecting the divergences between all the comments given by a rater and the overall scored obtained, we derive a quality rating about the rater. Then the rater's quality variable is fed back into the rating of the performance scores, as a weight on his comments.

In order to avoid distortions we require a sufficient confidence value derived from the number of submitted comments before factoring in these results.

5.3 Economic Incentives

5.3.1 Truth Revelation Incentives

Given the uni-directional rating design of the system, we need explicit incentives which will entice clients to submit comments about their experiences with the services, and moreover, to report on these truthfully. We create this incentive for each client by publishing, as a part of the reputation system, a list of clients who contributed comments and the number

of comments they did present. We expect that a client who is listed as a rater can expect to be treated well by the service providers, because his listing is a signal to them that in future he also will submit reports on their performance. The number of comments that a client has submitted reveals whether a client, relative to his activity level, has supplied more than just a token commentary.

To entice truth revelation from the clients, we have to apply another measure to our reputation system. We considered the possibility of publishing the rater reputation that our system calculates in the process of aggregating scores. There, a rater's reputation reflects his rating accuracy and the influence assigned to his comments by the reputation system. Making this rater reputation public would increase the rater's incentive to provide high ranking commentary, because this would send a stronger signal to the providers. However, the rater reputation should be kept undisclosed by the reputation system, since knowing this would also create an incentive for the clients to modify or even fabricate their comments in a targeted way. They could do so by submitting comments that reflect the currently published scores for the services. So actually, in order to promote truthful comment submissions, the reputation system adopts a policy of striking clients off its list of raters if they submit erratic or modified comments. We can recognise such clients from their particularly low rating reputations. We should point out that the clients' reputations and their influence on the ratings will decline and remain unnoticed outside of the reputation system before we strike them off the rater register. We strike them off once the coherency of their ratings falls below a certain low standard. For example, in chapter 7, we will show in figure 7.4 how inaccurate raters can be identified through the reputation system. With this policy, raters have no incentive to submit untruthful comments. Instead, they have an incentive against this, particularly because they are not able to gauge how high their own rating reputation is. Rating reputations depend on all the other comments and thus are only known by the reputation system.

Another advantage in adopting this policy of striking off random or malicious raters is that doing so reassures providers that the reputation system is meaningful and that they should accept the published rankings. This acceptance should increase their willingness to improve their service quality where necessary. Service quality improvements should actually be self-evident if the reputation system creates competition over quality aspects.

5.3.2 Financing the Operations of the Reputation System

Operating such a reputation system needs to be funded in some way. However, the way we choose to fund the reputation system has implications for the incentives put forth by the reputation system. With a reputation system, we aim to resolve some of the market inefficiencies, which presumably would yield benefits that could be tapped for refinance. Clients benefit directly from the higher reliability of services from better providers, and the providers benefit from the market's ability to drive "lemon"-providers out of business and the therefore generally higher confidence by the clients in the services market. Some providers may benefit by charging a premium for delivering service with higher reliability.

Possibly the easiest route to obtain the funds to run the reputation service would be to demand subscription fees from the listed providers, as these could afford to write off the

fees as a form of advertisement for themselves. The other incentive for the providers to pay for a subscription would be to lock foul competitors out of the market, as they now would be identifiable through the reputation system. Effectively they would be forming a guild of "accredited" providers and thereby maintain a clean market for the clients. However, we would choose against funding the system by the providers, as this would create the wrong incentives for the operation of the reputation service. The operators of the reputation system would lack the incentive to promote transparent performance comparisons between providers and thereby promote competition among them, but would have an incentive to set a high bar for the level for market entry. Our reputation system seeks to aid the clients' choice to make the market in general more efficient, and a provider funded system would naturally follow different goals.

The economically cleaner solution would be to demand payment from the clients and thereby provide a direct incentive for the reputation system to cater to the needs of the clients' choice. A client funded reputation system does not have an incentive to alter the objectivity of its provider reputation rankings and if it is subject to competition itself, where clients choose a reputable reputation system, the reputation system also has a disincentive to accepting bribes (or advertisements) from the providers. However, requiring clients to pay is likely to meet with resistance, as it can be expected that some of the clients would rather not use such an advisory service at all than have to pay for it, even if such could be shown to pay off in the long run. Moreover, clients could undermine this revenue model by forwarding or even publicising the reputation system's ranking lists. Further, the actual pricing model would be a sensitive issue, as a fixed fee might be too high for some small one-time client and yet negligible for a high-turnover one. Addressing this through discriminatory pricing schemes is likely to introduce distortions and using micro-payments that scale proportionally with the volume of use is likely to introduce unreasonably high transaction overheads. A specific problem is system startup, as the value of the reputation system increases along with the number of gathered clients' review comments. Therefore, one could not use the client funding to cover the startup phase. On a general level, it may turn out that the clients' individual per-transaction-profit from using the reputation recommendation system is not sufficient to pay for the running cost of the system, even if their amortised profits over time, due to improvements of the whole market, would do so. This is not an unlikely case and one common solution is to call for a government sponsored funding solution.

Where a system's net benefits need to be amortised either over time or aggregated over a number of market participants, it is common to seek government sponsorship, as the system meets the criteria to be considered an infrastructure element. It is our anticipation that the reputation system would be such a candidate due to all the complications stated in our discussions about client funding and rejecting provider funding on the ground of incentive incompatibility.

Another possible solution is the mix of government and client funding. In that case one would draw on client charges for enquiries about high stakes services that yield client benefits clearly outweighing the charges and government funds for the remainder, in particular any startup periods. In the UK's energy market, the government supports such price comparisons in order to make the market more dynamic and transparent. Even in the absence of direct government funding for a reputation system, it may well be neces-

sary to introduce government regulations that protect the reputation reporting system. Otherwise, providers not only can lack cooperation with such a reputation system setup, but could actively obstruct its work, for example by insisting their clients not disclose their experiences that stem from using the provider's service. Such obstructions would most likely be expected from providers who have built their reputations through other means, such as advertising, or sheer size and now attempt to protect their advantageous profile.

5.4 The Rating Mechanism

In this section we describe the mechanism for compiling scores and the commentator ratings as introduced in section 5.2.2.

5.4.1 Definitions

We have a set of k *service providers* $P = \{p_1, .., p_k\}$ and l *raters* $R = \{r_1, .., r_l\}$ who rate the providers on the performance they experienced as clients. The providers perform at a certain 'real' performance level and this is the value we ideally would like to obtain as a final score for this provider in the reputation reporting system. However, this 'real' level is an unknown variable, possibly even unknown to the provider himself. Every time a client requests services from this provider, the client experiences a probabilistic value of this performance. However, these individual 'experiences' may be observed slightly differently by each individual client. The individual differences arise from a number of reasons such as them applying different measurements, or aggregating their experiences on different scales.

At a time t, rater $r_A \in R$ decides to make a *comment*: $c_{t,r_A \mapsto p_B} \in C$, about provider $p_B \in P$, reporting his observations in form of a numerical value: $val(c) \in \mathbb{R}$. This model places no constraints on the point of time at which or frequency with which any rater may submit a comment. The following calculations are performed for all raters and providers, however in order to simplify the notation, we will demonstrate the calculations for an exemplary rater r_A and an exemplary provider p_B. As the comments are collected at a central point in the recommendation system, we define all comment time stamps t to be distinct. Raters can submit more than one comment about a provider and these can be distinguished by the time the comment was made.

Initialisation

After collecting a sufficient number of comments[2] on a provider: $|\{c_{t,r_* \mapsto p_B}\}|$, we can calculate an initial score for this provider. In our terminology '$*$' represents any client or provider applicable. Before entering the iterative loop, we initialise the provider scores to

[2]It is possible to develop a confidence value that indicates how many comments are sufficient to achieve statistical integrity. However, we decided to omit developing and analysing such a variable.

obtain the *initialisation vector* $s^{(0)}_{p_{1,..,k}}$. This initialisation simply averages the values of all the comments:

$$\forall\, p_B \in \{p_1, .., p_k\} :$$
$$s^{(0)}_{p_B} = \begin{cases} \dfrac{\sum\limits_{c \in C'} val(c)}{|C'|}, & C' = \{c_{t, r_* \mapsto p_B} \in C\} \\ 0, & if \quad C' = \emptyset. \end{cases} \tag{5.1}$$

C' is the set of all comments made by any rater about provider p_B and if this set is empty, we assign a score of nil. The (0) represents the iteration number[3], indicating the initialisation procedure at this point, where we simply average the received comments.

Local Scores

Next, we calculate the score each rater would assign by himself for each provider. For each provider, we take the collection of raters having submitted comments on this provider and then for each rater within this set we average his comments on this provider to obtain a local score $local.s_{r_A \mapsto p_B}$. Here, C' is the set of all comments made by rater r_A at different times about provider p_B and if this set is empty, $C' = \emptyset$, we assign a score of nil, $local.s_{r_A \mapsto p_B} = 0$:

$$\forall\, r_A \in \{r_1, .., r_l\} : \forall\, p_B \in \{p_1, .., p_k\} :$$
$$local.s_{r_A \mapsto p_B} = \begin{cases} \dfrac{\sum\limits_{c \in C'} val(c)}{|C'|}, & C' = \{c_{t, r_A \mapsto p_B} \in C\} \\ 0, & if \quad C' = \emptyset. \end{cases} \tag{5.2}$$

5.4.2 The Iterative Loop

Rater Reputations

After calculating initial and local scores for all providers, we enter the iterative loop and calculate for all the raters their reputations $q_{r_A \in R}$: The reputation is calculated from the difference between rater A's comments and the above calculated initial scores $s^{(0)}_p$. Then we calculate the standard deviation of this difference. Of this difference we take the norm by dividing it by the number of comments submitted by this client. This yields our reputation value, except that, as it stands, a good rater with a high rating reputation

[3]In our notation, where variables (e.g. s) are indexed by the iteration number (e.g. n), such as $s^{(n)}$, the iteration number is kept in parenthesis in the superscript in order to distinguish these from power operations.

would be assigned a smaller value q. In order to correlate good reputations to high reputation values, we invert this final reputation value. C' is the set of all comments made by rater r_A about any provider and if this rater did not submit any comments at all, he is assigned a rater reputation of nil, $q_{r_A} = 0$:

$$\forall\, r_A \in \{r_1, .., r_l\} :$$

$$q_{r_A}^{(n)} = \begin{cases} \dfrac{|C'|}{\sqrt{\sum\limits_{c \in C'} \left(s_{p_*}^{(n)} - val(c)\right)^2}}, & C' = \{c_{t,\, r_A \mapsto p_*} \in C\} \\ 0, & if \quad C' = \emptyset. \end{cases} \tag{5.3}$$

Influence Shares

Following, we use the raters' reputation values as a weight on their comments, and therefore obtain an updated score for the provider ratings.

This weight each rater obtains is directly proportional to their reputation value q, and is represented as their share of influence $infl$ on the reputation scores. From the above collection of all raters $\{r_1, .., r_l\}$ of the providers $\{p_1, .., p_K\}$, we take their reputation values $\{q_{r_1}, .., q_{r_l}\}$, and divide 100% of available influence into slices $\{\mathit{infl}_{r_1}, .., \mathit{infl}_{r_l}\}$, proportionally to these reputation values. Additionally we introduce what we will label the rating model's *selectivity weight* variable w, which reinforces the selective effect of the model for $w > 1$ and dampens the model effect for $0 < w < 1$. For $w = 0$ the model effect is entirely disabled and the resulting scores are result of plainly averaging the input comments. This selective weight variable is essential to the utility of the rating model, as we need it to adjust the model's balance between selection and inclusiveness to any certain market scenario. In case the model was run on an empty input set without any comments submitted to the system, we assign all influence values to null, $\mathit{infl}_{r_A} = 0$:

$$\forall\, r_A \in \{r_1, .., r_l\} :$$

$$\mathit{infl}_{r_A}^{(n)} = \begin{cases} \dfrac{\left(q_{r_A}^{(n)}\right)^w}{\sum\limits_{i=1}^{l} q_i^w}, & C \neq \emptyset \\ 0, & C = \emptyset. \end{cases} \tag{5.4}$$

Updating Scores

Finally, we compute an updated performance score rating $s_{p_B}^{(n+1)}$ for the next iteration $n + 1$ by summing up the averaged comments from 5.2 multiplied with the influence shares from the previous equation (5.4). Note that if in the previous equation (5.4) all

influence values have been set to null, $infl_{r_A} = 0$, the resulting scores automatically also are all null:

$$\forall\, p_B \in \{p_1, .., p_k\} :$$
$$s_{p_B}^{(n+1)} = \sum_{i=1}^{l} \Big(local.s_{r_i \mapsto p_B} \times \frac{infl_{r_i}^{(n)}}{\sum\limits_{j \in R'} infl_j^{(n)}} \Big),$$
$$R' = \{ r_* \in R \,|\, \exists\, c_{t, r_* \mapsto p_B} \}. \quad (5.5)$$

In the sum of the local scores, we divide the influence values $infl_{r_i}^{(n)}$ by the sum of all influence values of the raters that made comments about this provider p_B. If all raters commented on this provider, then the sum of influence values will be 1, and if only one rater r_A commented on this provider, this sum will be equal to his influence value and thus the score for this provider will reflect 100% of this rater's comment value $local.s_{r_A \mapsto p_B}$ on provider p_B. This construct is necessary if not all the raters have commented on this provider, to then translate the influence values into the relative shares of the raters who did comment on this provider.

5.4.3 Convergence

This concludes iteration n, the next will continue at step (5.3), with these now updated provider scores and raters' influence values. This iterative process will continue, until the updated differences of the calculated performance scores fall below a convergence threshold:

$$\delta > \max_{i=1,..,k} \| s_{p_i}^{(n+1)} - s_{p_i}^{(n)} \|. \quad (5.6)$$

Practically, the convergence threshold is satisfied, when the largest update of any provider score, from one iteration to the next, falls below δ. We considered a more complex convergence criteria that is proportional to the absolute values of the scores. However, as we discuss in section 6.4.2 in the next chapter, convergence is usually very rapid, and we found a fixed threshold of $\delta = 0.0001$ to be practically suitable in most conceivable scenarios. Further, in section 6.4.1 we prove that this algorithm converges in all cases. The algorithm is robust such that if one runs this rating algorithm with no or only one comment as input, the algorithm converges immediately in the first iteration.

Multiple Convergence Points

Note that one could start with a different initial score vector in equation (5.1) and would in most cases reach the same result. We choose this averaging initialisation vector to allow for a faster convergence in most cases. The other cases, where the result depends

on the initialisation vector, have multiple convergence points. If a particular setting of data and algorithm parameters can yield more than one convergence point, we consider the resulting reputation scores useless. In such a case the model is not able to make a reasonable recommendation. In section 6.4.4 of the next chapter we analyse the case of multiple convergence points further.

5.4.4 Confidence Value

If only a subset of raters has submitted ratings on a particular provider, the resulting score of the provider is only based on the comments from these raters. This may mean that a resulting score can be derived from raters who have a low overall rating reputation. There is little to be done about the score in such a case, since there are no other ratings that we have more confidence in. However, from equation (5.5) we can calculate the sum influence values that we used to compile a resulting score and therefore obtain a confidence value on this score. The confidence value $\mathit{confidence}_{p_B}$ is calculated for every provider, after the algorithm has converged:

$$\forall\, p_B \in \{p_1, .., p_k\} :$$

$$\mathit{confidence}_{p_B} = \sum_{j \in R'} \mathit{infl}_j,$$

$$R' = \{r_* \in R \,|\, \exists\, c_{t,\, r_* \mapsto p_B}\}. \quad (5.7)$$

The confidence value $\mathit{confidence}_{p_B} \in [0, .., 1]$ is a measure that states how many per cent of the total accumulated rater reputation participated in deriving the score for this provider p_B.

5.5 Conclusions

In this chapter we started out by giving the motivation for our approach to developing a reputation/recommendation model, that is based on the idea that some raters' comments are more valuable than others' and in so doing seek to maximise the consensus among the raters. Due to the adoption of numerical scales for the measure of reputations, we were able to build a base for our reputation aggregation function that interrelates ratings by the same rater for different rating aspects. Since the outcome recommendations of our aggregation function depends on the reputations of the raters, and vice versa, we obtain an iterative algorithm.

This leaves us to establish whether this algorithm actually converges and how effective the aggregation function might be. Moreover, is the model actually able to produce meaningful outcome recommendations when it is discriminating between raters? Further, as our model introduces the selectivity weight variable, what effect does this variable have on the model's outcome recommendations? We address all these questions in the next

chapter with a set of micro-level experiments, whereby we look at small-scale scenarios that allow us to manually trace the steps of the rating model from the comments to the outcome recommendation scores.

Chapter 6

Analysis of the Reputation Model

In this chapter, we analyse the technical capabilities of the rating model that we developed in the previous chapter. In sections 6.1 to 6.3 we use small-scale scenarios to understand how the model exactly derives its outcome scores. One of the design intentions of the model was to be able to identify and discriminate raters, who seem to contribute less to the overall consensus. Therefore, we integrated the selectivity weight variable into the model, which adjusts the strength of the model's focus on discrimination versus inclusiveness. In these case scenarios we can understand how one can adjust this variable, depending on the situation, such that the level of outcome scores meet the expectations.

Following, in section 6.4 we focus on the technical feasibility of the model algorithm. Since this algorithm is iterative, we need to ensure that it converges (section 6.4.1), and that it does so at a reasonable rate (section 6.4.2). We also explain how different convergence points make for different results, which are reachable through different starting vectors.

In the next chapter (7) we will discuss application relevant issues of the reputation model and algorithm.

6.1 First Example

In order to gain understanding of the basic functionality of the reputation model and how different settings for w affect the scores, we show an example scenario in table 6.1. Each rater is assumed to have submitted exactly one comment about each provider. The top half of the table lists the values of the comments c submitted by the raters $r_{[E..K]}$ about the providers $p_{[A..D]}$. The lower half of the table shows the scores $s(w)$ obtained from running our reputation model on these submitted comments, with different settings for the selectivity weight variable $w = [0.1, 0.9, 1.0]$. The comment values are chosen such that three of the raters (r_E, r_G, r_H) share a relatively high degree of agreement on the ratings $(p_A = 1.0, p_B = 2.0, p_C = 3.0, p_D = 4.0)$ of the providers and the other four raters (r_F, r_I, r_J, r_K) diverge in each different ways. While r_F shares the same agreement as the three coherent raters on three of the four provider ratings, he diverges strongly for

	p_A	p_B	p_C	p_D	$in(w = 0.1)$	$in(w = 0.9)$	$in(w = 1.0)$
r_E	1.0	2.0	3.0	4.0	15.41	41.69	60.12
r_F	1.0	2.0	3.0	1.0	13.44	2.06	1.12
r_G	0.9	1.9	2.9	3.9	15.64	30.18	19.60
r_H	1.1	2.1	3.1	4.1	15.16	16.93	13.88
r_I	0.5	1.5	3.5	4.5	14.25	5.46	3.35
r_J	2.5	4.0	1.0	2.0	12.91	1.63	0.88
r_K	0.0	0.0	5.0	4.0	13.17	2.05	1.18
$s(w = 0.1)$	0.99	1.92	3.08	3.41			
$s(w = 0.9)$	0.96	1.95	3.02	3.92			
$s(w = 1.0)$	0.98	1.97	3.02	3.96			

Table 6.1: Example scenario showing comments submitted by raters r, about providers p, the resulting model scores s and the influence share in, both depending on the chosen selectivity weight w.

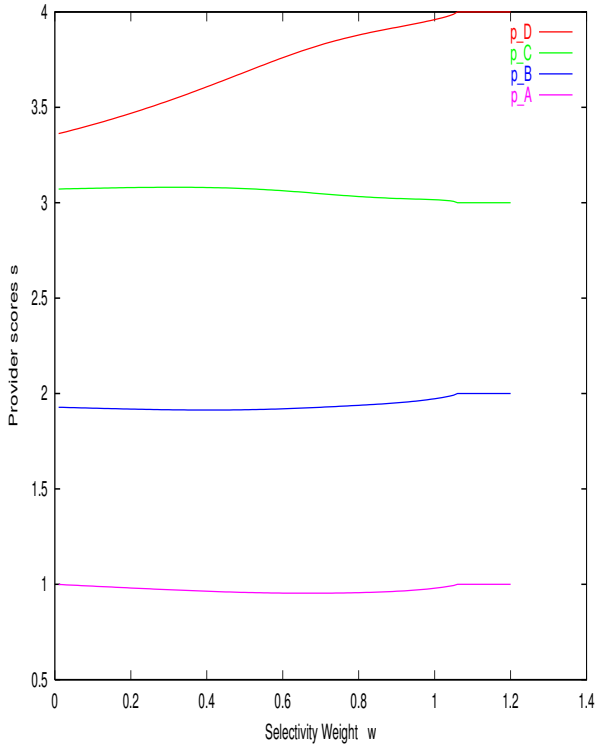

Figure 6.1: Scores for the Example Scenario Dependent on the Selectivity Weight.

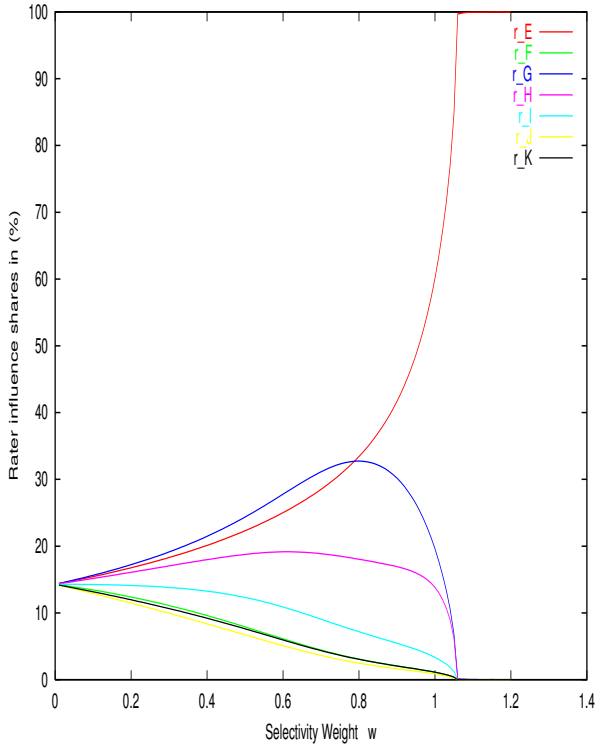

Figure 6.2: Influence Shares for the Example Scenario Dependent on the Selectivity.

the fourth provider. Then, r_I still follows the ratings of the coherent raters, but with a much higher variance, but r_J and r_K diverge in nearly random fashion. Our expectation of a meaningful selection of scores by a rating aggregation method would be that the chosen scores come out to be the ones of the three providers with the high degree of agreement. We also expect that the influence of r_F and r_I is rather diminished, but r_J and r_K with their obvious irrelevance to other rater's comments are removed from the consensus altogether.

In the part of the table with the scores we can see that for settings of $w = (0.9, 1.0)$, the scores come out to meet our expectations as they are close to the coherent raters' comments. In the part about the influence shares of each rater we also can see that our expectations are roughly met. Figure 6.1 shows in more detail how the scores are dependent on the settings on w. The example is set up in a way such that r_E reflects the average scores the closest. The scores for providers p_A, p_B and p_C, do not vary greatly with different settings for w, this is because the differences in the submitted scores cancel each other out. For p_D on the other hand, the average derived from all submitted comments diverges by about 20% from the score calculated by the model. To explain how the model yields different values for p_D, it is best to look at the influence values *in* on the right side of table 6.1 and figure 6.2. With $w = 0$ the scores end up nearly perfectly averaged and all raters share the same influence on the scores. Where with $w = 1.1$ the scores fall onto the rating submitted by rater r_A only, as he obtains 100% of the influence share. The influence turns in favour of r_A because he is the closest to the average of the majority of the raters. With a less extreme setting for the selectivity weight factor, such as $w = [0.7, .., 1.0]$, we obtain a balancing between locking out the influence of deviating raters and the raters who are close to the average. In this case, about three raters share the majority influence (r_E, r_G, r_H), and four raters are increasingly marginalised (r_F, r_I, r_J, r_K) with higher settings for w. Correspondingly, with loosing their influence, the scores approach the average of the other three. This can be seen on the score value s for p_D which depends more on the distribution of influence than the other providers' scores. Further, even if the setting for w is not yet that high that only one rater counts, the resulting score $s_D(w = 1.0) = 3.96$ for provider p_D is relatively close to the ideal score as defined in the premise of the scenario and is reached for the selectivity weight $s_D(w = 1.1) = 4.00$.

From this example, we can see that the rating model is able to discriminate against deviating rater's comments and find a rating that is supported by a larger subgroup of raters with similar comments. We also can see that by adjusting the selectivity weight w we are able to balance between how many of the raters are discredited and by how much they are so.

6.2 Two Opposing Groups

In order to understand the properties of the rating model we confront it with some very divergent rating scenarios. We expect the model to arbitrate between the diverging comments even if these are so contradictory that no meaningful consensus is apparent to an

	p_A	p_B	p_C	p_D
r_E	1.0	2.0	3.0	4.0
r_F	1.5	2.5	3.5	4.5
r_G	0.9	1.9	2.9	3.9
r_H	1.1	2.9	3.1	3.4
r_I	5.0	6.0	7.0	8.0
r_J	5.0	6.0	7.0	8.0
r_K	5.0	6.0	7.0	8.0

Table 6.2: Comments Submitted by Two Opposing Groups of Raters

observer. However, by tracing the model behaviour in detail when it is pushed to the limits and kept in a very constraint setting, allows us further to predict its behaviour for larger statistical scenario analysis. We investigate the cases of two and four commenting blocks.

After convincing ourselves that the model can be used to take single raters out of the rating pool, we are interested in how the model deals with two groups competing with opposing rating suggestions. Table 6.2 shows the comments submitted by two groups of raters with the values of their comments falling into two opposing clusters. The first group with raters $r_{[E,F,G,H]}$ has one more rater than the second group with raters $r_{[I,J,K]}$, that however have a perfectly low variance, which should again work in their favour. In the development of scores in figure 6.3, we can see that with an increasing effect of the selectivity variable, the larger group dominates the resulting model scores. Because this small three-rater group submitted identical comments, their graphs of the influence shares show as only one line in figure 6.4. Although the second group displays perfectly low variance, it is further away from the overall average, which is the most critical parameter in determining the dominant group. Pushing the selectivity variable to a maximum level results in rater r_F's scores to dominate in the larger group. This is due to r_F's scores being closer to the overall average, including the second, smaller cluster of raters.

To understand how the iterative process of the rating algorithm works, we look at the changes from one iteration to the next in this scenario. Figure 6.5 shows the scores updates during the algorithm's convergence for this scenario with the selectivity weight variable chosen to $w = 0.85$. The progress of the corresponding influence shares is in figure 6.6. We notice that the algorithm converges after 33 iterations and should point out that for most other settings of w would converge even faster. Further, the losses of influence of the minority group are attributed to one rater only in the dominant group. The losses are not evenly distributed among the dominant group, because r_F is overall with his comments significantly closer to the comments of the minority group than his other group members. Concluding, we can remark that the rating model when presented with two opposing rating groups is able to force a decision in favour of what received the most support. Similarly, if we are more interested in a more inclusive reputation scores, we can achieve this with varying settings for the selectivity weight variable.

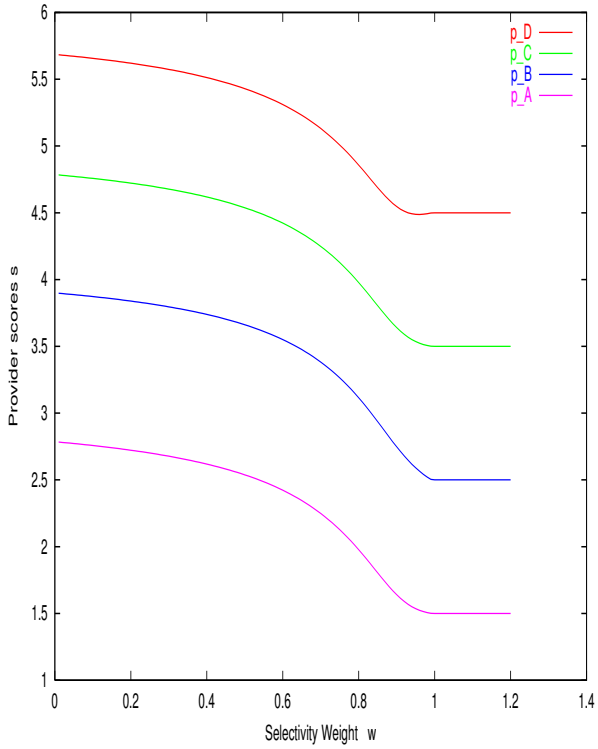

Figure 6.3: Scores with Two Opposing Groups of Raters.

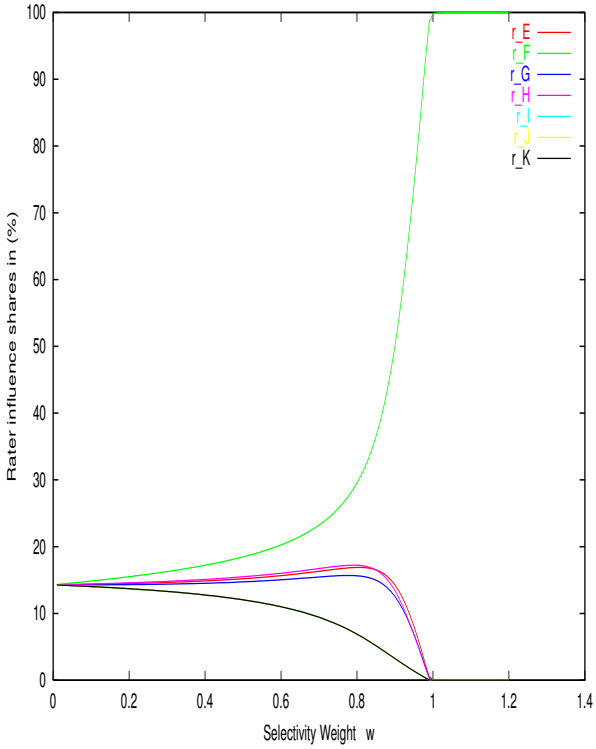

Figure 6.4: Influence Shares with Two Opposing Groups of Raters.

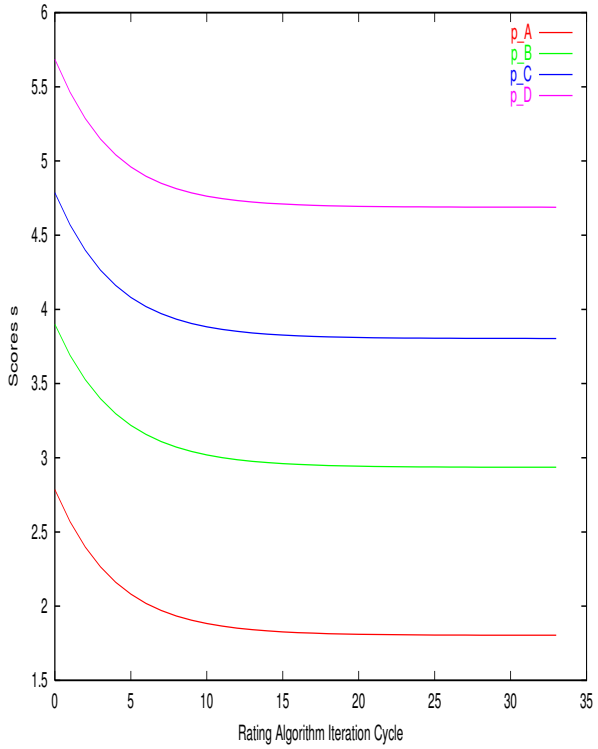

Figure 6.5: Iteration Updates by the Rating Algorithm of the Scores in the Two-Group Scenario for $w = 0.85$.

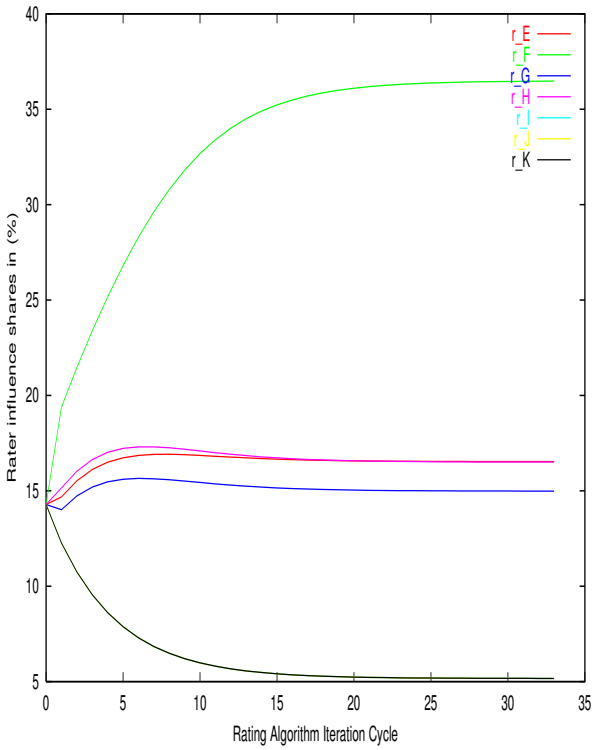

Figure 6.6: Corresponding Iteration Updates of the Influence Shares.

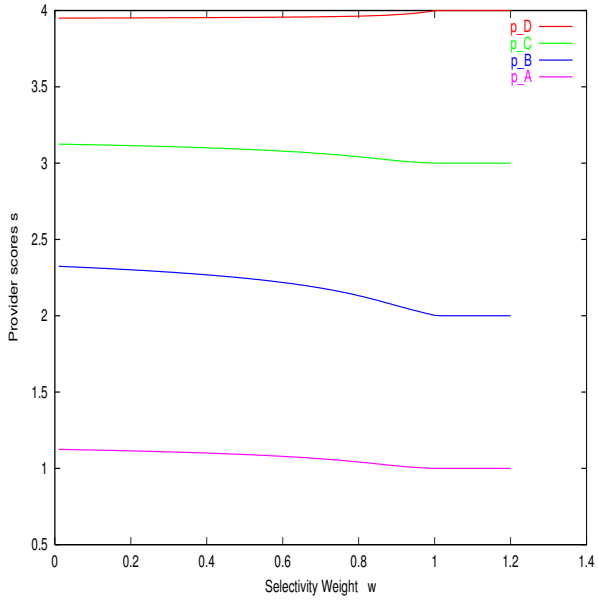

Figure 6.7: Scores with only the larger of the two opposing groups of raters present.

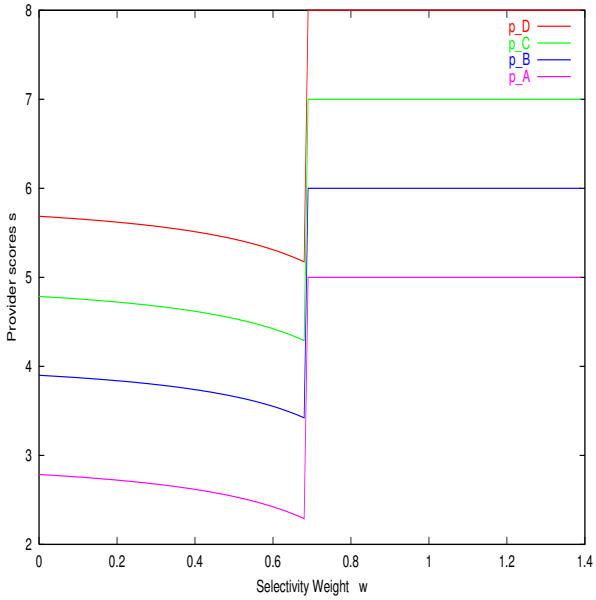

Figure 6.8: Scores for the same two group comments, but algorithm Initialisation Vector on the smaller group.

6.2.1 One Group Removed From The Two Opposing Ones

In order to verify the effect of the second cluster's influence, despite being dominated by the larger cluster of raters, we look at this same example, with the second cluster's comments removed. Only comments from raters $r_{[E,F,G,H]}$ from table 6.2 are included, and in figure 6.7 we can see that rater r_E instead of r_F now becomes the dominant rater when the selectivity variable is applied with a high setting. Therefore, we can conclude that, though the dominated cluster's comments are neglected in the resulting scores, these do help to determine which of the raters in the resulting dominant cluster is dominating the remaining raters of his own cluster when the factor w is set to a higher level. As we see in the resulting scores for the rated providers, the selection of the dominant rater within one group of raters is significant for the resulting scores.

6.2.2 Alternate Convergence Point

In this section, we further investigate the impact of the presence of the second, smaller group of raters. What if we start our algorithm's computation with an initialisation vector that favours this second smaller group? We replace the scores of the initialisation vector $s^{(0)}_{p_{1,..,k}}$ from equation (5.1) in section 5.4 by the values of the comments that were submitted by raters $r_{[I,J,K]}$. This should in certain cases allow the algorithm to yield different results. In figure 6.8 we see that for $0 \leqq w \leqq 0.68$ the algorithm's converges to the same output scores as with the original initialisation vector in figure 6.3. Contrary, for $w \geqq 0.69$ a second convergence point exists and the scores converge entirely on the comment values of the smaller group.

As soon as the second convergence point exists, the results from the reputation model are questionable. The intuition behind this is, that if for the same comments and selectivity weight, we are able to obtain more than one possible outcome, there is no model-inherent way to determine which of the outcomes to choose. The motivation for choosing simple averaging as an initialisation vector in equation (5.1) was to speed up the convergence. If, however, choosing a different starting point leads to a different outcome, this also casts doubt on the model's ability to yield any conclusion in this scenario. Therefore, we suggest limiting the settings of w for this scenario to the range of the unique solution scores: $0 \leqq w \leqq 0.68$. And if one is interested in the outcome scores with the strongest possible application of the rating model, one would choose the selectivity weight to $w = 0.68$.

6.3 Many Diverging Rater Groups

Now we examine the case of four groups of raters, each with different offsets in their rating patterns. While the example with the two opposing groups (section 6.2) was intended to investigate the effective model outcome where the overall opinion is a contest, in this case the input comments to the model can be considered overall divergent and erratic. Looking at the comment values of this table, it is not possible to extract some coherent scores the providers deserve or any that one would expect this model to find. Although the comment values within one group are placed fairly close together, there is no overlap between the groups' comments for any of the rated providers. Table 6.3 shows these four groups that contain between two and four raters each. In figure 6.9 we can see that the model output scores with increasing selectivity factor $w = [0, .., 1]$ stay very close to the plain averaged scores where $w = 0$. This development is to be expected, because the model is eliminating the influence of the groups with the larger offset gaps to the overall average at the same rate. For settings of $w > 1$, the model scores fall very sharply onto the comment values of the one rater, who is closest to the overall average. This sharp turning point at $w = 1$, which for example does not occur in the example scenario of section 6.1 (figure 6.1), is due to the high divergence of the comments submitted by the raters. Effectively, these output scores, either simply averaging ($0 < w < 1$) or selecting one rater ($w > 1$), are as much meaningful as the input is, given how little common ground the comment values contain.

In figure 6.10 we evaluate the different convergent points of this scenario. We start the algorithm with the initialisation vector set to every possible rater's comment values. To keep it clear, we show only the scores for provider p_A, the other providers are very similar at for the given offset differences. This comparison shows that there can be as many convergence points as there are different rater comment sets, given a high enough setting of w, when the algorithm will rapidly converge on the comment set of this rater. Nevertheless, for a reasonable range of $0 \leqq w < 0.7$, the model is able to reach a unique convergence point, independent of the initialisation vector setting, even if this largely is only an averaging of all received comments.

While the model's resulting scores of this experimental scenario are neither particularly

	p_A	p_B	p_C	p_D
r_E	0.0	1.0	2.0	3.0
r_F	0.1	1.1	2.1	3.1
r_G	1.8	2.8	3.8	4.8
r_H	1.9	2.9	3.9	4.9
r_I	2.0	3.0	4.0	5.0
r_J	2.1	3.1	4.1	5.1
r_K	2.9	3.9	4.9	5.9
r_L	3.0	4.0	5.0	6.0
r_M	3.1	4.1	5.1	6.1
r_N	3.9	4.9	5.9	6.9
r_O	4.0	5.0	6.0	7.0
r_P	4.1	5.1	6.1	7.1

Table 6.3: Comments Submitted by Four Opposing Groups of Raters

informative nor surprising, it is worth noting that the model responds in a neutral fashion to very divergent comment input values. This makes the model overall more generally applicable.

6.4 Algorithm Convergence

In this section, we investigate how well the model algorithm converges. As this algorithm is an instance of multi-dimensional convergence, any formal analytical analysis would be extraordinarily difficult, and therefore we use simulations and challenging scenarios to perform this analysis. First, we argue why the algorithm normally converges, except for in special deadlock scenarios (section 6.4.1), and secondly, we assess the speed of this convergence in a challenging setting(section 6.4.2). We continue by explaining the criteria for convergence (section 6.4.3) and describing the different directions the convergence can take (section 6.4.4).

6.4.1 Conditions for Algorithm Convergence

In this section we discuss the conditions for the convergence of our iterative algorithm. We would like to prove that this algorithm converges in all situations, but there may be some conditions under which it may not converge. Here, we will explore the necessary conditions for non-convergence of the algorithm, to convince ourselves that most of the times the algorithm will converge.

First of all, if score updates were monotonic from iteration to iteration, convergence would be trivial, but, as we can see in figures 6.14, 6.15 and 6.16 from the scenario of section 6.4.2, score updates are not in all cases monotonic.

What range of values can the scores assume? The scores themselves are limited by

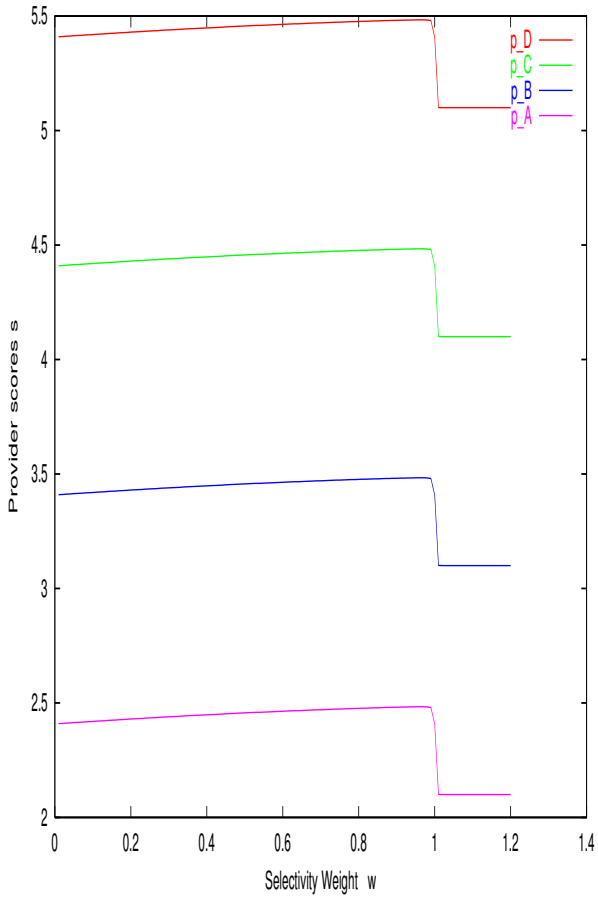

Figure 6.9: Scores with four opposing groups of raters.

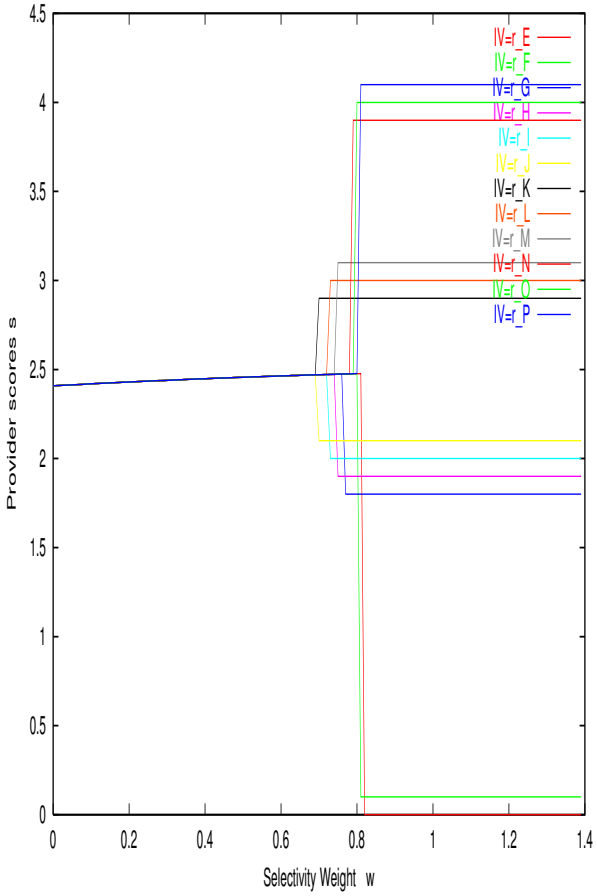

Figure 6.10: Scores for provider p_A depending on the Initialisation Vector (IV) supplied by comments of each rater $r_{E,..,P}$.

the range of values of the comments. For any rated provider p_B, the outcome scores s_{p_B} cannot lie outside of the minimal and maximal comment value $c_{t,r_* \mapsto p_B}$ given for this provider: $s_{p_B} \in [min(c_{t,r_* \mapsto p_B}), .., max(c_{t,r_* \mapsto p_B})]$. From this observation, the score updating footprint of our iterative algorithm can either follow a path that leads to eventually ever smaller update step sizes, or otherwise follow a cyclic path. A third path is not possible, because of the limited range the scores can assume. In the first case, the ever smaller update step sizes, our algorithm would consider this to be a case of convergence, as soon as the update step size underpasses a set threshold. The second case would not lead to convergence, but instead an infinite loop.

What is necessary for the scores to cycle continuously? First, the comment values have to be arranged in a circular contradictory fashion, such that the comment values cancel each other mutually out. This could work as follows: The current iteration's scores raise the influence of a particular set of raters, which alters the scores of the subsequent iterations in such a way that a second set of raters gains influence. This second set of raters then influence the scores to move into a different direction. Continuing this process transitively, the scores may end being moved back to the initial point. Such a scenario is possible and would resemble an instance of the Arrow impossibility theorem.

How likely is it to reach such an infinite loop? First of all, the comment values have to cancel each other out perfectly. A slight imbalance will not lead to an infinite loop. But, actually, constructing such a set of comments is not very difficult. However, the difficulty then comes in finding the right initialisation vector $s_{p_B}^{(0)}$ that leads into this infinite cycles. Without the correct corresponding initialisation vector the algorithm will converge to an result that is nearest to this initialisation vector. We were not able to produce such an corresponding initialisation vector for a minimal system of mutually contradictory comment values leading to infinite cycles. Note that it is less hard to find a statically stable initialisation vector that supports the contradictory state, but in this case the algorithm would stop immediately, because the influence would not be redistributed at all. In conclusion, we find that the iterative algorithm may not converge in all cases, but practically such cases are rare and hard to find.

6.4.2 Rate of Convergence

In addition to assuring the convergence of the reputation model, the rate of this convergence is important as it might interfere with the scalability of the algorithm if the rate is too slow. In general, the rate of conversion depends very much on the circumstances of the rating scenario and in most cases the algorithm will converge after a very few iterations. The rating algorithm will take more iteration cycles for convergence if the model output scores are significantly different from the plain average of all comments.

In order to evaluate the convergence rate, we set up the following scenario that is attempting to challenge the algorithm and is characterised by figures 6.11 and 6.12. The situation is deliberately set up to be divisive to such a degree that it does not have any other useful outcome or application. The purpose of this scenario is to explore and demonstrate the convergence performance of the rating algorithm when the model is pushed to its technical limits. The scenario is made up of 20 providers, who are rated by 100 normal distribution randomised raters with a high standard deviation of $\sigma = 25.0$. There are two equal size groups of 50 raters, where one of the groups adds an offset of $+100$ to all their ratings. As a result we obtain two competing groups of equal commenting power and the model scores only slowly tip in direction of the comments of one of the groups. Figure 6.11 shows that depending on further parameters like the selectivity weight factor w, the resulting model scores swing in favour of one or the other of the two groups. We observe that for $w = [0, .., 1.22]$ the model scores remain on the plain average between both groups. For $w = [0, .., 1]$ this should be expected, as in this area the effect of the model is dampened and geared in direction of plain averages. With $w > 1$, the selective effect of the rating model progressively increases and at about $w = 1.25$, the scores start to tip in direction of one of the groups. This polarisation continues, except that at $w = 1.78$, the conditions under which this randomised scenario is run all of a sudden favour the other group. Although the initialisation vector is the same, the default starting average, the scores flip and gravitate in direction of the other group. Note, that the scores that can be obtained at this point are not particularly useful beyond measuring the rate of convergence itself. The discussion of the convergence points continues in section 6.4.4.

At this point, we want to analyse the rate of convergence of the algorithm under this

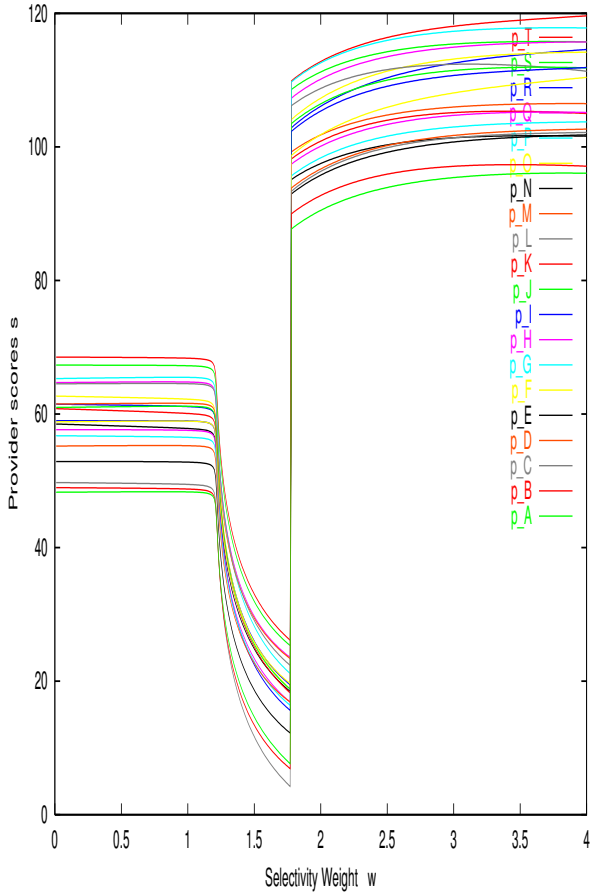

Figure 6.11: Reputation Model Scores for the Divergent Rating Scenario Examining the Convergence Rate.

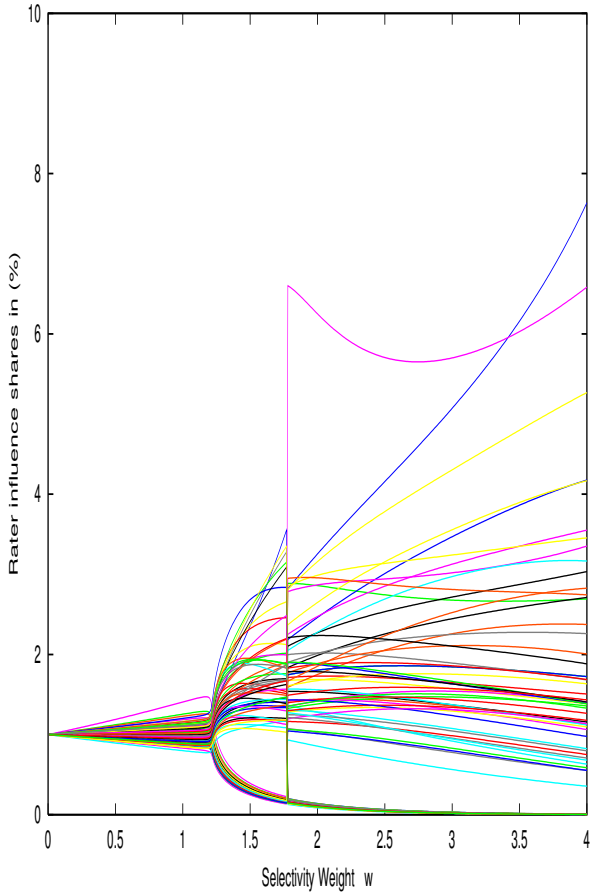

Figure 6.12: Influence Shares for the Divergent Rating Scenario Examining the Convergence Rate.

divergent scenario. Figure 6.13 displays how many iterations the rating model requires to converge, depending on the selectivity weight factor w. It is apparent how the difference in convergence rates correspond to the different model score outcomes from figure 6.11. For a setting of $w = [0, .., 1.0]$ we know that the model's influence is dampened, thus the scores converge rapidly, as they stay near to the initial averaging of the comments happening before the first iteration. The rate of convergence then slows down dramatically to spike at $w = 1.22$. The scores at that point move away from the average only slightly. In these conditions, scores are updated by a small amount just above the convergence threshold for a long time, sliding to a stable point, the model effect is not strong enough for sharp shifts or to force a decision between the commenting blocks. Beyond this, the model requires about 30 iterations for $w = [2, .., 4]$, as in this area all the effective scores diverge much from the average and such requires a considerable number of updates, even if the model effect is relatively strong at this point. At $w = 1.77$ we observe a small moderate reduction in the convergence rate, coinciding with a change in direction, the model's outcome scores are taking. We highlight this in figure 6.14, where we show the largest of all score updates and its direction in a given iteration cycle for different values of the selectivity weight factor. We notice that the direction flips between $w = 1.77$ and $w = 1.78$ and that with higher values of w the larger model score updates happen sooner

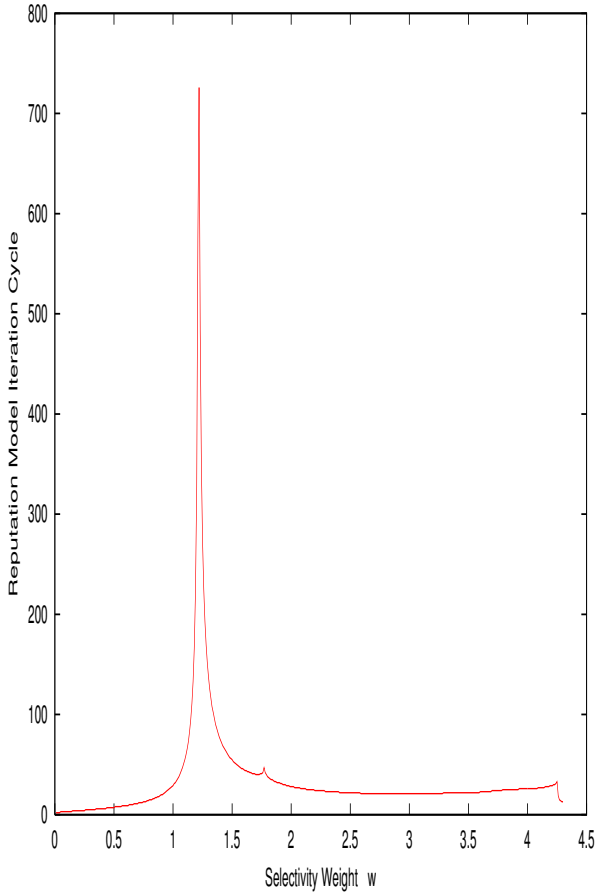

Figure 6.13: Iteration Cycles Required to Converge for the Divergent Rating Scenario, Depending on the Selectivity Weight.

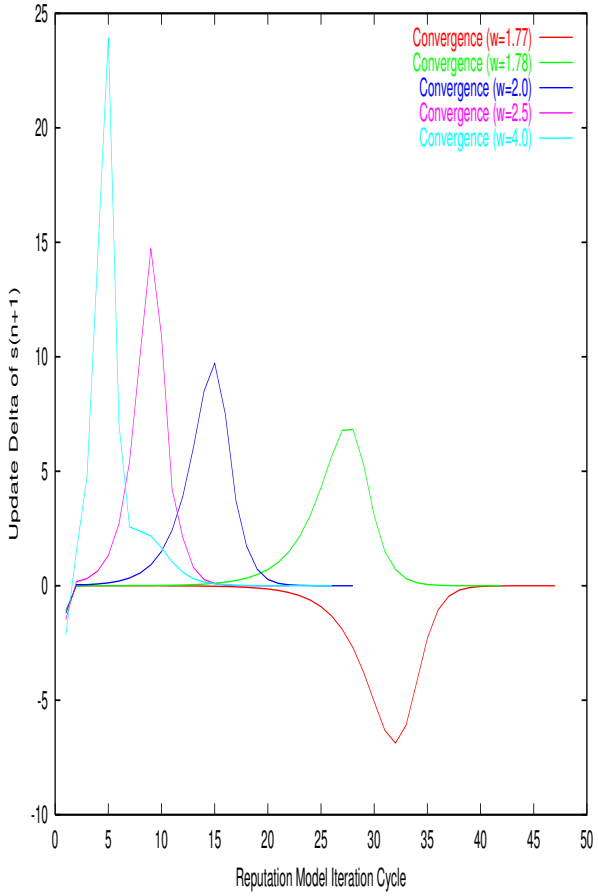

Figure 6.14: The Largest Update Step Sizes of Scores per Iteration for Selected Selectivity Weight Values.

because of the model effect being more pronounced.

Although the algorithm can in certain cases take several hundred iterations to converge, this only happens for specific model settings and particularly challenging commenting scenarios. In the vast majority of cases, the algorithm converges very rapidly and even in the few cases where it takes some more iterations to converge, progress is steady enough to ensure predictable timely convergence.

6.4.3 Convergence Threshold

A determining factor in measuring the rate of convergence in the above section is the criteria by which we determine when to stop the iteration. We chose a simple convergence threshold, and if the largest iteration step falls below this threshold, we consider the convergence terminated. We found a threshold of 0.0001 to be adequate, because by our observations, in this case the iterative approximate result does not differ much from its theoretical convergence limit for $threshold \to 0$.

One theoretical limitation remains, if for example the convergence starts out with

iteration steps that are below the convergence threshold, the algorithm stops, even if subsequent iteration steps would be more significant before approaching the limit. It would have been possible to consider such cases and develop a convergence criteria that also takes the gradient of iteration steps into account. One could require that the maximum iteration step has to having been decreasing over the last few iterations, as well as being below the convergence threshold. However, as we did not encounter any such cases even in our contrived scenarios of the above section, which attempts to create such a problem, we decided to keep the convergence criteria simple. For example, in figures 6.15 one can see how the iteration steps of the scores remain very small initially, only to gain a visible magnitude after about 18 iterations and to converge again after another 20 further iterations, in order to finally fall below the convergence threshold after a total of 47 iterations. If this limitation were a problem it should occur in a case like this, which is set up to stretch out the iterative process.

6.4.4 Multiple Convergence Points

Depending on the setting of the selectivity weight factor w, any given scenario set of comments has one unique or multiple possible sets of scores to converge to. In fact, for $w = 0$ all scenarios have exactly one convergence point, this being the simple average of all comment values, and for a very high setting of w, as many convergence points exist as there are raters submitting comments in the system. To reach all these convergence points it usually is necessary to alter the initialisation vector of equation (5.1). Figure 6.10 demonstrates how one can reach all the possible different convergence points, by initialising the rating algorithm with each different rater's comment values.

If starting from the same averaging initialisation vector, and continuously changing the setting of the selectivity weight factor w, usually also results in continuous shifts of the convergence point. Commonly, when changing from one to the next convergence point, transitions are continuous. However, discontinuous jumps are possible if the two convergence paths are connected by discontinuous gradients. Such dramatic rifts are visible in the example of the previous section 6.4.2 with the very divisive scenario setup. The setup of comments provides for two equally influential voting blocks with their coherence differing only by a minor random statistical amount. For settings of $w = [0; ..; 1.22]$ that encourage compromising results, the medium between both blocks is chosen, when at $w = 1.22$ the first shift happens, although a continuous one, but from this point onwards the scores converge in direction of one of the voting blocks. The more prominent shift with a discontinuity occurs at $w = 1.78$, where the convergence assumes a slightly different slope and therefore moves in the opposite direction. Figure 6.15 for $w = 1.77$ and figure 6.16 for $w = 1.78$ show the different slopes for these two borderline cases. How this transition comes to be is more clearly visible when comparing the slope of the influence shares during the iterative convergence for $w = 1.77$ in figure 6.17 and for $w = 1.78$ in figure 6.18. The same colour graphs belong to the same rating clients, thus one can see how from a very similar initial redistribution of influence shares, the path of convergence takes opposite directions. The very same clients who dominate in setting A, are marginalised in setting B and vice versa.

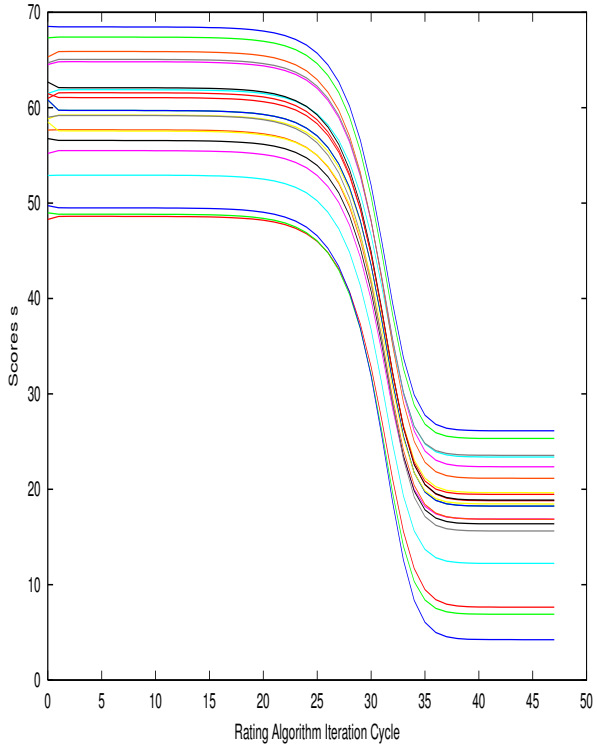

Figure 6.15: Slope of Convergence of Scores for Divergent Example with $w = 1.77$, Converging to Voting Block A.

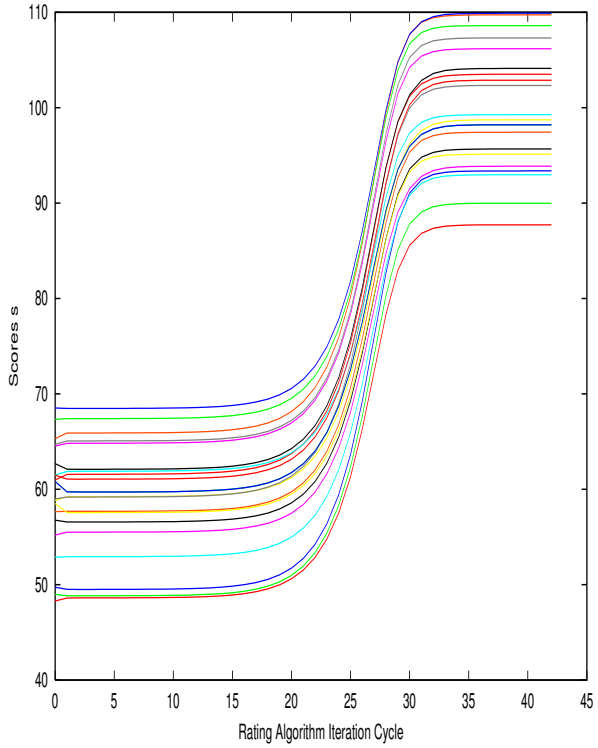

Figure 6.16: Slope of Convergence of Scores for Divergent Example with $w = 1.78$, Converging to Voting Block B.

Note that this analysis is mainly made for technical illustration purposes, as we will discuss in the next chapter, how we will discard resulting scores, if more than one convergence point exists for the current setting of w.

6.5 Conclusions

In this chapter we were able to demonstrate that our rating model, as presented in chapter 5, is able to select the scores derived from a majority group (sections 6.1 and 6.2). When presented in section 6.3 with a more convoluted situation of four groups of opinions, it is able to select scores that seem to be in line with the one group that might be the most amenable to all groups. On the same case scenario we show how choosing a different initialisation vector can be used as a vehicle to gear the outcome scores in the direction of certain of the rating groups. However, all the outcome scores significantly depend on the selectivity weight variable, which turns out to be a useful parameter for adjusting the model's degree of "discriminatory selection" versus "equalising inclusiveness".

Further, we demonstrate (1) the technical feasibility of the iterative algorithm, in that it normally converges (section 6.4.1), show that (2) it does so usually very rapidly (section 6.4.2), and (3) explain how the convergence can take different directions to different convergence points, depending on the starting vector one has chosen to use (section 6.4.4).

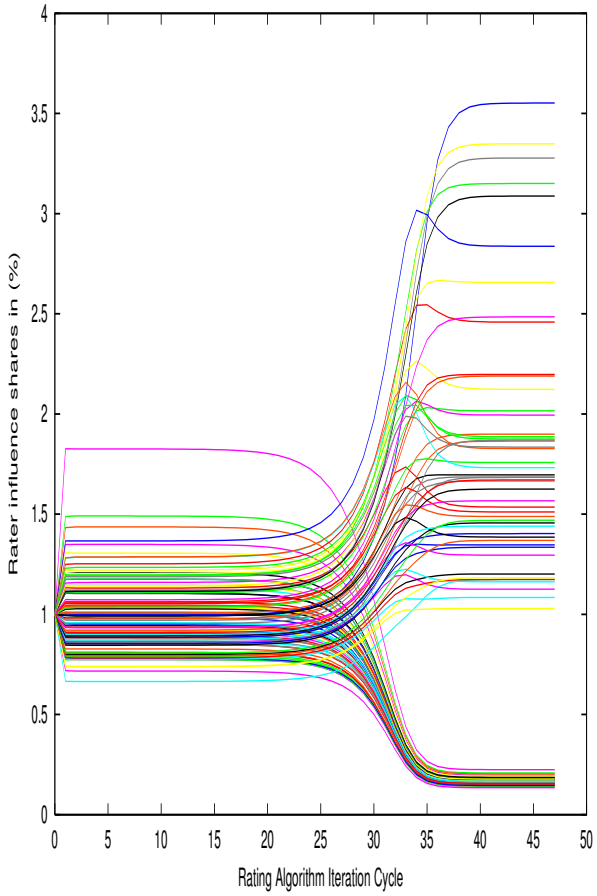

Figure 6.17: Slope of Influence Shares During Convergence for the Divergent Example with $w = 1.77$, Converging to Voting Block A.

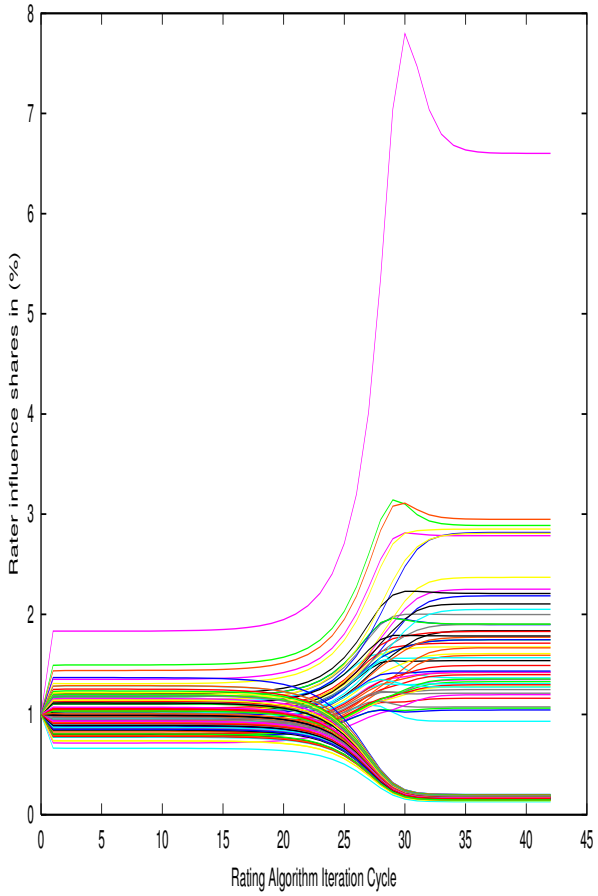

Figure 6.18: Slope of Influence Shares During Convergence for the Divergent Example with $w = 1.78$, Converging to Voting Block B.

In the next chapter we turn to the actual use of the aggregation model and present a series of scenarios that analyse the model's contribution when the scenario is too large for a human observer to readily see the intended solution, as it was possible in the cases of the first three sections of this chapter. We will use statistical models of scenarios with one set of raters being "normal" and one "deviant" set of raters with a different agenda. We are interested to see if the model under adjustment of its selectivity variable is able to identify the different sets of raters, and if we are able to set the model conditions such that in all of the scenarios we are able to marginalise the effects of raters who we set out to be "deviant". We also discuss further application possibilities without modelling them explicitly in case studies.

Chapter 7

Applications of the Reputation Model

After having assessed the technical properties of the reputation aggregation model in the previous chapter, we turn in this chapter to application-oriented case scenarios with larger numbers of clients that display statistically modelled behaviour. The challenges for this reputation aggregation model are raters with "deviant" debilities or agendas, because if all raters were to be cooperative, an aggregation function with simple averages would be sufficient. Therefore, we model our threat cases in this chapter with two sets of raters: One "normal" set that behaves cooperatively and a "deviant" set of raters that resembles the threat. Since we do not limit the range of what "deviant" behaviour could be, we arrange a few different types of threat scenarios, and then push the challenge it poses further to see where the model's limit is to contain the threats. This approach motivated our choice to work with synthetic threat models, because these allow us to create threat models of arbitrary strength.

The threat cases we investigate start with a case where the "deviant" raters on average agree with the "normal" raters, but have a higher variance in their comments (section 7.2) and then we increase the numbers of raters with this high variance. In section 7.3 we raise the challenge of the threat model with two *Byzantine style*[1] manipulation attempts, where a minority of raters attempts to "rig" the outcome recommendations according to their own agenda. In section 7.4, we consider a different kind of threat model, in which the providers do not deliver the same performance to all clients. Does this pose a general problem for our rating aggregation model, since the raters then report different levels of performance? The common question for all these threat scenarios is whether the aggregation model is able to respond equally well when it has to deal with larger numbers of comments, since in the last chapter we limited ourselves to small-scale scenarios.

Following, we discuss issues that are relevant for the practical operation of the reputation system, beginning with section 7.5, in which we explain how to adjust the crucial selectivity weight parameter, then in section 7.6, where we describe how the outcome

[1]We labelled this threat model *Byzantine style*, because it contains a malicious collective of raters, analogous to the Byzantine Fault model in computer security, which contains a bribed collective of generals in a distributed synchronisation problem (see section 6.2.1.1 in Anderson's book [5]).

recommendations can be customised for specific users and in section 7.7, where we discuss the possibility of aging comments over time. Further, in section 7.8 we discuss some limitations of the reputation aggregation model, and continue with a review of the reputation aggregation function from a point of social choice theory. We conclude with a section (7.10) that describes a range of potential applications in different areas to show how versatile the aggregation model's use could be.

7.1 Statistical Analysis

The scenarios of the following two sections (7.2 and 7.3) contain 100 raters and 20 providers. In these scenarios, the providers are set to perform at a certain average performance level, which is characterised by a numeric value. The goal of our rating model is to identify these performance levels from the comments submitted by the raters. The scenario assumes that raters are not able to capture the performance level of a provider perfectly, because performance is compiled from the quality of interactions over a time period. But these scenarios assume that the providers will on average perform to a certain measurable level, therefore comments are drawn from a random variable with a normal distribution where the mean equals this performance level. While each provider will have a different performance level to show the figures more clearly, one should note that the choice of the actual level is irrelevant to the rating model, only the deviations from this level are relevant.

7.2 Different Rating Accuracy

7.2.1 Few Poor Raters

First, we confirm that the rating model is able to discern raters with a weak rating consistency, similar to the limited example of section 6.1. Therefore, we set up a scenario with sets of raters of different rating accuracy. Figures 7.1 and 7.2 show a scenario where a group of 90 raters has a low standard deviation of $\sigma = 0.25$ and a second group of ten raters has a relatively high standard deviation of $\sigma = 1.0$. In figure 7.2 we can identify the two distinct bunches of graphs, the top bunch corresponding to the group of low deviation raters and the lower bunch relating to the ten raters with the high deviation. Starting from a selectivity weight factor $w \mapsto 0$ where all raters have the same influence, with increasing w, these two bunches separate quickly and for $w > 1.5$, the high deviation group's influence is negligibly low. At the same time this influence is handed to the more accurate raters of the first group. For the corresponding scores, however, we see in figure 7.1 that these are very stable and therefore independent of w, because the number of raters with the high standard deviation ($\sigma = 1.0$) is proportionally small. Moreover these deviating raters' probability distributions have the same mean as the other raters.

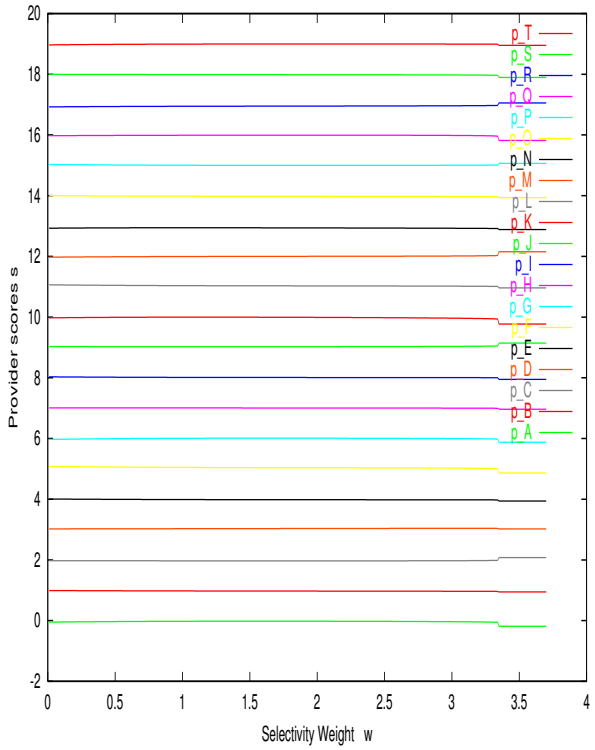

Figure 7.1: Scores for a Mix of 90 Raters with $\sigma = 0.25$ and Ten Raters at $\sigma = 1.0$.

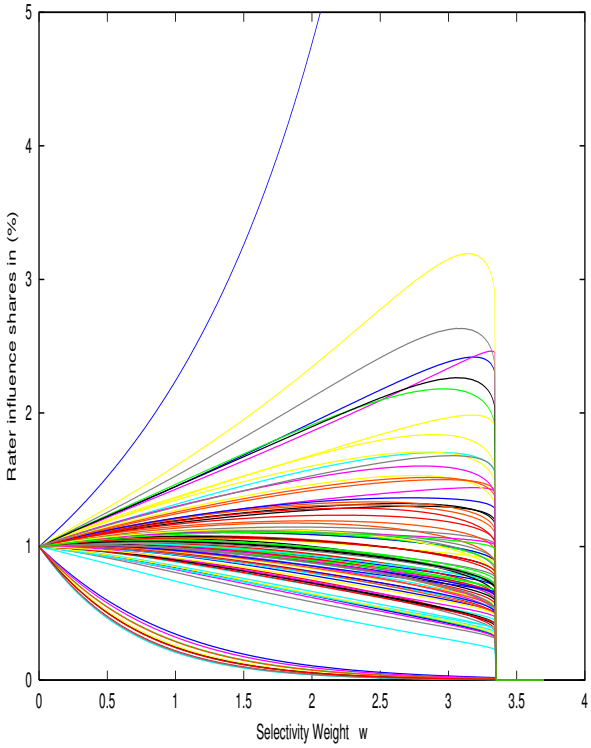

Figure 7.2: Influence Shares for a Mix of 90 Raters with $\sigma = 0.25$ and Ten Raters at $\sigma = 1.0$.

7.2.2 A Large Majority of Poor Raters

Next, we want to compare how the rating model responds to a larger number of raters with a high standard deviation, seeing as the group of such raters in the previous example made up only 10% of all raters, which is far from dominant. Figures 7.3 and 7.4 show the results for a scenario where the tables are turned from the previous one, this time ten of the raters have a standard deviation of $\sigma = 0.25$ and the other 90 have the high deviation of $\sigma = 1.0$. Again, we can identify ten graphs that diverge significantly from a large bunch of 90 other ones, that lose their influence gradually. In this scenario, one would choose a selectivity factor of about $w = [2.5, .., 3.0]$ in order to put the balance in favour of the ten raters with a low deviation. In doing so, we would ensure that the scores are only drawn from raters with the higher accuracy. Just as in the previous example, the resulting scores in this scenario do not vary much in relation to the selectivity factor and the corresponding influence distribution. In this case, that is to be explained with these 90 raters collectively not diverging on the rating scores, as they have the same mean as the others.

Both of the above demonstrations show how raters with inaccurate ratings are marginalised. While in these demonstrations the scores are not negatively impacted by the presence of these inaccurate raters, it is a vital aspect of the rating system that such raters are discriminated reliably, because we expect such comments to be misleading and to have an impact when these inaccurate comments do not cancel each other out in a high number

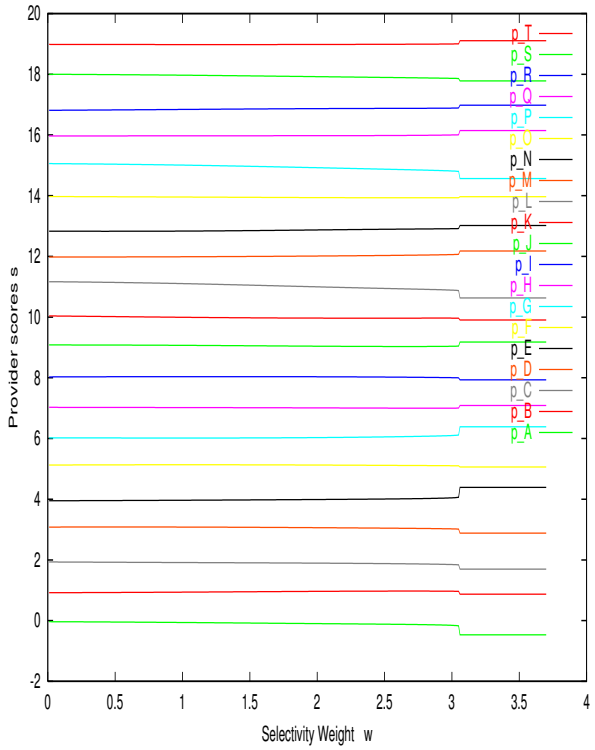

Figure 7.3: Scores for a Mix of Ten Raters with $\sigma = 0.25$ and 90 Raters at $\sigma = 1.0$.

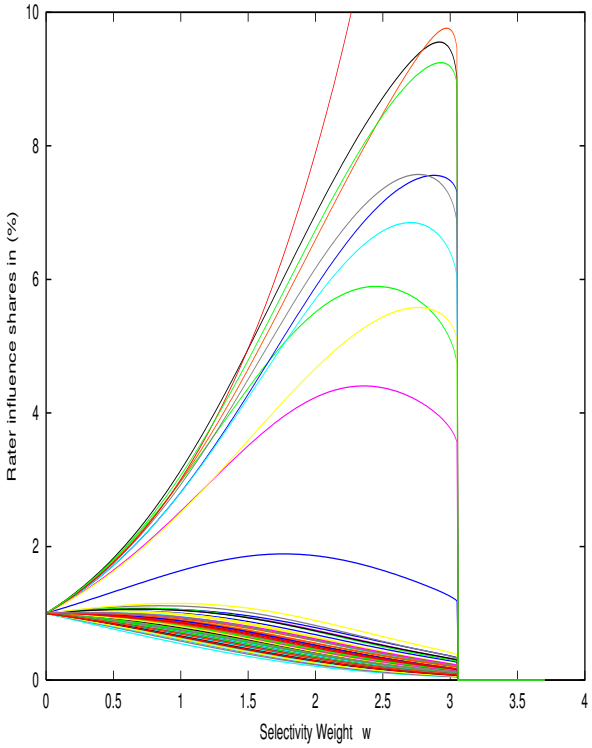

Figure 7.4: Influence Shares for a Mix of Ten Raters with $\sigma = 0.25$ and 90 Raters at $\sigma = 1.0$.

of rating cases.

7.3 Filtering Deviating Commentators

The most important threat model of "bad" ratings is a malicious collective of raters who attempt to influence the score outcome in a coordinated and directed fashion. In order to make this set of deviating raters a strong challenge, these raters will have the same low deviation ($\sigma = 0.25$) as the raters with the "correct" comments.

7.3.1 Malicious Rater Collective Manipulating All Scores

In figures 7.5 and 7.6 we show the effects of a scenario with 40 of the 100 raters applying a deviating commenting agenda. These 40 raters apply an offset of -2.5 to all their ratings of all providers[2], where the other raters maintain the same rating pattern as in the above examples. In figure 7.5, we see how the scores are affected by this deviation, because for

[2]The offset value of -2.5 was chosen such that one could visibly recognise the impact of the deviators on the scores, with the combined scores effectively lowered by an offset of -1. One could also question if a lower absolute value for such a deviation would amount to a threat model or would have to consider such comments valid and rightfully include these in the scores.

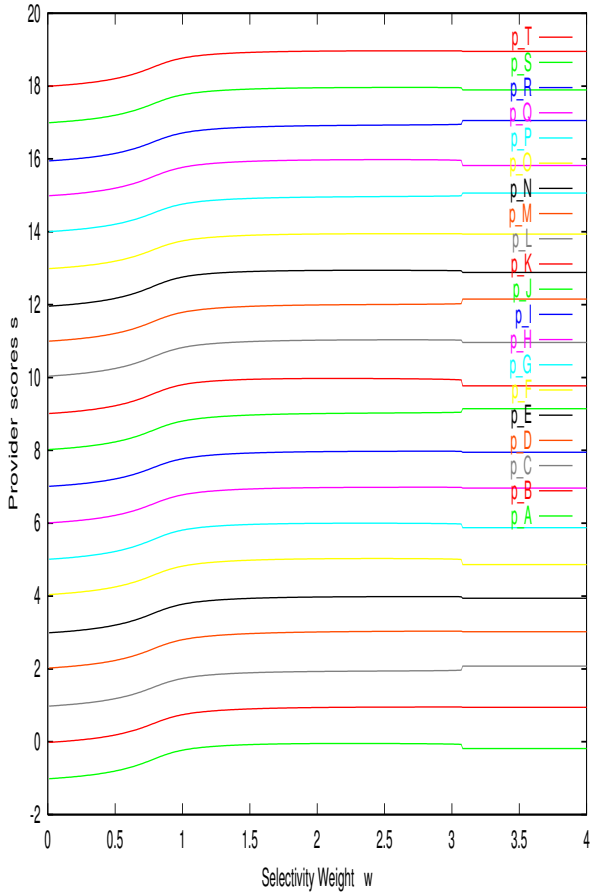

Figure 7.5: Scores for a Scenario with 40 Raters Deviating with a Rating Offset (−2.5) Applied to All Providers.

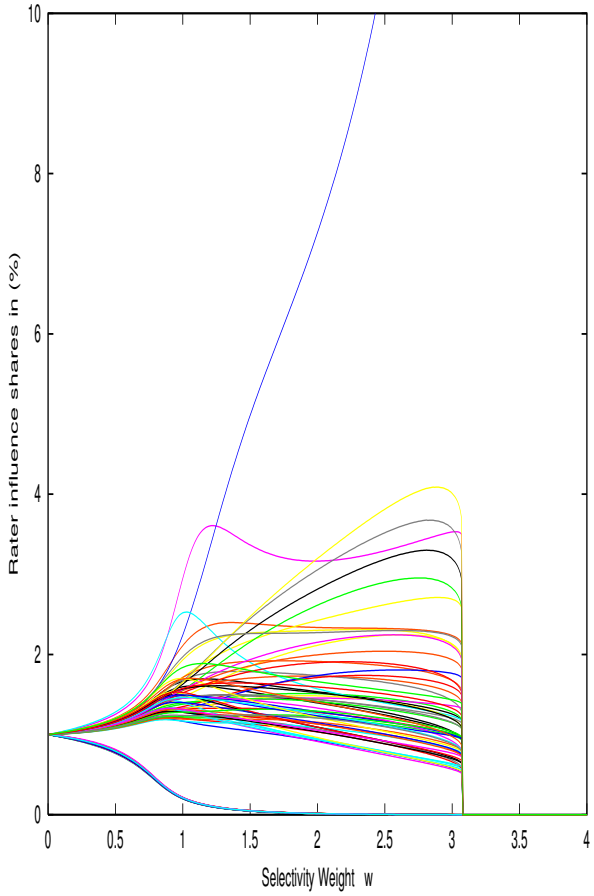

Figure 7.6: Influence Shares for a Scenario with 40 Raters Deviating with a Rating Offset (−2.5) Applied to All Providers.

$w \leqq w < 0.5$ the scores are lowered by one unit from the "correct" values. Increasing w makes the model more selective and for a value of $w > 1.0$ the impact of the deviating raters on the scores and their influence on these is mostly removed.

Evaluating figure 7.6, one could wonder about another effect: While the impact of the deviators is removed, some raters of the other group are gaining a higher influence share, which they lose after the influence of the deviators is entirely removed. In this transition phase, the resulting model scores are shifted and some of the 60 "good" raters happen to be slightly closer in their rating bias to the deviators. After the deviators are entirely removed, the resulting scores move toward the average of all of these 60 "better" raters.

At this point, we would like to determine an appropriate setting for w in this scenario, such that deviators are rightfully disabled, but the other raters form a meaningful aggregation. Similarly to the simple examples in previous chapter (section 6.2.2), we search for a second convergence point. We replace the initialisation vector in equation (5.1) with the comment values of the deviators. Running the altered reputation model on the same scenario yields the scores seen in figure 7.7. For values of $0 \leqq w \leqq 1.36$, the scores and the influence distribution (figure 7.8) are identical to the results with the general initialisation vector (figures 7.5 and 7.6). At $w = 1.37$, the second convergence point is available, the

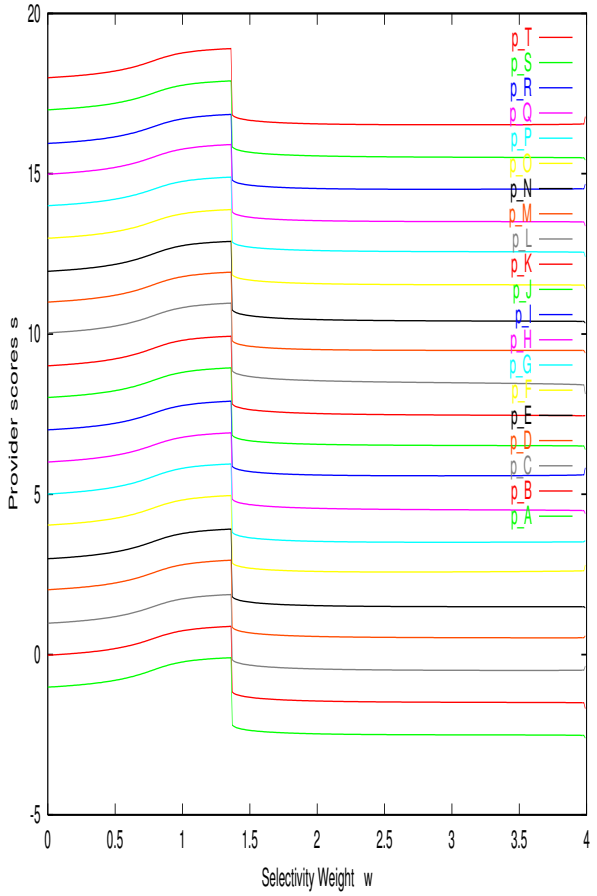

Figure 7.7: Scores for same scenario with the Initialisation Vector set to the deviators' comments.

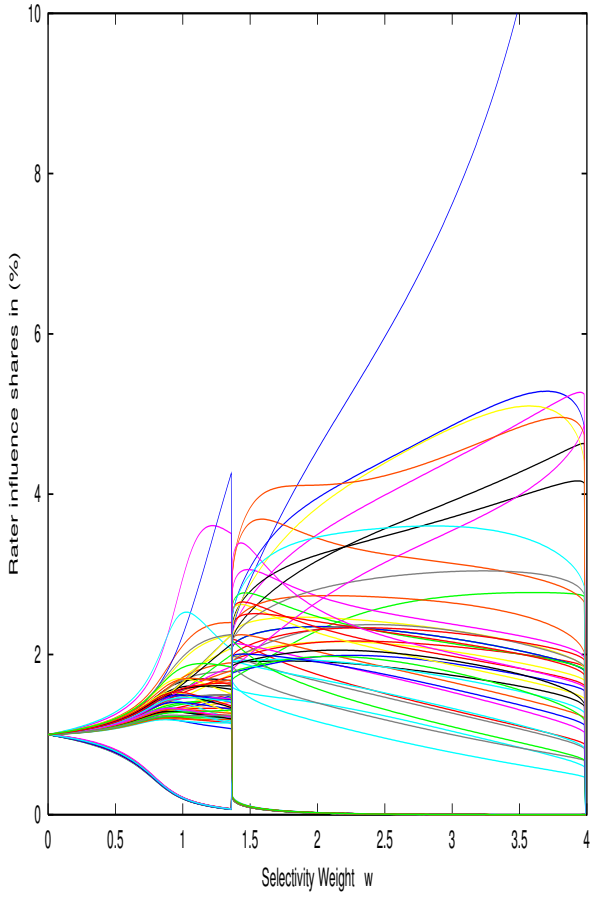

Figure 7.8: Influence Shares with the Initialisation Vector favouring the deviators' and their convergence point.

scores follow a discontinuous jump to the scores of the deviator group and the influence distribution is reversed for both groups. With the existence of the second convergence point, the solution is not unique and it is debatable which of the convergence points should be chosen. Therefore we choose to disregard results that offer two convergence points.[3] Within the range of possible unique model solutions, we want to maximise the rating model's selectivity effect in order to enable the model to filter out the deviating raters. Choosing $w = 1.36$ satisfies both of these criteria and yields us model solution scores that are practically free of influence taken by the deviators, seeing as collectively their influence amounts to only 2.79% at this point.

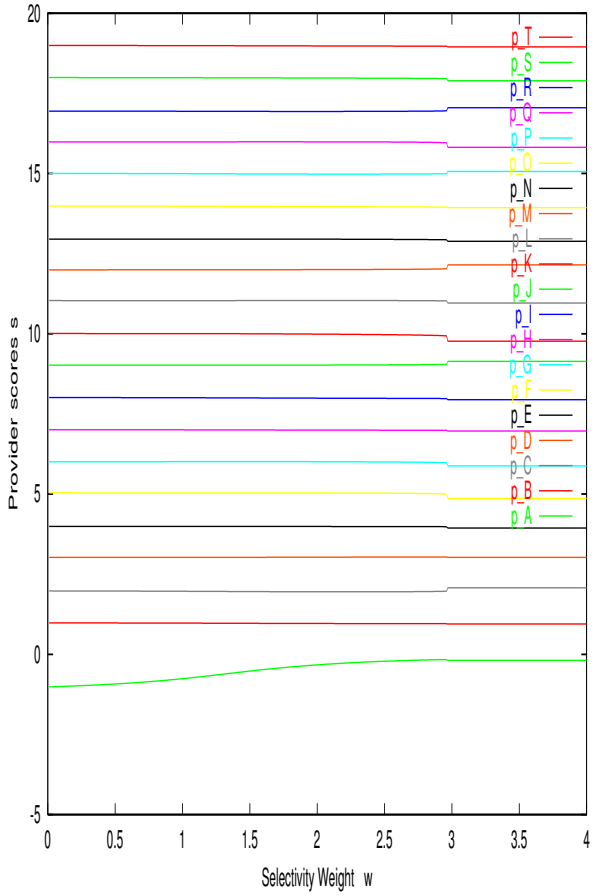

Figure 7.9: Scores for 40 Raters Deviating with a Distinct Rating Offset (−2.5) on Provider p_A Only.

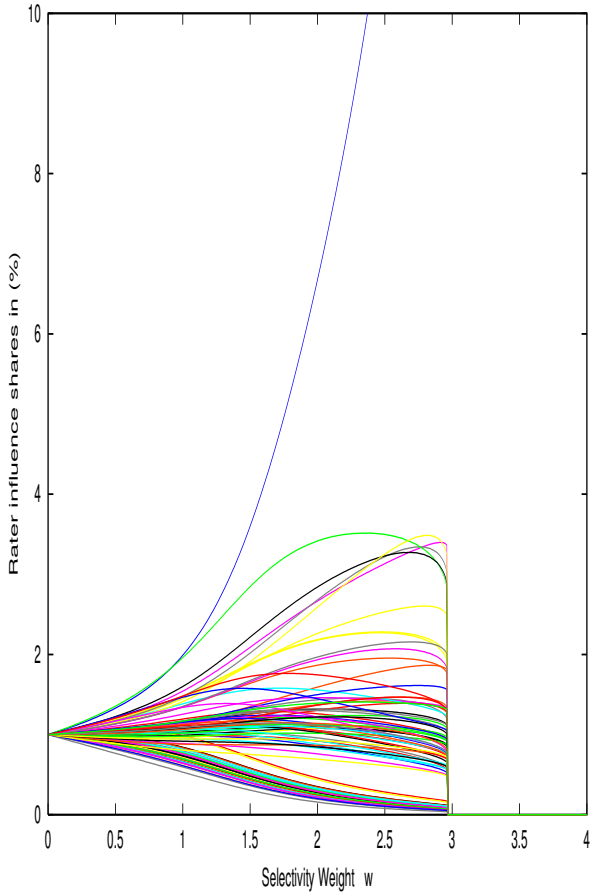

Figure 7.10: Influence Shares for 40 Raters Deviating with a Distinct Rating Offset (−2.5) on Provider p_A Only.

7.3.2 Malicious Rater Collective Manipulating One Provider Score

While a threat model with a malicious collective of 40% of the raters applying a rating bias is large fraction, it is not the most challenging threat case, because in the previous section the deviators applied this bias to all providers that are to be rated. It is much harder to identify and deselect these deviators if they attempt to modify the rating of only one of the providers. In this scenario we again have 40 deviators applying similarly a rating offset of −2.5, however only to provider p_A. In figure 7.11 we see how the scores are affected by this deviation, because for $0 \leqq w < 0.5$ the score of p_A is lowered by one unit from the "correct" value. Increasing w makes the model more selective and eventually for $w > 2.5$ the rating scores reflect the rating pattern of the 60 "good" raters only. From figure 7.12 we see though, that the gap between the graphs of the 40 deviating raters

[3]We can choose to take into account results that allow for more than one convergence point, if we have a method for choosing the desired initialisation vector to reach the desired convergence point. One such method is to take a client-specific view, and select the convergence point according to this client's comments, as discussed in section 7.6.

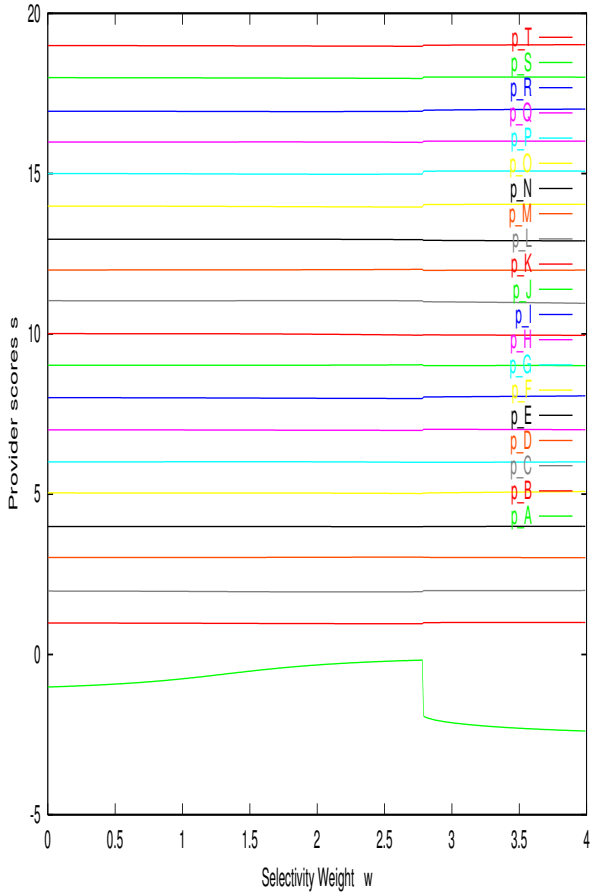

Figure 7.11: Scores when algorithm initialised in favour of second convergence point, belonging to the deviators.

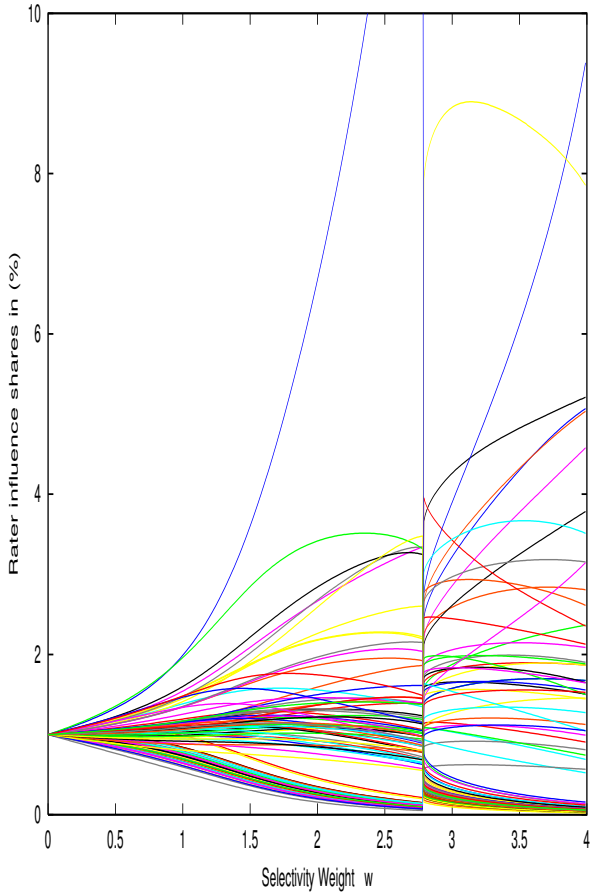

Figure 7.12: Influence Shares with Initialisation Vector favouring deviators' convergence point.

(represented by the graphs close to 0 in the range of $2 < w < 2.78$) and the 60 other ones is very narrow. This shows that the rating model is pushed to its limit, although it still does achieve the goal of removing the effect of the deviators for a selectivity weight value of about $w > 2.5$.

Also in this threat scenario, we want to determine the appropriate setting for the selectivity weight variable w. As in the previous section, we replace the initialisation vector by the comment values of the deviators to yield figure 7.11. In this case, the second convergence point is reachable for $w > 2.78$. Choosing $w = 2.78$ is a satisfying result in terms of deselecting the deviators, as the results are mostly free of their influence, which amounts to only 4.41% at this point.

Summarising, we are able to claim that the rating model performs well in both of the above threat scenarios. Even for a large deviating group resembling 40% of all raters and minimising the deviators' exposure, the model is able to identify the deviators and remove their impact on the scores.

7.4 Deliberately Inconsistent Provider Performance

The scenarios we discussed in the previous sections assume that providers display a consistent performance behaviour. This does not imply that they have to deliver identical performance every time, but we assume that clients base their ratings on a number of transactions with a service and that statistically the clients experience on average a similar performance from a provider and that this average is characteristic for this provider. Deriving a reputation with the model under such assumptions for a provider who displays a high variability in his performance would still result in the scores correctly reflecting his average performance.

However, what happens if a provider does not deliver a varied performance by statistical chance, but rather has opted to treat a specific set of clients with a specifically worse or better performance level? For example, provider p_A could decide that he delivers a guaranteed-level 99.99% of the agreed performance terms to a majority of 75% of preferred customers and delivers as low as only 50% of the agreed performance terms to the minority of 25% of the remaining contracting clients, when he needs to use these clients as a best-effort-serviced buffer for his performance demand variances. With the clients then submitting their observations as comments, this would result theoretically in a performance score of 87.49%, if applying a plain average. We would argue that this does not represent the performance of the provider well at all. Such a result neglects the good and the bad of this provider's strategy. In order to make this scenario statistically more realistic and relevant, the majority group experiences a normal distribution with no performance above 100% and a very low standard deviation of $\sigma = 0.25$, simulating the guaranteed performance level. The minority group's normal distribution of comments on the other hand will never fall below 50% and displays an extremely high standard deviation of $\sigma = 25.0$, reflecting the best effort characteristic of their service experience. Figure 7.13 compares the model's score outcomes for the two possible convergence points of p_A's rating and practically, with the statistical circumstances mitigating p_A's performance bias, a plain averaging of comments results in a flattering rating score of 91.4%. The remainder of this scenario contains again 100 raters, with 25 reporting the lower performance and a total of 20 providers, where the other 19 all behave consistently and predictably. While the first convergence point is found from the overall average as the initialisation vector, the second is derived from an initialisation vector formed from the comments the minority group is submitting. Similar to the previous scenarios, when we set the selectivity weight high enough, beyond levels where it mediates between the two groups, the second convergence point appears, that is for $w \geqq 1.85$. At that point, we can clearly identify the two general performance levels (99.99%; 50%) this provider is supplying to the two groups of clients. We can individually identify which of the clients belong to either of these two groups by observing the exact reversal of the influence shares held by either group for the different convergence points as we can see for $w = 1.85$ in figure 7.14. Ironically though, when increasing the selectivity weight, eventually for $w \geqq 2.83$ the resulting scores centre on one of the minority group raters who happens to be very close again to the plain average which we obtained with $w = 0$.

From the point of view of the scores, this scenario is identical to a set of clients with a rating bias, as we described in section 7.3. By analysing the submitted comments it is

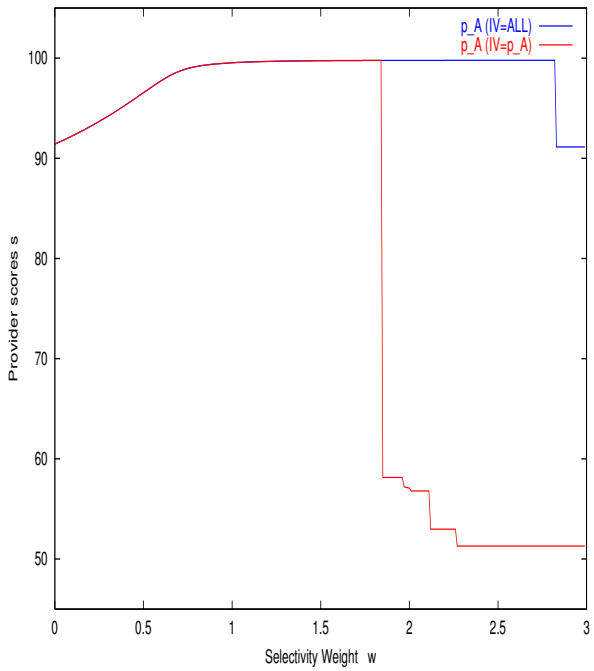

Figure 7.13: Scores for both convergence points where provider p_A is deliberately delivering inconsistent performance.

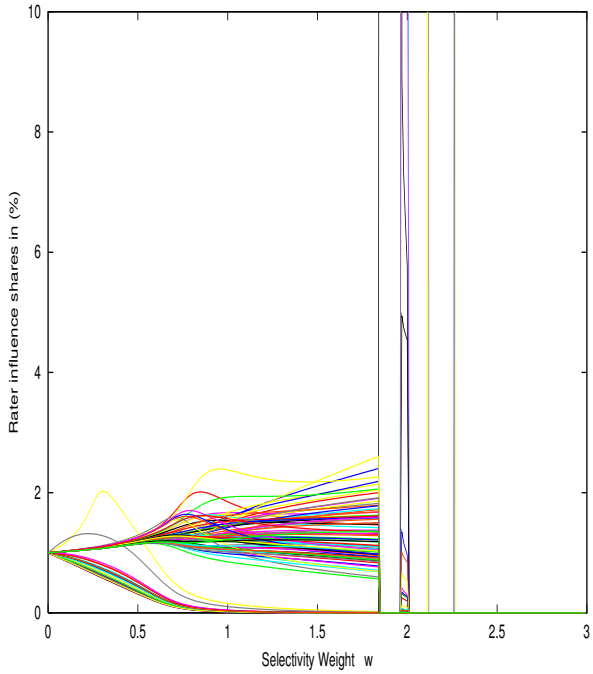

Figure 7.14: Influence distribution with initialisation vector set on comments of minority group and revealing the second convergence point from $w \geqq 1.35$.

not possible to distinguish between a provider actually treating a client differently than others or a client turning out a different rating than the performance he has received. The model however is able to reveal these divergences, particularly if these are part of a general pattern and not random flukes. However, to tell which of the both deviances we are dealing needs to be resolved externally. Thus, the operator of the rating model has to anticipate the kind of threats that are possible or likely to occur in his rating scenario. With knowing which of the two threat cases one is dealing with, one can choose the appropriate convergence point as the effectively reported scores. Alternatively, if one were paranoid about both threat cases at once, one could choose to punish both sides. The affected provider by choosing the disadvantageous convergence point for his reputation scores and the rating clients by offering them a the lower of the two possible payoffs from the economic incentives that are part of the framework, as described in section 5.3.

7.5 Choosing the Appropriate Model Selectivity Weight

As we are able to observe in the previous sections, the resulting scores depend very much on the settings for the model's selectivity weight w. Assigning a value to w ultimately is an application specific task and depends on the intentions of the rating model's operator. Here we list some general rules as to what effect a certain setting of w will have on the scores:

- Settings of w that allow for a second or more convergence points, when supplied with different initialisation vectors, are not desirable.
- Resulting model scores that allow for alternate convergence points can only be accepted if one has a method choosing the desired convergence point and the corresponding initialisation vector. The next section (7.6) provides an example for such an approach.
- If undesirable raters are a critical threat to the rating situation, one will choose a higher setting for the selectivity weight variable w, as in this way the model's discriminatory effect is reinforced. Therefore, commonly one will choose w to be as high as possible, within the range of unique solutions.
- If the goal is to find a non-compromising solution, effectively electing one all dominant rater, one can set the selectivity weight to an extremely high value.
- If we expect a systemic high variance of the submitted comments, we want to dampen the discriminatory effect of the model by choosing a lower setting for w, close to 0.

7.6 Client-Specific Rating Customisation

We are able to produce model scores specifically tailored for one rating client. In doing so, we assume that this client most of all trusts his own ratings and secondarily the other raters' ones. We achieve this customisation by using this client's comments as the initialisation vector, instead of $s^{(0)}_{p_{1,..,k}}$ in equation (5.1) to enter the iterative loop at equation (5.3). We expect this to have an effect on the slope of the rating algorithm and where possible lead to selection of different of the possible convergence points. This approach however presumes that we (1) chose to obtain model scores with settings of w that are so selective that more than one convergence point exists and (2) w is not so selective that the model scores are identical to the comment values of the rating client who supplied the initialisation vector. Such an approach is mostly useful, if the range of settings of w with one unique convergence point is deemed not sufficiently selective and one requires a stronger discrimination.

7.7 Aging Comments

What happens to comments submitted over time and should we emphasise more recently submitted comments? We are able to extend the reputation model to factor in a decay for the comments. The aging would be integrated into equation (5.2), where we compile the local scores $local.s$. At present the local scores simply take the average of the comment values one rater has submitted on a particular provider. Taking the time of submission into account, we can give more relative weight to his more recent comments over the older ones. A special case would be to expire previous comments altogether as soon as a new one is submitted about a provider.

Note that it is not possible to apply aging of comments on a more general level, particularly aging of comments in relation to those submitted by other raters. This cannot be done, because the premise of the rating model is, firstly, to determine a reputation for a rater and the quality of his comments. The rating model must then compile scores based on these rater reputations. It would be conflicting to base the scores on the reputations and at the same time the relative age of the comments. Further, we would weaken the rating model's notion of the rater reputations.

7.8 Limitations of the Rating Model

In case one uses our reputation model for an ongoing rating board that is continuously updated as raters submit new comments, a limitation of the model is that a rater could look up the current model computed performance scores for a provider, and then to rate and submit exactly this score as a comment value. This strategy would not change the provider's score as it stands at the moment. But it will prove problematic if more comments are added so that the model becomes biased in favour of such previous scores. More critically though, this rater will earn a higher rater reputation through this strategy, and hence his other comments on other providers also will be given a greater share of influence. In order to disable this cheating possibility, in the design of the rating scenario, it is necessary to decouple any possible bijective relationship between comments and scores. Other than requiring that all comments have to be received before scores are calculated, one way to achieve such a one-way functional relationship is by introducing a temporal decoupling, whereby the scores reflect the situation of a past period. Section 7.7 describes an instance of such a temporal decoupling and updating.

If one considers scenarios such as the *Sybil Attack* [25][4], most reputation systems can break down if the attack is just severe enough. If one expects such an attack, our model can be instrumental in recognising the presence of the attack and partitioning the agents into clusters of similar rating behaviour. If, however, one were to deploy a reputation/recommendation model for business purposes and services, we would expect protection from the worst kinds of fraud through means external to our aggregation function, such as the authentication of raters.

7.9 Social Choice

Social Choice Theory is a subdiscipline of economics that builds a theory for fairness of voting schemes. If one proposes a new Social Welfare Function, which aggregates the preferences of individuals, social choice suggests a number of desirable aggregation properties, that can either be satisfied or violated by the function. The theory originates with the dilemma stated by Kenneth Arrow's impossibility theorem [7]. This impossibility

[4]In the *Sybil Attack* the attacker replicates his identity unlimited numbers of times and thereby is able to undermine defense mechanisms such as replication. Douceur [25] shows that this attack can practically only be prevented with a trusted agency that certifies identities.

theorem puts forth four properties for Social Welfare Functions that seem to be reasonable to be expected and then proves that no Social Welfare Function can satisfy all four for more than three voters.

If we evaluate our reputation aggregation model with the same theoretical properties as in Arrow's problem, our model would be able to satisfy three out of these four properties. However, there is little value in evaluating our model from a Social Choice perspective at all, since the premise of our model contradicts the fundamental beliefs of Social Choice Theory. Social Choice assumes that "everybody's vote counts", whereas we assume that some raters are less able than others to make valuable comments and therefore redistribute the influence each rater has. This discriminatory approach that we chose implies that our reputation model cannot be applied to the same problems as Social Welfare Functions. Our model is better suited for more "dictatorial" applications.

7.10 Further Application Areas of the Reputation Model

In this section, we present applications of the reputation model beyond electronic services. One of such further applications is the academic peer review process, which we will discuss in detail in the next chapter along with a case example.

7.10.1 Stock Market Predictions

Where stock market analysts are assigned the roles of raters, and their ratings take the form of comments, meta-predictions can be made of future stock prices. The rating model seeks for a common opinion among the analysts' ratings, and also displays how much influence each analyst has on the final rating. Difficulties arise when only a single analyst is right, and that analyst's comment is nonetheless refuted by the common opinion. It is worth pointing out that the application of our rating model to stock market predictions is only useful where the history of the analysts' predictions is not available for comparison with the actual stock value history.

7.10.2 School Grading

In a school system where the exams of students are evaluated by more than one examiner, and each examiner evaluates more than one set of exams, this rating model yields insights into the students' work, and more importantly, their assessors' ability to examine. Here, the examiners take the roles of raters and the scores they assign to each exam are comments. One could combine the examiners' comments by averaging them, as is often done. Applying this rating system would allow the examination authorities to base their final results on less deviant gradings. Altogether, this approach would prevent biased grading, where an examiner applies his own grading scale to the exams, or else just inconsistently overrates some students and underrates others.

This application would, for example, be possible in the grading of the final grammar school exams in the German state of Baden-Würtemberg. There, each exam is independently evaluated by two different teachers at different schools. Where the two teachers' assessments differ by more than one grade point, the Ministry of Education appoints a third examiner to evaluate an exam and arbitrate its results. Or, if these assessments differ by less than one point, the average of the two is taken for the final score. By applying our rating system to this scenario, one allows those teachers who are, in general, more consistent with their grading to have a stronger influence on the overall score. On top of this, the Ministry of Education would probably be curious to find out which of their teachers deliver more consistent grading than others.

7.10.3 User Movie Ratings

Many web sites operate general rating lists such as the Internet Movie Data Base [47]. Users are allowed to submit comments on how high they rate a certain title on a scale of [1,..,10]. A general problem with simply adding up these kind of comments is that one has to rely on the vast majority of users actually submitting reasonable comments. One will, however, often find users submitting deliberately extreme votes in order to promote or discredit a certain title they favour. Applying our rating model would allow the rating web site operators to reduce the influence of extreme voting in a controlled fashion. One would choose a balancing value for the selectivity weight variable, such that credible raters' comments are not affected. Alternatively, if not applying our rating model, one would have to resort to heuristic or other methods of filtering the comments in order to remove skewing effects from the rating scores. However, the main objection to applying our rating system to movie reviews is that the purpose of our rating model is to establish a ranking between the rated entities and to achieve this goal assumes objective reviews at the expense of subjective opinions. Such a rigourous approach may not be necessary for movie reviews, unless the outcome is used to award some kind of a prize that is based on a ranking.

7.11 Conclusions

In this chapter we demonstrated that the rating aggregation model is successfully able to recognise and filter out various forms of "deviating" rating behaviour. If some of the raters are having a weak rating accuracy, in the form of a high rating variance, then the aggregation model is able to gradually recognise and contain these raters (section 7.2), even if these weak raters resemble 90% of all raters. Next, if a set of raters colludes and collectively attempts to influence the outcome scores with their biased agenda, the model is able to recognise and filter out this set of raters (section 7.3), even if this collective only applies a minor rating bias to one of the providers and otherwise supplies consistent ratings for the other providers. Moreover, this filtering still works when the malicious collective makes up almost half of the rating population (40%). In order to achieve these results we did not apply any special model conditions, such as a tailored starting vector (see section 7.6), but could achieve these with the regular, unique solutions the model

is able to generate. In section 7.4 we modelled a situation whereby the raters are well-behaved, but the providers have a bias towards a set of these client raters and demonstrate that the aggregation model is equally effective at recognising such a situation.

Comparing the results of these threat scenarios with the small-scale ones of the previous chapter, it appears that the larger set of comments actually strengthens the model's ability to respond to "deviant" rating behaviour, even if the relative proportions of deviance are similar.

Throughout these experiments, we could see that the necessary level of the selectivity weight, to counter the presented threat, is an useful indicator of how challenging the present threat is. For the tricky challenges, we have to set the selectivity much higher to remove the deviator's influence, since then the gap between the deviators and the normal raters in their rating profile is narrow. However, it is not a problem to set the selectivity weight higher in such cases, since any alternate convergence point does not appear until the selectivity has reached these high levels, and therefore the results are not ambiguous. In being useful as an indicator, the selectivity weight is analogous to the "undercutting sensitivity" variable that we used as an indicator in chapter 4.

To assist any practical use of the rating aggregation model, in section 7.5 we were able to describe a set of rules on how to set the selectivity weight, since this appears to be a crucial factor in the aggregation model's performance. Further to this end, we showed in section 7.6 how the model can be customised for particular clients, gave account of the scope for aging comments over time (section 7.7), and listed in section 7.8 the most relevant limitations for the general use of the model.

Finally, we described in section 7.10 three rating scenarios beyond e-services applications, where the reputation aggregation model's properties could make for a useful evaluation tool. The next chapter will present another application scenario beyond e-services, the academic peer review process, and we will discuss the merits of applying our reputation model, as well as evaluating it in a case example.

Chapter 8

Reputations Applied to the Academic Peer Review Process

In this chapter, we examine the practical conditions of applying the reputation model, as presented in chapter 5, to an area outside of electronic services, the academic peer review process. In the first part of this chapter, we discuss the merits of using the reputation model for this application in general (section 8.1) and describe the ideal conditions for doing so (section 8.1.1). Further, we elaborate about the general contributions the model makes (section 8.1.2) and contrast this with the objections one could have in general (section 8.1.3). In the second part of this chapter, we present a case example (section 8.2) from an actual workshop's peer review assessment, where we analyse the acceptance recommendations (section 8.2.1) with our reputation model (section 8.2.2). Finally, in section 8.2.3 we contrast the recommendations our reputation model produces with the workshop's actual acceptance decisions. Throughout this chapter, we will demonstrate how one ought to apply the methods developed in the previous chapters, such as mapping reputation aspects onto an "objective" scale (section 8.1.1) and selecting the appropriate level for the "trust sharpening variable" (section 8.2.1).

8.1 Theory of Reputations for Peer Reviews

This recommendation/reputation model is a very useful tool to aid the academic peer review process. Using this model allows insight into the quality of the reviews made, the common consensus among the reviewers and lends some advice on how to rank the papers, and therefore which ones to accept or reject. To simplify our further discussion, we will talk about a journal review process, although this recommendation model could be used equally well for any other peer reviewed venue, like conferences and workshops, research grants, or academic positions.

In the academic peer review process, it is common to have between two and four reviewers for a submission of academic work. Not all reviewers assess all the submissions in the pool of work to assess, but each work should be reviewed by more than one assessor. The intention is to reach a common consensus on which submissions are to be accepted

and which are not.

At this point, it is very common to obtain some very contradictory reviews for the same submission. One reviewer completely rejects the work and finds many flaws in it, while another praises the genuine contribution of the work and strongly recommends its acceptance. Practically every author of academic work has some personal stories to tell about such scenarios.This situation is rather unsatisfactory for the authors who find their work in such a dispute, but also does not make it easier for the editor to make the final call on whether something is acceptable or not.[1]

Since these divergent commentaries are an inherent feature of the academic peer review process, we think that it would be helpful to formalise some aspects of the process of finding the consensus on accepting papers. We think that it would be beneficial to acknowledge that not every reviewer is equally good at assessing the merits of a paper (some simply know the area better than others), and to use our recommendation model that assigns quality measures to raters, depending on how good they are at finding a common consensus.

8.1.1 Peer Review Process Parameters

In order to make our recommendation system work for the case of the academic peer review process, we need to align the steps accordingly, so that the recommendation system actually produces meaningful suggestions.

Accordingly to the terminology we used in these chapters, the journal editor corresponds to the operator of the recommendation/reputation system, the reviewers of the papers are represented by the rating clients and the papers are equivalent to the services on offer by the providers. To be specific, a paper is assessed in terms of several aspects (i.e. relevance to the journal, contribution, language, overall acceptability) and in such a case, every one of these aspects is an independently rated entity for the recommendation system. This means that every aspect is equivalent to what we previously represented as one provider. While this is the only way to represent such a structure in our system, it carries the additional benefit that it increases the number of rated variables and therefore also increases the reliability of its recommendations. Our system benefits from a tighter coupling between the rating relations, which we achieve with this multiplication of rated entities.

Further, our model for the recommendation system calls for objective variables. At first sight, it may seem impossible to bridge this conflict with peer reviews, since every reviewer is allowed to have his own view on the contributions of the paper, and this view does not easily relate to a numeric value. However, we would like to contest such an opinion, since the paper that is to be rated is identical for every reviewer and should have some kind of inherent "true" value for the public, even if the magnitude may differ depending on the point of view of the rater. Reviewers may still have some differences in

[1]Some peer-review venues will only accept submissions that have all the reviewers in agreement with positive comments. Also in such a case one has to find a ranking of the assessed work, in this case with the divergences and the line between acceptability at higher levels with more subtle differences.

judgement, but the whole point of the review process is to come to a consensus on which of the papers are more deserving than others to be accepted for a journal/conference. We, then, are faced with the task of assigning numeric values to rating aspects in such a way that every reviewer has the chance to grade the paper on the same scale. One objective scale by which the reviewers can rate papers involves describing the paper review aspects with detailed, transparent levels, as for example with this paper review form below. In this, we ask the reviewers to complement their descriptive analysis with a numeric rating. In parentheses we show the values of our rating scale and the reviewers are free to choose the appropriate value for the paper they are rating. Furthermore, the reviewers can also choose intermediate numbers if they consider the aspect to fall in between the available criteria (e.g. a 3.4 on a scale of 7):

- Technical merits of the paper
  - **Poor** — obvious flaws throughout **(1.0)**
  - **Mediocre** — major parts of the methodology lack merit **(2.0)**
  - **Average** — parts of the methodology could be improved **(3.0)**
  - **Solid research** — applying state of the art **(4.0)**
  - **Genuine** — innovative methodology **(5.0)**
- Writing quality — orthography and grammar
  - **Poor** — many mistakes throughout **(1.0)**
  - **Mediocre** — mistakes at regular intervals **(2.0)**
  - **Adequate** — occasional mistakes **(3.0)**
  - **Good** — correct orthography, occasional grammar mistakes **(4.0)**
  - **Excellent** — correct throughout **(5.0)**
- Writing quality — style
  - **Poor** — stylistic mistakes all over, incoherent arguments **(1.0)**
  - **Mediocre** — lacking style, difficult to follow arguments **(2.0)**
  - **Adequate** — plain style, sometimes difficult to follow arguments **(3.0)**
  - **Good** — adequate style, readable arguments **(4.0)**
  - **Excellent** — good style, interesting to read **(5.0)**
- Relevance to the journal
  - **Irrelevant** — irrelevant to the journal **(1.0)**
  - **Marginal** — marginally relevant to the extended scope of the cfp **(2.0)**
  - **Appropriate** — relevant only when considering the extended journal scope beyond the core cfp issues **(3.0)**
  - **Relevant** — relevant to one or more of the cfp issues **(4.0)**
  - **Highly relevant** — relevant to one or more the cfp issues and also the theme at large **(5.0)**
  - **Perfectly relevant** — perfectly in line with the cfp theme and the its issues **(6.0)**
- Contribution to the field of study
  - **Useless** — clearly does not contribute anything — solution to a non-problem — unclear if anything has been done to improve on existing knowledge **(1.0)**

  - **Repetition** — has been done identically before — establishing the fact that the experiment is repeatable — backing up existing knowledge with new data **(2.0)**
  - **Minimal** — minimal incremental improvement on existing knowledge **(3.0)**
  - **Small** — small but distinct incremental improvement **(4.0)**
  - **Average** — average incremental improvement **(5.0)**
  - **Substantial** — above average incremental improvement **(6.0)**
  - **Novel** — novelty beyond incremental changes **(7.0)**
- Contribution to academic discussions
  - **Unworthy** — is not worthy any discussion **(1.0)**
  - **Minor** — could be interesting to be discussed by a small minority group **(2.0)**
  - **Average** — could yield some discussions in general **(3.0)**
  - **General** — expect to lead to discussions in general **(4.0)**
  - **Controversial** — expect this to be a highly debated contribution **(5.0)**
- Practical relevance
  - **Inapplicable** — solution is not applicable **(1.0)**
  - **Far fetched** — practical implications would be far fetched **(2.0)**
  - **Possible precursor** — practical implications could be reached with further applied research **(3.0)**
  - **Limited** — touches some current practical issues as it is **(4.0)**
  - **Good** — provides a solution to a current practical problem **(5.0)**
  - **Excellent** — solves a whole class of current problems **(6.0)**
- Overall Rating
  - **Strong reject** — clearly unacceptable **(1.0)**
  - **Reject** — too many weak points, not enough strengths in this paper **(2.0)**
  - **Possible reject** — seems to be too weak to accept, but has some strengths and depends on other submissions **(3.0)**
  - **Borderline** — the weaknesses and strengths are balanced **(4.0)**
  - **Possible accept** — the paper has enough strong points, but also some weak ones **(5.0)**
  - **Accept** — the paper has enough strong points, weaknesses can be fixed or are less relevant **(6.0)**
  - **Strong Accept** — the paper is bursting with strong points **(7.0)**

With such an explicit description at hand, it is possible to reduce these aspects of discussions about acceptance to linear scales and numeric values. Note, that these numeric values are not supposed to obliterate any more differentiated comments, but merely formalise some of the available judgements. More differentiated descriptive comments are still solicited and are considered by the editor when making his ultimate decisions. Further to this, a journal editor may decide that not all of the above listed aspects are necessary, depending on the focus of the journal's acceptance policy.

The scales of the rating aspects should be adjusted to suit the particular needs of the journal; for example, if it has a particular focus, this could be included as a separate aspect for the ratings. Similarly, if any one particular aspect is considered to be far more indicative for acceptance by the journal, we can replicate this aspect in the model and thereby increase the influence of the ratings on this aspect over the influence of the other aspects.

Further, our model also maintains its full utility, even if the raters do not submit comments on all the aspects of a paper. Omitting one or more comments on one aspect may weaken the confidence in the results of this aspect, but it does not impair the general practicability of the model. Actually, if a rater is uncertain about how to rate a certain aspect, he should not do so at all. In this case, he does not influence this aspect and neither is his rater reputation diminished, should his uninformed guess otherwise diverge much from the common consensus.

Since the recommendation model is potentially able to yield more than one convergence point for its calculations (see section 6.4.4), we would gauge that it is sufficient to take into consideration only the results for those cases of settings, that have one unique convergence point. Alternatively, if these settings do not provide for enough differentiation, but instead yield too much plain averaging, the editor can use his own review comments as a starting vector to seed the model's iterative process. In so doing he can select the convergence point most closely related to his reviews and trust his own judgement more than those of the other reviewers.

8.1.2 Contributions of the Recommendation System

The main contribution of this recommendation system for the review process is that it equips the editor of the journal with a tool that aggregates the opinions of the reviewers, while anticipating divergences in reviewers' comments. The strength of the model is that it both equalises the divergences within the comments for one paper and also equalises the divergences between the collected comments of one rater and the comment collections of his peers. The model can thus maximise the agreement on rating aspects as much as possible, and transfer this convergence configuration on to the less agreeable aspects.

We should point out that this recommendation system is only a support tool for the editor to make up his own mind, and to make the review process more transparent (in the first place to himself, and if he decides to publish the results, also for the rest of the community). He still can, and should, inspect the additional descriptive parts of the analysis to make his ultimate decisions. In addition, the recommendation model is helpful to the editor because it produces a confidence value (see section 5.4.4), which states just how much of the total accumulated rater reputation was employed to assess one particular paper. With this confidence value at hand, the editor might find that the reviews of one particular paper have been done on weak grounds and decide to have this paper reviewed by an additional reviewer with a high reputation value.

One claimed benefit of using this model is that it would be fairer to aggregate the reviewers' comments with the model, rather than simply averaging ratings. It would be

more fair, as every paper is not reviewed by every reviewer and therefore whether a paper receives sufficiently unbiased attention, or whether it just happened to be unlucky to be reviewed by some of the less able reviewers is left to chance. Using the model, though, the editor could recognise review patterns for the individual review board members and group them into classes (similar to how this is described in section 6.3). We guess that the editor might find classes, such as the very conscientious reviewers, who pay great attention to accuracy and detail, and show a high degree of agreement among their comments. On the other hand, another class the editor might identify could be the self-interested ones, who see little merit in anything that is not their own work or at least that is related to their topic.

A further nice point of using our recommendation system is to obtain a value for the rating reputation of the reviewers. The editor can find out who of his reviewers is more in line with the common consensus and who is less so. This may be useful for the editor, who might want to use this for the selection of his future reviewers. Besides this, the rating reputation value gives us an immediate incentive for reviewers to assess the papers well. Assessing a paper well in this context means that the reviewer finds the common consensus on the judgement of a paper (and that might be to decline the acceptance). At present, the reviewers already have this kind of incentive, since the editor gets to see and compare their commentaries, but ranking the reviewers makes this incentive more immediate.

Altogether, using such a tool does not imply invention of new mechanisms, since for example the editor might already be able to see that one paper's reviews have been both weak and diverging and therefore have it reassessed. Mainly, though, the tool makes the whole process more transparent. The transparency of the review process will be useful for the editor himself to support his decisions, and it will also allow the editor to provide authors with both the qualitative feedback that currently is given and also a methodical assessment to account for why one paper was ranked over others. Making the review decision process more transparent will improve the confidence in this process of authors and the community at large, and could possibly also do so for the non-academic community (e.g. government funding bodies).

8.1.3 Discussion of Possible Objections

We also expect some objections against employing this tool in the academic review process. One objection could be that editors claim they are better able to make their decisions based on the descriptive review analysis that authors provide presently. The idea behind such a claim would be that the descriptive approach allows for more freedom to express an opinion and therefore reflects the assessment of a paper more closely than the narrow numeric scales allow for. Further, an experienced editor will point out that he knows by experience the rating reputation of his reviewers and can therefore assign higher importance to the highly reputed reviewers. Our contention with these arguments is that our model would not take anything away from the descriptive aspects of a review, or force an editor to ignore more subjective, experience-based elements of the paper assessments. Our model only seeks to formalise the agreement on those aspects that can be placed on

linear scales and already in practice are governed by conventions.[2]

Another contention with the rating model might be that it discourages reviewers from making bold statements (strong viewed ratings) at the top and bottom of the scale. Since their influence is diminished if they diverge frequently from the common consensus, they might feel compelled to always water-down their opinions, resulting in a "middle-of-the-road-approach". We contend that this should not be a problem, because if the common consensus on a paper is that it is "Poor in its technical merit", then it also lowers the relative influence of a reviewer, if he rates it "Mediocre" for the sake of avoiding bold statements. Thus, it always is most influential to have an opinion that the other peers can agree with. Promoting such strategies is not to the detriment of the peer review process, but actually strengthens it, because peer-review is the process that seeks academic consensus. Divisive arguments also have their merit in the academic world, however are not helpful when it comes to ranking one paper over another.

8.2 Example Peer Review at the PET Workshop

In this section, we present a case study of a peer review process with review comments from the "Workshop on Privacy Enhancing Technologies" (PET) 2004, Toronto, Canada 26 - 28 May. In their review process, the program committee (PC) was asked to rate each reviewed paper's acceptability on a scale from one to six, and the workshop chairs kindly permitted us to use this data in an anonymised form to run our reputation model on it.

Contrary to the idealised development of peer review parameters, above in section 8.1.1, this workshop's review process has only one aspect rated on a numeric scale, namely the overall acceptability recommendation of a particular paper submission. While we believe that a greater wealth in rated aspects strengthens the selection of the review process, and therefore recommend deconstructing the independent review aspects, this does not need to imply that the reputation model is useless when only one such aspect is available.

8.2.1 Review Data

The raw review comments can be seen in tables 8.1 and 8.2, where the 19 program committee members, r0-r18, rated 49 paper submissions, p0-p48, on a scale from one to six. Each of the reviewers assessed between three and eleven papers, altogether 141 comments were submitted, such that most papers, 39 of them, were assessed by three reviewers, four papers ended up with four reviewers, and nine were assessed by two reviewers only. While the average score for all comments is 3.3, in some cases the reviewers took the liberty to assign a comment value of zero. Since our model does not apply any restrictions on the scales and we are not concerned with fairness, we will allow these values, as if the full scale

[2]For example it is a common convention that reviewers will make a recommendation on the acceptance and in doing so, choose one of a few definitive, conventional statements: "Reject" - "Borderline" - "Accept", possibly with the combination of weak/strong qualifiers to fine-tune the placement on the recommendation scale.

had been zero to six. In these tables, the reviewers are shown in a random order, and the papers are sorted by the average comment values they received from the reviewers. This average comment value is shown in table 8.2 in the "Peer Review" section. Left of this column are asterisks (*) marking those papers that were finally accepted by the program committee for presentation at the workshop.

Whether a paper actually got accepted was not directly determined by the averages of the comment values. Instead, very high average comments were accepted, very low ones rejected, and the large part of the middle ground was debated individually by the whole program committee.

8.2.2 Reputation Model Analysis

By running our reputation system on this review data, we obtained the graphs in figures 8.1-8.3. These figures show the results of the reputation system, over the full range of settings for the trust sharpening factor $w = [0, .., 1.1328671]$, any higher setting thereof would produce ambiguous results, leaving the range of unique convergence points. The resulting values at the maximal setting of the trust sharpening factor, $w_{max} = 1.1328671$, were then included as extra columns and rows in tables 8.1 and 8.2. This value for the trust sharpening factor, $w_{max} = 1.1328671$ is as close to the limit of alternate convergence points appearing, as we were able to obtain numerically. As we can see in figures 8.1-8.3, some values change at a high rate, when approaching this fault line with the alternate convergence points, and the convergence of the reputation algorithm slows down.

We obtain a graph of the reputation system scores s for each of the 49 papers over the range of applicable settings of the trust sharpening factor, and display these graphs in figures 8.1 and 8.2, in units of sevens graphs in seven separate grids. This is done in an effort to be able to identify each of the individual colour coded graph. Correspondingly, underneath the scores, we display the confidence values for each of the score graphs at the same settings. The confidence values represent the aggregated influence shares of the reviewers that participated in the review of a particular paper. Unsurprisingly, the graphs of the confidence values then mirror the slopes of the reviewers' influence graphs. This is the first instance in our analyses that we were able to make use of this confidence value, since in all previous cases, all assessors submitted comments about all rated entities and this value then had been by default 100%.

For the development of the scores of papers p0-p48 in figures 8.1 and 8.2, we can see that for $w = 0$ the papers are sorted by score, since this represents the average of the comments submitted by the reviewers. By increasing w, the effects of the selectivity of our reputation system increase and we can observe some changes in the ordering the reputation would recommend, depending on the actual setting of w. For the graphs of the confidence values, we can see that they originate in exactly three points for $w = 0$. These three origin levels, $[0.105, 0.158, 0.211]$, represent the number of reviews this paper received, namely whether it was reviewed by two, three, or four reviewers. This correspondence reflects the fact that, at a point when all reviewers share the identical influence (for the average), more reviewers assessing a paper increases the confidence in the resulting average score. These confidence values diverge more for more aggressive settings of the trust sharpening

factor w, and the slopes can become steep when approaching the fault line that arises when more than one convergence point is reachable.

In figure 8.3 we show the influences each reviewer reached, for clarity broken down in three grids, depending on the setting of w. This influence, measured in percent, represents the reputation a reviewer obtained in our reputation system, in terms of how well his comments match the overall consensus. Naturally, for $w = 0$, when all scores are taken in plain averages, all reviewers reach identical influence. With increasing settings of w, the influence shares diverge, and at the highest possible setting, the two top reviewers, r0 and r4, together gather 47.2% of all influence. These two reviewers share only one paper, p37, that both of them reviewed and both rated it with the same comment value of 3, which allows for this shared acquisition of influence, since they otherwise would compete for influence. In general, one can recognise a correlation between the number of papers reviewed by a program committee member and the influence share he then is able to obtain in our system. This correlation is a useful property, since it increases the credibility of the resulting scores obtained from the reputation system.

8.2.3 Discussion of the Results

When analysing the scores in figures 8.1 and 8.2, the most interesting points are the intersections between the score graphs of each paper. Such an intersection resembles a change in the ranking of the papers, as recommended by our reputation system. Most notably, among the higher rated papers p0-p20, our reputation system raises the ranking of p8, p11 and p12 significantly and their graphs traverse several such crossings. Considering the individual comments, in all three cases, the papers received two very favourable comments and one that would not promote acceptance. It turns out that in these cases, the reviewers with the less favourable comments gradually lose their influence, and thereby the score of the paper rises. Curiously, though, of these "rising" papers, only paper p12 was also accepted by the program committee.

Out of this group of rated papers, one stands out by getting downgraded, p7, which crosses the graphs of three papers that start out lower. Looking at the comments, this paper loses score ranking, because its highest grade, a 6, which is an exceptional grade, has been issued by a rater who achieves only little influence (r11). Consequently, the paper turned out not to be accepted by the program committee, and among the rejected papers it is the one with the highest average comment values.

The most dramatic change in the ranking through the application of the reputation system happens outside the range of acceptable papers, for p41. It raises 22 places to rank 20 and the reputation system's score rises as high as 3.61, compared to the 2.33 of its averaged comment values. This rise is based on the fact that rater r0, who gains much influence in our system, gave it a much higher value than the other two raters that commented this paper. A similar rise for similar reasons happens for p33, which gains 16 places. In neither case the paper came close to the "acceptable" range, which is good, since it shows that even the most dramatic changes do not incur unreasonable consequences. In figure 8.4 we plot the distribution of the maximum changes in the ranking that our reputation system suggests. Despite the limited size of the data sample,

	r0	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11	r12
p0		5											
p1				4									5
p2	5												
p3		5							5				
p4						6							
p5												4	
p6						5					4		
p7					4					3		6	
p8		6			5								
p9								5	5				
p10							4					4	
p11	5	5											
p12			4		5					3			
p13						4		4.5					
p14								5		4	2		
p15											3		
p16	4							4					
p17						3	5						3
p18	3								4				
p19							3			4			
p20		4	4	2	4								
p21	3												
p22						2					3		
p23		4						6					0
p24												3	
p25				3									
p26				4									
p27			4		3								
p28	2												4
p29										2	3		
p30						5							
p31				3			4		2				
p32	2		4										
p33					4							4	
p34				3									
p35										4			1
p36				4			2						
p37	3	2			3								
p38										3			
p39								3.5		2	2		
p40							4				1		2
p41	4							1				2	
p42							4						0
p43									3		2		
p44							2		2				
p45	2	2											
p46						1	2						
p47									1		1		
p48				1	1				1				
	r0	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11	r12
infl	21.0%	6.6%	1.8%	3.5%	26.2%	1.8%	3.6%	1.7%	7.0%	3.3%	5.3%	2.3%	1.2%
Rank	2	4	14	8	1	15	7	16	3	9	5	12	18

Table 8.1: Peer review comments (raters r0-r12) and reviewers' influence for the maximal trust sharpening factor.

the distribution appears to be reasonably shaped, resembling a bell-curve to a rough approximation, and the peak of 7 papers represents papers that are assigned the same ranking with our reputation system as they were in the case of averaged values. Further, the bulk of ranking changes stays within 8 ranking slots up or down. Thus, on average, our reputation system suggests a balanced and selective redistribution of the ranking, and does not produce indiscriminate recommendations. This property is important to increase the credibility of our system.

One contributing factor for the peak of the distribution of changes in figure 8.4 being at distance $\emptyset$ is, that for some papers the scores are not influenced by the reputation system. This is the case for p0 and p45, and is due to the fact that for these papers, all reviewers submitted identical comment values. It is plausible that in such a case, a recommendation system should not alter the scores. Since it is a case that does not appear often in this review scenario, it is not a limiting factor here.

When looking at the ranking of actually accepted papers, we see a strong correlation between this and the average score ranking; however, it is not a direct correlation. The

	r13	r14	r15	r16	r17	r18	Peer Review Accepted	Peer Review Average	Reputation System Score	Reputation System Rank	Reputation System Confidence
p0					5		*	5	5.00	1	8.9%
p1		5					*	4.67	4.54	8	7.6%
p2		5	4				*	4.67	4.96	2	25.0%
p3		4					*	4.67	4.82	4	16.6%
p4				3			*	4.5	4.19	10	4.6%
p5				5			*	4.5	4.55	7	5.1%
p6				4			*	4.33	4.18	11	9.9%
p7								4.33	4.04	13	31.7%
p8				2				4.33	4.95	3	35.6%
p9						3	*	4.33	4.34	9	13.0%
p10							*	4	4.00	14	5.9%
p11					2			4	4.77	5	29.9%
p12							*	4	4.73	6	31.3%
p13						3		3.83	3.56	21	7.8%
p14					4		*	3.75	3.29	27	12.6%
p15					4	4		3.67	3.55	22	11.9%
p16					3		*	3.67	3.91	15	24.9%
p17								3.67	4.09	12	6.6%
p18					4			3.67	3.31	26	30.3%
p19				4.5		3		3.63	3.53	23	13.9%
p20								3.5	3.82	17	38.1%
p21				4				3.5	3.12	29	23.7%
p22	5							3.33	3.11	30	8.6%
p23								3.33	3.87	16	9.5%
p24					2	5		3.33	3.71	19	8.9%
p25		3				4		3.33	3.40	25	10.7%
p26	3			3				3.33	3.45	24	7.6%
p27				3				3.33	3.06	32	30.8%
p28	4							3.33	2.22	41	23.5%
p29				4.5				3.17	3.08	31	11.4%
p30						1		3	2.20	42	6.1%
p31								3	2.76	35	14.1%
p32								3	2.16	44	22.8%
p33					1			3	3.78	18	30.7%
p34		1	5					3	2.50	38	7.5%
p35					3		*	2.67	3.14	28	6.7%
p36						2		2.67	2.61	36	11.3%
p37								2.67	2.88	34	53.7%
p38				3		2		2.67	2.59	37	10.3%
p39								2.50	2.25	40	10.3%
p40								2.33	2.19	43	10.1%
p41								2.33	3.61	20	25.0%
p42						3		2.33	3.01	33	9.0%
p43					2			2.33	2.48	39	14.7%
p44								2.00	2.00	45	10.7%
p45			2					2.00	2.00	45	28.6%
p46								1.50	1.66	47	5.4%
p47	2							1.33	1.10	48	13.8%
p48								1	1.00	49	36.7%

	r13	r14	r15	r16	r17	r18
infl	1.4%	2.9%	1.1%	2.8%	2.3%	4.3%
Rank	17	10	19	11	13	6

Table 8.2: Peer review comments (reviewers r13-r18), the outcome and the reputation system's scores for the maximal trust sharpening factor.

divergence between both is comparable with the difference between the average scores ranking and the reputation system's ranking, as it is shown in table 8.2 for the maximum trust sharpening value w_{max}. This suggests, that the magnitude of ranking reordering due to our reputation system is at an acceptable level. There are some papers that were accepted by the program committee that also in our system received favourable scores; however, there are equally many that were accepted, while not being favoured by our system. This just reinforces the suggestions made previously in section 8.1.2, namely that this reputation system can be useful to act as a recommendation tool, to give advice on which papers need further assessment. From the review scores alone, it is not possible to derive the true value of a paper. Paper p35 for example received poor comment values, but was accepted by the program committee. This paper stands out in our reputation system, in as much as it gains 9 places in the ranking with our system. This could be taken as an indicator by the program chair to scrutinise the paper further by having it assessed again. Apparently, the paper also did get another chance in the program committee discussion, since its numeric scores would have been far from acceptable.

Revisiting the influence shares attained by each reviewer in figure 8.3 shows that r0 and r4 stand out dramatically for $w \mapsto w_{max}$. It is the nature of this reputation system, that it is able to produce such sharply selective results. If such an imbalanced influence distribution is not desirable, it is possible for the committee chair, who sets the parameters of the reputation model, to choose a more modulating heuristic. Such a modulating heuristic could be that "the aggregated influence of the top third of the reviewers shall be equal to the aggregated influence of the rest of the reviewers". Such a heuristic then leads to the choice of smaller, more balanced values for the trust sharpening factor w.

While the choice of the value for the trust sharpening variable w is crucial for the reputation system's recommendations, some trends can be observed to be independent of this variable. Notably, reviewer r15 appears to lose a lot of his influence, even for lower values of w, compared to many other reviewers where most of the losses or gains happen only for $w > 1$. The reputation system downgrades the influence of this reviewer, r15, mainly for two reasons. First, this reviewer's comments diverge significantly from the other reviewers' comments, and secondly he supplied only three paper reviews, which weakens his position additionally, since he then has a good chance to obtain a relative high variance.

While reviewers r5, r7, r12, r13 share a similar fate as does r15, in contrast to these r2 shows a different slope for the influence development. Reviewer r2 only loses his influence for the higher settings of w, not from the onset. In this case, the loss is not due to a disagreement with many other review comments, but mainly due to one significant disagreement with reviewer r0 over the rating of paper p32. Since reviewer r0 is one of the two reviewers who amass a large part of all influence available when $w \mapsto w_{max}$, then r2 shares sink correspondingly. This degradation happens because the effective system scores for p32 move away from r2's comment value. This effect to some degree taints the scores we derive for paper p32 with our system, because this paper was assessed by only two reviewers, who appear to disagree significantly. Since even the higher one of the two review comments would not be sufficient for acceptance, it would probably not be worth expending the effort to reassess this paper an additional time.

Reviewer r8 attains a high reputation and influence, but not as high as the two extreme cases r0 and r4. The reason why r8 does not gain as much as these other two for $w \mapsto w_{max}$, is that he competes with r0 over the review of p18. It just happens that r0 overall maintains more of his influence and therefore "wins" this competition. However, this effect is limited to the area close to $w \mapsto w_{max}$.

Finally, we observe that this data set is sufficiently large for our reputation system to make a useful contribution to the recommendations of the ranking of the papers. A larger data set, particularly covering more than one assessment aspect per reviewed paper, would improve the reliability of our reputation system's output scores. In this review case, the assessment aspect we were able to look at was about the acceptability of the paper. This aspect is naturally a controversial one, and it would be beneficial for our system to also include some less controversial review aspects, such as the readability of the paper. Adding these less controversial aspects would allow the system to cross-link agreement (and disagreement) between the controversial and uncontroversial aspects. We expect that doing so would dampen the overall variance, and hence the resulting influence distribution

would not be as sharp; possibly more meaningful though. In such a case, the reputation system would have more justification for downgrading certain reviewers. Effectively, this also would allow the program chair to operate the reputation model at a higher, less compromising setting of w, to derive the paper ranking. Concluding, we can say that our reputation system is able to contribute to an alternate ranking of the papers, but since the data set is thin, with only one review aspect available, using a mediating setting for the trust sharpening value in the range of $0.8 < w < 1.05$ would be advisable.

8.3 Conclusions

In this chapter we were able to give an extensive exposition and discussion of how our reputation system can be applied to the academic peer review process. We described what steps need to be added to the review process to enable the reputation system to contribute its recommendations. We showed in detail how in the academic peer review process one could map qualitative assessment criteria onto a linear numeric scale and argued why doing so is "fair", since the goal of the process is to reach consensus on a ranking. We also explained how these recommendations from the reputation system should be used as an advisory method without detracting from the usual acceptance decision process.

We also were able to analyse the case of an actual workshop's peer review comments with our reputation system. We found the reputation system's recommendations to be useful, despite the fact that this example contained the data for only one assessment criterion, and were able to explain the system's behaviour given the comments at hand. Our analysis yielded insights into questions such as: which paper should be further assessed, which raters are generally contributing more to a consensus than others, and finally what value to choose for the trust sharpening factor, in order to generate a new ranking of the assessed papers.

In the next chapter we will discuss how the results of this and the previous chapter's analyses can be seen in combination with the results from all the previous chapters, in particular the supply-function pricing model analysis in chapter 4.

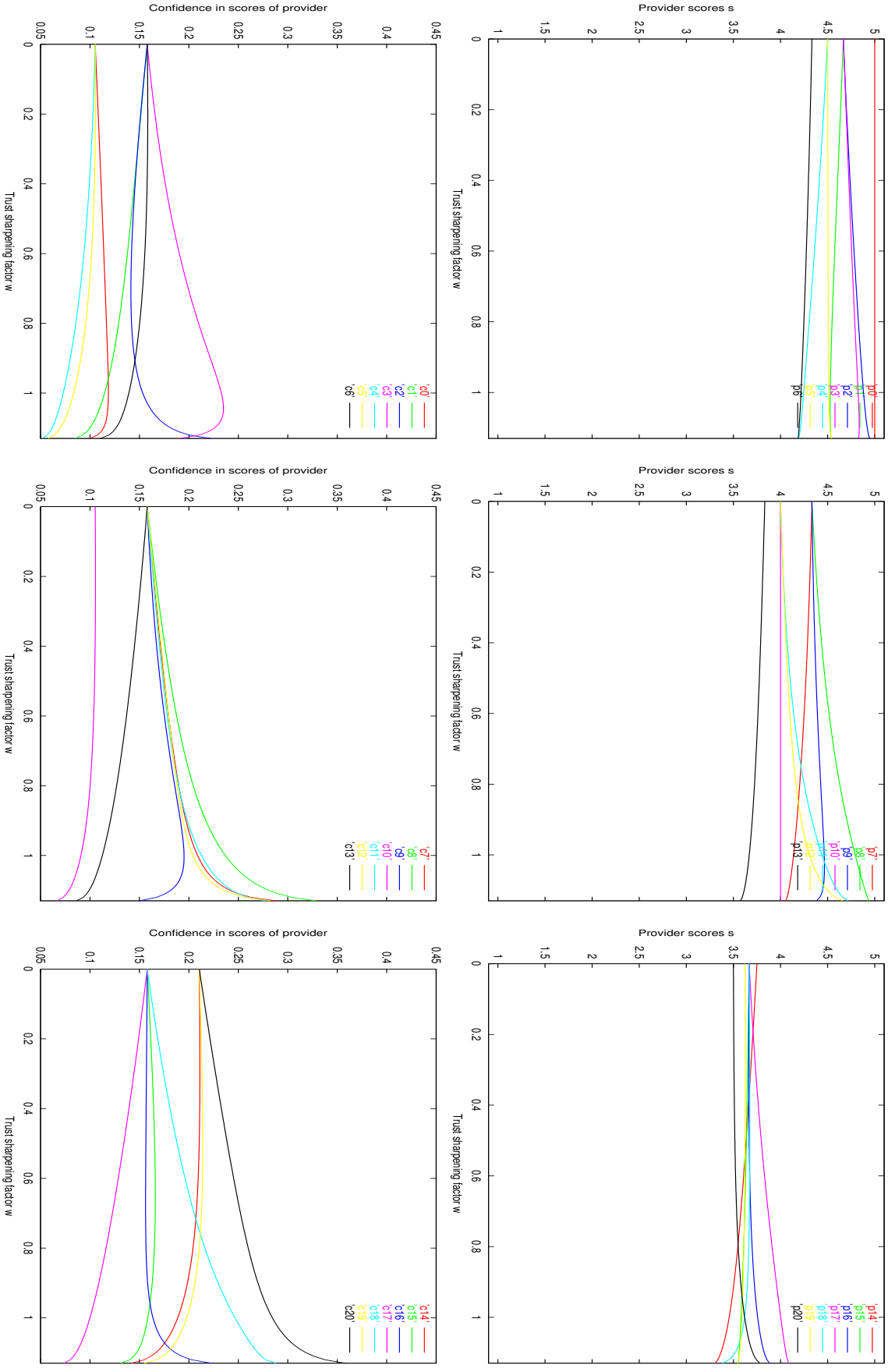

Figure 8.1: Reputation system scores and their confidence (papers p0-p20).

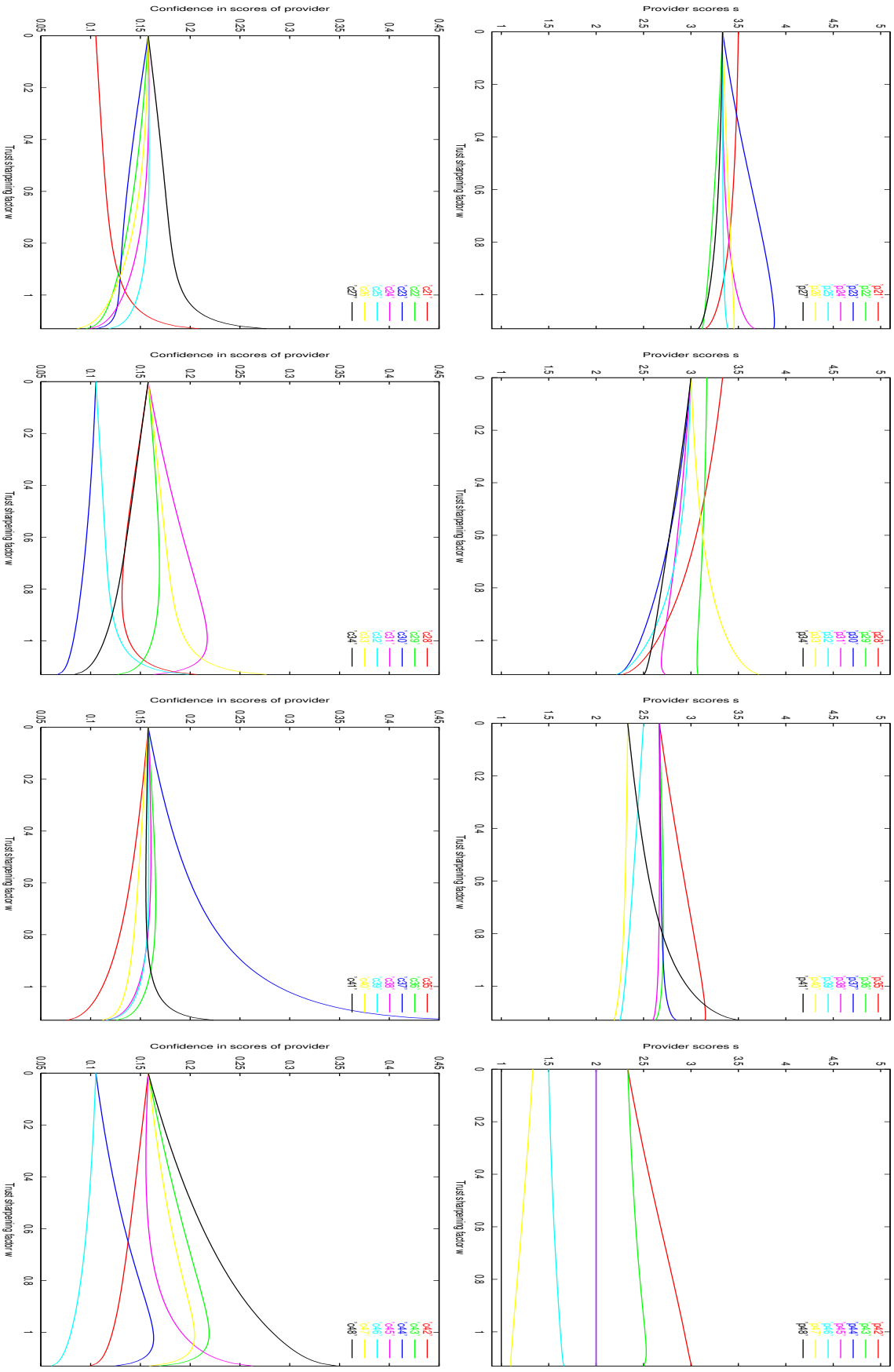

Figure 8.2: Reputation system scores and their confidence (papers p21-p48).

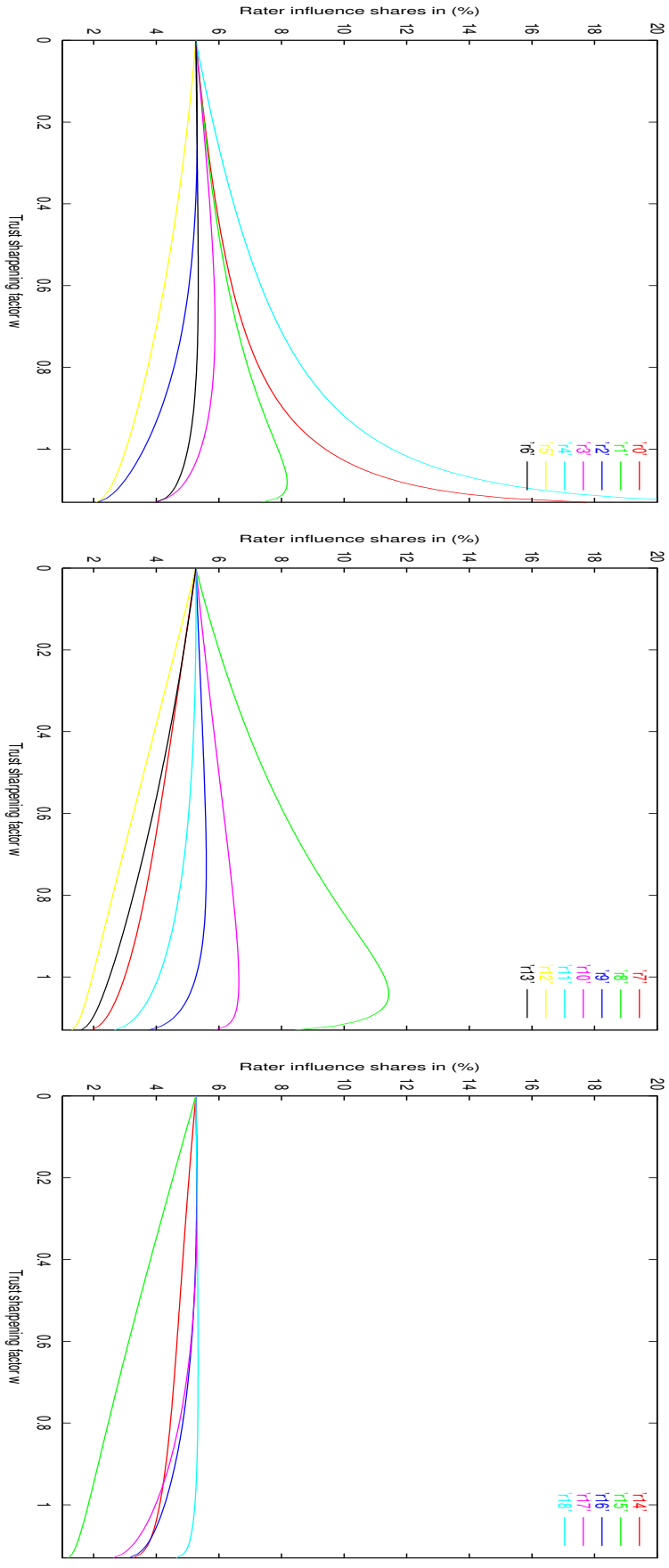

Figure 8.3: The reputations of the reviewers.

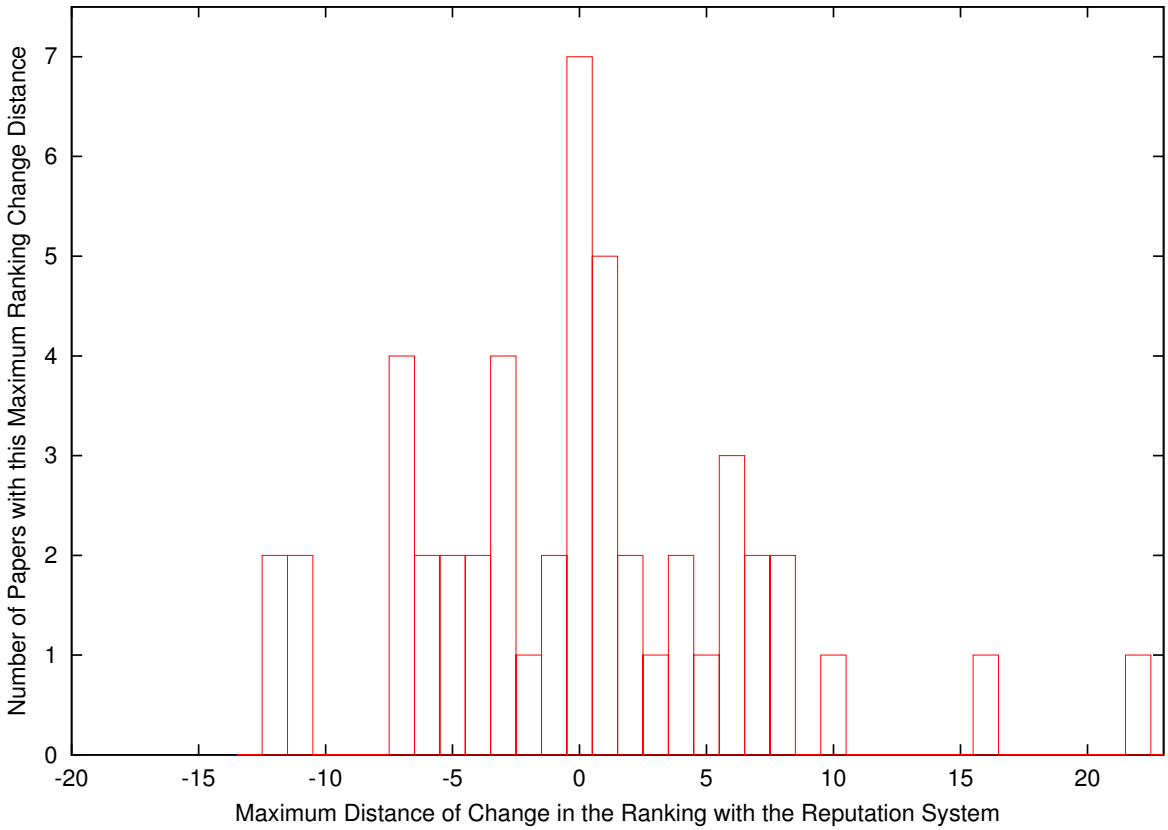

Figure 8.4: The distribution of the maximal distances of ranking change due to the application of our reputation system.

Chapter 9

Conclusions

In this thesis, we presented and analysed two models, a pricing model and a reputation aggregation model, that address problematic issues of market-based services allocations. Our hypothesis from the introduction was that market-based services require cooperation to flourish fully. Though the two models address very different issues of market-based services allocations, and are therefore also concerned with very different types of cooperation, both models are successful at promoting cooperation and both successfully apply enforcement methods to suppress deviation from this cooperation.

In the remainder of this chapter, we reflect on the contributions of (section 9.1) the pricing model, (section 9.2) the reputation aggregation model, and then in section 9.3 elaborate on the conclusions that can be drawn from the methods used in the analysis of both models on the general topic of cooperation and deviation.

9.1 Pricing Model

The pricing model we were able to develop in this thesis provides a solution for the problem with e-services, where perfect competition and excess supply lead to prices at level zero. To achieve its means, the model uses microeconomic theory to create incentives to make rational agents cooperate. But we explicitly exclude the possibility of collusion based on punishment strategies, or using prices in other periods with higher resource demand, in order to maintain prices above competitive levels.

To model this pricing at a technical level we use supply-functions, which have not been previously applied to pricing of e-services. Not only have supply-functions not been used in this context, but also they were previously only used in settings with a central auctioneer who matches supply and demand to achieve the equilibrium. We extended this model to our scenario with continuous trading and a decentralised, bilateral market. This extension relies on continuous updates of the model beliefs about the current market demand and to respond to the competitive situation.

In our analysis of the pricing model, we investigated a number of questions, which we will summarise here:

- Are providers able to maintain prices above marginal variable cost?

  We found this to be the case in general, and in particular for a low number of competing providers the model works reliably well. With large numbers of providers, the effect of raised prices is present, but less pronounced, since in such a situation, the effect on the prices from one individual provider withholding some of his supply is diluted (section 4.1.2, figure 4.3).

- Is the model able to defend against "deviators", who know about the supply-function model, but attempt to exploit it? Is deviation able to break the cooperation?

  We demonstrated that the model is able to make the worst-case "deviation", the undercutting-strategy, unprofitable in all circumstances, by raising the model's undercutting-sensitivity parameter (section 4.2, figure 4.5).

- What are the losses due to a high sensitivity to thwart deviators?

  The providers stand to loose up to 10% of their profits if the undercutting-sensitivity is set very high, and no deviator is present. With a deviator present, the prices drop quickly and losses are high, but this is the intention of the model's response strategy (section 4.2.1, figure 4.6).

- Are the providers able to gain sufficient information from the trading to update the model timely enough in order to respond to changes in demand or competitors' behaviours?

  We found that the information the model obtains from the trading is sufficient to respond to demand changes and to respond to deviant (undercutting) pricing competitor providers. Only when the relative numbers of clients per provider become very few, then any request and sale is a statistically exceptional event, and then responses of the model become erratic (section 4.1.1, figure 4.1).

- Is the pricing model legal, or would its form of cooperation constitute to being collusion?

  The model would be considered legal, since it does not involve any collusion. One way to explain how the providers cooperate, is that they all simulate the market and have chosen the same strategy that gives them the benefits of a Nash-Equilibrium (section 3.1).

All together, the pricing model is able to satisfy all the proposed challenges and is a valuable contribution to the make revenue streams for resources more reliable and less erratic, since providers are not forced to singly derive their revenue from variations in client demand.

9.2 Reputation Aggregation Model

The reputation aggregation system that we present in this thesis seeks to address the question of how to achieve consensus when aggregating comments and it provides a mechanism

that recognises and diminishes the influence of comments that contribute less to the overall consensus. While the model in itself does not attempt to conclusively determine what "consensus" is, its mechanism is able to provide a continuum of solutions between a perfectly equalising aggregation and a dictatorial one, by adjusting the model's selectivity parameter. Since the exact notion of "consensus" will always be application-specific, it is remarkable that the model, through its range of unique solutions (with only one convergence point), is able to supply results that could be considered well founded, independent of the actual application-specific situation.

The main contribution of this reputation aggregation model is that it provides, through the introduction of "rater reputations", a straight-forward way to determine priorities for some comments over others, and derive a rater's reputation singly from the proximity of this rater's comments vis-a-vis all the other comments. In so doing, the model derives outcome scores that are based on the comments from the largest subgroup of raters with the highest degree of consensus.

Similar to the pricing model, in the course of our analysis of the reputation aggregation model, we posed a number of questions where we collect the answers here:

- Can the model produce meaningful outcome recommendations when it discriminates between raters?

  The rating model is able to select the scores derived from a majority group and selects scores that seem to be in line with the one group that might be the most amenable to all groups (sections 6.1 and 6.2).

- Does the iterative algorithm of the reputation aggregation function actually converge reliably and how fast is it doing so?

  The iterative algorithm normally converges and it does so usually very rapidly (except for special conditions, section 6.4).

- Is the reputation model able work meaningfully with few or many comments available?

  The reputation aggregation function is very scalable in that it is able to produce meaningful results for just a handful of comments, while also being able to produce more stable results when hundreds of comments are available (sections 6.1-6.2 and sections 7.2-7.4).

- Is the rating model able to deal with coordinated "deviant" rating behaviour?

  The rating aggregation model is able to recognise and filter out various forms of "deviating" rating behaviour:

  Raters with a weak rating accuracy, even up to 90% of the rating population (section 7.2).

  A Byzantine-style rating collective with a deviant agenda of applying a minor rating bias, even if this is nearly half of the rating population (40%, section 7.3).

  A set of providers with a bias towards certain clients (section 7.4).

- How does the selectivity parameter of the model influence the results, and how to choose its setting?

  When set to zero, the selectivity parameter leads to one unique solution where all comments are simply averaged, and for a high enough setting the model will produce as many dictatorial solutions, derived from one user's comments, as there are raters present. In between these extremes the parameter yields a continuum of results that mediate between "discriminatory selection" versus "equalising inclusiveness".

  If one seeks to maximise the kind of consensus that it is intended by the model, we would recommend the selectivity parameter to be set at highest possible setting, that still is within the range unique solutions (section 7.5). If one seeks for higher settings of the parameter and more selectivity than available with the unique solutions then one can choose one of the application-specific or client-oriented approaches, which describe how to select the initialisation vector and conversely the convergence point (section 7.6).

- Is it possible to assume that raters are sharing the same "objective" view of the rated aspect and use a numerical scale to represent reputations?

  We showed how one could map qualitative assessment criteria onto a linear numeric scale and explained that for the goal of reaching consensus on a ranking, this is a "fair approach" (section 8.1.1).

We conclude that the reputation model is able to handle all the presented threat models and is a valuable tool for the aggregation of comments, when one is seeking consensus on establishing a ranking of the rated aspects.

9.3 Cooperation and Deviation

Throughout this thesis we worked on two models that aim to incite cooperation in the e-services market. The means used to achieve the cooperation are very different in both models, since the problems each model aims to solve are equally different. The pricing model seeks cooperation on price levels among the providers through the choice of pricing strategy, whereas the reputation model is a service that explicitly facilitates cooperation among clients, to aggregate their experiences and filter out poor performers.

Both models use incentives to achieve cooperation, but do so in different ways. The pricing model gives direct micro-economic incentives by promising additional revenues. In contrast, the reputation model only indirectly gives the incentive to report truthfully. The incentive comes almost as an afterthought, not without first scrutinising the comments. This filtering within the reputation model is necessary because the players are not immediately affected on their revenue part.

The threats for both models are players who deviate from the cooperation that the models are trying to achieve. Both models have a mechanism with which they respond to a threat, in the pricing model the challenged players change their strategy to match the perfectly competitive strategy of the deviator and in the reputation model outliers'

comments are eliminated. Both models have a variable parameter that moderates the strength of this response to a threat. These model parameters turned out to be essential to achieve the effectiveness that we did, since the threat cases require application-specific analysis and responses. Moreover, in both cases the level of this response parameter that is necessary to counter a threat, turned out to be a useful indicator for the severity of the threat in the present situation. With this indicator we can quantitatively compare different market situations and threat cases.

Finally we conclude, that it is possible to develop appropriate models for market-based services allocation, that promote cooperation, by constructing systems of incentives that reward cooperation. In order to complete the system of incentives, one needs to provide for the ability to respond to deviation threats with balanced, application-specific measures.

Appendix A

Protocol Simulation

This appendix describes the simulator we built to implement the service request protocol, that was introduced in the previous chapter. First we explain discrete event simulations and further particular design choices we made, followed by a description of its technical implementation and finally we analyse its performance. Two elements in the implementation of a non-parallel discrete event simulator are particularly relevant for speeding up its performance, (1) the thread switching and (2) the event queue implementation. Both are discussed in detail and are the focus of the performance evaluation.

This simulator is used to implement the pricing scenario of chapter 3 and yields the data analysis that we discuss in chapter 4. Further, it provides the framework for the reputation model presented in chapter 5 and its analysis in chapters 6 and 7.

A.1 Approach

A.1.1 Application or Simulation?

In order to analyse the service request protocol we chose to implement it in a simulation. Alternatively, we could have implemented it in the form of a prototype application and then run many clients and servers, executing the protocol in order to analyse its capabilities. However, our goal of this research was to focus on the effects of scenarios with large sets of clients and servers. While it would not be impossible to set up a testbed with such dimensions, it is much harder to observe and extract the overall market properties of such a scenario, than it is when being confined to one simulation.

A.1.2 Discrete Event Simulation

In the design of our simulation it was our intention to be able to model technical elements of the request protocol and the applications involved and therefore decided to use discrete events to implement the simulation. A discrete event simulation [58] normally simulates

many objects, which frequently map to real world objects. These *simulation participants*[1] contain some state and all interactions between these participants occur through time-tagged events, where the events reflect the proceedings as these would happen in wall-clock-time.[2]

Figure A.1 shows the structural elements of a discrete event simulation, which contains besides the events also objects that participate in the simulation. An event contains a time and a procedure that is associated with this event. The simulation works by placing all existing events in an queue, the *event-queue* ordered by the time of the events. Note that this time can and usually will reflect some measure of wall-time, but it might as well just be some counter that is used to note the precedence of events.

The simulation then proceeds by taking the first event from the queue and advancing simulation time to the time associated with this event. The simulator then executes the procedure that belongs to this event, with the just adjusted time. While this procedure could also be a generic independent function, it will normally be one of the methods of one of the simulation objects. This object method operates on its data sets, given the current simulation time and data effected from previous event-procedure invocations from other objects. The object methods also can generate new events that are being inserted by the simulation into the event queue and will invoke some object's method at a specific time in the future. Causality prevents the simulator from accepting events into the queue that occurred in the past. The simulation will normally continue until the event queue is empty.

A.2 Core Simulator Implementation

A.2.1 Discrete Event Simulation Packages

As there are many existing implementations of discrete event simulators available, we first evaluated if we could make use of one such simulator instead of implementing one from scratch ourselves. For this evaluation we tested two very different simulators with a simple micro-benchmark.

GTW

We first evaluated GTW[3] [32], a high performance parallel discrete event simulation package developed by Richard Fujimoto, because of its excellent performance properties. It uses a time-warp algorithm to achieve optimistic parallel execution. Parallel execution of any sort requires synchronising the shared data, which however limits the scalability of any parallel application. For discrete event simulations this is particularly challenging

[1] In order to be able to better distinguish between software objects and simulation objects we call the latter simulation participants.

[2] Simulation researchers use the term *wall-clock-time* to refer to the "real time" when meaning the time that we can see on the clock that is hanging at the wall in the room.

[3] *G*eorgia *T*ech *T*ime *W*arp.

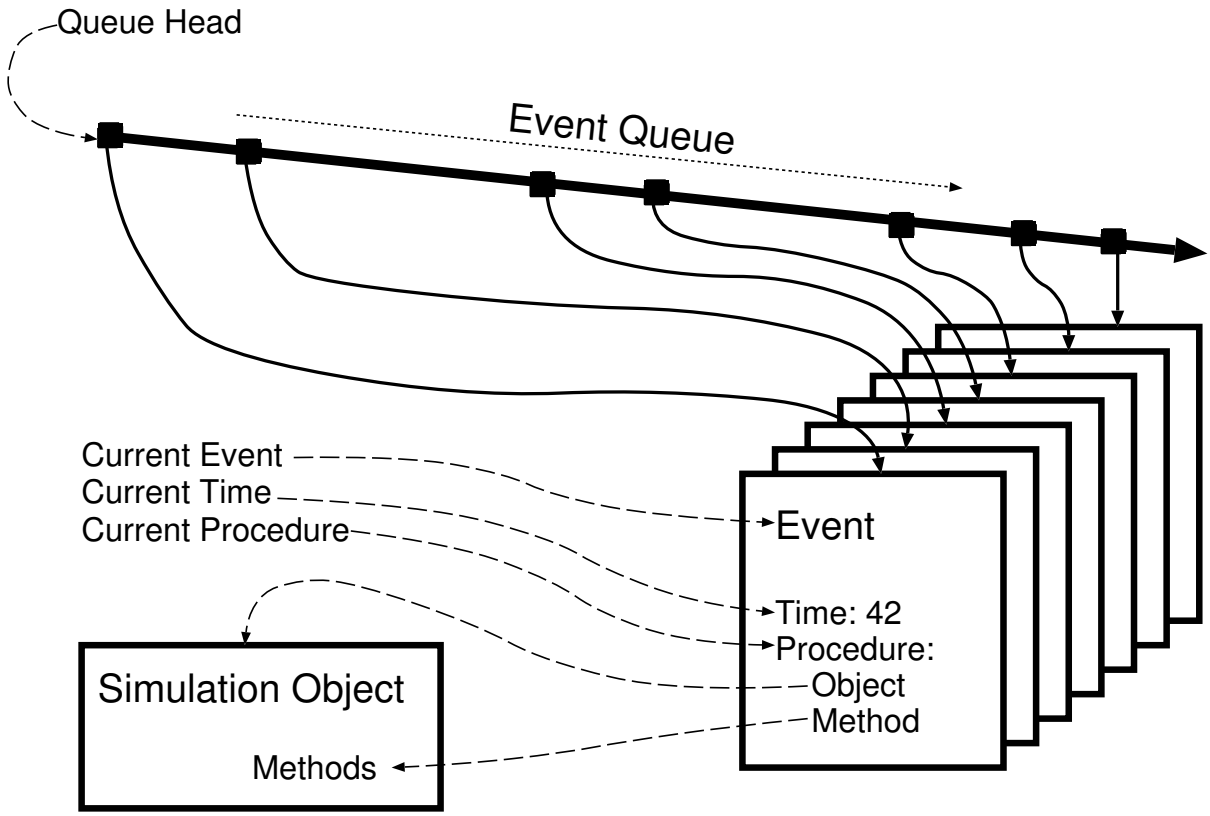

Figure A.1: The structure of a discrete event simulator.

due to their high degree of interaction between processors, where conservative approaches, that rely on strict synchronisation, cause delays at a high rate. Time-warp fares better, as it optimistically continues executing processors, even if these might execute events out of order. When it detects that events were executed out of order and these events are causally related, then it undoes the execution of these events by rolling back to the previously correct state. This allows for high performance, particularly when running the simulation on a parallel distributed machine such as a cluster with commodity SMPs, as opposed to using one large symmetric shared memory parallel machine only. The execution of our micro-benchmark on GTW turned out to be very fast, but we decided against GTW for practical application implementation reasons. GTW's rollback implementation implies that dynamic memory allocation cannot be automatically rolled back, but has to be carefully considered separately, or better substituted by using entirely static application implementations. We did not want to limit our simulation implementation a priori with such a paramount constraint.

JavaSim

We then turned to the abundance of discrete event simulators implemented in Java, as we favoured this language for its ease of implementation. One of the more promising candidates appeared to be *JavaSim* [61]. We implemented our micro-benchmark to find out that it required somewhat tedious semantics for activation of the simulated process-

ing entities. But we finally decided against using it because of its extraordinarily low performance. It is our speculation that this lack of performance stems from the way it implements its synchronisation and context switching between simulation participants.

A.2.2 DES Implementation Approach

Due to the experiences made with the micro-benchmarks in the previous section A.2.1, we decided to implement our own simulator from scratch and name it *DES*. In doing so we are able to modify the simulation's functionality like the event queue and the semantics of the simulated processing entities. We chose Java for implementing DES, because of its portability and its popularity for distributed protocols. We also decided against parallelising the simulator as this would increase the implementation efforts dramatically and the lack of locally available parallel machines would not have rewarded such a development.

A.2.3 Communicating Processors

DES uses Java threads to model simulated objects and then executes the events that belong to this object in the context of this object's thread. Instead of this approach we could also have implemented the simulator without a separate thread for each object, but use a functional event-state-machine model. Such an approach has performance advantages, as it can be implemented without the overhead of context switches between the threads, and simply use a function call to enter a separate object's event procedure. However, as we intended to implement the protocol simulation in the form of communicating processors, threads lend themselves better to this implementation model. Communicating processors not only contain state in the form of associated data, but also the program counter pointing to the current position of the execution thread. Using thread objects in the simulator facilitates this current position pointer. Figure A.2 illustrates an exemplary conversation between a client and a server. Both are following a linear execution path and whenever they issue a blocking receive message call, the scheduler will hold the execution of this thread. When time progresses to the point where the party that is waiting for a message is supposed to receive this message event, the scheduler will resume this party's thread at the very point where it was halted. This is not withstanding the fact that one can model communicating processors with the event-state-model, but when doing so one would also have to integrate the thread's current state into the model or implement such through a separate simulation model layer.

A.2.4 Coroutines

The most suitable way to implement communicating processers is to use *coroutines*, as they are found implemented for example in BCPL [64]. BCPL-style Coroutines are controlled through the primitives "callco", "resumeco" and "cowait" that transfer execution to other coroutines. A related concept to coroutines are *continuations*, as it is possible to implement coroutines using continuations [43]. One can also implement oneself coroutines

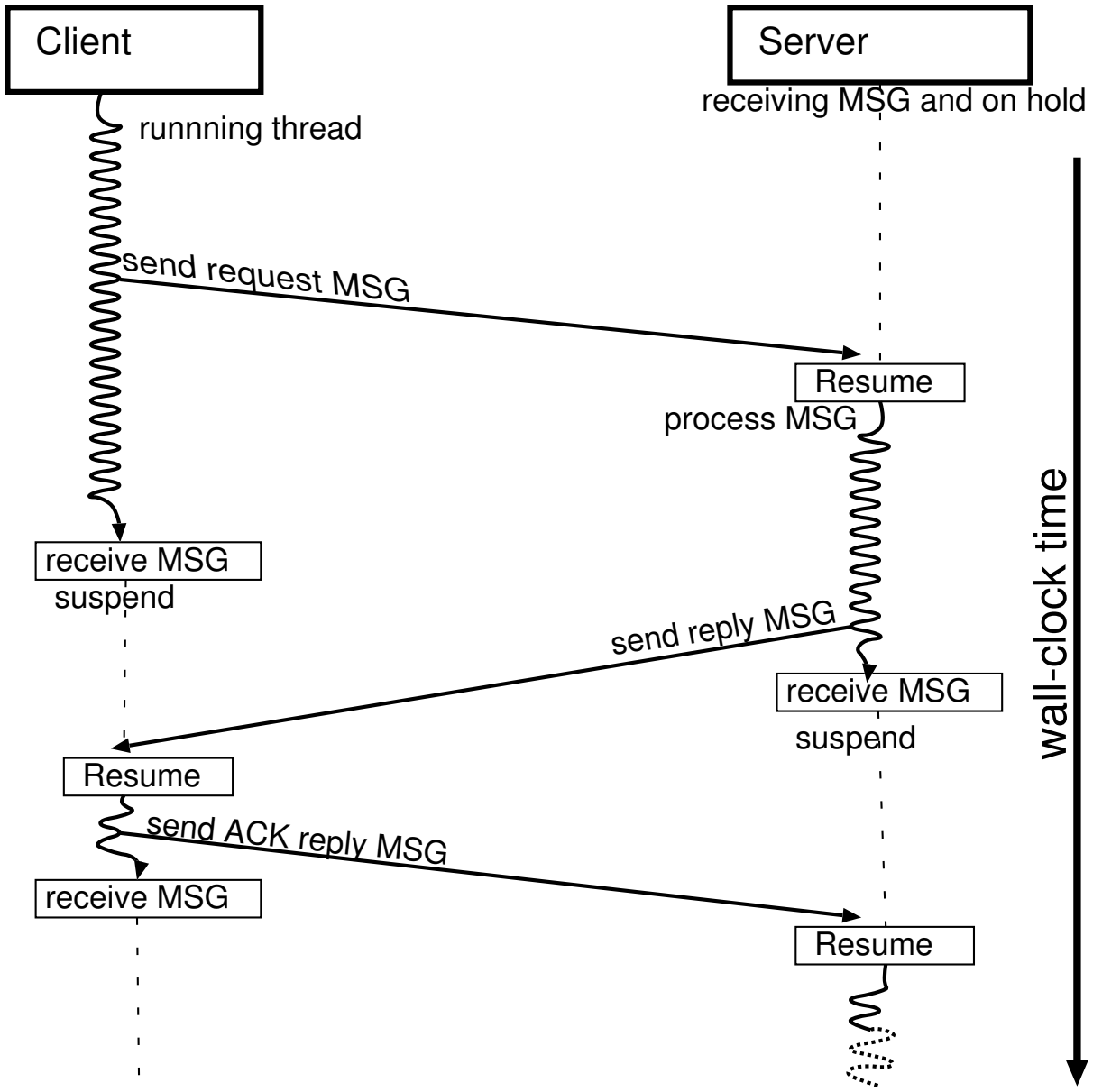

Figure A.2: Simulation of communicating processors.

for a thread supported language such as Java or C++ [85]. However, due to performance related reasons explicated in section A.3.4 we decided on a more low-level thread control using direct primitives.

A.2.5 Random Number Generator

A crucial module in a simulator is the pseudo random number generator. Its efficiency may effect the execution speed, but most of all its pseudo random properties are pivotal to the simulation and whether its results are meaningful. A good pseudo random number generator must be efficient and generate a neatly uniform distribution with a very long cycle before it yields the same numbers again. We adopted a Java implementation of the *Mersenne Twister*, developed by M. Matsumoto and T. Nishimura [62], which is proven to perform very well and is available under the GNU license.

A.3 Thread Switching

A.3.1 General Principle

After fetching a new event from the queue, the simulator usually has to switch to a different thread. This is the case most of the time, unless two subsequent events are associated with the same simulation object. Figure A.3 illustrates the sequence of steps involved in switching from Object A's thread to Object B's, which is halted at present:

1. *Object A's* thread is running.
2. A's thread invokes the `receive_message()` function.
3. The function enters the *scheduler* in the discrete event simulator module.
4. The scheduler retrieves the next event from the event queue.
5. The scheduler adjusts the current time to the time of the new event.
6. The scheduler then schedules the thread of the object B, belonging to the new event, by setting the thread from on hold to `resume()`.
7. The last action taken in the context of object A's thread is to `suspend()` itself.
8. Now the Java scheduler takes action and switches context and control to the only thread that can be run, object B.

At the start of the simulation we limit the number of concurrently running threads explicitly to one, so that we can control the scheduling in the simulator explicitly and disable Java's multitasking. We also have to prevent the Java scheduler from switching control to object B's thread already at step (6), as this would lead to race conditions and unpredictable thread execution sequences, because it were possible for object A's thread to be rescheduled later and immediately then executes the left-over `suspend()` command.

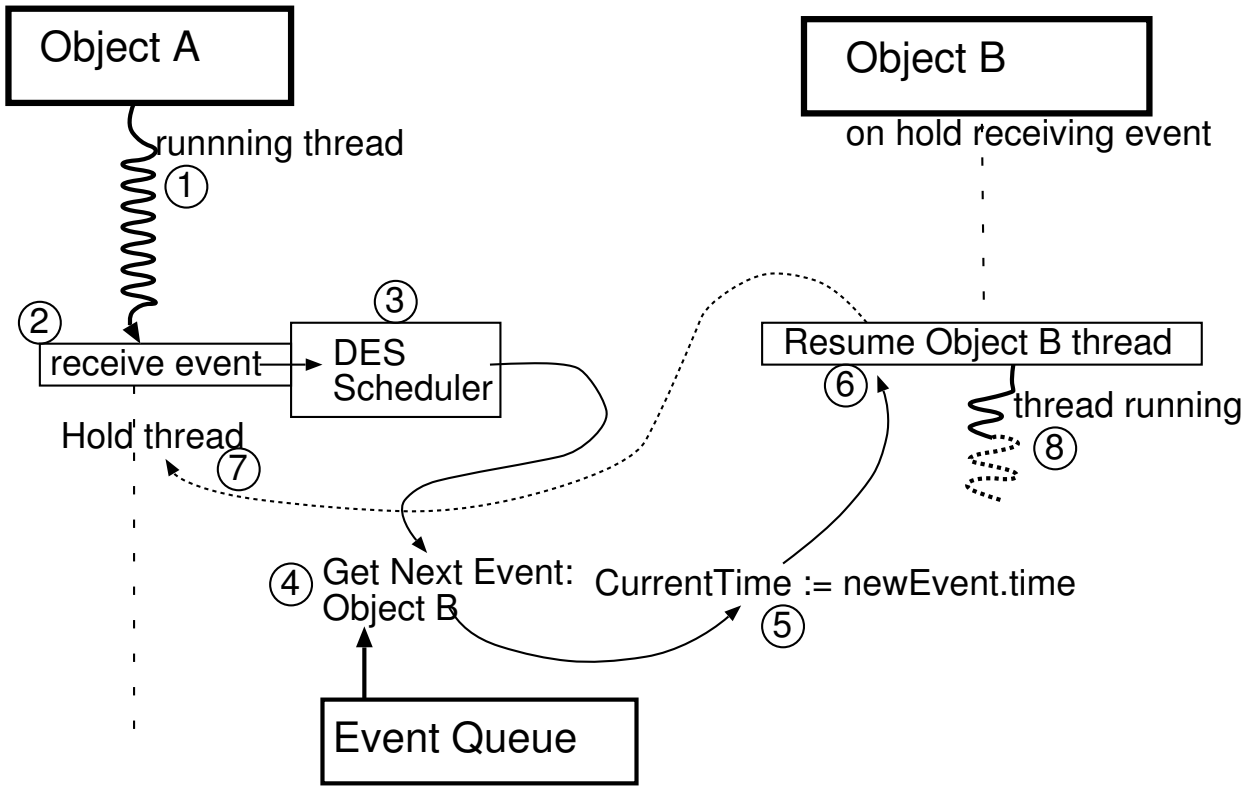

Figure A.3: Switching execution threads.

A.3.2 Synchronisation Through Locks

We can facilitate the control transfer between the Java threads with different methods. An elegant way is the use of conditional variables and critical regions where all but one of the threads are made to wait on a conditional variable. Activating this conditional variable transfers control to the selected thread. The use of a critical region then ensures that steps (6) and (7) are executed atomically.

A.3.3 Explicit Thread Scheduling

An alternative thread control method is to explicitly use `Thread.resume()` to reactivate the next object and to halt the current one, even though the method is deprecated. Figure A.4 shows the procedure implementing the thread switching using the explicit `Thread` directive. First, if the next simulation process we want to switch to is identical to this one we simply return to this thread's execution. In order to ensure in our implementation, that steps (6) and (7) are executed in combination, we add a `Thread.yield()` after the `Thread.suspend()`. In doing so, we ensure that control is handed back to the previously running thread, in case the Java thread scheduler in the mean time already had got active and had started running the newly woken up thread while the previously running thread is still able to run. One would expect this case to occur only rarely, but if so, at this point we resolve the potential race condition and the previously running thread will at

```
public static Event Schedule(SimulationProcess Receiver) {
    PrepareNextEvent();
    if (EventToSchedule.EventRecipient != Receiver) {
        EventToSchedule.EventRecipient.resume();
        Thread.currentThread().suspend();

        Thread.yield();
    }
    return EventToSchedule;
}
```

Figure A.4: Context switching implementation.

this point run into the `Thread.suspend()`. We should point out that this construct is not safely portable between different Java implementations, as the actual scheduling effects of the above commands are not specified in the Java standard. Especially `Thread.yield()` may be interpreted as a suggestion rather than an imperative and a construct relying on it is bound to run into race conditions in different Java runtime environments.

A.3.4 Implementation

Despite the warnings we just presented, we chose to implement the thread synchronisation in our simulator using the direct `Thread.resume()/.suspend()/.yield()` directives. We verified that the Java runtime environments we used actually complied with our assumptions on the thread scheduler behaviour and that our simulation would run reliably without race conditions. Our motivation for favouring this less elegant and less robust implementation was to maximise performance. This higher performance is related to the thread handling implementation of the Java runtime system.

A.3.5 Java Threads

We gain orders of a magnitude faster simulation execution when using a Java runtime environment that does not implement threads through mapping them onto operating system supported threads. Operating system threads, or sometimes also called *kernel threads*, involve significantly higher context switching overheads compared to application library threads. Threads that are implemented by an application extension, such as the Java runtime environment, which sometimes are also called *green threads*, require only few instructions and no operating system call for a context switch. The main advantage of using kernel threads is to have these scheduled independently of each other and signalled by the operating system. In doing so one can built more robust multitasking applications. However, our simulator seeks to achieve the opposite, not external operating system scheduling, but rather explicit execution control within the application, which is our simulation.

A well performing alternative to threads managed in the programming language as opposed to kernel threads is to have so called lightweight threads, semi-managed by the system in an application library. Sun's Solaris provides such lightweight threads, called *LT*s [88], which are supported by the operating system and are listed and scheduled in an application library. For operating system scheduling, *LWP*s (*L*ight *W*eight *P*rocess) are used and the LTs are scheduled on top of these. A Java runtime system that uses LTs to mange their threads would be very suited to our discrete event simulation. Windows ($NT|2000|XP|etc.$) has a similar lightweight thread facility called *fibers* [65]. However, by our experience, these are less accessible and more intricate to employ than Solaris LTs.

We found Kaffe [48] to be a very efficient Java runtime environment with low overheads for thread handling and other operations, and also not using kernel threads. Also for performance reasons we avoided the use of critical regions, as these slow down the execution of the Java program.

A.4 Event Queue

A.4.1 Performance Critical Functions

The second performance critical element in the simulator is the implementation of the event queue. The elements, namely the events, in this priority queue are ordered by their invocation time. The two operations that have to be fast for the operation of this priority queue are (1) `Extract-Min`, to obtain and remove the event with the "nearest" time, and (2) `Insert` to place a new event into the correct time position. For our initial implementation we used a simple flat list, but it was obvious that while `Extract-Min` (1) would be instantaneously fast, a linear-time operation for `Insert` (2) would not scale well for large simulation scenarios.

A.4.2 Priority Queue Theory

Using a binary tree with log-time behaviour on average for the element retrieval/inserting operations would be a first improvement, but to increase scalability significantly we considered using heap structures. With a binary or a binomial heap we would obtain log-time behaviour on average for the retrieval and inserting operations. Better though, implementing a Fibonacci heap (see chapter 20 in Introduction to Algorithms [16]) would provide log-time performance for `Extract-Min` and constant time for `Insert`. Fibonacci heaps achieve this performance only amortised over a larger number of invocations. The `Extract-Min` operation also triggers the `Consolidate` procedure that reduces the total number of trees in the heap and re-links the trees that each root in the root list has a distinct value. As a result, the amortised constant time is achieved at a high level. Further to this, from experience implementing fibonacci heaps [46], we know how complicated the implementation and verification is. The fact that fibonacci heaps require amortisation for their performance is not an impediment to using them in a discrete event simulation, as the total simulation runtime is significant, not a response time for certain function

invocations. However, the implementation complications and its high constant running cost lead us to decide against using fibonacci heaps.

A.4.3 Event Horizon

Concept

The event horizon is the state-of-the-art in any good discrete event simulator. When new events are scheduled, instead of feeding these into the main priority queue, we add them to an unsorted temporary holding queue. We also track the time of the earliest event in this temporary queue. When the next event to be processed happens to be in this temporary queue we say that the event horizon has been crossed and sort this queue before merging it now into the main priority queue.

Advantages

The main advantage is that inserting a new event into this temporary queue is a constant time operation and depending on the implementation of the main priority queue, doing a collective insertion of the new elements can be much more efficient. How advantageous this approach is depends on the time distribution of the newly arriving events. Practically though, in most simulation scenarios, the majority of new events are not inserted in the very near future and it is worthwhile to build this event horizon. Our implementation contains an event horizon temporary queue that is sorted with mergesort, as this is particularly efficient for linked lists.

A.4.4 Qheap

Among the different suggestions for binary heaps for priority queues, we chose to adapt *Qheap*, suggested by Steinman [86]. Instead of using binary trees to describe heaps, this structure only uses linked lists and benefits from low overheads.

Fundamentally, Qheap is a sorted linked list with a fixed maximum number of elements. The fixed length limit is kept short enough, to the point where a straight insertion into a sorted list is faster than the overheads of heap operations.[4] A *meta-item* is a construct that lets a list of items appear as one item in the list. Meta-items can also be nested. When the sorted temporary list is inserted into this Qheap, the whole list is inserted into the heap as one meta-item which derives its sorting key value from the smallest key in its list of elements. If the inserted list is longer than the fixed length limit, it is broken into maximum length pieces. By meta-sizing sublists and nests of lists into meta-items, we ensure that each of the lists themselves are sorted. In this manner, the Qheap is actually a recursively linked list data structure that closely relates to the heap property.

[4]Practically, the limit is set to 40 elements and only minor performance differences show in the range between 20 and 80 elements.

The only drawback of this construction is that when items are to be removed, and the top item is a meta-item, this requires untangling, which however is fairly simple. It involves removing the top item from the meta-item, remetasising the rest of the list and inserting the new meta-item into the Qheap at the correct position. Because heaps are known to have worst case $log_2(n)$ amortised behaviour, this data structure should never break down. As shown in section A.8.2 this heap construction is efficient and scalable.

A.4.5 Events

Event Creation

In order to ensure that events are unique and conform to correct formatting in the queue, we chose to allow only the simulator part to create events that are inserted into the queue. A simulation participant makes a call to the simulator's function `SendEvent` with parameters for the destination participant, the time the event is to set to be delivered, the event type and a message object pointer to contain anything the participants wishes to signal to the other participant. The simulator then verifies the applicability of these parameters, and if so creates an event containing these. When the event is triggered, the destination participant is handed this object while it is deleted from the simulator's queue.

Efficiency versus Safety

It might have been more performance efficient if the simulation participants were allowed to create and format their own events, as they could reuse an existing old object and just reformat it. This would save some activity of the Java runtime system, particularly at the side of the garbage collector. However, we decided against this, as it would allow for corruption of queued events from misbehaving simulation participants. We could check the correctness of the event object at the point of its submission to `SendEvent`, but subsequently the participant still holding a reference to this object could cause corruptions, possibly by attempting to reuse an object that he considers available, where actually it is in use. One could create a list with available events and then maintain a list of free events and the ones in use, but this would require more explicit discipline by the programmer making use of the simulator.

A.5 Reservation Protocol Implementation

A.5.1 Simulation Startup

When the simulation is starting up, the simulation description and launching module in its initiation launches all the simulation participants, in our case the clients and servers. These clients and servers inherit their thread properties and simulation functionality from

the simulation package and are created in a halted state. In the initiation, all other simulation parameters are set and files for post-simulation-analysis are created. After the initiation, the launching module sends a special wake-up event to all the simulation participants. At the end, the launcher hands control to the simulation scheduler, which now works trough the event queue, that contains all the wake-up events. One by one, the simulation participants start up and work trough their initial processing that happens before wall-time is started. In their initial processing, the participants set up relevant parameters and set themselves up for future behaviour by sending themselves events for a specific wall-time. After all participants are done with preprocessing, simulation time starts to advance as the event queue feeds wall-time relevant events to the participants who continue with their simulation behaviour by communicating with other participants and themselves through creating new events. The simulation continues until either a set simulation end time is reached, or the simulation runs out of events to process.

A.5.2 Simulation Message Communication

In general, the participants act out the protocol as specified in chapter B. Clients repeatedly request for some resources and in so doing, go through the five phases of the request protocol. Participants send messages to themselves to simulate timeouts associated with the protocol communication. Clients initiate the request on a set-up timeout and at any point in the negotiation message exchange, both parties wait for replies within a set timeout window. To make the timing of the protocol communication realistic, we simulate message transmissions with randomised delays. Within the format of protocol execution, we individualise the behaviour of the simulation participants, by having different classes of them, by deriving subtypes with modified behaviour and by passing them different parameters that trigger different action.

A.6 Simulation of Clients

The resource seeking clients have a function that simulates their demand input, which is usually drawn from a probability distribution matching a suitable application demand. This demand input feeds into the client activating its resource request procedure `MakeRequest`, which is printed in figure A.5. `MakeRequest` first waits for the time set up by the demand, to then send one request to all servers. After collecting their replies that came back within the timeout period, the client selects what it considers the best offer - in the simplest form it chooses the cheapest price for the resource request. If the selected and notified winning server accepts as well, and issues a contract ticket, the client is able to make use of the resource.

We implemented the clients to run `MakeRequest` synchronously, as we did not see the need for a client to make independent requests concurrently. In order to handle asynchronous messages and also to filter out obsolete events, the requests for particular events are made by calling through the `FetchNextEvent(ExpectedEventType)` function. `FetchNextEvent` discards events of an unexpected type and if it receives a connection

```
protected void MakeRequest() {
    AwaitRequestTime();
    SendRequests();
    CollectReplies();

    if (NoOffersRcved > 0) {
       if (AcceptBestOffer()) {
            if (RequestSuccessful = ReceiveTicket()) {
                PrepareConnection();
            }
        }
    } else {
        WinningBid = null;
        RequestSuccessful = false;
    }
}
```

Figure A.5: The clients' resource request procedure.

close event, it aborts the whole request procedure.

A.7 Simulation of Servers

A.7.1 The Processing Loop

At startup, the servers are initiated with descriptors of the resources they are managing and from then on wait for requests from clients. Upon receiving a request, the server enqueues this request to process it as soon as it is on top of its processing queue. This construct allows us to simulate a certain processing time for the servers and thereby reflect the load on the server from processing these queries. In processing a query, the server verifies the resource availability for the requested time to then generate an offer to the client with the corresponding negotiation details. At this point the server holds the resource for the client until a certain timeout and if the client accepts, the server finally books the respective time slots in the resource roster.

Figure A.6 prints the code of the servers' `ProcessingLoop`, which is waiting for requests from clients and processes them immediately if possible, or enqueues these if more than one request is received concurrently. We obtain the simulated processing time for processing a request from the procedure `GetProcessingTime`. This loop continues to process the requests it has enqueued previously until it is idle again.

```
public void ProcessingLoop () {
    Event current = FetchNextEvent();
    while (true) {
        double ProcessingTime = CurrentTime + GetProcessingTime();

        while (ProcessingQueue.length > 0 || null != current) {
            Event EnqueueThis = Hold (ProcessingTime);
            while (ProcessingTime > CurrentTime) {
                ProcessingQueue.PutEnd(EnqueueThis);
                EnqueueThis = FetchNextEvent();
            }
            ProcessRequest (current);

            if (ProcessingQueue.length > 0) {
                current = ProcessingQueue.GetFront();
                ProcessingTime = CurrentTime + GetProcessingTime();
            } else {
                current = null;
            }
        }
        current = FetchNextEvent();
    }
}
```

Figure A.6: The servers' processing loop.

A.7.2 Processing an Event

The processing of a request itself is shown in figure A.7. We first check if we missed the deadline of the request set by the client, as this could happen if the server's request queuing got backlogged too far. Then we check for available resource capacities, and in a first step of loading the resources more efficiently, we locate the highest loaded resource that is able to satisfy the request. If we find such a resource, we decide to make an offer to the client with the conditions formulated by a separate function. Otherwise, we send a declining message to the client.

A.7.3 Message Handling

The server also contains a function called `FetchNextEvent`, which similar to the clients' version filters out outdated or otherwise unexpected event messages. It also multiplexes between the different client sessions that the server is concurrently connected to. It passes the different protocol message events to the appropriate server handler functions.

```
protected void ProcessRequest (Event Request) {
    MsgRequest MReq = (MsgRequest) Request.Content;
    if (MReq.RequestTimeout <= CurrentTime) {
        return;
    }
    boolean AnyOffer = false;
    for (int i = 0; i < NoResources; i++) {
        if (ResArr[i].ResTypeClass == MReq.ResTypeClass) {
            double avail = ResArr[i].Capacity - ResArr[i].MaxLoad(MReq);
            if (avail >= MReq.ResShare && avail < Minavail) {
                AnyOffer = true;
                OfferRes = ResArr[i];
                Minavail = avail;
            }
        }
    }
    if (AnyOffer) {
        OfferConditions OCond = MyPrice.CalcOffer(...);
        Booking NewBooking = OffeRes.CreateNewBooking(Request, OCond);
        MsgOffer MOffer = FormatOffer(MReq, NewBooking, OCond);
        SendEvent(Event.OFFER, MsgLatency(), Request.Source, MOffer);

        MsgSrvTout MTout = FormatTimout(NewBooking);
        SendEvent(Event.SRV_TOUT, CurrentTime + OfferTOut, this, MTout);
    } else {
        MsgOffer MOffer = EmptyOffer(MReq);
        SendEvent(Event.OFFER, MsgLatency(), Request.Source, MOffer);
    }
}
```

Figure A.7: Processing a Request.

A.7.4 Resource Management

Resources belong to the server, but are managed in a separate class and contain functions for adding and removing of bookings of the resource at a particular time interval. These booking sheets are flexible, as they treat time a continuous variable with no requirements to slotting of time. The resource also contains functions to handle certain events, like a server's offer reservation timing out, a client connecting to the resource, maintaining statistics about usage and checking availability.

A.8 Performance Evaluation

In this section we investigate the performance of the our implementation with regard to the two most performance relevant aspects of a discrete event simulator, the event queue and the thread switching.

A.8.1 The Micro-Benchmark

Event Ping-Pong

For both performance evaluations, we set up a minimal benchmark application that operates the simulator with limited parameters. Our benchmark application has the parameters p for the number of simulation participants, i for the number of initial events and n for the number of events to be processed. The benchmark first creates all the simulation participants, then fills the event queue with the initial events. The event times are chosen from a uniformly distributed random number distribution with a fixed maximum value of T, the events are assigned to randomly chosen simulation participants. Then the timed benchmark starts, and the simulation participants process these initially inserted events. For every retrieved event, they send out a new event, again randomly assigned to one of the simulation participants with uniformly distributed time and the same maximum time T. The benchmark comes to an end, when a simulation participant recognises that it processes the n^{th} event and signals the benchmark to interrupt execution and to stop the timing clock.

Memory Activity

The benchmark code itself avoids any calls for memory allocation during its timed run and reuses the old message objects. However, implicitly there are new objects created, because every call to `SendEvent` leads to the simulator allocating memory for the new event. Thus, it is unavoidable that during the timed run of the benchmark the Java garbage collector might get active to collect the orphan event objects. This might lead to performance variances that can only be amortised over a larger number of function calls and an extended timing period.

Simulation Phases

Due to its simplicity, this benchmark ensures that its execution time is mostly spent in the simulation module itself, not the benchmark code. Further, it ensures that events are retrieved from the heap at an average depth, as this would occur in a practical simulation, not at extreme cases, such as the top or bottom of the heap.[5] Another aspect in which

[5] A more detailed study would apply several different event time density distributions, ranging from balanced ones to differing front and tail heavy ones. For the purpose of our analysis we considered the most common case to be sufficient.

the benchmark is realistic is that for every retrieved event we insert a new one. Usually a discrete event simulation contains three phases, a buildup, during which the event queue increases, the main processing phase in which it is more or less constant, and a wind-down phase during which the event queue is getting shorter until the simulation completes with no further events to process. With our set up of the time measurement, we only measure the second phase. However, our benchmark measures an equal number of insertion and event extraction operations, as is done during the course of a usual discrete event simulation. We could have chosen to simulate the buildup and wind-down phases of the simulation too, but for the performance most critical is the middle phase with the long event queue, which is why we focus on this part for the event queue performance. Moreover, with this construct we keep the length of the event queue constant after its initiation, which makes the performance measurements more transparent.

Hardware Equipment

The numbers of events to be process was chosen to be high enough to ensure a benchmark runtime of at least a minute for every measurement taken. Taking longer measurements minimises variation in operating system behaviour and time measurement inaccuracies. These performance numbers were obtained on a Toshiba laptop with a 2.2 GHz pentium4 CPU, 512 MB RAM, running Windows XP, using Microsoft's J++ compiler and JView as the Java runtime environment and promoting the executing process to real-time in the operating system's scheduling.

A.8.2 Event Queue

Figure A.8 shows the rate of events processed by the benchmark, dependent on the length of the event queue. In order to isolate the performance of the event queue, we set this benchmark to run with only one simulation participant. In so doing, we have only one executing thread and no thread switches in the measurement. The performance of the graphs uses the metric "events per second". This actually includes two event queue operations for every processed event, namely `Extract-Min` and `Insert`. We chose not to measure either of these operations in isolation, as they both are dependent on the length of the queue and it is then not possible to measure a large number of operations successively, average the time taken, and measure the operation's speed relative to a certain queue length.

We can see that the performance of the event queue scales very well, even for long queue cases. As the scale on the x-axis is displayed in log-scale, and the graph's performance degradation can be approximated linearly, we can conclude that the performance of this benchmark follows in the average case log-time. In our implementation, `Extract-Min` is on average a constant-time operation and `Insert` is on average a log-time operation. Therefore, the observed log-time behaviour of the benchmark executing both operations in combination satisfies our expectations. With larger numbers of events in the event queue, our performance measurements are influenced by runtime and operating system activities. For more than one million events in the queue, the operating system extends the available

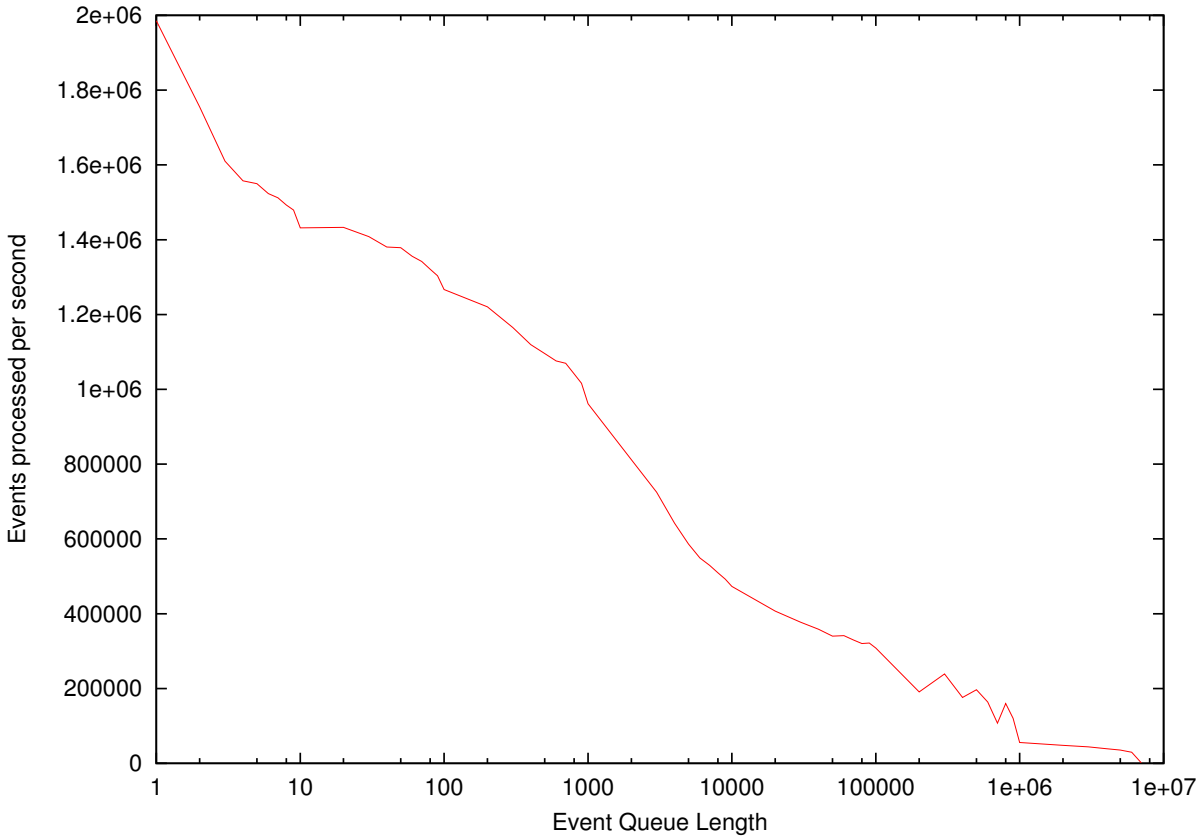

Figure A.8: Event queue processing benchmark.

RAM through swapping. For seven million events in the queue, the benchmark runs out of RAM altogether and slows down by more than an order of a magnitude (30000 $\frac{Events}{s}$ to 1205 $\frac{Events}{s}$) due to constant swapping. It is our expectation, that given a larger RAM, the event queue's performance would scale up to larger queues with continued log-time behaviour.

A.8.3 Thread Switching

Here, we measure if our implementation of thread switching, as described in section A.3.4, is efficient and scalable. In figure A.9 we show the rate of thread switches per second, depending on the number of treads used by the system, using our micro-benchmark. In order to isolate the time spent in the thread switching activity, the event queue only ever holds one event, which makes its operations instantaneous. The rate of thread switching declines with larger numbers of threads present, because the system also needs to manage the threads in lists and run queues.

The performance of this way to control the simulation threads scales well. For less than 50 simulation threads the switch time is almost constant and for less than 1000 simulation threads the degradation is only 35% from the peak. Even for several thousand threads, the performance does not decline dramatically and at 7000 threads it still maintains 35% of the peak performance. At this point however, the system is running out of resources

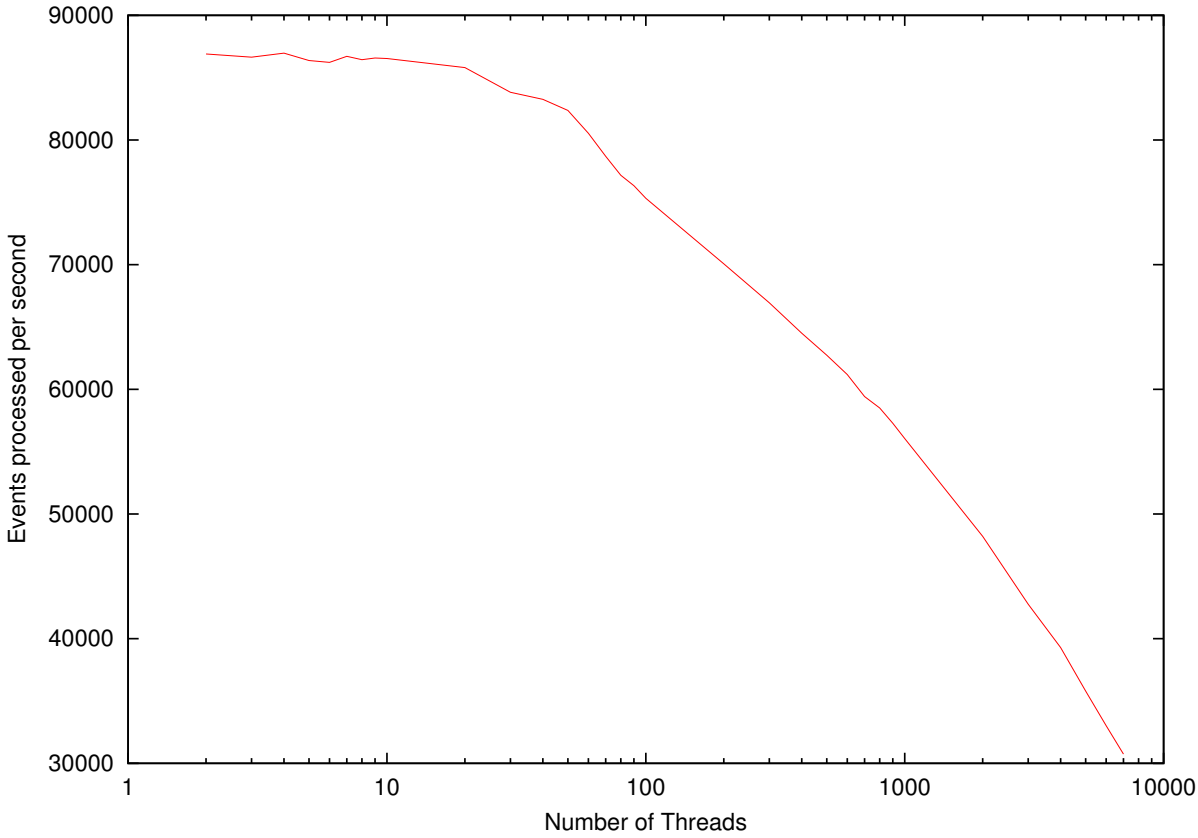

Figure A.9: Thread switching benchmark.

and it is barely able to create further extra threads.

Appendix B

The Scenario and Language of the Allocation Protocol

Since the allocation protocol presented in chapter 3.2 is integral to the construction of the pricing model in that chapter, we present in this appendix the settings and further possible implementation details for this allocation protocol. First, we give an overview of the market scenario (section B.1) for this protocol, with its participants being the providers of resources (section B.3) and clients with their applications (section B.2) who seek these resources.

Next, we describe the scenario's participants at different levels of technical abstraction. We start with the highest level and give some examples for what kind of organisations the clients could be and what kind of applications these have (section B.2). Correspondingly, we give some examples for provider companies and what kind of resources they might offer to the clients (section B.3). At the next level, we briefly discuss how the participants' objectives are represented by software agents (section B.4). At the most technical level (section B.6), the providers make themselves accessible through web services technology, which we introduce in section B.5. This enables the clients to discover (1) the available resources and (2) the communication format, defines (3) the message formats for the interaction between the parties involved, and finally binds the parties together to facilitate (4) the communication for the service transactions between clients and providers.

The remainder of this chapter presents the format of the protocol (section B.7). There, we describe the framework of transactions between both parties, which sets the contractual constraints at the level of the interacting software agents. To illustrate how these transactions might be implemented with greater detail at the communication level of web services, we include some possible extensions.

The main feature of this allocation protocol is that it enables the clients to negotiate with all the providers at the same time and to choose the best offer. This facility instigates competition between providers of services and is a fundamental element for the pricing model in chapters 3 to 4 and the reputation model in chapters 5 to 7.

A secondary feature of this allocation protocol is that it is able to allocate the services as a reservation for a specific time period. This is a necessary feature for allocating

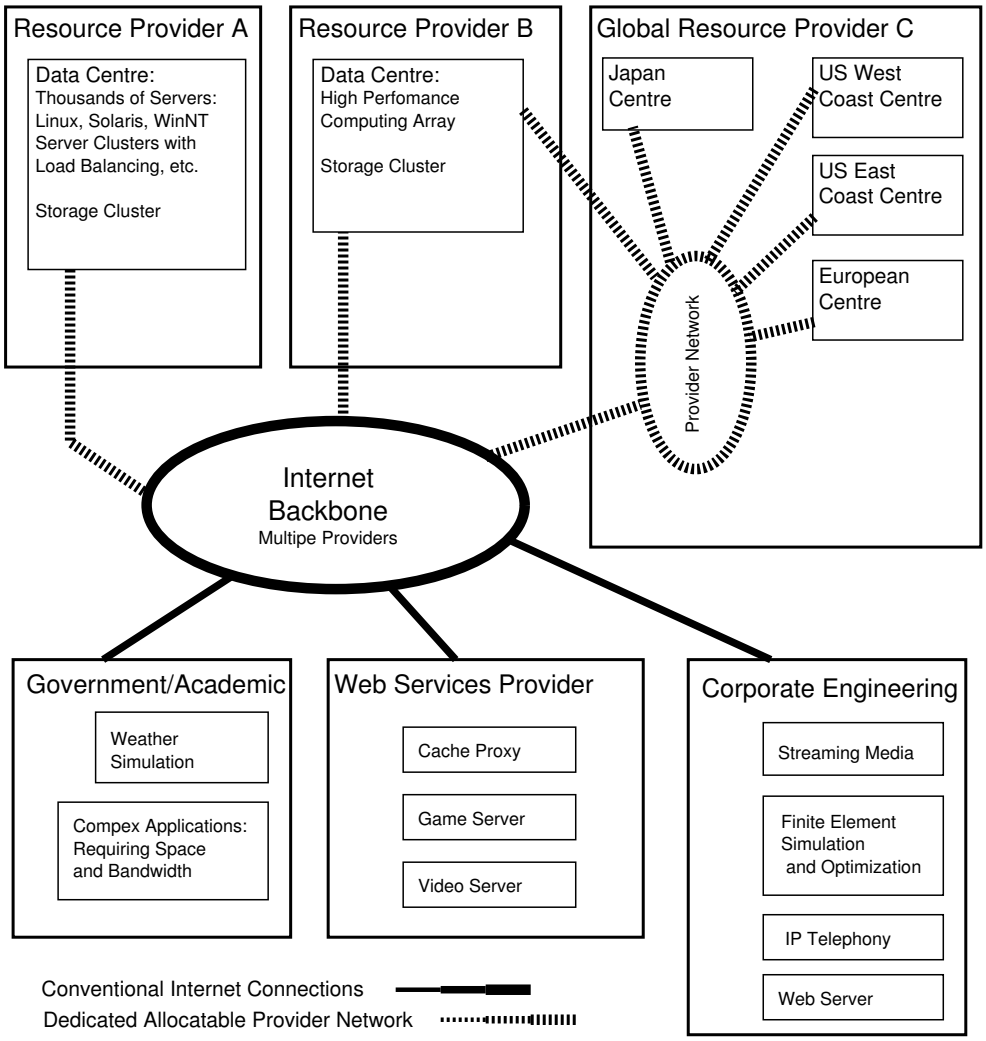

Figure B.1: Example Application Scenario

resources as a service and is essential to the pricing model. The reputation model does not require this feature and is applicable to any kind of service, not only resources, but moreover is applicable to any kind of entity that can be rated.

Finally, we discuss how security concerns are addressed through the protocol design and web services security (section B.9).

B.1 Scenario Overview

Figure B.1 shows a small scenario with some instances of providers and clients. Practically, there should be hundreds or thousands of different participants, also with more variety than we show here. On the side of the clients, who demand resources, we might find organisations such as Internet Services Providers (ISPs), corporate IT departments and data processing departments, but also government departments and universities that will benefit from allocating their resources automatically through web services. In section B.2

we provide some examples for the applications these clients could run to motivate their resource usage.

Providers are data centres that advertise their services to be accessed by the clients. Most commonly, these data centres are set up by companies that try to make a profit from operating this facility. However, some of these data centres may also be set up by other organisations, such as the government/universities in the case of super-computing centres and Grid infrastructure [91], or an ISP offering (integrated) networking and processing services. We also would expect that some providers may in turn be clients of other resources themselves. Extending such relationships transitively leads us to a supply chain of services and resources.

Traditionally, Internet traffic is not charged individually, nor is it possible to allocate the resources to individual streams, therefore we drew the Internet backbone as one large cloud in our diagram. When client applications use the Internet as a general resource, as a web server for example would, they have to take into account the performance properties of the Internet as it is available. However, certain Network connections are being allocated and charged to individual entities. These are for example the connections from the data centre to the Internet backbone and also dedicated lines for between data centres.

B.2 Client Applications

Here we list some of the possible applications and their particularly demanding characteristics in their resource usage profile. This list does not claim to be all inclusive or even representative, but rather is supposed to suggest the existence of potential demand for the ensuing developments. Moreover, all these applications would require very different levels of provider cooperation and standardisation in the user communities. Therefore, these applications would have differing chances to be successfully hosted by an automatically allocated provider:

1. *Http servers* require mainly high bandwidth for storage and Internet access and medium amounts of storage capacity.
2. *CGI processing servers* work in connection with http servers and require substantial cpu processing resources as well as low-latency data access bandwidth, since such applications usually are used to search through databases.
3. *Search Engines* require low latency for the data access and possibly need to store large amounts of data for a long time.
4. *Cache proxy servers* require high bandwidth for the storage data access and Internet access, as well as large amounts of long term data storage.
5. *Media Servers* delivering video/audio streams, mainly require high bandwidth with a guaranteed minimum bandwidth for the Internet access and possibly large amounts of long term data storage.
6. *Distributed game servers* require low latency Internet access and medium amounts

of cpu processing capacity.

7. *Supercomputing Applications* are mainly simulations and other parameter optimisation applications. Examples include weather simulations, finite element engineering simulations, fluid dynamics simulations and discrete event simulations used for strategic decision making (military and economic), or traffic optimisation problems.

   The resource needs for supercomputing applications tend to be excessive as it is always possible to increase the fidelity of the results by expanding the data set, increasing the depth of computation steps, or reducing the granularity of the computation steps. To increase processing speed, supercomputing applications require many cpus (typically dozens to hundreds and up to several thousands) and high bandwidth with low latency interconnection networks between the computation nodes as well as sufficient disk storage.

   Traditionally, a supercomputer's computation nodes were interconnected with a specially designed backplane bus, but today with the cost efficiency of commodity workstations clusters, supercomputers use network type high performance interconnects such as Fiber Channel, Myrinet and Scalable Coherent Interface (SCI). Researchers adapt the programming model of the applications to tolerate the proportionally higher communication latency versus cpu processing time.

   Temporarily, these applications may require a lot of storage during the processing time and then require suitable high bandwidth Internet access to transport the input and output data to the client. Some supercomputing applications work integrated in combination with several such large scale simulations (i.e. a weather simulation integrated into a stratospheric chemistry simulation), in which case one needs a high bandwidth interconnect between the separate simulations, particularly if these are run at separate locations.

8. *Cycle Crunching Applications* are exhaustive search applications, i.e. code breaking or genome sequencing. These seek for endless computing resources, but one of their main motivations is low cost. Such applications mainly seek for many CPU cycles and contrary to Supercomputing Applications, such crunching jobs are not interacting tightly. Because of the low demands on interaction latency, these applications are easy to schedule.

B.2.1 Allocation Strategies

The clients will need individualised strategies to allocate the appropriate resources, and for our purposes these strategies are unlimited beyond the request protocol and its format. The request protocol does not put any constraints on the particular needs of individual applications, since the variation in the actual requirements is rather wide as can be seen in the differing demand profiles above.

B.2.2 Platform Independent Applications

For some of the applications presented above, the code could be installed by the provider and the clients only have to supply the data, such as in the case of a http server. However, the majority of applications assume that the clients supply their own code to be run by the hosting providers. In order to make this possible, the providers will be able to cater for a set of popular platforms, such as Apple, Linux, Solaris, Unix (manufacturers' flavours), Windows [NT, 2000, XP]. These platforms depend on the hardware the provider has available; however, he can choose to offer virtual platforms to gain flexibility with the assignment of his hardware. Possible virtual platforms include:

- *Java* [87] is Sun's development of a virtual machine platform.
- *.Net* [63] is Microsoft's development of a virtual machine platform and interoperability technology.
- *VMware* [93] makes the Intel-based systems virtual and provides (remote) control tools.
- *XenoServers* [75, 9] is a research effort that virtualises machine functionality at the lowest level and allows applications to be sent to the remote host including their own operating system.

To create portable cluster applications we can virtualise the communication with protocol standards such as the Message Passing Interface (MPI) [40] and Java Remote Method Invocation (RMI).

B.3 Resource Providers

Resources are managed by a broker service, which is run on the site of the resource providers (i.e. a data centre) in the form of a web service. The allocation/reservation protocol's format, or also referred to as the demand description language, recognises three primary kinds of resources: computation, networks and storage. Any application will need all three resources, though depending on its characteristics, the particular focus and combination of resource demand may be shaped very differently, as we have seen in the above section. The primary types of resources are broken down further in more detailed categories to match the differing demand profiles by the applications.

1. *Computation* in the form of server type processing units. At the low end, these units start with cheap PC-based servers and scale up to more robust technology from various manufacturers such as Dell, HP, IBM, SUN, etc., containing several CPUs and can be rack mounted to form larger clusters. These clusters may consist of tens, hundreds, up to thousands of machines and in order to maintain high throughput will be interconnected with high bandwidth networking, e.g. ATM [56], Gigabit Ethernet [33] and Fibre Channel [21].

   At the higher end, such clusters can function as supercomputers if they are able to support low-latency requiring applications (High Performance Computing, Database

Clusters), and therefore are interconnected with low-latency network interfaces such as Myrinet [68], Scalable Coherent Interface (SCI) [36] and Scramnet [89]. To meet demand from certain applications, some servers will be equipped with additional amounts of memory and storage. High performance for supercomputing application with strong demands on even tighter communications coupling may then be served with dedicated symmetric shared memory parallel (SMP) machines from vendors such as Cray (X1), HP, IBM (SP), NEC, SUN (HPC) and SGI (Origin).

2. *Network bandwidth*: At present, in the Internet in general we are not able to reserve bandwidth from end to end. However, in a data centre, a provider is able to allocate specific amounts of access bandwidth to a client and their application. This access connection links the data centre's servers to the backbone of the Internet. Depending on the technical scenario, the Internet provider who is supplying this access connection, may also be able to grant transmission guarantees within the limits of his own network. For example, a media server application also needs to reserve sufficient bandwidth to deliver the streams from the streaming servers to the backbone of the Internet.

   Further, the network providers are able to offer dedicated channels between interconnection points their network is linked to. For example, a grid computing application could allocate a transatlantic link with special QoS parameters to connect two supercomputing sites for a meta-computing [31] architecture.

   Another potential application for bandwidth reservation is bandwidth trading between Internet Service Providers engaged in Internet transport. To reserve network bandwidth, ATM [56] and RSVP [10] are suitable technologies.

3. *Storage* for longer term data deposit on storage arrays: Applications may need in connection with their computational requirements additional space to store larger volumes of data. To serve their purpose the storage arrays are connected to the servers with high throughput I/O access (e.g. Fibre Channel [21]). Depending on the kind of expected applications catered for by the computing architecture, the storage arrays will vary in their access bandwidth, access latency as well as reliability and redundancy.

B.4 Negotiating Agents and Brokers

In order to facilitate the automated allocation of resources from providers to client applications, we need several intermediary management software modules. Since these software modules are autonomous in their operations and are set up to serve the purposes of their operator, while only being limited by the constraints of the communication protocol (section 3.2) that facilitates the platform for negotiation with the other parties, such a module is often called an agent[1]. Client applications employ an agent to seek for the resources.

[1]The motivation for using the term agent is in the personalisation of the software component. This does not necessarily imply use of concepts like mobile code as sometimes anticipated by artificial intelligence researchers and similarly feared by security researchers.

The providers of these resources match this negotiation with a brokering agent.

The allocation and negotiation protocol specifies some constraints on how the client agents can format their requests, however beyond the formatting, we do not assume any limitations on the side of the client. In fact, we would expect that clients employ some methods from artificial intelligence in order to use past experience to decide where to request services from, which service to choose, which prices are acceptable and which options to favour. Further, we expect that every client will have different agent technology, and even if they would employ the same methods and algorithms, we expect them to parameterise them differently and use individual input data.

Similarly, the brokering agent, responsible for the management of the resource provider's services, will be bound by the formatting of the communication protocol and any contractual ties that yield from the negotiation with the client. However, we will not assume any further constraints on the behaviour of the providers' agents, such that these are free to implement methods that maximise their own benefit. With this goal at hand, the brokers will implement appropriate tools that estimate the market demand at any given time, implement pricing policies that maximise their profit, given the market demand. Further, the brokering agent will adjust the resource scheduling policies in accordance with the economic policies that maximise his profit. For example, this may mean that the resource broker arranges for overbooking of scheduled slots in anticipation of cancelled contracts and also with the statistically calculated possibility of having to therefore fail on contract terms at some time. In this case the provider would have to take into account the economic impact of the contract penalties from undelivered service versus the gain from overbooking and additionally the loss in reputation with respect to fulfilling his contractual promises.

Since an agent's cost of decision time is negligible compared to a human's, the rate at which these negotiations are done can be very high. Hence, the time scale for allocation periods can be very low.

B.5 Open Services Platform - Web Services Technology

We need an enabling technology platform for our protocol (section 3.2). Our protocol assumes that the following list of operations can be performed through existing technology, without any human invention and in a secure fashion: (1) Effortless discovery of resource providers - practically zero search cost. (2) Efficient negotiation with the providers - automated matching of available items and preferences. (3) Accessing and operating the controls of the resource autonomously - automated API interaction and resource control.

Many such service platform architectures exist, notably CORBA [70], TINA [90], ANSA [6] and Web services. While many such platforms would have provided the necessary functionality to enable our protocol's purpose, we chose to focus on Web services to describe the potential implementation of our protocol. We favour Web services, because at present, out of all the service platforms, it appears most likely to achieve standardised

and broad-reaching adoption.

In the next section we will discuss how the individual building blocks of Web services facilitate the individual actions of the protocol, where here we introduce the general functionality of Web services technology and explain the relevance of these functions for our protocol.

The availability of these operations is a prerequisite for our protocol, but moreover, the economic assumptions of the pricing model in chapter 3 draw on these technical parameters. Without the premise of zero search cost, we could not assume perfect competition for our scenario, which leads to the economically challenging situation, where providers potentially do not obtain any revenue, and our pricing model counters this situation. Further, an automated reputation aggregation model, as we present in chapter 5, would not be very necessary if one were to assume that human operators are part of the allocation decision process, since these might take over such a function involuntarily. Web services, through the various protocols that encompasses them, provide the technical platform to solve each of the above business interaction requirements. Many of these Web services protocols are still under development, either through further standardisation or refining and adding of features, and need to further evolve to practically solve the above stated requirements. The Web services protocols are based on XML and its derivatives (i.e. XML schemas), which makes them self-descriptive and human readable. The Web services protocols are based on XML [35], to make for a self-descriptive human readable format. As a good introductory example, Gottschalk et. al. describe the practical deployment scenario of Web services to facilitate an online market of sailboat parts [37]. To give a base for our exposition of how Web services integrate with the protocol presented in the following section, we here briefly introduce the main functionality of the three basic Web services protocols, UDDI, WSDL and SOAP.

B.5.1 UDDI — Universal Description, Discovery, and Integration

UDDI [18] provides the registry piece in the Web services protocol stack and facilitates the first one of the three prerequisite operations stated above. It performs thee functions, namely the (1) *publish* function, which is concerned with how a Web services provider registers itself and its services, (2) the *find* function deals with how a client application finds and matches the description of a Web service, and (3) *bind* deals with how a client application connects to and interacts with the Web service.

In brief, the UDDI Business Registry works as follows: (1) Standards bodies provide the UDDI registry with descriptions of various kinds of services. (2) Service providers submit descriptions of their services. (3) The UDDI registry assigns unique identifies to each service. (4) Clients query the registry to discover services. (5) Clients connect to the provided Web services.

A UDDI registry stores three kinds of data. *White pages* contain information such as the name of the business, contact information and other descriptions. *Yellow pages* contain information that classifies the company, based on standard industry classification

mechanisms, i.e. NAICS, UNSPSC. The *green pages* contain technical information about the service, such as information about business processes, service descriptions and binding information about the services.

A UDDI registry can be deployed in various scenarios, depending on the market in which it is mediating. The most popular scenario is that of a (1) *global business registry*, as found at `UDDI.org`, which is operated by IBM, Microsoft and HP. Such registries have a general purpose and are open to serve any kind of market. However, UDDI registries do not have to be installed only in open markets, but can be employed in a closed (2) *business-to-business marketplace*. In such a deployment, both publish and find operations are restricted to the legitimate businesses registered with the marketplace. The kind of marketplace then provides value added services like service monitoring, which ensures firstly, that participants in the registry have been vetted by a rigourous selection procedure and secondly, that the registry entries conform to market standards. Yet another UDDI deployment is a (3) *Portal UDDI*, where the publish operation is limited to one provider, namely the operator, and the find operation is open to external clients. Lastly, an (4) *Intranet deployment* of a UDDI registry allows different departments of a single organisation to integrate their operations. All the private deployment scenarios, numbers (2-4), have an advantage over the open global registry deployment, number (1), since they are able to restrict how the service is described and can screen the approved businesses. In the case of the open global registry, one will have to provide these same assurances through a set of additional methods, such as reputation services.

B.5.2 WSDL — Web Services Description Language

WSDL [98] is a standardised language, developed to describe Web service interactions. It provides for a client who wants to send a SOAP message to a provider with a description of how to format this message. Such a description is achieved through a composition of definitions. A *service* is defined by a set of one or more network endpoints, so-called *ports*. Each port is associated with a specific *binding*. It is the binding that defines how an abstract set of operations and messages are bound to a port, according to a specific protocol. A binding maps a specific protocol to a *port-type*. A port-type is composed of one or more *operations*. Operations represent an abstract set of things that the service can "do". Each operation is composed of a set of abstract *messages*. Messages represent the data that is communicated during the operation. Each message contains one or more pieces of data, which are defined by *types*.

B.5.3 SOAP — Simple Object Access Protocol

SOAP [97] is the protocol that defines the format of the messages which were described by the WSDL description. SOAP defines the message syntax, the data payload and types, in addition to the message purpose. It does all this while providing methods for serialising data and formalising how these messages should be exchanged over HTTP, even though its applications could substitute these definition methods for its own. The SOAP design is modular and extensible, so that it can, for example, allow one to replace the transport

protocol HTTP with SMTP. With this set of definitions, clients can send messages to a Web service to invoke one of the service's operations and receive a reply. SOAP also defines a model for message exchanges, which can be used to model RPC-style request-response invocations, or alternatively more sophisticated models with unidirectional message chains through endpoints and definitions for endpoint processing behaviour.

B.6 Facilitating the Allocations With Web Services

In the previous sections we explained why clients might want to allocate external resources from providers, what kind of resources these are, and the roles both parties are taking. In this section we explain how Web services technology builds the technical platform for clients and providers to communicate with each other. We introduced Web services in general in the previous section (appendix B.5), here we focus on how its building blocks fit together with our allocation protocol.

In order to establish the relation between the client and the provider, we install a third party, which is a well known entity that is recognised by all clients and providers. To establish such registry services in the web services world, we have UDDI directories (Universal Description, Discovery and Integration [18]), which the client will turn to when attempting to allocate a resource. The registry might be a general one, or one that specialises on the kind of resources the client is seeking for, in either case, the UDDI will have a standard for the description of resources that the client is looking for. By searching through the yellow pages of the UDDI directory, the client obtains a list of providers offering a service that matches his resource requirements. In the white pages of the directory he can find out more about the provider, for example the location, because the client's application might depend on particular network connectivity constraints. In the green pages, the client can obtain the technical information for every service he is interested in. There he will find a WSDL (Web Services Description Language [98]) description, or a URL pointing to such a description.

Having obtained this WSDL description from the UDDI, the client is able to start negotiating with the resource provider. This negotiation will involve message passing on the base of SOAP (Simple Object Access Protocol [97]) invocations and will yield all the parameters for the client and provider to actually complete their business transaction. Here, they will settle the price and other conditions, such as the QoS level if the service allows for variable performance levels. Likewise, if both parties are not able to agree on the same terms, they abort their negotiation. If both agree and commit on the service contract, then the provider will pass a WSDL description to the client that allows the client to connect to the service and make use of it.

When a client makes use of a web service, he interacts with the service through SOAP messages. The WSDL description that the client obtained from the previous negotiation with the provider, describes the operations the client can invoke as part of this web service. If for example our web service's function is to run an application on a supercomputer, then the client uses SOAP's RPC-style invocations over HTTP to transmit the application code as well as the corresponding input data. Further SOAP invocations initiate the

execution of the application and finally also arrange the retrieval of the output data of the application.

B.7 The Allocation and Negotiation Language

The *allocation language* describes the message contents of the five phases of the message protocol (section 3.2). Each of the messages shown in Figure 3.1 have their own language and possible variants of replies depending on the keys used in the requests.

The messages are encoded through SOAP messages with a variable depth of message detail. In practical terms this means that a request has a list of primary description elements, some of which are optional, each of which may have secondary description elements where some or all may again be optional, and this recurses down to the leafs of the syntax tree. For illustration purposes we mention some example qualifiers in [brackets]. The listed keys are not meant to be inclusive or representative and in practice, the actual format of each message would involve several detailed description standards.

Phase 1: Client Sends Requests for Bids to Resource Providers.

1. *Resource Description* - [E.g. Web Server — Network Connection — CPU Time] - This is the identifier of a resource or service.
   (a) *Architecture* - [E.g. Apache — Linux x86] - The version of the resource or service.
   (b) *Quantity* - [E.g. 4 CPU — 10 GByte] - The number or amount of resources requested.
   (c) *Performance* - [E.g. Web Stone 20 Mbit/s — 850 MHz — 625 Mbit/s] - The speed of the service.
   (d) *Quality of Resource Service* - [E.g. Max latency = 0.5 s in 99% of transmissions] - Further desired contract clauses on service performance (There can be a list of desired performance and QoS keys).
2. *Requested Time* - The details for the advance reservation.
   (a) *Start* - [E.g. 20 Oct 2000 14:00] - The time for starting the use of the resource.
   (b) *End* - [E.g. 20 Oct 2000 15:00] - The time when the client does not need the resource anymore (This is optional because a contract can be open ended, e.g. for allocating storage).
   (c) *Duration* - [E.g. 600 s] - If the resource is only needed for a certain window in the period given above. In general, clients should allow for a generous end time and state the estimated duration (e.g. for a large data transfer), since in this case the resource broker gains flexibility when scheduling this request, and he may be able to give a more competitive price.

3. *Request Timeout* - [E.g. 20 Oct 2000 13:05] - This is how long the client is prepared to wait to collect offers from resource providers. This period could be a measure in the order of seconds, hours, or possibly days if the start time is far into the future. The aim of the design is that the protocol is able to support a wide range of the time scales of advance notices (i.e. as in [96]). The resource provider is not guaranteed that the client is waiting until the end of the timeout, but the client has to expect that most providers will try to make the offer decision as late as possible.
4. *Maximum Request Valuation* - [E.g. $ 1.50] - This is the maximum in the budget of the client, indicating how important this service is. If the provider is running an auction on the service, he will use this figure. The resource provider can make an offer beyond this valuation and also can use it to speculate on a higher price to be obtained, but has to expect that he might be out-bid by other resource providers.

Phase 2: Resource Providers send Offers and the Conditions to the Client.

1. *Resource Description* - [E.g. Web Server — Network Connection — CPU Time] - This is the description for a resource or service, and should match the requested description. Here the provider states that actual available configuration details.
   (a) *Architecture* - [E.g. Apache 1.3.12 — Linux RedHat 7] - The actual version of the service offered.
   (b) *Quantity* - [E.g. 2 CPU — 12 GByte] - The number or amount of resources offered. This can be more or less than the request, depending on whether the provider can satisfy the request or not.
   (c) *Performance* - [E.g. Web Stone 23.2 Mbit/s — 1000 MHz — 625 Mbit/s] - The actual speed of the resource.
   (d) *Quality of Resource Service* - [E.g. Max latency = 1.0 s in 99% of transmissions] - This states the reply to the qualities as requested by the client. The resource provider may decide to offer less or more quality than requested depending on the conditions (e.g. load).
2. *Requested Time* - The details on when the client can use the service.
   (a) *Start* - [E.g. 20 Oct 2000 14:30] - The time for starting the use of the service. The broker can offer a different time than requested, in the hope that this also suits the client.
   (b) *End* - [E.g. 20 Oct 2000 15:45] - The end time of the service allocation. This can be open ended if a special continuous rental is arranged.
3. *Request Timeout* - [E.g. 20 Oct 2000 13:05] - This is the time for the offer to be valid. The client has to decide within this deadline if he wants to accept this offer as a winning bid. The resource broker needs to keep this deadline small, so that he can offer the resource to other clients. On the other hand the offer should be valid until some time after the client's request is timing out.

4. *Cost* - [E.g. $ 1.25] - This is the quote for the client, if he decides to accept this offer. The resource broker needs to calculate this quote competitively, since there will be other brokers also making offers.

Phase 3: Client is Selecting the Winning Offer to the Resource Provider.

This is just the indication that the client agrees on the terms of the resource provider. There is a timeout on this award, which should be insignificant in the general case, because it is in the interest of the resource provider to settle the negotiation as soon as possible.

Phase 4: Resource Provider Sends Contract Acceptance and is Issuing the Electronic Ticket to the Client.

The generated electronic ticket ensures that only the client who negotiated the deal can use the services of the resource. The ticket contains information about the location of the resource in the form of the corresponding URL of this service. The URL points to the WSDL [98] description of the corresponding SOAP [97] invocations to operate the resource. One of these SOAP transactions will demand the electronic ticket.

Phase 5: Client Sends Electronic Payment and Acknowledgement of Electronic Ticket to the Provider.

On receipt of the payment the resource provider activates the electronic ticket at the resource.

Phase 6: Client Uses the Resource.

The client uses SOAP messages to connect to the specified resource, transmits the ticket and uses the resource with the provided parameters to control the resource. At the same time, the management system of the resource provider will have notified all the necessary parts of the resource to allow the client's transaction to continue within the agreed parameters.

B.8 Complex Allocations

Complex Allocations are extensions of the allocation language, with the objective to describe the actual resource requirements of the clients in more detail. This detailed description can be exploited to increase the efficiency of any allocation, and therefore lead to a maximisation of the welfare of clients and providers. Here, we describe two such allocation language extension, bundling and price functions. While bundling is an important and fundamental concept, price functions are just one of many kinds of optimisations that can be implemented to improve allocation efficiency.

With bundling requests, clients are able to specify a set of resources that they want to allocate at a certain price for the whole bundle. A client's valuation for the bundle of resources may differ dramatically from his valuation of each of the individual resources. E.g. the client may ask for time on a supercomputer for his scientific simulation, but only if he also can get hold of sufficient network bandwidth afterwards to move the data in a timely manner, because otherwise he might as well use a slower machinery in the first place. Therefore, it is not possible to derive market mechanisms for the case of allocating resources individually and achieve the same economic efficiency as bundling does, when the clients require complementing resources. Further, bundling can be a convenient way of representing a large range of complex allocation scenarios. For example, we can use fixed size slots to represent the allocation time of a resource and then request and assign bundles of these slots. In such an approach it is easy to combine a multitude of complexities, by reducing these to a bundling problem. However, bundling does create serious computational complexities, since problems such as optimal resource allocation for the provider and optimal resource demands for the clients become NP-complete problems. In practice, one would address these complexity problems with the design of efficient approximation algorithms, as has been done by Zurel and Nisan for combinatorial (bundling) auctions [100].

Another opportunity to increase economic efficiency is to allow clients to specify their resource per price preferences through functions. Practically, such a function could be a piecewise linear relation between how much he is willing to pay and the quantity of resources obtained. For example, a client could ask for a minimum of 10 CPUs, which are worth to him up to $ 2.00 for the specified time, and for every additional CPU he would be willing to spend $ 0.10 extra, up to a maximum of $ 4.00. With this more detailed valuation function, the provider is able to maximize his profit and allocates the resources accordingly.

B.9 Security

In this section we describe how security is addressed in our protocol on the basis of Web services security. This description is not exhaustive and does not cover all the necessary security aspects, but addresses some of the general design implications of Web services security.

B.9.1 Web Services Security

Web services security is at present still an open research area in itself, a joint white paper by IBM and Microsoft about a roadmap is the current state-of-the-art [45]. In order to ensure security properties, such as encryption and signatures, standards such as SOAP-Security, XML Digital Signature, XML Encryption and SSL are available, due to the natural extensibility of the core XML-based Web services model. Existing standards for authentication, such as X.509, public-key certificates, Kerberos and password digests also can be integrated into Web services security models. It can be assumed that with the

correct use of these standards, these basic security properties can be met.

However, a more general problem for Web services security is the management of access control in a web of services and a web of policies that govern the access to the services. In order to be able to create security formats and access mechanisms for this management problem, a number of security elements have been introduced:

1. A *Security Token*, which is a representation of security-related information, i.e. a X.509 certificate, Kerberos ticket, SIM card token, user name, and it can be signed by an issuer of the token.
2. *Claims* are statements about a *subject*.
3. A *Proof-of-Possession* might for example be the private key associated with a security token that contains a public key.
4. The *Web Service Endpoint Policy* specifies all the claims that are required in order to process messages, e.g. proof of user or group identity.
5. The *Claim Requirements* are tied to a message or action (parts or whole). E.g. a service may require a requestor to prove authority for purchase amounts greater than a stated limit.
6. The *Intermediaries* pass on messages from the initial requesters, for example to perform actions such as routing the message or adding headers, security tokens and encryption.
7. An *Actor* is an intermediary or endpoint and processes a SOAP message.

Combining these security elements, we are able to construct security mechanisms in which:

1. A Web service can require that an incoming message proves a set of claims (e.g. name, key, capability). The set of required claims forms a policy.
2. A requester can send messages with proof of the required claims by associating security tokens with the message.
3. When a requester does not have the required claims, he can try to obtain these by contacting other Web services. Such other Web services, which are named security token services, will in turn require their own set of claims. Security token services broker trust between different trust domains.

The central threat for web services security is that "trust is transitive" and while this applies not only to web services, it does apply particularly well to these. One of the big promises of web services is that these form business chains, get combined and reused in a fluid, dynamic fashion. This collides with the promise that a web service behaves according to limited, secure behaviour and does not mishandle the trusted data. Will a service behave as promised? This is a question that might be addressed through penalty constructions, whereby deviations of any kind incur some form of repercussions. However, this does not prevent the misbehaviour in the first place. Therefore we advocate the use of reputation systems to obtain an indicator on how likely a service provider is to fulfil

the promised security properties and other performance aspects.

B.9.2 Allocation Protocol Security

In order to limit the possibilities for fraud of any kind, we require that all the participants (providers and clients) are registered at a certificate authority that issues cryptographically secure PKI-keys. This will prevent the participants from many illegal behaviours and makes the contracts they negotiate enforceable. For example, if a provider cashes in on a contract, but then later does not fulfil all the contractually negotiated terms, he can be legally held liable and possibly be bared from further trading as a resource/service provider.

B.9.3 Trusting the Client Applications

One concern for the security of the providers is that they do not trust the clients' applications. In most of the cases of potential client applications in section B.2, we assume that the clients supply their own code for the providers to execute on their machines. Without a thorough inspection, the providers cannot, and should not trust the code supplied by the clients. The code could misbehave by causing the machine to crash because of faulty programming, or it could exceed the agreed resource demands and hog the resource, or it could be malicious and either harm the operations of the provider or other installations. In any of these cases, the provider will want to guard himself against such potential misbehaviour of the client code. This can be done best through visualisation of the access to the hardware, as we described earlier in section B.2.2. Through visualisation, the providers are able to explicitly police the resource control of the client applications without having to inspect the and trust the code. For example, VMware [93] supplies tools that are able to virtualise an IBM mainframe machine, such that it is able to securely host thousands of virtual Linux machines. And XenoServers supply low-level mechanisms for isolating (and accounting) of application resource access [9].

Bibliography

[1] Karl Aberer and Zoran Despotovic. Managing Trust in a Peer-2-Peer Information System. In *Proceedings of the Ninth International Conference on Information and Knowledge Management (CIKM)*, 2001.

[2] Blaise Allaz and Jean-Luc Vila. Cournot Competition, Forward Markets and Efficiency. *Journal of Economic Theory*, 59:1–16, 1993.

[3] Yair Amir, Baruch Awerbuch, and R. Sean Borgstrom. A Cost-Benefit Framework for Online Management of a Metacomputing System. In *Proceedings of the ACM International Conference on Information and Computation Economies (ICE-98)*, October 1998.

[4] A. Anastasiadi, S. Kapidakis, C. Nikolaou, and J. Sairamesh. A Computational Economy of Dynamic Load Balancing and Data Replication. In *Proceedings of the ACM International Conference on Information and Computation Economies (ICE-98)*, October 1998.

[5] Ross Anderson. *Security Engineering*. Wiley, 2001.

[6] Architecture Projects Management Ltd., http://www.ansa.co.uk. *ANSA/ODP*.

[7] Kenneth J. Arrow. *Social choice and Individual Values*. Yale University Press, 1963.

[8] Yannis Bakos and Chrysanthos Dellarocas. Cooperation Without Enforcement? A comparative analysis of litigation and online reputation as quality assurance mechanisms. MIT Sloan Working Paper No. 4295-03, Massachusetts Institute of Technology (MIT), Sloan School of Management, March 2003.

[9] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the are of virtualization. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP19)*, 2003.

[10] R. Braden, L. Zhang, D. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP). Version 1 functional specification, Internet Draft, Internet Engineering Task Force, 1996.

[11] Sviatoslav Braynov and Tuomas Sandholm. Incentive compatible mechanism for trust revelation. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Bologna, Italy, July 2002.

[12] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[13] Sonja Buchegger. Coping with Misbehavior in Mobile Ad-hoc Networks. PhD Dissertation, Ecole Polytechnique Fegerale de Lausanne, January 2004.

[14] A. Chavez, A. Moukas, and P. Maes. Challenger: A Multiagent System for Distributed Resource Allocation. In *Proceedings of the First International Conference on Autonomous Agents '97*, Marina Del Ray, California, USA, 1997.

[15] Mao Chen and Jaswinder Pal Singh. Computing and using reputations for internet ratings. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, October 2001.

[16] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. McGraw-Hill, 1990.

[17] Dan Cosley, Seve Lawrence, and David M. Pennock. Referee: An open framework for practical testing of recommender systerms using researchindex. In *Proceedings of the 28th VLDB Conference*, Hong Kong, China, 2002.

[18] Cross-industry initiative, http://www.uddi.org. *Universal Description, Discovery, and Integration*.

[19] Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, Pierangela Samarati, and Fabio Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks). In *Proceedings of the ACM CCS'02*, Washington, DC, USA, 2002.

[20] Ketil Danielson and Judith Molka-Danielsen. Admissions and Preemption in Auction-based Reservation Networks. In *Proceedings of the ISQE Workshop*, MIT, Boston, MA, USA, December 1999.

[21] Paul Davis, Mike Austin, and Alan Fearday. Optimizing Data Storage Performance for Fibre Channel Networks. White paper, Systran Corporation, 2003.

[22] Chrysanthos Dellarocas. Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In *Proceedings of the ACM Conference on Electronic Commerce*, pages 150–157, Minneapolis, Minnesota, USA, 2000.

[23] Chrysanthos Dellarocas and Paul Resnick. Online reputation mechanisms a roadmap for future research. In *Proceedings of the First Interdisciplinary Symposium on Online Reputation Mechanisms*, Cambridge, MA, USA, April 2003.

[24] Roger Dingledine, Nick Mathewson, and Paul Syverson. Reputation in privacy enhancing tehcnologies. In *Proceedings of the 12th conference on Computers, Freedom and Privacy*, San Francisco, California, USA, April 2002.

[25] J. R. Douceur. The sybil attack. In *First International Workshop on Peer-to-Peer Systems (IPTPS '02)*, Cambridge, MA, USA, March 2002.

[26] Boris Dragovic, Steven Hand, Tim Harris, Evangelos Kotsovinos, and Andrew

Twigg. Managing trust and reputation in the xenoserver open platform. In *Proceedings of the 1st International Conference on Trust Management*, Heraklion, Crete, May 2003.

[27] Boris Dragovic, Evangelos Kotsovinos, and Peter Pietzuch. Xenotrust: Event-based distributed trust management. In *Proceedings of the Second IEEE International Workshop on Trust and Privacy in Digital Business (DEXA-TrustBus'03)*, Prague, Czech Republic, September 2003.

[28] D. F. Ferguson, C. Nikolau, and Y. Yemini. An Economy for Flow Control in Computer Systems. In *Proceedings of the INFOCOM*, 1990.

[29] Domenico Ferrari, Amit Gupta, and Giorgio Ventre. Distributed Advance Reservation of Real-time Connections. In *Proceedings of NOSSDAV'95*, 1995.

[30] Peter C. Fishburn and Andrew M. Odlyzko. Dynamic behavior of differential pricing and quality of service options for the internet. In *Proceedings of the ACM International Conference on Information and Computation Economies (ICE-98)*, 1998.

[31] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. In *Proceedings of the Workshop on Environments and Tools for Parallel Scientific Computing, SIAM*, Lyon, France, August 1996.

[32] R. M. Fujimoto. Parallel Discrete Event Simulation. In *Proceedings of the Winter Simulation Conference*, pages 19–28, December 1989.

[33] GigaBit Ethernet Alliance. *10 Gigabit Ethernet Technology Overview*, May 2002. White Paper.

[34] Emanuele Giovannetti and Cristiano A. Ristuccia. Estimating Market Power in the Internet Backbone Using Band-X data. Cambridge Working Papers in Economics, Department of Applied Economics, University of Cambridge, June 2003.

[35] Charles F. Goldfarb and Paul Prescod. *The XML Handbook*. Prentic Hall, 2000.

[36] Damian M. Gonzalez, Alan D. George, and Matthew C. Chidester. Performance modeling and evaluation of topologies for low-latency SCI systems. White Paper, Dolphin Inc., 2001.

[37] K. Gottschalk, S. Graham, H. Kreger, and J. Snell. Introduction to Web services architecture. *IBM Systems Journal*, 41(2), 2002.

[38] Richard J. Green. and David M. Newbery. Competition in the british electricity spot market. *The Journal of Political Economy*, 100(5):929–953, October 1992.

[39] A. Greenberg, R. Srikant, and W. Whitt. Resource Sharing for Book-ahead and Instantaneous-request calls. In *Proceedings of the 15th International Teletraffic Congress*, 1997.

[40] William Gropp, Ewing Lusk, and Anthony Skjellum. *Using MPI: Portable Parallel Programming with the Message Passing Interface*. MIT Press, Boston, MA, second

edition, 1999.

[41] Minaxi Gupta, Paul Judge, and Mostafa Ammar. A reputation system for peer-to-peer networks. In *Proceedings of NOSSDAV'03*, Monterey, California, USA, June 2003.

[42] R. H. Guttman, A. G. Moukas, and P. Maes. Agent-mediated Electronic Commerce: A Survey. Technical report, Software Agents Group, MIT Media Laboratory, 1998.

[43] Christopher T. Haynes, Daniel P. Friedman, and Mitchell Wand. Obtaining Coroutines With Continuations. *Journal of Computer Languages*, 11(3/4), 1986.

[44] HiveCache Software, Inc., http://www.mojonation.net. *Mojonation Technical Overview*, September 2000.

[45] IBM and Microsoft. *Security in A Web Services World: A Proposed Architecture and Roadmap*, April 2002. White Paper.

[46] Peter Ilberg and Joerg H. Lepler. Implementation and Performance Evaluation of Fibonacci Heaps. CS 6155 Project Report, Georgia Institute of Technology, Atlanta, GA, December 1996.

[47] Internet Movie Database Inc., http://www.imdb.com/. *Internet Movie Data Base.*

[48] Kaffe.org, http://www.kaffe.org. *Kaffe, a Java Virtual Machine Implementation.*

[49] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the Twelfth International World Wide Web Conference (WWW)*, Budapest, Hungary, 2003.

[50] M. Karaul, Y. A. Korilis, and A. Orda. A Market-Based Architecture for Management of Geographically Dispersed, Replicated Web Servers. In *Proceedings of the ACM International Conference on Information and Computation Economies (ICE-98)*, October 1998.

[51] Frank P. Kelly. On tariffs, policing and admission control for multiservice networks. *Operation Resarch Letters*, 15(1), 1994.

[52] Jeffrey O. Kephart and Amy Greenwald. Shopbot economics. In *Proceedings of the Fifth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, University College London, London, UK, July 1999.

[53] Georg Kirchsteiger, Muriel Niederle, and Jan Potters. Public versus private exchanges. In *Tilburg, CentER Discussion paper*, July 2001.

[54] Paul D. Klemperer and Margaret A. Meyer. Supply function equilibria in oligopoliy under uncertainty. *Econometrica*, 57(6):1243–1277, November 1989.

[55] M. Kumar and S. I. Feldman. Internet Auctions. In *Proceedings of the 3rd USENIX Workshop on Electronic Commerce*, 1998.

[56] Othmar Kyas. *ATM Networks.* International Thomson Publishing, May 1995.

[57] Spyros Lalis, Christos Nikolaou, Dimitris Papadakis, and Manolis Marazakis.

Market-driven Service Allocation in a QoS-capable Environment. In *Proceedings of the ACM International Conference on Information and Computation Economies (ICE-98)*, October 1998.

[58] Averill M. Law and W. David Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, Boston, MA, third edition, 2000.

[59] Joerg H. Lepler and Karsten Neuhoff. Resource pricing under a market-based reservation protocol. In *Proceedings of the Second International Workshop on Internet Charging and QoS Technologies (ICQT)*, volume 2511 of *Springer LNCS*, pages 303–314, Zurich, Switzerland, October 2002.

[60] Joerg H. Lepler and Karsten Neuhoff. Resource reservation with a market-based protocol: what prices to expect? *Computer Communications*, 26(13):1434–1444, August 2003.

[61] M. C. Little. JavaSim User's Guide. Documentation, University of Newcastle Upon Tyne, 1999.

[62] M. Matsumoto and T. Nishimura. Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, January 1998.

[63] Microsoft, Inc., http://www.microsoft.com/net. *.Net*.

[64] Ken Moody and Martin Richards. A coroutine mechanism for BCPL. *Software Practice and Experience*, 10:765–771, February 1980.

[65] MSDN Library, Microsoft Corporation. *Platform SDK: DLLs, Processes, and Threads: Fibers*, 2004.

[66] Lik Mui, Ari Halberstadt, and Mojdeh Mohtashemi. Notions of reputation in multi-agents systems: A review. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Bologna, Italy, July 2002.

[67] Tracy Mullen and Michael P. Wellman. The Auction Manager: Market Middleware for Large-Scale Electronic Commerce. In *Proceedings of the Third USENIX Workshop on Electronic Commerce*, Boston, MA, USA, September 1998.

[68] Myricom Inc. *Overview of Myrinet*, 2004. White Paper.

[69] Y. Nishibe, K. Kuwabara, T. Suda, and T. Ishida. Distributed Channel Allocation in ATM Networks. In *Proceedings of GLOBECOM '93*, December 1993.

[70] Object Management Group, http://www.corba.org/. *Common Object Request Broker Architecture*.

[71] Nagao Ogino. Connection Establishment Protocol Based on Mutual Selection by Users and Network Providers. In *Proceedings of the ACM International Conference on Information and Computation Economies (ICE-98)*, October 1998.

[72] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital

Library Technologies Project, 1998.

[73] David C. Parkes. Optimal Auction Design for Agents with Hard Valuation Problems. In *Proceedings of the IJCAI'99 Workshop on Agent Mediated Electronic Commerce (AmEC-99)*, Stockholm, Sweden, July 1999.

[74] David M. Pennock, Eric Horvitz, and C. Lee Giles. Social choice theory and recommender systems: Analysis of the axiomatic foundations of collaborative filtering. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, 2000.

[75] Dickon Reed, Ian Pratt, Paul Menage, Stephen Early, and Neil Stratford. Xenoservers: Accountable Execution of Untrusted Programs. In *Proceedings of the Seventh Workshop on Hot Topics in Operating Systems (HotOS-VII)*, March 1999.

[76] Ori Regev and Noam Nisan. The POPCORN Market - an Online Market for Computational Resources. In *Proceedings of the ACM International Conference on Information and Computation Economies (ICE-98)*, October 1998.

[77] D. Reininger, D. Raychaudhuri, and M. Ott. Market Based Bandwidth Allocation Policies for QoS Control in Broadband Networks. In *Proceedings of the ACM International Conference on Information and Computation Economies (ICE-98)*, October 1998.

[78] Paul Resnick. Trust Among Stangers in Internet Transactions: Empirical Analysis of eBay's Reputation System. In *NBER Workshop*, 2001.

[79] Paul Resnick, Richard Zeckhauser, Eric Friedman, and Ko Kuwabara. Reputation systems. *Communications of the ACM*, 43(12), 2000.

[80] Jakka Sairamesh, Donald F. Ferguson, and Yechiam Yemini. An approach to pricing, optimal allocation and quality of service provisioning in high-speed packet networks. In *Proceedings of the Fourteenth Annual Joint Conference of the IEEE Computer and Communication Societies*, volume 3, 1995.

[81] Tuomas Sandholm. Issues in Computational Vickrey Auctions. International Journal of Electronic Commerce. *Special issue on Intelligent Agents for Electronic Commerce*, 4, 2000.

[82] Amartya Sen. *Handbook of Mathematical Economics*, volume 3, chapter Social Choice Theory. Elsevier Science Publishers, 1986.

[83] Andrei Serjantov and Ross Anderson. On dealing with adversaries fairly. In *Workshop on the Economics of Information Security*, May 2004.

[84] Brian Ninham Shand. Trust for resource control: Self-enforcing automatic rational contracts between computers. PhD Dissertation, Cambridge University, August 2004.

[85] Ajai Shankar. Implementing Coroutines for .NET by Wrapping the Unmanaged Fiber API. *MSDN Magazine*, September 2003.

[86] Jeffrey S. Steinman. Discrete-Event Simulation and the Event Horizon Part 2: Event List Management. *ACM SIGSIM Simulation Digest*, 26(1):170–178, July 1996.

[87] Sun, Inc., http://java.sun.com. *Java Virtual Machine*.

[88] Sun Microsystems Inc. *Performance Documentation for the Java HotSpot VM: Threading*, 2004.

[89] Systran Corporation. *SCRAMNet+ Shared Memory – Speed, Determinism, Reliability, and Flexibility for Distributed Real-Time Systems*, 2001. White Paper.

[90] TINA Consortium, http://www.tina.com. *Telecommunication Intelligent Network Architecture (TINA-C)*.

[91] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maguire, T. Sandholm, D. Snelling, and P. Vanderbilt. *Open Grid Services Infrastructure*. Global Grid Forum, http://www.gridforum.org, 2003.

[92] Hal R. Varian. *Intermediate Microeconomics: A Modern Approach*. W. W. Norton & Company, New York, NY, Sixth edition, 2003.

[93] VMware, Inc., http://www.vmware.com/. *VMware Server*.

[94] Carl A. Waldspurger, Tad Hogg, Bernardo A. Huberman, Jeffrey O. Kephart, and W. Scott Stornetta. Spawn: A Distributed Computational Economy. *IEEE Transactions on Software Engineering*, 18(2), February 1992.

[95] X. F. Wang, K. Hosanagar R Krishnan, and P. K. Khosla. Equilibrium reputation mechanism for mobile agent based electronic commerce. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Bologna, Italy, July 2002.

[96] Damon Wischik and Albert Greenberg. Admission Control for Booking Ahead Shared Resources. In *Proceedings of IEEE Infocom*, 1998.

[97] The World Wide Web Consortium, http://www.w3.org/2000/xp/Group/. *Simple Object Access Protocol*.

[98] The World Wide Web Consortium, http://www.w3.org/TR/wsdl. *Web Services Description Language*.

[99] P.R. Wurman, W.E. Walsh, and M.P. Wellman. Flexible double auctions for electronic commerce: Theory and implementation. *Decision Support Systems*, 24, 1998.

[100] Edo Zurel and Noam Nisan. An Efficient Approximate Allocation Algorithm for Combinatorial Auctions. In *Proceedings of the ACM Conference on Electronic Commerce (EC-01)*, Tampa, Florida, USA, October 2001.