École Centrale de Nantes **Université de Nantes** **École des Mines de Nantes**

ÉCOLE DOCTORALE STIM

« SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DES MATÉRIAUX »

Année 2008

N° attribu par la bibliothque

Allocation de Requêtes dans des Systèmes d'Information Distribués avec des Participants Autonomes

THÈSE DE DOCTORAT

Discipline : Informatique

Spécialité : Bases de Données

Présentée
et soutenue publiquement par

Jorge-Arnulfo QUIANÉ-RUIZ

*Le 22 Septembre 2008 l'*UFR *Sciences & Techniques, Université de Nantes,*
devant le jury ci-dessous

Président	:	Pr. Georges Gardarin	Université de Versailles St Quentin
Rapporteurs	:	Christophe Sibertin-Blanc, Pr	Université Toulouse I
		Georges Gardarin, Pr	Université de Versailles St Quentin
Examinateurs	:	Sylvie Cazalens, MC	Université de Nantes
		Philippe Lamarre, MC	Université de Nantes
		Ioana Manolescu, CR INRIA	INRIA Saclay-Île-de-France
		Patrick Valduriez, DR INRIA	INRIA Nantes

Directeur de thèse : Patrick Valduriez
Co-encadrant: Philippe Lamarre

Laboratoire: **L**ABORATOIRE D'**I**NFORMATIQUE DE **N**ANTES **A**TLANTIQUE.
UMR CNRS 6241. 2, rue de la Houssinire, BP 92 208 – 44 322 Nantes, CEDEX 3.

N° 366-371 ED-STIM

A SATISFACTION-BASED QUERY ALLOCATION FRAMEWORK FOR DISTRIBUTED INFORMATION SYSTEMS

Allocation de Requêtes dans des Systèmes d'Information Distribués avec des Participants Autonomes

Jorge-Arnulfo QUIANÉ-RUIZ

⋈

favet neptunus eunti

Université de Nantes

Jorge-Arnulfo QUIANÉ-RUIZ

A Satisfaction-Based Query Allocation Framework for Distributed Information Systems

IV+XLII+128 p.

This document was edited with `these-LINA` LaTeX2e class of the "Association of Young Researchers on Computer Science (LOGIN)" from the University of Nantes (available on : `http://login.irin.sciences.univ-nantes.fr/`). This LaTeX2e class is under the recommendations of the National Education Ministry of Undergraduate and Graduate Studies (circulaire n° 05-094 du 29 March 2005) of the University of Nantes and the Doctoral School of « Technologies de l'Information et des Matériaux(ED-STIM) ».

Print : thesis.tex – 09/02/2009 – 16:28.

Abstract

In large-scale distributed information systems, where participants (consumers and providers) are autonomous and have special interests for some queries, query allocation is a challenge. Much work in this context has focused on distributing queries among providers in a way that maximizes overall performance (typically throughput and response time). However, participants usually have certain expectations with respect to the mediator, which are not only performance-related. Such expectations mainly reflect their *interests* to allocate and perform queries, e.g. their interests towards: providers (based on reputation for example), quality of service, topics of interests, and relationships with other participants. In this context, because of participants' autonomy, *dissatisfaction* is a problem since it may lead participants to leave the mediator. Participant's *satisfaction* means that the query allocation method meets its expectations. Thus, besides balancing query load, preserving the participants' interests so that they are satisfied is also important. In this thesis, we address the query allocation problem in these environments and make the following main contributions. First, we provide a model to characterize the participants' perception of the system regarding their interests and propose measures to evaluate the quality of query allocation methods. Second, we propose a framework for query allocation, called S_bQA, that dynamically trades consumers' interests for providers' interests based on their satisfaction. Third, we propose a query allocation approach, called $\$_bQA$, that allows a query allocation method (specifically S_bQA) to scale up in terms of the numbers of mediators, participants, and hence of performed queries. Fourth, we propose a query replication method, called S_bQR, which allows supporting participants' failures when allocating queries while preserving participants' satisfaction and good system performance. Last, but not least, we analytically and experimentally validate our proposals and demonstrate that they yield high efficiency while satisfying participants.

Keywords: distributed information systems, query allocation, mediation, autonomous participants, participants' satisfaction, scale up, participants' failure

Résumé

Nous nous intéressons aux systèmes d'informations où les participants (clients et fournisseurs) sont autonomes, c.à.d. ils peuvent décider de quitter le système à n'importe quel moment, et qu'ils ont des intérêts particuliers pour certaines requêtes. Dans ces environnements, l'allocation de requêtes est un défi particulier car les attentes des participants ne sont pas seulement liées aux performances du système. Dans ce contexte, l'insatisfaction des participants est un problème car elle peut les conduire à quitter le système. Par conséquent, il est très important de répondre aux attentes des participants de sorte à ce qu'ils soient satisfaits. Dans cette thèse, nous abordons ce problème en apportant quatre contributions principales. Primo, nous fournissons un modèle pour caractériser la perception des participants par rapport au système et proposons des mesures qui permettent d'évaluer la qualité des méthodes d'allocation de requêtes. Secundo, nous proposons une méthode d'allocation de requêtes, S_bQA, qui permet d'équilibrer à la volée les intérêts tant des clients que des fournisseurs en se basant sur leur satisfaction. Tertio, nous proposons $\$_bQA$: une version économique de S_bQA qui permet de passer à l'échelle en nombre de médiateurs, de participants, et par conséquent, de requêtes traitées. Quarto, nous proposons S_bQR : une méthode de réplication de requêtes qui permet de supporter les pannes éventuelles des participants, tout en préservant leur satisfaction.

Mots-clés : systèmes d'information, allocation de requêtes, médiation, participants autonomes, satisfaction des participants, passage à l'échelle, panne des participants

ACM Classification

Categories and Subject Descriptors : H.2.4 [**Database Management**]: Systems—*Distributed databases, Query processing*; H.4.0 [**Information Systems Applications**]: General.

General Terms : Design, Information Systems, Management, Performance, Reliability.

Acknowledgements

First of all, I would like to express my gratitude to my advisors, Professors Patrick Valduriez and Philippe Lamarre, who provided invaluable guidance throughout my years at University of Nantes and that without them this thesis would not have been possible. Their persistent pursuit of perfection and their deep insights into various subjects have always been an inspiration to me. It is them who introduced me to the databases and distributed systems area. It is them who helped me to become a researcher. Most importantly, it is them who made me understand that a good researcher is one who loves what he does as research. Simply, it has been a great honor working with them.

I would also like to thank other members of my thesis committee : Professors George Gardarin and Christophe Sibertin-Blanc for their valuable and detailed comments on my thesis, which helped me to improve it ; Sylvie Cazalens and Ioana Manolescu, for their fruitful comments on my thesis. I would especially like to thank Sylvie Cazalens. I had the opportunity to work closely with her during my Ph.D. I have learnt a lot from her impeccable attitude towards research while endless discussions with her have helped shape a lot of my ideas. In fact, she is also a main contributor of the solution we present in the third chapter of this dissertation.

I am very thankful to the Mexican National Council for Science and Technology (CONACyT) for their financial support that without it the possibility of doing this thesis would not have happened.

I have had the chance of making a lot of friends over the years at University of Nantes who made my life as a graduate student quite enjoyable : Reza Akbariania, Eduardo Almeida, Guillaume Blin, Cedric Coulon, William Dedzoe, Marcos Didonet, Rabab Hayek, Manal El-Dick, Fabrice Evan, Lorraine Goeuriot, Sandra Lemp, Jorge Manjarrez, Vidal Martins, Jean-Marie Normand, Wenceslao Palma, Antoine Pigeau, Guillaume Raschia, Ricardo Soto, Siloé Souza, David Tlalolini, Mounir Tlili, Anthony Ventresque, Juliana Vermelho. I want to thank all of them for being as they are.

I am blessed to have a truly loving wife's family who live in the beautiful "Pay Basque" region of the south of France. Since I landed in France they immediately took me as one of them. They always helped us, my wife and me, for several administrative procedures, finding a flat, and much more more things that we were confronted in France. Most important, I would like to thank them for all the beautiful moments we had together. In particular, I want to thank "Amatxie", my wife's grand-mother, a lovely person who watch and pray for us from heaven.

I especially want to thank all my family in Mexico. My parents knew instilled in me that the most precious treasure than a person can have is the knowledge, which helped me to set up a life-long goal in pursuing knowledge. It is their love and belief in me that gave me the force to overcome every seemingly insurmountable difficulty, and continue fighting for reaching my next dream. I owe them my life. I want to thank my sister and brother for being with me always I need them. I want to give my special thanks to my parents-in-law for taking me as their new child and their support.

Finally, I want to thank my wife, Mariana Quiané, and my son, Killian Quiané-Haro. I thank my wife for her help, comprehension, and support during these past few years. She was always there to encourage me when I needed it most. I thank my son whose birth motivated much more myself to pursue my dreams. To them, I dedicate this work.

Table of Contents

Table of Contents VII

List of Tables XI

List of Figures XIII

List of Examples XV

Extended Abstract in French XVII

— Body of the Dissertation —

Introduction 1

1 Participants Characterization and Measures 7
1.1 Problem Statement . . . 8
1.2 A Usual Characterization of Providers . . . 8
1.3 Satisfaction Model . . . 9
1.3.1 Participants' Adequation . . . 10
1.3.2 Participants' Satisfaction . . . 12
1.3.3 Provider Intention-based Profit . . . 14
1.3.4 Query Allocation Method Efficiency . . . 15
1.3.5 Discussion . . . 16
1.4 System Measures . . . 17
1.5 Related Work . . . 18
1.6 Chapter Summary . . . 19

2 Satisfaction-based Query Allocation 21
2.1 Problem Definition . . . 22
2.2 Consumer's Side . . . 23
2.3 Provider's Side . . . 23
2.4 Mediator's Side . . . 24
2.4.1 Scoring and Ranking Providers . . . 25
2.4.2 Regulating the System . . . 26
2.4.3 Query Allocation Principle . . . 27
2.4.4 Communication Cost . . . 27
2.5 Discussion . . . 29
2.6 Experimental Validation . . . 30
2.6.1 Setup . . . 30
2.6.2 Baseline Methods . . . 31
2.6.3 Results . . . 32

2.7 Related Work . . . 40
2.7.1 Data Mediator Systems . . . 41
2.7.2 Multi-Agents . . . 41
2.7.3 Web Services . . . 42
2.7.4 Load Balancing Approaches . . . 44
2.7.5 Economic Approaches . . . 45
2.8 Chapter Summary . . . 46

3 Scaling Up Query Allocation 49
3.1 Problem Statement . . . 50
3.2 Use of Virtual Money . . . 51
3.2.1 Flow of Virtual Money . . . 53
3.3 Provider's Side . . . 54
3.3.1 Computing Bids . . . 54
3.3.2 Bidding in the Presence of Several Mediators . . . 55
3.4 Mediator's Side . . . 56
3.4.1 Computing Providers' Level . . . 56
3.4.2 Invoicing Providers . . . 57
3.4.3 Communication Cost . . . 59
3.5 Cost of Federating Mediators . . . 59
3.6 Experimental Validation . . . 60
3.6.1 Setup . . . 60
3.6.2 Results . . . 61
3.7 Related Work . . . 65
3.7.1 Peer-to-Peer Networks . . . 65
3.7.2 Grid-based Networks . . . 67
3.7.3 Multi-Agent Networks . . . 68
3.7.4 Small-World Networks . . . 69
3.7.5 Summary . . . 70
3.8 Chapter Summary . . . 71

4 Dealing with Participants' Failures 73
4.1 Problem Definition . . . 74
4.2 Satisfaction Model for Faulty Participants . . . 75
4.2.1 Consumer Satisfaction . . . 75
4.2.2 Provider Satisfaction . . . 76
4.2.3 Global Satisfaction . . . 77
4.3 Non Systematic Query Replication Based on Satisfaction . . . 81
4.4 Experimental Validation . . . 84
4.4.1 Setup . . . 84
4.4.2 Results . . . 85
4.5 Related Work . . . 88
4.5.1 Query Replication . . . 89
4.5.2 Rollback-Recovery Protocols . . . 92
4.5.3 Concluding Remark . . . 94
4.6 Chapter Summary . . . 95

Conclusion **97**

— Appendixes —

A The S$_b$QA Prototype **103**
A.1 S_bQA's demo : A BOINC example 103
A.2 S_bQA within Grid4All 105
A.2.1 Grid4All Example Application 106
A.2.2 Selection Service Specification 106

Notations **111**

Bibliography **113**

List of Tables

1 Fournisseur ayant les capacités de traiter la requête d'Emma. XVIII
2 Paramètres des simulations. XXXVI
3 Reasons of the provider's departures for a workload of 80% of the total system capacity. XL
4 Reasons of the provider's departures for a workload of 80% of the total system capacity. XL
5 Reasons of the provider's departures for a workload of 80% of the total system capacity. XLI

— Body of the Dissertation —

6 Providers for $eWine$'s query. 4

2.1 Simulation parameters. 31
2.2 Reasons of the provider's departures for a workload of 80% of the total system capacity. . . 37

3.1 Virtual money balance along a sequence of mediations. 54
3.2 $\$_b QA$ in a (a) competition and an (b) imposition case, with $\omega = 0.5$ and $n = 2$. 57

4.1 Simulation parameters. 84

— Appendixes —

List of Figures

1 Valeurs possibles de ω en fonction des satisfactions. XXVI
2 Number of results vs query's criticity when a consumer requires five results. XXVIII
3 Satisfaction des fournisseurs avec une charge croissant de 30% à 100% de la capacité du système ; agents non autorisés à quitter le système. XXXVII
4 Satisfaction des clients avec une charge croissant de 30% à 100% de la capacité du système ; agents non autorisés à quitter le système. XXXVII
5 Utilisation des fournisseurs avec une charge croissant de 30% à 100% de la capacité du système ; agents non autorisés à quitter le système. XXXVIII
6 Départs des fournisseurs avec une charge exprimée en fonction des capacités initiales du système ; agents autorisés à quitter le système. XXXVIII
7 Départs des clients avec une charge exprimée en fonction des capacités initiales du système ; agents autorisés à quitter le système. XXXIX
8 Temps de réponse avec une charge exprimée en fonction des capacités initiales du système ; agents autorisés à quitter le système. XXXIX

— *Body of the Dissertation* —

9 Overview of Query Allocation in Distributed Systems with Autonomous Participants. 2

2.1 Tradeoff between *preference* and *utilization* for getting providers' *intention*. 24
2.2 The values that ω can take. 26
2.3 SQLB system architecture. 28
2.4 Participants' satisfaction results for a workload range from 30 to 100% of the total system capacity when participants are captive. 33
2.5 (a) and (c) : query load balancing results for a workload range from 30 to 100% of the total system capacity when participants are captive, (d) and (e) : allocation efficiency results for different workloads, and (f) : ensured response times. All these results are with captive participants. 34
2.6 Impact on performance of providers' departure. 35
2.7 Participants' departures. 36
2.8 Quality results for a workload range from 30 to 100% of the total system capacity when participants are captive and for three kinds of providers : *(i)* when they are interested only in their preferences (the *preference-based case*), *(ii)* when they are just interested in their utilization (the *utilization-based case*), and *(iii)* when their utilization is as important as their preferences (the *normal case*). 38
2.9 Quality results for a *workload* range from 30 to 100% of the total system capacity when participants are *captive* and : (a)-(c) providers compute their intentions based on their preferences and utilization (the *normal case*) and (d)-(f) providers compute their intentions based on their preferences (the *preference-based case*). 39
2.10 Performance results with captive participants. 39
2.11 Cycle of a Web service invocation. 43

3.1 A x-redundant VO with 3 mediators. 51
3.2 General system architecture. Site m denotes the mediator, c a consumer, and p a provider. . . 52
3.3 Quality results in mono-mediator VOs for different workloads and with captive participants. 62
3.4 Impact on performance of providers' departure. 62
3.5 Quality results with captive participants for different workloads in a x-redundant VO with 8 mediators. 63
3.6 Performance results (a) and (b) for different workloads in a x-redundant VO with 8 mediators and captive participants, and ; (c) for a workload of 60% of the total system capacity with 50 consumers and 100 providers (set1), 100 consumers and 200 providers (set2), 200 consumers and 400 providers (set3) and 400 consumers and 800 providers (set4). 64
3.7 Grid Organization. 68
3.8 Acquaintance topology forms. 69

4.1 Number of results vs query's criticity when a consumer requires five results. 76
4.2 Results with faulty participants and different workloads. 86
4.3 Results with different query criticity values and different workloads. 87
4.4 Results with a high-probability of failure and different workloads. 88
4.5 Passive redundancy model. 90
4.6 Active redundancy model. 91
4.7 Recovery line, rollback propagation, and domino effect. 93
4.8 Logging for deterministic replay. 94

— *Appendixes* —

A.1 Some S_bQA GUIs. 104

List of Examples

— Body of the Dissertation —

1 Participants' Preferences 1

— Appendixes —

Extended Abstract in French

Les systèmes d'information distribués font souvent l'hypothèse que leurs participants[1] sont autonomes, c'est-à-dire qu'ils sont libres de rejoindre ou de quitter le "système"[2] à n'importe quel moment et sans avoir à en référer à qui que ce soit. La motivation des participants à intégrer le système peut être liée à l'espoir que ce système peut répondre à leurs attentes et leur permettre d'atteindre leurs objectifs. Le départ quand à lui est souvent consécutif à la déception. Pour le bon fonctionnement de ces systèmes, il est donc primordial que la répartition des tâches soit attentive aux attentes des participants, tant des clients que des fournisseurs, de sorte à ce qu'ils soient *satisfaits*, autant que faire ce peut.

De nombreux travaux ont été menés dans le contexte de l'allocation de tâches : de la recherche des fournisseurs pouvant réaliser une tâche [LH04, NBN99], à l'allocation d'une tâche de sorte à maximiser ou minimiser certains critères comme la répartition de charge et le temps de réponse [ABKU99, GBGM04, MTS90, RM95, SKS92]. Cependant, les attentes des participants se joignant au système ne sont pas nécessairement restreintes aux performances. Les clients peuvent manifester un certain intérêt concernant la qualité des résultats (si tous les fournisseurs ne donnent pas les mêmes réponses). Parmi les tâches qu'ils peuvent traiter, les fournisseurs peuvent avoir des préférences pour certaines. Tous peuvent avoir des préférences concernant les agents avec lesquels ils traitent. L'*intention* qu'a un fournisseur de traiter une tâche est donc le résultat de la combinaison de plusieurs critères qui lui sont propres.

De même, l'*intention* d'un client à voir sa tâche traitée par tel ou tel fournisseur est aussi le résultat de considérations personnelles. Cette approche soulève un problème. Si aucun fournisseur ne manifeste d'intérêt pour une tâche donnée, elle risque de ne pas être traitée. C'est de fait ce qui se produit dans différents systèmes basés sur des techniques de micro-économie [FNSY96, FYN88, SAL$^+$96]. Dans un tel cas, le traitement de la tâche doit être imposé à un ou plusieurs fournisseurs, ce qui les mécontentera (sauf à les dédommager). D'un autre coté, si la tâche n'est pas traitée, c'est le client qui sera mécontent.

Les problèmes inhérents à l'allocation de tâches sont donc de nature differente. D'abord, les attentes des participants peuvent-être contradictoires. Ensuite, le fait que nous considérons que les tâches doivent être traitées par le système, même si les fournisseurs ne souhaitent pas les traiter pour des raisons qui leurs sont propres, peut introduire du mécontentement. Enfin, trop de mécontentement conduit les participants à quitter le système ce qui peut avoir des conséquences sur les fonctionnalités offertes par le système. Le départ de fournisseurs peut conduire à perdre des fonctionnalités, et le départ de clients est une perte de source de travail pour les fournisseurs.

A notre connaissance, ce problème n'a pas été adressé dans son ensemble. Les mécanismes de médiation qui effectuent l'allocation de tâches ne tiennent compte ni des *intentions* des participants, ni de leur *satisfaction*. Les contributions majeures de cette thèse sont donc :

- La proposition d'un mécanisme de médiation (S_bQA) qui s'adapte immédiatement et automatiquement aux attentes des participants. Ce mécanisme utilise les *intentions* exprimées par les participants pour définir leur *satisfaction*. Il utilise ensuite ces deux notions pour allouer les tâches. Pour

[1] Tout au long de ce document, le terme participant fait référence à la fois aux fournisseurs et aux clients.

[2] Le "système" peut désigner soit le système d'information distribué, soit plus localement un médiateur [Mil02, RS97].

Table 1 – Fournisseur ayant les capacités de traiter la requête d'Emma.

Fournisseurs	**Charge**	**Intention**	**Cons. Int.**
Mark	15%	Oui	Non
Robert	43%	Non	Oui
Johnson	78%	Oui	Non
William	85%	Non	Oui
Mary	100%	Oui	Oui

atteindre une certaine équité, S_bQA pondère les intentions des différents participants en fonction de leurs satisfactions respectives.

- Une analyse des techniques d'allocation du point de vue de la satisfaction.
- Une validation expérimentale comparant S_bQA à d'autres techniques existantes ($Capacity\ based$ et *Mariposa-like*) qui montre la supériorité de notre approche.

Pour illustrer le problème des systèmes d'information distribués avec des participants autonomes, considérons par exemple un système incluant des centaines de scientifiques (biologistes, docteurs en médecine, généticiens...) travaillant sur le génôme humain. Ils sont répartis sur la planête et ils partagent leurs informations. Chaque site, qui représente un scientifique, déclare ses capacités au système et gère localement ses préférences et intentions.

Considérons un scénario simple. Emma (Dr. en médecine) vient de découvrir un gène responsable d'une maladie de la peau. Elle interroge le système pour trouver des liens éventuels avec d'autres maladies. Pour une vue plus générale, elle souhaite avoir des réponses de plusieurs collègues, disons 2 pour simplifier l'exemple.

Dans un premier temps, le système doit identifier les fournisseurs capables de traiter la requête. Un algorithme de *matchmaking* [SKWL99] permet de résoudre ce premier problème. Supposons que pour cet exemple, il y en ait 5. La seconde étape consiste à obtenir les intentions de ces fournisseurs par rapport à cette requête (supposées binaires dans cet exemple). Le tableau 6 regroupe les différentes données de cet exemple.

Mary est la plus chargée (elle n'a plus de ressource disponible). Robert et William ne désirent pas traiter cette requête pour des raisons qui leurs sont propres. D'un autre coté, pour des raisons de confiance envers leurs résultats, Emma ne souhaite pas que Mark ou Johnson traitent sa requête.

Quoi qu'il en soit, à la demande d'Emma, le système doit choisir deux fournisseurs pour leur allouer la requête. Mark et Robert sont les moins chargés. C'est donc à eux que les méthode basées sur la répartition de charge alloueraient la requête. Cela aurait pour conséquence de mécontenter Robert et Emma. Répétées, de telles décisions pourraient conduire ces participants à quitter le système. Ici la seule réponse correcte du point de vue des intentions est Mary. Malheureusement, cette allocation n'est pas satisfaisante du point de vue de la répartition de charge. De plus, Emma a demandé à ce que la requête soit envoyée à deux scientifiques. C'est donc un cas qui génèrera du mécontentement d'un coté ou de l'autre.

Plusieurs questions restent donc ouvertes : *Que doit faire le système dans ce cas ? Doit-il privilégier les intentions du client (ici Emma) ? les intentions des fournisseurs ? Doit-il prendre en compte la charge des fournisseurs ?* Dans cette thèse nous répondons à ces questions, mais aussi nous proposons un modèle qui permet d'analyser le comportement d'un système de ce type. Les notions présentées peuvent aussi

servir à une méthode de médiation dans ses prises de décisions.

Travaux Précédents

Dans le contexte des systèmes d'information distribués à grande échelle, de nombreuses approches se sont concentrées sur le problème de l'allocation de tâches avec comme objectif les performances du système, sans aucune considération des intentions des participants, sauf à considérer qu'ils sont eux-mêmes exclusivement intéressés par les performances. Par exemple, les propositions [RM95, SKS92] allouent chaque tâche entrante aux fournisseurs qui sont les moins chargés parmi ceux qui peuvent traiter la tâche.

Les techniques de médiation utilisant une approche économique peuvent prétendre prendre en compte les intentions. Mariposa [SAL+96] est l'un des premiers systèmes utilisant des techniques de microéconomie pour la gestion des informations dans un système réparti. Cette proposition est basée sur un système de vente aux enchères. Pour schématiser, les clients payent les fournisseurs pour qu'ils traitent leurs tâches. Le prix établi par les fournisseurs est calculé en fonction de leurs préférences et de leur charge de travail, ceci afin de garantir un certain équilibre de la charge de travail au sein du système. Une fois les offres publiées, un broker sélectionne les fournisseurs ayant les offres les plus basses. En revanche, nos expérimentations montrent que Mariposa ne garantit pas un bon équilibre de charge de tâches. En outre, si aucun fournisseur ne souhaite traiter la tâche, elle ne le sera tout simplement pas.

La médiation, dite *médiation flexible*, proposée dans [LCLV07] est aussi basée sur des aspects économiques. Là encore, les fournisseurs font des offres pour obtenir des tâches. Ces offres sont alors équilibrées par leur qualité estimée (ou réputation). Le prix que doit payer un fournisseur pour obtenir la tâche dépend de sa réputation. Contrairement à Mariposa, lorsqu'une tâche n'intéresse personne, cela conduit à la "réquisition" de fournisseurs auxquels on impose de la traiter. Un mécanisme de compensation financière est alors mis en œuvre. Cette compensation augmente les possibilités de ces fournisseurs à faire valoir leurs choix dans les prochains tours. Dans cette approche, ce sont des mécanismes économiques qui sont utilisés pour réguler le système, la satisfaction des participants étant supposée être une propriété induite.

Dans [QRLV06], nous avons proposé une technique de médiation basée sur la satisfaction des fournisseurs, mais ni la satisfaction des clients, ni leur intentions ne sont prise en compte. Dans [QRLV07a], nous avons proposé une stratégie pour prendre en compte ces différentes notions, mais aucune méthode pour faire la fusion des *intentions* des clients et fournisseurs n'est proposée. En outre, les stratégies que nous proposons dans [QRLV07a] peuvent être utilisées pour améliorer les résultats des travaux précedents.

Notions Préliminaires

Le système que nous considérons est constitué d'un ensemble de fournisseurs P, d'un ensemble de clients C et d'un ensemble de médiateurs M. Ces ensembles ne sont pas nécessairement disjoints, un même agent pouvant jouer plusieurs rôles. Les fournisseurs peuvent être hétérogènes en termes de capacités, ne disposant pas tous des mêmes ressources, mais aussi en termes de données. Ce dernier point signifie qu'ils peuvent donner des résultats différents les uns des autres pour la même tâche. Les tâches sont abstraites par un triplet $q =< c, d, n >$ tel que $q.c \in C$ est l'identifiant du client ayant émis la tâche, $q.d$ est la description de la tâche, et le paramètre $q.n$ représente le nombre de fournisseurs que le client veut voir traiter la tâche. Considérons par exemple une application de commerce électronique. Dans ce

cas, la tâche correspond à un appel à propositions. Donc, lorsqu'un client fait un appel à propositions, il peut souhaiter limiter le nombre de réponses à n. De même, les fournisseurs ne souhaitent généralement pas répondre à tous les appels d'offres.

La tâche peut être exprimée de manière textuelle, logique, dans un langage spécifique tel que SQL, XQuery, *etc.* Le problème de "matchmaking", consistant à identifier les fournisseurs pouvant travailler avec cette requête est en dehors du champ de cet article (voir [LH04, NBN99] pour plus d'informations). Nous nous contentons de supposer que nous disposons d'un processus permettant d'identifier les fournisseurs adéquats de manière idéale, i.e. sans faux positif ni faux négatif.

Les clients confient leurs tâches à un médiateur $m \in M$ dont le rôle est d'allouer chaque tâche q à $q.n$ fournisseurs. P_q représente l'ensemble des fournisseurs associés au médiateur m (n'apparaissant pas dans la notation pour éviter de l'alourdir) pouvant traiter la tâche q. Cet ensemble de fournisseurs est obtenu via un processus de "matchmaking" supposé correct.

L'allocation d'une tâche q est formalisée par un vecteur $All\overrightarrow{oc}_q$ de longueur N tel que :

$$\forall p \in P_q,\ All\overrightarrow{oc}[p] = \begin{cases} 1 & \text{si la tâche est allouée à } p \\ 0 & \text{sinon} \end{cases}$$

Dans le cas où le nombre de fournisseurs pouvant traiter la tâche est insuffisant par rapport au nombre de fournisseurs demandés par le client, ils doivent tous la traiter. Ceci impose donc que $\sum_{p \in P_q} All\overrightarrow{oc}_q[p] = min(q.n,\ N)$ où $N = ||P_q||$.

Modèle de Satisfaction

Notre attention s'est portée sur deux caractériques des participants qui permettent de comprendre comment ils peuvent percevoir le système dans lequel ils interagissent.

La première de ces caractéristiques est l'*adéquation*. En fait, deux adéquations doivent être considérées. a) *adéquation du système par rapport à un participant* e.g. un système dans lequel un fournisseur ne peut trouver aucune requête correspondant à ses attentes n'est pas adéquat pour ce fournisseur ; b) *adéquation d'un participant au système* e.g. un client qui émet des requêtes qui n'intéressent aucun fournisseur n'est pas adéquat par rapport au système. A travers ces notions il est possible d'évaluer si un participant a une chance d'atteindre ses objectifs dans un système. À moins d'avoir une connaissance globale du système, un participant ne peut déterminer lui même ce que les autres pensent de lui. Aussi, nous considérons l'adéquation d'un participant au système comme une caractéristique globale (cf. Section).

La seconde caractéristique est la *satisfaction*. Comme pour l'adéquation, deux sortes de satisfaction peuvent être consédérées : a) la *satisfaction d'un participant vis-à-vis du système* e.g. un client qui reçoit des résultats de fournisseurs qu'il ne souhaitait pas solliciter n'est pas satisfait ; et b) la *satisfaction d'un participant par rapport au système de médiation* e.g. un fournisseur devant traiter des requêtes qu'ils ne désirait pas met en cause le système de médiation lorsqu'il constate qu'il existe des requêtes lui convenant mieux, mais ne lui étant pas allouées. Ces deux notions de satisfaction peuvent avoir un impact important sur le système dans la mesure où elles peuvent fonder une décision de départ d'un participant.

Nous supposons que les participants ont une mémoire limitée et qu'ils ne mémorisent donc que leurs k dernières interactions avec le système[3]. Nous allons donc définir les différentes notions présentées ci-dessus par rapport à la mémoire des participants. Deux remarques supplémentaires. Il est évident que

[3]Notons que k peut être différent d'un participant à l'autre. Cependant, dans un souci de simplification, nous supposerons ici que ce paramètre est identique pour tous les participants.

ces notions évoluent au cours du temps, mais pour éviter d'alourdir les notations, le temps n'apparaîtra pas. Enfin, ces notions peuvent être définies soit à partir des préférences des participants, soit à partir de leurs intentions. Si les définitions formelles sont similaires, les valeurs obtenues présentent quelques différences. Pour des raisons de place, nous ne pouvons en présenter ici qu'une seule version. Dans la mesure où les préférences sont souvent considérées comme des données privées, ce sont les intentions affichées auprès des médiateurs qui serviront de base à nos définitions.

Caractérisation locale d'un client

Un client est caractérisé à partir des informations qu'il peut obtenir du système. Intuitivement, les caractéristiques présentées ci-après sont utiles pour répondre à des questions de la forme "Dans quelle mesure mes intentions correspondent à celles des fournisseurs pouvant traiter mes requêtes ?" – *adéquation d'un client par rapport au système* – "Dans quelle mesure les fournisseurs ayant traité mes dernières requêtes me satisfont ?" – *Satisfaction d'un client* – "La méthode d'allocation des requêtes me satisfait-elle ?" – *Satisfaction d'un client par rapport à l'allocation* –. Ces notions seront basées sur la mémoire d'un client qui sera notée IQ_c^k.

Adéquation

L'*adéquation du système pour un client* caractérise la vision du système qu'a le client. Dans le scénario présenté dans l'introduction, le système est relativement adéquat pour Emma car bon nombre des fournisseurs lui conviennent. Plus formellement, l'adéquation du système par rapport au client c et pour une requête q, notée $\delta_s(c, q)$, est définie comme étant la moyenne des intentions de c par rapport à l'ensemble des fournisseurs pouvant traiter q (P_q). La valeur de cette notion est volontairement amenée dans l'intervalle $[0..1]$.

$$\delta_a(c, q) = \Big((\frac{1}{||P_q||} \sum_{p \in P_q} \overrightarrow{CI}_c q[p] \ ) + 1 \Big) \Big/ 2 \tag{1}$$

L'*adéquation du système par rapport à un client* c, est alors définie comme la moyenne des adéquations pour les k dernières requêtes.

Definition 1. *Adéquation du système par rapport à un client.*

$$\delta_a(c, q) = \frac{1}{||IQ_c^k||} \sum_{q \in IQ_c^k} \delta_a(c, q)$$

Plus la valeur est proche de 1, plus le client considère le système comme adéquat.

Satisfaction

La satisfaction d'un client c concernant le traitement d'une de ses requêtes q, notée $satFunction(c, q)$ est liée aux fournisseurs auxquels sa requête a été allouée ($\widehat{P_q}$). La moyenne semble une technique intuitive. Cependant, elle ne permet pas de prendre en compte le souhait d'un client d'avoir plusieurs résultats de fournisseurs différents. Par exemple, dans le scénario de l'introduction, Emma a demandé 2

fournisseurs. Si le système ne lui en alloue qu'un seul la satisfaction d'Emma ne peut être totale, même si ce fournisseur est parfait. L'équation suivante tient compte de ce point.

$$\delta_s(c,q) = \Big((\frac{1}{n} \sum_{p \in \widehat{P_q}} \overrightarrow{CI}_c q[p]\ ) + 1 \Big) \Big/ 2 \tag{2}$$

où n abbrège $q.n$. Les valeurs de $\delta_s(c,q)$ sont dans l'intervalle $[0..1]$.

La *satisfaction* d'un client c est alors obtenue en faisant la moyenne des satisfactions par rapport aux k dernières requêtes traitées.

Definition 2. *Satisfaction d'un client*

$$\delta_s(c) = \frac{1}{||IQ_c^k||} \sum_{q \in IQ_c^k} \delta_s(c,q)$$

Cette notion de satisfaction ne tient aucun compte du contexte. Elle ne permet donc pas au client d'évaluer les efforts consentis par le système d'allocation pour le satisfaire. Par exemple, en reprenant le scénario de l'introduction, suppose qu'Emma a une intention de 1 (resp. 0.9, 0.7) pour que la requête soit allouée à Robert (resp. William et Mary). Allouer la requête à William est dans l'absolu satisfaisant. Cependant, il existe un autre fournisseur dans le système qui serait encore plus satisfaisant. La *satisfaction d'un fournisseur par rapport au système d'allocation*, notée $\delta_{as}(c)$ (définition 18) permet de rendre compte des efforts effectués en ce sens par la méthode d'allocation. Cette satisfaction prend ses valeurs dans l'intervalle $[0..\infty]$.

Definition 3. *Satisfaction d'un client par rapport à la méthode d'allocation*

$$\delta_{as}(c) = \frac{1}{||IQ_c^k||} \sum_{q \in IQ_c^k} \frac{\delta_s(c,q)}{\delta_a(c,q)}$$

Si la valeur ainsi obtenue est supérieure à 1, le client peut en conclure que la méthode d'allocation agit en sa faveur. Par contre, si cette valeur est proche de 0 la méthode défavorise le client.

Caractérisation locale d'un fournisseur

Cette section est consacrée à la caractérisation d'un fournisseur. Intuitivement, nous cherchons à répondre à des questions de la forme : "dans quelle mesure les requêtes émises sur le système correspondent aux intentions du fournisseur ?" – *Adéquation du système* –; "dans quelle mesure les dernières requêtes que le fournisseur a eu à traiter lui conviennent ?" – *Satisfaction du fournisseur* –; "la méthode d'allocation est-elle statisfaisante ?" – *Satisfaction du fournisseur par rapport à la méthode d'allocation* –. Ces caractéristiques seront définies par rapport aux intentions exprimées par le fournisseurs sur les k dernières requêtes qu'il est capable de mémoriser ($\overrightarrow{PI}_p k$).

Adéquation

L'*adéquation du système par rapport à un fournisseur* aide ce fournisseur à déterminer si le système dans lequel il évolue correspond à ses attentes. Par exemple, dans notre scénario, on peut considérer que le système est adéquat par rapport à Marc dans la mesure où la seule requête émise par Emma correspond à ses intentions. Cependant, il est difficile de conclure en ne considérant qu'une seule requête. Une moyenne est plus informative.

Definition 4. *Adéquation du système par rapport à un fournisseur*

$$\delta_a(p) = \begin{cases} \Big((\frac{1}{||PQ_p^k||} \sum_{q \in PQ_p^k} \overrightarrow{PI}_p k[q]\ ) + 1 \Big) \Big/ 2 \\ 0 \qquad\qquad\qquad\qquad\qquad si\ PQ_p^k = \emptyset \end{cases}$$

Les valeurs que peut prendre cette adéquation sont dans l'intervalle $[0..1]$. Plus la valeur est proche de 1, plus le système est adéquat par rapport au fournisseur concerné.

Satisfaction

Contrairement à l'adéquation, la satisfaction d'un fournisseur ne dépend que des requêtes qu'il a eu à traiter. En revenant encore une fois au scénario de l'introduction, et en supposant que le système alloue la requête d'Emma à Robert, Robert ne sera pas satisfait car il ne souhaite pas la traiter. La *satisfaction* d'un fournisseur, $\delta_s(p)$, est donc définie comme étant la moyenne des satisfactions obtenues sur les requêtes traitées par le fournisseur (SQ_p^k) parmi les k dernières requêtes (PQ_p^k). La valeur est ramenée sur l'intervalle $[0..1]$. Plus la valeur est proche de 1, plus le fournisseur est satisfait.

Definition 5. *Satisfaction d'un fournisseur*

$$\delta_s(p) = \begin{cases} \Big((\frac{1}{||SQ_p^k||} \sum_{q \in SQ_p^k} \overrightarrow{PI}_p k[q]\ ) + 1 \Big) \Big/ 2 \\ 0 \qquad\qquad\qquad\qquad\qquad si\ SQ_p^k = \emptyset \end{cases}$$

Avec cette définition, un fournisseur peut évaluer s'il obtient des requêtes lui permettant d'atteindre ses objectifs, ou au moins, satisfaisant ses intentions. D'un autre coté, les efforts déployés par la méthode d'allocation pour l'aider peuvent aussi l'intéresser. Nous définissons la *satisfaction d'un fournisseur par rapport à la méthode d'allocation* comme étant la ratio de sa satisfaction sur son adéquation (définition 20). Les valeurs sont dans l'intervalle $[0..\infty]$.

Definition 6. *Satisfaction d'un fournisseur par rapport à la méthode d'allocation*

$$\delta_{as}(p) = \frac{\delta_s(p)}{\delta_a(p)}$$

Plus la satisfaction d'un fournisseur par rapport à la méthode d'allocation est supérieure à 1 plus l'effort de la méthode d'allocation en faveur du fournisseur est important. A contrario, plus la valeur est proche de 0, plus la méthode est pénalisante pour le fournisseur.

Caractérisations des participants du point de vue du système

Les participants, tant les fournisseurs que les clients, sont ici caractérisés d'un point de vue global. L'objectif est de pouvoir répondre à des questions de la forme : "Dans quelle mesure les requêtes d'un client correspondent aux attentes des fournisseurs" – *Adéquation d'un client par rapport au système* – "Dans quelle mesure un fournisseur répond-il aux attentes des clients ?" – *Adéquation d'un fournisseur par rapport au système* –

L'*adéquation d'un client par rapport au système* permet d'évaluer si ce client correspond aux attentes des fournisseurs. En reprenant notre scénario, la requête d'Emma est adéquate au système car une grande

partie des fournisseurs sont prêts à la traiter. En accord avec cette intuition, l'adéquation d'une requête q d'un client c, notée $\delta_a(c,q)$, est définie comme la moyenne des intentions déclarées par les fournisseurs. Les valeurs sont ramenées dans l'intervalle $[0..1]$.

$$\delta_a(c,q) = \Big((\frac{1}{||P_q||} \sum_{p \in P_q} pi_p(q)\ ) + 1 \Big) \Big/ 2 \tag{3}$$

L'adéquation du client par rapport au système est simplement définie comme la moyenne de ces valeurs.

Definition 7. *Adéquation d'un client par rapport au système*

$$\delta_a(c) = \frac{1}{||IQ_c^k||} \sum_{q \in IQ_c^k} \delta_a(c,q)$$

L'*adéquation du fournisseur par rapport au système* permet d'évaluer si les clients sont intéressés par ce fournisseur. En revenant à notre scénario, Emma ne souhaite pas que Mark traite sa requête. Cela ne joue pas en faveur de Mark. L'adéquation d'un fournisseur par rapport au système, $\delta_a(p)$, est définie comme la moyenne des intentions montrées à son égard par les clients sur les k dernières requêtes proposées. Les valeurs sont ramenées entre $[0..1]$. Plus la valeur est proche de 1, plus le fournisseur est adéquat.

Definition 8. *Adéquation d'un fournisseur par rapport au système*

$$\delta_a(p) = \begin{cases} \Big((\frac{1}{||PQ_p^k||} \sum_{q \in PQ_p^k} \overrightarrow{CI}_c q[p]\ ) + 1 \Big) \Big/ 2 & \\ 0 & if\ PQ_p^k = \emptyset \end{cases}$$

Mécanisme de Médiation

S_bQA est un mécanisme d'allocation fondé sur la prise en compte des *intentions* et de la *satisfaction* des participants. Cela lui permet de s'adapter immédiatement et automatiquement aux changements d'*intentions* des participants. Par exemple, le système de médiation ne tiendra compte des performances (temps de réponse et répartition de charge) que si les participants en tiennent eux mêmes compte dans l'expression de leurs intentions.

Nous présentons ici S_bQA en deux phases. La première partie décrit la technique d'évaluation d'un score pour chaque fournisseur correspondant à la pertinence de lui allouer la tâche. La deuxième partie présente l'algorithme général de S_bQA.

Intentions des participants

Les *intentions* des participants sont exprimées sur l'intervalle $[-1..1]$. Une intention positive traduit le souhait que le choix devienne réalité, souhait d'autant plus important que la valeur est proche de 1. Au contraire, une intention négative traduit le souhait de ne pas voir cette possibilité se réaliser, souhait d'autant plus important que la valeur est proche de -1. Une valeur de 0 traduit une indifférence.

Il est de la responsabilité d'un participant de calculer ses propres intentions en combinant les critères qu'il juge utile de considérer (e.g. charge, préférences, temps de réponse, réputation, expériences passées,...).

La manière dont les participants calculent leurs *intentions* est considérée comme une information privée à laquelle le système ne peut accéder. Cependant, il ne faut pas s'y tromper, cela a un impact direct sur le comportement global du système. Par exemple, si les participants manifestent tous un intérêt marqué pour des temps de réponse les plus faibles possibles, la médiation tenant compte de leurs intentions devrait conduire à l'obtention d'un système performant du point de vue des temps de réponse. Tel ne sera pas le cas si les participants s'intéressent à la qualité des réponses sans aucune considération pour le temps. La médiation doit permettre d'adapter le comportement global du système aux attentes des participants.

Pertinence d'allouer une tâche â un fournisseur

Étant donné une tâche q et un fournisseur p, la pertinence d'allouer cette tâche â ce fournisseur est évaluée et quantifiée en considérant les deux points de vue en présence. Le point de vue du client c est obtenu en considérant son *intention* de voir sa tâche traitée par p et le point de vue du fournisseur p est obtenu en considérant son *intention* de traiter la tâche q de c. La confrontation de ces deux points de vue pourrait être directe, mais dans un souci d'équité, nous avons choisi de permettre qu'un point de vue soit privilégié par rapport à l'autre.

Definition 9. *Évaluation d'un fournisseur*

$$scr_q(p) = \begin{cases} \left(\overrightarrow{PI}_q[p]\right)^{\omega}\left(\overrightarrow{CI}_c q[p]\right)^{1-\omega} & \\ \qquad si\ \overrightarrow{PI}_q[p] > 0 \wedge \overrightarrow{CI}_c q[p] > 0 & \\ -\Big(\left((1-\overrightarrow{PI}_q[p])+\epsilon\right)^{\omega}\left((1-\overrightarrow{CI}_c q[p])+\epsilon\right)^{1-\omega}\Big) & \\ \qquad sinon & \end{cases}$$

À différence du vecteur $\overrightarrow{PI}_p$ qui stocke les *intentions* du fournisseur p pour taiter les dernières tâches qui lui ont été proposées, le vecteur $\overrightarrow{PI}_q$ stocke l'*intention* de chaque fournisseur $p \in P_q$ pour traiter la tâche q (mêmes informations notées différemment suivant les points de vue adoptés : fournisseur ou mediateur). Le paramètre $\epsilon > 0$ est une constante habituellement valuée à 1. Son rôle est seulement d'éviter le passage à zéro. ω est une variable qui prend ses valeurs dans l'intervalle $[0..1]$; elle traduit le fait qu'un point de vue peut être privilégié par rapport ? l'autre. Lorsque ω vaut $0,5$, les deux points de vue sont considérés avec une égalité parfaite. Lorsque ω vaut 0, le point de vue du fournisseur est totalement occulté pour ne prendre en compte que celui du client. L'inverse est vrai quand ω vaut 1.

Pour chaque évaluation S_bQA calcule la valeur de ω en fonction de la *satisfaction* des agents dont on confronte le point de vue. L'idée est de privilégier le point de vue de l'agent qui est le moins satisfait. ω représente donc la différence de satisfaction entre le fournisseur et le client en présence. La Figure 2.2 illustre les valeurs que ω peut prendre selon la *satisfaction* des clients et fournisseurs.

$$\omega = \Big(\big(\delta_s(c) - \delta_s(p)\big) + 1\Big) \Big/ 2 \tag{4}$$

Une fois quantifiées la pertinence d'allouer la tâche pour chaque fournisseur pouvant traiter la tâche, il n'est pas difficile d'ordonner les fournisseurs du plus pertinent au moins pertinent. Le résultat de cet ordonnancement est mémorisé dans un vecteur $\overrightarrow{R}_q$, où $\overrightarrow{R}_q[1]$ (respectivement $\overrightarrow{R}_q[N]$) est le fournisseur le plus (respectivement le moins) pertinent.

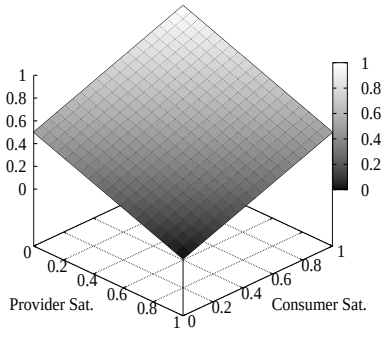


Figure 1 – Valeurs possibles de ω en fonction des satisfactions.

Algorithm 1: Allocation d'une tâche

Input : q, P_q
Output: $All\overrightarrow{oc}_q$

begin
  `// Intention du client`
  fork demander l'intention ? $q.c$;
  `// Ints. des fournisseurs`
  foreach $p \in P_q$ **do**
    fork demander l'intention ? p ;
  waituntil $\overrightarrow{CI}_c q$ et $\overrightarrow{PI}_q$ ou un *timeout*;
  `// Éval. des fournisseurs`
  foreach $p \in P_q$ **do**
    évaluer p par rapport $\overrightarrow{CI}_c q$ & $\overrightarrow{PI}_q$;
  `// Ordre des fournisseurs`
  rank P_q, $\overrightarrow{R}q$, par rapport $scr_p(q)$;
  `// Selection des fournisseurs`
  for $i = 1$ **to** $min(n, N)$ **do** $All\overrightarrow{oc}[\overrightarrow{R}_q[i]] \leftarrow 1$;
  for $j = min(n, N) + 1$ **to** N **do** $All\overrightarrow{oc}[\overrightarrow{R}_q[j]] \leftarrow 0$;
end

Principe

Pour allouer une tâche q, dans un premier temps, le médiateur doit être capable de déterminer l'ensemble de fournisseurs qui ont la capacité de traiter cette tâche (i.e. l'ensemble P_q). Un grand nombre de travaux ont déjà porté sur ce problème, voir par exemple [LH04, **?**]. Aussi nous considérerons ce problème comme étant résolu par des techniques que nous ne présentons pas ici.

Les grandes étapes permettant d'allouer la tâche q à $q.n$ fournisseurs parmi ceux de l'ensemble P_q sont présentées dans l'algorithme 3 . Après avoir demandé en parallèle et obtenu les *intentions* des

différents participants concernés par cette allocation (ligne 2)[4], l'évaluation de chaque fournisseur est calculée (ligne 7) comme indiqué dans la section précédente. Une fois triés (ligne 8), ceux qui sont considérés comme étant les plus pertinents sont sélectionnés (ligne 9). Finalement, tous les participants à cette médiation sont informés du résultat (lignes 9 et 10). Cet algorithme peut être optimisé de bien des manières, mais notre but est ici d'en présenter une version facilement compréhensible.

Réplication de Requêtes

Aujourd'hui, Internet offre de nombreuses possibilités à grande échelle de calcul distribué. Une des plus récentes solutions à grande échelle de l'informatique est l'utilisation de milliers, voire des millions de non fiables, les ordinateurs personnels autonomes (les fournisseurs) connecté à Internet pour échanger des informations ou de calcul des ressources les uns avec les autres. Fournisseurs de mettre leurs fonctionnalités de calcul ou de ressources au service d'autrui (les consommateurs) pour des raisons de collaboration ou pour leurs propres avantages. D'une part, les services Web sont un exemple clair de la concurrence massive de calcul distribué lors de leur invocation encourt un coût monétaire. D'autre part, quelques exemples de coopération massive de calcul distribué sont les projets SETI@home et distributed.net. En effet, dans ces environnements les participants ont des intérêts envers des requêtes ainsi que l'autonomie de quitter et rejoindre le système à volonté. Par exemple, un participant, don de ses ressources de calcul à plusieurs projets de recherche, mais désire d'effectuer en moyenne plus de requêtes de certains projets spécifiques que des autres.

Le fait de considérer à grande échelle, ouvert (pour les participants autonomes) des systèmes distribués a une autre conséquence : la possibilité de participants, ou, plus généralement, dysfonctionnement des participants. En fait, comme l'échelle d'un système distribué est augmentée du nombre de participants, la possibilité que l'un d'eux est soumis à l'échec augmente également. Études des participants dans la disponibilité à grande échelle systèmes distribués, tels que Overnet, Napster et Gnutella montrent qu'il existe un important roulement en raison de l'échec. En conséquence, dans ce contexte, l'utilité des applications distribuées est de plus en plus limitée par la disponibilité plutôt que de performance. Ce problème de traiter avec les participants des échecs a été largement étudié par plusieurs travaux dans les systèmes distribués. En raison de l'autonomie, cependant, un fournisseur peuit être malicieux, c'est-à-dire de être byzantine, et, par conséquent, il peut erroner ou tout simplement ne retourner pas des résultats, pour une requête. C'est pourquoi certains systèmes distribués font la réplication de la même requête (c'est-à-dire qu'elle crée de sauvegarde des requêtes pour une requête) sur plusieurs fournisseurs pour comparer leurs résultats. Il est, par exemple, la politique de SETI@home. Par conséquent, la réplication d'une requête peut répondre à deux objectifs : comparer les résultats des requêtes de différents fournisseurs et à soutenir d'éventuels fournisseurs échecs. Dans ce chapitre, nous nous concentrons sur celles-ci et l'ancien rapport aux travaux futurs. En effet, la réplication requête a un coût qui ne devrait pas être négligée car elle car elle peut rapidement utiliser toutes les ressources de calcul dans le système. Le système de point de vue, de recherche nécessite la réplication soit plus puissant fournisseurs ou d'autres fournisseurs. Les participants de point de vue, il n'est pas évident qu'un participant a la même intention, et donc la même satisfaction, à être utilisés comme source primaire que comme source de sauvegarde. À notre connaissance, aucun modèle tolérance de fautes a traité avec les intentions et de la satisfaction des participants, par conséquent, aucune requête de reproduction technique est appropriée pour des systèmes d'information distribués avec des participants autonomes qui ont des intérêts à envers les requêtes.

[4] Un timeout évite les attentes trop longues.

Modèle de satisfaction qui considère les pannes des participants

Rappelez-vous que dans le chapitre , nous avons proposé un modèle pour caractériser les participants de leurs intentions à long terme, qui définit déjà les définitions de certains des participants satisfaction. Toutefois, nous n'avons pas considéré dans ce modèle que les requêtes ont différent criticits pour un consommateur, que les résultats produits par un fournisseur ne peuvent pas être renvoyés à un consommateur en raison de la panne du consommateur ou du fournisseur. Ce dernier point implique, en d'autres termes, le modèle que nous avons proposées suppose qu'étant donné une requête q, les résultats de chaque fournisseur de P_q est retourné à le consommateur. Dans cette section, nous libérons cette hypothèse et proposons des définitions de satisfaction qui d'envisagent la possibilité que seuls les résultats d'un ensemble $\widehat{\widehat{P_q}} \subseteq P_q$ de prestataires sont retournés au consommateur en raison des pannes des participants. En outre, nous proposons une définition satisfaction globale, qui considère la probabilité de non-participants, devrait caractériser le bonheur de tous les participants concernés par l'attribution d'une requête.

La satisfaction des consommateurs Comme défini dans le chapitre , il est par le biais de sa satisfaction que le consommateur peut évaluer s'il est, ou non, les résultats qu'il attend du médiateur. Considérant que le consommateur désire des résultats différents pour une requête, nous avons défini la satisfaction du consommateur afin que plus des résultats il aura, plus il sera satisfait. Toutefois, ce n'est pas toujours le cas lorsque les fournisseurs tombent en panne et que les requêtes ont different valeurs de criticits. Par exemple, un consommateur, exigeant deux résultats pour une requête critique basse, il peut être plus satisfaits de recevoir un seul résultat d'un fournisseur envers qu'il a une intention de 1 que de recevoir les résultats de deux fournisseurs à qui il a une intention de 1 et -1, respectivement. Cela dépend de la criticité de la requête pour recevoir autant de résultats dont le consommateur a besoin. Intuitivement, si une requête a une criticité $\gamma = 1$ (respectivement $\gamma = 0$) signifie que le consommateur ne serait pas satisfait du tout si il n'a pas reu tous les résultats dont il a besoin (resp. signifie que la satisfaction du consommateur dépend fortement du nombre de résultats, il reoit). Pour tenir compte de cela, soit $\widehat{\widehat{P_q}}$ l'ensemble des fournisseurs dont les résultats sont retournés au consommateur, nous modifions la satisfaction coefficient $\frac{1}{n}$ de l'équation 1.3 comme suit,

$$\frac{1-\gamma}{n-\gamma\cdot||\widehat{\widehat{P_q}}||} \tag{5}$$

Nous illustrons le comportement de ce coefficient de satisfaction au-dessus de la Figure 4.1. Observez que le plus critique est une requête et le nombre de résultats reues diminue, le coefficient de satisfaction diminue, ce qui conduit également à une diminution de satisfaction. Il est intéressant de noter que, lorsque la criticit d'une requête prend la valeur de 1, le coefficient de satisfaction toujours les valeurs zéro si le nombre de résultats n'est pas requis par le consommateur. Ensuite, compte tenu de l'Équation 4.1 et le fait que les fournisseurs tombent en panne, nous définissons la satisfaction du consommateur comme suit.

Definition 10. *Consumer Satisfaction Concerning a Single Query Allocation (revisited)*

$$\delta_s(c,\widehat{\widehat{P_q}}) = \begin{cases} \frac{1-\gamma}{n-\gamma\cdot||\widehat{\widehat{P_q}}||}\cdot\Big(\sum\limits_{p\in\widehat{\widehat{P_q}}}(\overrightarrow{CI}_q[p]+1)/2\Big) & if\ \gamma<1 \\ \frac{1}{n}\cdot\Big(\sum\limits_{p\in\widehat{\widehat{P_q}}}(\overrightarrow{CI}_q[p]+1)/2\Big) & otherwise \end{cases}$$

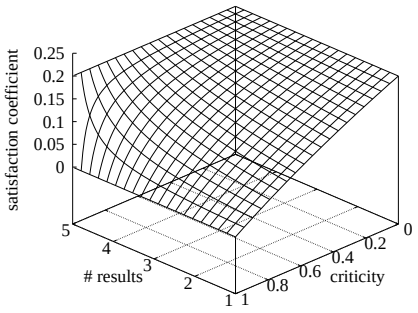


Figure 2 – Number of results vs query's criticity when a consumer requires five results.

La satisfaction d'un fournisseur Comme on l'a noté à ce jour, un fournisseur peut évaluer, par le biais de sa satisfaction, si le médiateur, il attribue ces questions qui répondent à ses intentions. Inversement à un consommateur, le fait qu'une requête a une haute criticité, ou non, n'a pas d'influence sur la satisfaction d'un prestataire. A son tour, le fait que le fournisseur effectue une requête et ses résultats ne sont pas retournés au consommateur peut impacter de manière significative sur sa satisfaction (en fonction de son coût). La raison en est que le prestataire est généralement égoistes et, par conséquent, le fait de dépenser les ressources de calcul à effectuer des requêtes dont elle recueille aucun avantage ne répond pas à leurs intentions à toutes. Ainsi, compte tenu de cela, nous définissons à nouveau la satisfaction d'un prestataire $p \in P_q^{ok} \cap P_q$ comme suit.

Definition 11. *Provider Satisfaction Concerning a Single Query Allocation*

$$\delta_s(p, \widehat{P_q^r}, \widehat{\widehat{P_q}}) = \left| \begin{array}{ll} \big(\overrightarrow{PPI}_p[q] + 1\big)/2 & if\ p \in \widehat{\widehat{P_q}} \\ \big(-\overrightarrow{PPI}_p[q] + 1\big)/2 & if\ p \in (P_q \backslash \widehat{P_q^r}) \cap P_q^{ok} \\ \big(1 - \overrightarrow{PC}_q[p]\big)/2 & if\ p \in \big(\widehat{P_q^r} \backslash \widehat{\widehat{P_q}}\big) \cap P_q^{ok} \end{array} \right.$$

Rappelez-vous que vecteur $\overrightarrow{PPI}_p$ contient les intentions exprimées par p vers les k dernires requêtes proposé. L'idée qui derrièrre la définition ci-dessus est que si un fournisseur effectue une requête et ses résultats produits sont retournés au consommateur, sa satisfaction est relative à une telle allocation est alors basée sur son intention (ligne 1 de l'équation ci-dessus). Sinon, si un fournisseur n'xecute pas une requête, sa satisfaction concernant cette requête répartition est fondée sur les effets négatifs de son intention (ligne 2). Cela signifie que si un fournisseur négatif exprime une intention d'effectuer une requête et il n'est pas attribué la requête, il est satisfait avec le médiateur d'emploi parce qu'elle ne dépense pas au calcul des ressources pour effectuer une requête qu'il n'aime pas. Dans la définition ci-dessus, nous estimons également que les cas où un fournisseur effectue une requête et de ses produits résultat n'est pas retourné au consommateur (ligne 3). Dans ce cas, nous supposons que le prestataire n'est pas satisfait de l'exécution des requêtes pour rien. Ainsi, nous définissons la satisfaction du fournisseur sur la base de son coût pour effectuer une requête. Nous traduisons le coût des valeurs dans l'intervalle $[0..0, 5]$, ce qui signifie que le prestataire a toujours un faible taux de satisfaction dans de tels cas.

global de satisfaction Nous faisons précisément dans cette section la satisfaction globale en ce qui concerne une requête d'allocation. L'un des principaux objectifs lorsqu'ils traitent avec des fournisseurs d'indisponibilité dans la requête des allocations est de créer des requêtes de sauvegarde afin que les réponses à court temps de réponse sont assurées pour les consommateurs. Dans autonome des systèmes distribués, l'attribution de sauvegarde des requêtes n'est pas une tâche facile dû à autonomie des participants. Jusqu'à présent, nous avons défini la requête des problèmes d'allocation dynamique et autonome des systèmes distribués comme une maximisation de la satisfaction globale. Toutefois, selon les définitions des sections 4.2.1 et 4.2.2, les intentions des participants sont contradictoires, c'est lors de la création de sauvegarde des requêtes consommateurs améliore la satisfaction (en assurant leur satisfaction). Nous définissons la satisfaction globale par l'examen de ce point contradictoires. Dans ce but, nous considérons la probabilité de défaillance des participants.

Tout d'abord, il convient de noter que nous supposons que les erreurs ne sont pas relationées. Ainsi, la probabilité qu'un participant i ne pas échouer dans une unité de temps est de $1 - \ failProbabilityi$. Soit t_q le délais requis par un fournisseur p (qui ne figure pas dans la notation pour des raisons de clarté) pour traiter une requête q. Par conséquent, la probabilité $\mathcal{A}_i^(t_q)$ que i ne pas échouer dans un intervalle de temps discret t_q (c'est-à-dire que toujours être disponible au cours du temps intervalle (t_q) est donnée par l'équation ci-dessous.

$$\mathcal{A}_i^{t_q} = (1 - f_i)^{t_q} \tag{6}$$

Compte tenu de cela, voyons d'abord caractériser la probabilité qu'une requête, être traités avec succès par au plus h fournisseurs dont les pires classé fournisseur a un rang r, soit la probabilité de trouver au plus h jusqu'à ce que les fournisseurs de rang r en vecteur $\overrightarrow{R}_q$ ne manquent pas, avant de retourner les résultats d'une requête.

Lemma 1. *La probabilité de succès $\mathcal{S}_q^h(\widehat{Pr_q})$ qu'une requête q a pour être traite par au plus h fournisseurs de $\widehat{Pr_q}$ est donnée par*

$$\mathcal{S}_q^h(\widehat{Pr_q}) = \sum_{\substack{P_q^{ok} \subseteq \widehat{Pr_q} \\ ||P_q^{ok}|| \leq h}} \Big(\prod_{p \in P_q^{ok}} \mathcal{A}_p^{t_q} \prod_{p \in \widehat{Pr_q} \setminus P_q^{ok}} (1 - \mathcal{A}_p^{t_q}) \Big)$$

Démonstration. La probabilité qu'un ensemble de fournisseurs avec succès effectue une requête est donnée par sa probabilité disponible. Par l'équation Equation 4.2, la probabilité de disponibilit d'un ensemble P_q^{ok} de fournisseurs dans l'ensemble $\widehat{Pr_q}$ est $\prod_{p \in P_q^{ok}} \mathcal{A}_p^{t_q} \cdot \prod_{p \in \widehat{Pr_q} \setminus P_q^{ok}} (1 - \mathcal{A}_p^{t_q})$. Par conséquent, la probabilité de succès d'une requête q à être effectuée par au plus h fournisseurs dans $\widehat{Pr_q}$ est donnée par la somme disponible probabilité de tous les différents ensembles P_q^{ok} in $\widehat{Pr_q}$ qui satisfont la contrainte $||P_q^{ok}|| \leq \ h$, donc $\mathcal{S}_q^h(\widehat{Pr_q}) = \sum_{\substack{P_q^{ok} \subseteq \widehat{Pr_q} \\ ||P_q^{ok}|| \leq h}} \Big(\prod_{p \in P_q^{ok}} \mathcal{A}_p^{t_q} \prod_{p \in \widehat{Pr_q} \setminus P_q^{ok}} (1 - \mathcal{A}_p^{t_q}) \Big)$. □

Permettez-nous maintenant de caractériser la probabilité que les résultats produits par un fournisseur et un ensemble de fournisseurs sont retournés au consommateur.

Lemma 2. *Soit x la cardinalit de $\widehat{\widehat{P_q}}$, i.e. $||\widehat{\widehat{P_q}}||$, etant donné une requête q, la probabilité $\mathcal{S}_q^a(\widehat{Pr_q}, x)$ que les résultats des fournisseurs de $\overrightarrow{R}_q[a]$ et d'autres $x - 1$ fournisseurs dans $\widehat{Pr_q}$ est retourné au*

consommateur $q.c$ est donnée par la formule

$$\mathcal{S}_q^a(\widehat{P_q^r}, x) = \begin{cases} \sum\limits_{\substack{\widehat{\widehat{P_q}} \subseteq \widehat{P_q^r} \\ ||\widehat{\widehat{P_q}}|| < x \\ \overrightarrow{R}_q[a] \in \widehat{\widehat{P_q}}}} \Big(\prod\limits_{p \in \widehat{\widehat{P_q}}} \mathcal{A}_p^{t_q} \prod\limits_{p \in \widehat{P_q^r} \setminus \widehat{\widehat{P_q}}} (1 - \mathcal{A}_p^{t_q}) \Big) \; if \; x < q.n \\ \sum\limits_{\substack{\widehat{\widehat{P_q}} \subseteq \widehat{P_q^r} \\ ||\widehat{\widehat{P_q}}|| = x \\ \overrightarrow{R}_q[a] \in \widehat{\widehat{P_q}}}} \Big(\prod\limits_{p \in \widehat{\widehat{P_q}}} \mathcal{A}_p^{t_q} \prod\limits_{\substack{p = \overrightarrow{R}_q[j] \\ j \leq \max(k) \\ \overrightarrow{R}_q[k] \in \widehat{\widehat{P_q}} \\ p \notin \widehat{\widehat{P_q}}}} (1 - \mathcal{A}_p^{t_q}) \Big) \quad else \end{cases}$$

Démonstration. Nous utilisons un raisonnement proche celui du Lemma 3. Globalement, étant donné une requête q, deux cas peuvent exister pour qu'un fournisseur de $\overrightarrow{R}_q[a]$ retourne son produit suite au consommateur $q.c$: (i) que moins de $q.n$ fournisseurs $\widehat{P_q^r}$ est disponible au cours intervalle de temps discret t_q, et (ii) que la même ou plus de qn fournisseurs de $\widehat{P_q^r}$ est disponibles en temps discret au cours de l'intervalle t_q, mais tout au plus $x - 1$ fournisseurs ont un score plus élevé que $\overrightarrow{R}_q[a]$.

Dans le premier cas, le fournisseur $\overrightarrow{R}_q[a]$ ne doit pas être disponible au cours de l'intervalle de temps t_q pour tre dans l'ensemble. Ainsi, la probabilité $\mathcal{S}_q^a(\widehat{P_q^r}, x)$ que les rsultats de $\overrightarrow{R}_q[a]$ soient retournés à qc , quand $x < q.n$, est donnée par la somme disponible probabilité de tous les ensembles différents $\widehat{\widehat{P_q}}$ dans $\widehat{P_q^r}$ qui satisfont la contrainte $\overrightarrow{R}_q[a] \in \widehat{\widehat{P_q}}$. Par conséquent, l'Équation 4.2 et la disponibilité pour le $x < q.n$ cas, $\mathcal{S}_q^a(\widehat{P_q^r}, x) = \sum\limits_{\substack{\widehat{\widehat{P_q}} \subseteq \widehat{P_q^r} \\ ||\widehat{\widehat{P_q}}|| < x \\ \overrightarrow{R}_q[a] \in \widehat{\widehat{P_q}}}} \Big(\prod\limits_{p \in \widehat{\widehat{P_q}}} \mathcal{A}_p^{t_q} \prod\limits_{p \in \widehat{P_q^r} \setminus \widehat{\widehat{P_q}}} (1 - \mathcal{A}_p^{t_q}) \Big)$.

Dans le second cas, à l'inverse du premier cas, fournisseur de $\overrightarrow{R}_q[a]$ doivent être disponibles au cours de l'intervalle de temps discret t_q, mais doit également avoir au moins les $q.n$ pire classement en $\overrightarrow{R}_q[a]$ pour tre dans l'ensemble $\widehat{\widehat{P_q}}$. Ainsi, la probabilité $\mathcal{S}_q^a(\widehat{P_q}, x)$ que les résultats de $\overrightarrow{R}_q[a]$ soient retournés à $q.c$, lorsque $x = qn$, est donnée par la somme disponible probabilité de tous les différents ensembles $\widehat{\widehat{P_q}}$ dans $\widehat{P_q^r}$ qui satisfont aux contraintes $\overrightarrow{R}_q[a] \in \widehat{\widehat{P_q}}$ et $\nexists \overrightarrow{R}_q[j] \in \widehat{P_q^r} : \; j < a$. Par conséquent, par l'Equation 4.2 et la disponibilité pour le cas $x = q.n$, $\mathcal{S}_q^a(\widehat{P_q^r}, x) = \sum\limits_{\substack{\widehat{\widehat{P_q}} \subseteq \widehat{P_q^r} \\ ||\widehat{\widehat{P_q}}|| = x \\ \overrightarrow{R}_q[a] \in \widehat{\widehat{P_q}}}} \Big(\prod\limits_{p \in \widehat{\widehat{P_q}}} \mathcal{A}_p^{t_q} \prod\limits_{\substack{p = \overrightarrow{R}_q[j] \\ j \leq \max(k) \\ \overrightarrow{R}_q[k] \in \widehat{\widehat{P_q}} \\ p \notin \widehat{\widehat{P_q}}}} (1 - \mathcal{A}_p^{t_q}) \Big)$. □

Ensuite, nous avons officiellement la satisfaction globale en ce qui concerne l'attribution d'une requête en théorème 6. Depuis la satisfaction globale est calculée par le médiateur, étant donné une requête q, nous considérons comme suit vecteur $\overrightarrow{PI}_q$ à représenter les intentions des fournisseurs de P_q, mais que $\overrightarrow{PI}_q[p] = \overrightarrow{PPI}_p[q]$.

Theorem 1. *La satisfaction globale $\Theta(\widehat{P_q^r})$ de l'affectation d'une requête q à un ensemble $\widehat{P_q^r}$ est,*

$$\begin{aligned}\Theta(\widehat{P_q^r}) \;=\; & \sum_{j=1}^{r}\Bigg(\mathcal{A}^{t_q}_{\overrightarrow{R}_q[j]}\cdot\Big(\mathcal{A}_c^{t_q}\cdot\mathcal{S}_q^{n-1}(\widehat{P_q^{j-1}})\cdot\overrightarrow{PI}_q[\overrightarrow{R}_q[j]] \;+\\ & \qquad\big(1-\mathcal{A}_c^{t_q}\big)\cdot\mathcal{S}_q^{n-1}(\widehat{P_q^{j-1}})\cdot\overrightarrow{PC}_q[\overrightarrow{R}_q[j]] \;+\\ & \qquad\big(1-\mathcal{S}_q^{n-1}(\widehat{P_q^{j-1}})\big)\cdot\overrightarrow{PC}_q[\overrightarrow{R}_q[j]]\Big)\Bigg) \;+\\ & \sum_{j=r+1}^{||P_q||}\mathcal{A}^{t_q}_{\overrightarrow{R}_q[j]}\cdot-\overrightarrow{PI}_q[\overrightarrow{R}_q[j]] \;+\\ & \mathcal{A}_c^{t_q}\cdot\sum_{j=0}^{n}\left(\frac{1-\gamma}{n-\gamma\cdot j}\cdot\sum_{a=1}^{r}\Big(\mathcal{S}_q^a(\widehat{P_q^r},j)\;\cdot\overrightarrow{CI}_q[\overrightarrow{R}_q[a]]\Big)\right)\end{aligned}$$

Démonstration. Pour plus de clarté, nous procédons à démontrer l'équation ci-dessus ligne par ligne. La satisfaction globale de l'affectation d'une requête q est la somme des prévisions de satisfaction des fournisseurs de P_q et la satisfaction des consommateurs devrait $q.c$. Nous avons d'abord se concentrer sur les fournisseurs. Compte tenu Definition 33, trois cas se produisent : (i) lorsque les fournisseurs sont dans l'ensemble $\widehat{\widehat{P_q}}$, (ii) lorsque le prestataire est dans l'ensemble $(\widehat{P_q^r}\backslash\widehat{\widehat{P_q}})\cap P_q^{ok}$, et **(iii)** lorsque le prestataire est dans l'ensemble $(P_q\backslash\widehat{P_q^r})\cap P_q^{ok}$. En effet, dans tous ces trois cas, un fournisseur de P_q doit être dans P_q^{ok} à calculer sa satisfaction. Cette probabilité est donnée par l'Équation 4.2, $\mathcal{A}^{t_q}_{\overrightarrow{R}_q[j]}$.

Pour qu'un fournisseur $\overrightarrow{R}_q[j]$ dans $\widehat{P_q^r}\cap P_q^{ok}$ soit dans $\widehat{\widehat{P_q}}$, le consommateurs $q.c$ doit être disponible au cours de l'intervalle de temps discret requis par $\overrightarrow{R}_q[j]$ pour traiter q, ce qui est donnée par l'Équation 4.2, $\mathcal{A}_c^{t_q}$, et que la plupart des autres $q.n-1$ fournisseurs avec un classement inférieur à j également soit dans la série $\widehat{P_q^r}\cap P_q^{ok}$, ce qui est donnée par le Lemme 3, $\mathcal{S}_q^{n-1}(\widehat{P_q^{j-1}})$. Donc, la probabilité que les résultats produits par les fournisseurs de $\widehat{P_q^r}$ est retourné à $q.c$ est,

$$\sum_{j=1}^{r}\Big(\mathcal{A}^{t_q}_{\overrightarrow{R}_q[j]}\cdot\mathcal{A}_c^{t_q}\cdot\mathcal{S}_q^{n-1}(\widehat{P_q^{j-1}})\cdot\overrightarrow{PI}_q[\overrightarrow{R}_q[j]]\Big)$$

ce qui est multiplié par l'intention de $\overrightarrow{R}_q[j]$ car ses résultats sont retournés à $q.c$. Cela prouve la première ligne de la satisfaction globale.

Maintenant, un fournisseur $\overrightarrow{R}_q[j]$ dans $\widehat{P_q^r}\cap P_q^{ok}$ ne soit pas $\widehat{\widehat{P_q}}$ pour deux raisons principales : D'abord parce que $q.c$ tombe en panne dans l'intervalle de temps discret t_q, et, deuxièmement, parce qu'au moins $q.n$ autres fournisseurs avec un classement inférieur à j dans $\widehat{P_q^r}\cap P_q^{ok}$. Par l'Équation 4.2 et le Lemme 3, nous avons que la probabilité que la première possibilité se produit est,

$$\sum_{j=1}^{r}\Big(\mathcal{A}^{t_q}_{\overrightarrow{R}_q[j]}\cdot\big(1-\mathcal{A}_c^{t_q}\big)\cdot\mathcal{S}_q^{n-1}(\widehat{P_q^{j-1}})\cdot\overrightarrow{PC}_q[\overrightarrow{R}_q[j]]\Big)$$

et que la seconde possibilité se produit est,

$$\sum_{j=1}^{r}\Big(\mathcal{A}^{t_q}_{\overrightarrow{R}_q[j]}\cdot\big(1-\mathcal{S}_q^{n-1}(\widehat{P_q^{j-1}})\big)\cdot\overrightarrow{PC}_q[\overrightarrow{R}_q[j]]\Big)$$

qui sont multipliés par le cot $\overrightarrow{R}_q[j]$ car ses résultats ne sont pas retournés à $q.c$ dans les deux possibilités. Cela prouve la deuxième et troisième lignes de la satisfaction globale.

Pour finaliser avec le fournisseur, nous allons maintenant examiner le cas où un fournisseur ne peut être attribué une requête. Par l'Équation 4.2, la probabilité qu'un ensemble de fournisseurs de rang $j > r$ soit dans $(P_q \backslash \widehat{P_q^r}) \cap P_q^{ok}$ est,

$$\sum_{j=r+1}^{||P_q||} \mathcal{A}^{t_q}_{\overrightarrow{R}_q[j]} \cdot -\overrightarrow{PI}_q[\overrightarrow{R}_q[j]]$$

qui est multipliés par l'intention ngative du fournisseur $\overrightarrow{R}_q[j]$. Cela prouve la quatrime ligne de la satisfaction globale.

Concernant un consommateur, pour calculer sa satisfaction, le consommateur doit être disponible dans un intervalle de temps discret t_q, $\mathcal{A}_c^{t_q}$. Les prévisions de satisfaction d'un consommateur $q.c$ concernant un fournisseur $\overrightarrow{R}_q[a] \in \widehat{\widehat{P_q}}$ est donnée par la multiplication de la probabilité du fournisseur $\overrightarrow{R}_q[a]$ et d'autres $j-1$ fournisseurs dans $\widehat{P_q^r}$ dans $\widehat{\widehat{P_q}}$ et l'intention de $q.c$ vers $\overrightarrow{R}_q[a]$. Ainsi, par le Lemme 4, la satisfaction attendue du consommateurs $q.c$ concernant un ensemble $\widehat{\widehat{P_q}}$ est donnée par la formule

$$\mathcal{A}_c^{t_q} \cdot \sum_{a=1}^{r} \left(\mathcal{S}_q^a(\widehat{P_q^r}, j) \cdot \overrightarrow{CI}_q[\overrightarrow{R}_q[a]]\right.$$

Par conséquent, par définition 32, la satisfaction attendue du consommateurs $q.c$ concernant tous les ensembles de $\widehat{\widehat{P_q}}$ dans $llocRPrividersqr$ est,

$$\mathcal{A}_c^{t_q} \cdot \sum_{j=0}^{n} \left(\frac{1-\gamma}{n-\gamma \cdot j} \cdot \sum_{a=1}^{r} \left(\mathcal{S}_q^a(\widehat{P_q^r}, j) \cdot \overrightarrow{CI}_q[\overrightarrow{R}_q[a]] \right) \right)$$

qui finalement se prouve la dernière (la cinquième) ligne de l'équation de la satisfaction globale. □

Une méthode basée sur la satisfaction pour faire la réplication non systématique de requêtes

Nous présentons dans cette section l'algorithme pour faire la réplication de requêtes nommée S_bQR. À l'inverse de plusieurs travaux qui crént un fournisseur de sauvegarde par requête ($n_b = 1$), S_bQR fait la réplication des requêtes dans le but d'accroître la satisfaction des participants. Ainsi, il ne réplique une requte si cela implique une augmentation de la satisfaction globale (voir Theoreme 6). L'algorithme 4 montre les principales étapes du processus de la réplication de requêtes. S_bQR reoit en entrée une requête q, les vecteurs $\overrightarrow{R}_q$, $\overrightarrow{CI}_q$, $\overrightarrow{PI}_q$ et $\overrightarrow{PC}_q$. Nous assumons que $\overrightarrow{R}_q$ est générée par une fonction de scorage comme celle de la Definition 26), mais, sans perte de généralité, le vecteur $\overrightarrow{R}_q$ peut être générée par une autre fonction de scorage comme celle d'utilisation $\mathcal{U}_t$.

Tout d'abord, pour définir le nombre n_b de fournisseurs de sauvegarde, S_bQR initialises n_b à zero et r au nombre de requûetes requise par le consommateur $Q.c$ (lignes 2 and 3, respectivement, de l'algorithme 4). En fixant la valeur de r à n, a veut dire que les n_b fournisseurs de sauvegarde sont considérer à partir de $\overrightarrow{R}_q[n+1]$ à $\overrightarrow{R}_q[||P_q||]$. Comme deuxième phase, il construit l'ensemble $\widehat{P_q^r}$ et $\widehat{P_q^{r+1}}$ (lignes 4-7). Finallement, S_bQR vérifie si la satisfaction globale par rapport l'ensemble $\widehat{P_q^{r+1}}$ est plus grande que celle concernant l'ensemble $\widehat{P_q^r}$ (ligne 8). Ce calcul est donné par le théorème 7. En cas écheant,

Algorithm 2: Réplication de requêtes basée sur la satisfaction

Input : q, $\overrightarrow{R}_q$, $\overrightarrow{CI}_q$, $\overrightarrow{PI}_q$, $\overrightarrow{PC}_q$
Output: n_b

```
1  begin
       // Variables setting
2      n_b = 0
3      r = n
       // Provider sets setting
4      for i = 1 to r do
5          add provider R_q[i] to P_q^r
6          add provider R_q[i] to P_q^{r+1}
7      add provider R_q[r+1] to P_q^{r+1}
       // Computing the number of backup providers
8      while Θ(P_q^r) < Θ(P_q^{r+1}) do
9          increment n_b by one (n_b = n_b + 1)
10         increment r by one (r = r + 1)
11         if there exists provider R_q[r+1] then
12             add provider R_q[r] to P_q^r
13             add provider R_q[r+1] to P_q^{r+1}
14         else
15             break loop ;
16 end
```

il incrémente le nombre de fournisseurs de sauvegarde et ajoute le prochaine mieux class l'ensembles fournisseurs $\widehat{P_q^r}$ et $\widehat{P_q^{r+1}}$. Alors, il recommence à partir de la ligne 8 jusqu'à $\Theta(\widehat{P_q^r}) \geq \Theta(\widehat{P_q^{r+1}})$ ou s'il n'y a plus des fournisseurs dans le vecteur $\overrightarrow{R}_q$ (lignes 9-15). Bien sûr, l'algorithme 4 peut être optimisé, mais notre objectif est de montrer quel sont les phases dans le processus de réplication de requêtes.

Validation

L'objectif principal est d'analyser comment les méthodes d'allocation tiennent compte des notions de satisfaction (avec un intérêt particulier pour S_bQA). Pour cela, nous avons procédé en deux temps. Dans une première étape, nous avons mesuré les *satisfactions* en considérant que les participants sont captifs (aucune possibilité de quitter le système). Puis dans un second temps, nous avons donné la possibilité aux participants de quitter le système pour étudier l'impact de cette autonomie.

Paramètres des simulations

En utilisant SimJava, nous avons développé en Java un simulateur pour représenter un système d'information distribué comme défini dans [LCLV07]. Pour toutes les méthodes que nous avons testées, la

configuration est la même (c.f. Tableau 2.1). Seule la technique d'allocation diffère.

Nous initialisons les participants avec une valeur de satisfaction de 0.5 qui évolue avec les 200 tâches entrantes et les 500 tâches proposées. Autrement dit, la valeur du paramètre k est 200 pour les clients et 500 pour les fournisseurs. Le nombre de clients, fournisseurs et médiateurs dans le système est 200, 400 et 1, respectivement. Nous avons affecté les ressources suffisantes au médiateur de sorte qu'il ne cause pas de goulot d'étranglement dans le système. L'*utilisation* d'un fournisseur p à un moment donné t (notée par la fonction $\mathcal{U}_t(p)$) dénote la charge de p à t. Inspirés de [GBGM04], nous supposons que les fournisseurs obtiennent leur *utilisation* comme l'équation 7 où la fonction $cost_p(q)$ dénote le coût de traitement de la tâche q par le fournisseur p et la fonction $cap(p)$ dénote la capacité de traitement du p.

$$\mathcal{U}_t(p) = \frac{\sum_{q \in Q_p} cost_p(q)}{cap(p)} \tag{7}$$

D'une part, nous supposons qu'un fournisseur calcule son *intention* pour traiter une tâche q comme dans [QRLV06] (voir équation 8). Cette technique permet à un fournisseur de prendre en compte à la fois son *utilisation* et ses *préférences* en portant une attention plus ou moins soutenue ? l'une ou l'autre en fonction de sa *satisfaction* actuelle. Pour cela, la satisfaction $\delta_s(p)$ utilisée ici est basée sur les *préférences*, notées $prf_p(q)$).

$$pi_p(q) = \begin{cases} (prf_p(q)^{1-\delta_s(p)})(1-\mathcal{U}_t(p))^{\delta_s(p)}, \\ \qquad si (prf_p(q) > 0) \wedge (\mathcal{U}_t(p) < 1) \\ -\Big(((1-prf_p(q)) + \epsilon\big)^{1-\delta_s(p)} \big(\mathcal{U}_t(p) + \epsilon\big)^{\delta_s(p)}\Big) \\ \qquad sinon \end{cases} \tag{8}$$

D'autre part, pour simuler une hétérogénéité élevée des *intentions* chez les clients, nous divisons l'ensemble de fournisseurs en trois classes selon l'intérêt des clients : ceux pour qui les clients ont un grand intérêt (60% des fournisseurs), un intérêt moyen (30% des fournisseurs) et un intérêt faible (10% des fournisseurs). Par simplicité, nous nommons ces groupes de fournisseurs les *très-intéressants*, *moyennement-intéressants* et *peu-intéressants* respectivement. Les clients obtiennent leurs *intentions* envers les fournisseurs *très-intéressants* entre $0,34$ et 1, envers les fournisseurs *moyennement-intéressants* entre $-0,54$ et $0,34$ et envers les fournisseurs *peu-intéressants* entre -1 et $-0,54$. Sans perte de généralité, nous pourrions utiliser d'autres mécanismes pour obtenir les *intentions* des clients (par exemple, en utilisant les langages *TCL* ou *Rush*). Nous nous basons sur les résultats présentés dans [SGG02] pour paramétrer l'hétérogénéité des capacités des fournisseurs. 10% des fournisseurs ont une capacité faible, 60% des fournisseurs ont une capacité moyenne et 30% des fournisseurs ont une capacité forte. Par simplicité, nous nommons ces trois groupes de fournisseurs les *très-capables*, *moyennement-capables* et *peu-capables*. Les fournisseurs *très-capables* sont trois fois plus puissants que les fournisseurs *moyennement-capables* et sept fois plus que les fournisseurs *peu-capables*. Finalement, nous produisons deux classes de tâches qui sont traitées par les fournisseurs *très-capables* dans un temps de 1.3 et 1.5 secondes, respectivement. Les tâches arrivent au système avec une distribution de Poisson, couramment utilisée dans des environnements ouverts [Mar02].

Dans cet article, nous ne considérons pas le problème de la bande passante et nous supposons que tous les fournisseurs disposent des mêmes capacités réseau. Finalement, nous supposons que les clients ne demandent qu'un seul résultat par tâche et que tous les fournisseurs dans le système peuvent satisfaire toute tâche entrante (pas de problème de "matchmaking").

Table 2 – Paramètres des simulations.

Paramètre	Définition	Valeur
nbConsumers	Nombre de clients	200
nbProviders	Nombre de fournisseurs	400
nbMediators	Nombre de médiateurs	1
qDistribution	Distribution dans laquelle les tâches arrivent au sytème	Poisson
iniSatisfaction	Satisfaction initiale	0.5
conSatSize	Valeur de k pour les clients	200
proSatSize	Valeur de k pour les fournisseurs	500
nbRepeat	Nombre de répétitions par simulation	10

Méthodes de référence

Capacity based

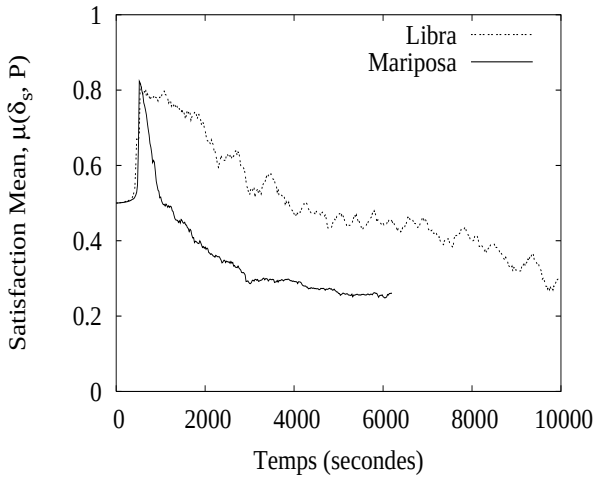
Dans le contexte des systèmes d'information distribués, les deux techniques les plus connues pour faire de l'allocation de tâches sont basées sur la charge [ABKU99, GBGM04] et la capacité [MTS90, RM95, SKS92]. Nous ne considérons pas les méthodes basées sur la charge car, à la différence de celles basées sur la capacité, elles supposent que tous les fournisseurs et toutes les tâches sont homogènes. Le principe des méthodes basées sur la capacité est d'affecter chaque tâche entrante aux fournisseurs qui sont les moins utilisés parmi ceux de l'ensemble P_q. $Capacity\ based$ s'est avéré meilleur que $Load\ Based$ dans des systèmes hétérogènes. Donc, nous comparons S_bQA à cette approche (que nous nommons $Capacity\ based$ pour simplicité) dans nos simulations.

Mariposa

Diverses approches économiques ont été proposées [FNSY96, FYN88, SAL$^+$96] pour faire de l'allocation de tâches. Mariposa [SAL$^+$96] est l'un des premiers systèmes utilisant des techniques de microéconomie. Mariposa a montré de bonnes performances dans des environnements ouverts et hétérogènes. C'est pourquoi nous l'avons implémenté et comparé à notre proposition. Dans Mariposa, chaque tâche entrante q arrive à un médiateur (broker) qui trouve l'ensemble P_q et demande à chaque fournisseur de P_q son *offre* pour traiter q. Les fournisseurs calculent leurs offres en fonction de leurs préférences et de leur charge actuelle. Une fois ces *offres* obtenues le médiateur alloue la tâche aux fournisseurs ayant fait l'*offre* la plus basse.

Résultats expérimentaux

Si les participants sont autonomes, ils peuvent quitter le système par *mécontentement* ou *famine*. Néanmoins, le choix du seuil de départ est très subjectif et peut dépendre de nombreux facteurs. Nous supposons que les participants dans le système supportent des seuils élevés de *mécontentement* et de *famine*. Un client décide de quitter le système par *mécontentement* si sa *satisfaction* est inférieure à 0.5, c'est-à-dire si les allocations ne lui sont pas favorables. D'autre part, un fournisseur décide de quitter


Libra
Mariposa
Satisfaction Mean, μ(δs, P)
Temps (secondes)
0
0.2
0.4
0.6
0.8
1
0
2000
4000
6000
8000
10000

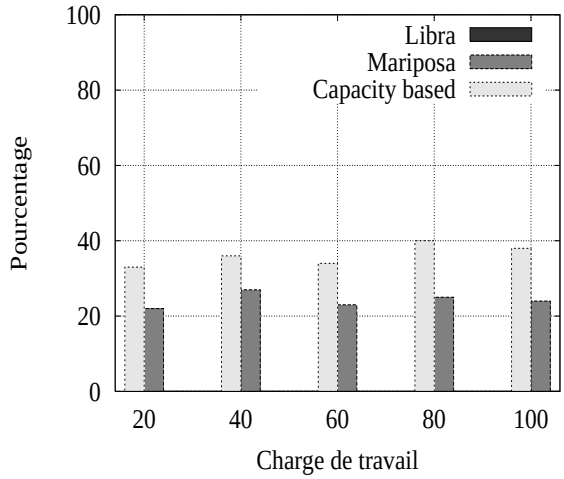


Figure 7 – Départs des clients avec une charge exprimée en fonction des capacités initiales du système ; agents autorisés à quitter le système.

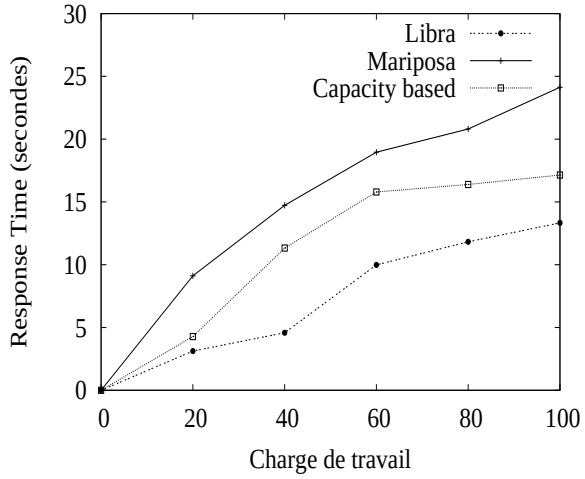


Figure 8 – Temps de réponse avec une charge exprimée en fonction des capacités initiales du système ; agents autorisés à quitter le système.

nous voyons que *Mariposa-like* a des problèmes pour garantir un bon équilibre des tâches tandis S_bQA a des performances proches à celles de $Capacity\ based$. On peut expliquer cela par le fait que les fournisseurs *très-capables* et *très-intéressants* monopolisent les tâches dans *Mariposa-like*, créant ainsi des famines chez les autres fournisseurs. Par ailleurs, nous avons pu voir pendant nos expérimentations que S_bQA a quelques difficultés à répartir la charge entre les fournisseurs, en particulier lorsque la charge totale du système est inférieure à 40%. En revanche, quand la charge de travail augmente S_bQA devient plus efficace car les fournisseurs commencent à s'intéresser à leur charge.

La figure 6 montre le nombre de départs chez les fournisseurs avec les trois méthodes. Nous voyons que $Capacity\ based$ et *Mariposa-like* perdent presque tous les fournisseurs pour toutes les charges de travail, sauf quand elle est en dessous du 30%, tandis que S_bQA perd uniquement 28% de fournisseurs

		S_bQA				$Capacity\ based$			
		low	med	high	total	low	med	high	total
	C.I.P.	1%	5%	13%		5%	16%	31%	
D	P.A.	2%	9%	8%	19%	3%	34%	15%	52%
	P.C.	13%	6%	0%		13%	30%	9%	
		low	med	high	total	low	med	high	total
	C.I.P.	0%	0%	4%		0%	0%	0%	
S	P.A.	4%	0%	0%	4%	0%	0%	0%	0%
	P.C.	2%	2%	0%		0%	0%	0%	
		low	med	high	total	low	med	high	total
	C.I.P.	0%	0%	6%		0%	0%	38%	
O	P.A.	0%	3%	3%	6%	3%	8%	27%	38%
	P.C.	1%	4%	1%		0%	18%	20%	

Table 3 – Reasons of the provider's departures for a workload of 80% of the total system capacity.

		S_bQA				*Mariposa-like*			
		low	med	high	total	low	med	high	total
	C.I.P.	1%	5%	13%		1%	7%	11%	
D	P.A.	2%	9%	8%	19%	0%	15%	4%	19%
	P.C.	13%	6%	0%		5%	12%	2%	
		low	med	high	total	low	med	high	total
	C.I.P.	0%	0%	4%		0%	2%	6%	
S	P.A.	4%	0%	0%	4%	3%	3%	2%	8%
	P.C.	2%	2%	0%		3%	5%	0%	
		low	med	high	total	low	med	high	total
	C.I.P.	0%	0%	6%		0%	0%	65%	
O	P.A.	0%	3%	3%	6%	1%	15%	49%	65%
	P.C.	1%	4%	1%		0%	30%	35%	

Table 4 – Reasons of the provider's departures for a workload of 80% of the total system capacity.

en moyenne. La figure 7 illustre les départs de clients. $Capacity\ based$ et *Mariposa-like* perdent en moyenne 38% et 25% de clients respectivement. Sur cette expérimentation, S_bQA montre encore une fois son avantage en ne perdant aucun client.

Finalement, nous évaluons l'impact des départs des participants sur le temps de réponse (figure 8). Le temps de réponse est défini comme le temps écoulé entre l'émission d'une tâche et la réponse. Nous pouvons voir ici que S_bQA assure de meilleurs temps de réponse. Nous constatons aussi que $Capacity\ based$ est meilleur que *Mariposa-like* car, comme vu précédemment, *Mariposa-like* surcharge les fournisseurs qui sont les plus capables et intéressants.

Tous ces résultats démontrent la grande adaptabilité de S_bQA aux attentes des participants, ce qui fait que S_bQA est fortement approprié aux environnements autonomes.

		S_bQA				$Capacity\ based$			
		low	med	high	total	low	med	high	total
D	C.I.P.	1%	5%	13%	19%	5%	16%	31%	52%
	P.A.	2%	9%	8%		3%	34%	15%	
	P.C.	13%	6%	0%		13%	30%	9%	
		low	med	high	total	low	med	high	total
S	C.I.P.	0%	0%	4%	4%	0%	0%	0%	0%
	P.A.	4%	0%	0%		0%	0%	0%	
	P.C.	2%	2%	0%		0%	0%	0%	
		low	med	high	total	low	med	high	total
O	C.I.P.	0%	0%	6%	6%	0%	0%	38%	38%
	P.A.	0%	3%	3%		3%	8%	27%	
	P.C.	1%	4%	1%		0%	18%	20%	

Table 5 – Reasons of the provider's departures for a workload of 80% of the total system capacity.

Discussion

Dans nos expérimentations, nous avons vu que les fournisseurs quittent le système par *mécontentement* dans $Capacity\ based$ et par *famine* dans *Mariposa-like*. Les fournisseurs qui décident de quitter le système avec ces deux méthodes sont pour la plupart ceux qui sont les plus capables et qui sont les plus demandés par les clients. Avec S_bQA, des fournisseurs quittent aussi le système, ceci principalement pour des raisons de *mécontentement* qui s'expliquent dans la très grande majorité des cas par des problèmes d'adéquation : ces fournisseurs sont considérés comme étant peu intéressants par les clients ou comme ayant de trop faibles capacités. D'autre part nous avons vu que les départs de clients ont aussi une certaine importance. En effet, si le nombre de tâches diminue, les fournisseurs ont moins de possibilités d'être satisfaits.

Conclusion

Nous avons considéré le problème de l'allocation de tâches dans des environnements ouverts où les participants ont des attentes particulières. Dans ce cadre, prendre en compte les *intentions* des participants de façon à ce que leur attentes soient satisfaites est crucial pour le bon fonctionnement d'un système. Dans cette thèse, nous avons proposé une méthode d'allocation de tâches (S_bQA) tout en considérant et satisfaisant les *intentions* des participants.

S_bQA diffère fortement des travaux précédents. Il arbitre entre les différents participants en se basant sur leurs *satisfactions*. Il favorise ainsi le point de vue des uns ou des autres, points de vue qui sont exprimés par les *intentions*. Cela a entre autre pour conséquence de réduire les problèmes de *famine* chez les fournisseurs.

Nous avons comparé S_bQA avec deux méthodes importantes ($Capacity\ based$ et *Mariposa-like*) et avons montré par expérimentation que S_bQA présente de nombreux avantages. Les résultats prouvent que $Capacity\ based$ et *Mariposa-like* perdent plus de 20% de clients alors qu'aucun client ne quitte le système avec S_bQA.

Dans l'approche exposée dans cette thèse, un médiateur quantifie la pertinence d'allouer une tâche â tel ou tel fournisseur en considérant l'*intention* du client et du fournisseur. Il favorise le moins satisfait des deux. D'autres solutions sont envisageables. Par exemple, il peut être naturel pour un médiateur de faire

en sorte qu'un fournisseur (resp. un client) soit satisfait du travail de médiation. Cela nécessite l'introduction d'une notion de satisfaction par rapport à la médiation, qui est différente de la satisfaction présentée ici. Quel que soit le contexte, un fournisseur recevant n tâches aura la même satisfaction. En revanche, sa satisfaction par rapport à la médiation sera d'autant plus forte que le contexte lui sera défavorable. Autrement dit, une technique de médiation n'a aucun mérite à satisfaire un participant lorsque ses désirs sont en adéquation avec son environnement. Nous pensons intégrer cette notion dans une version future.

Lors d'expérimentations[5] faisant intervenir plusieurs médiateurs, nous avons constaté, sous certaines conditions, des phénomènes d'auto-organisation : clients et fournisseurs partageant les mêmes intérêts se regroupent autour du même médiateur. Nous souhaitons explorer ce phénomène que nous n'avons pas observé avec les autres approches.

Enfin, le problème adressé dans cette thèse comme celui adressé par les approches économiques est de réguler un système tout en satisfaisant les participants. Nous comptons donc développer une version économique de S_bQA pour analyser en détail les apports spécifiques de l'économie.

[5] Les expérimentations en question ne sont pas celles présentées dans cette thèse.

Introduction

Over the last few years, a tremendous number of information sites, providing a variety of content and services, have emerged on the Internet to form large-scale distributed information systems. This is mainly fostered by the current requirements on scalability and availability of information. Information sites (the providers) are heterogeneous in terms of capacity and data. Heterogeneous capacity means that some providers are more powerful than others and can treat more queries per time unit. Data heterogeneity means that providers provide different data and thus produce different results for a same query. Queries are also heterogeneous, that is, some queries consume more providers' resources than others. Moreover, in these kind of large-scale systems, consumers and providers (which we refer to participants) are usually autonomous in the sense that they are free to leave the mediator at any time and do not depend on anyone to do so. Besides, in these environments, participants usually have special interests towards queries. In such environments, it is well known that query allocation is crucial for the well operation of the system because of participants' autonomy and heterogeneity.

In this thesis, we focus on query allocation in the context of large-scale distributed information systems with a mediator that allows consumers to access providers (information sites) through queries [Mil02, ÖV99, RS97]. Providers declare their *capabilities* for performing queries to the mediator and consumers pose queries to the mediator. Then, the main function of the mediator is to allocate each incoming query to a provider among those that can deal with each query (i.e., among the set of relevant providers). A simple solution is that the mediator returns the set of relevant providers for each incoming query and let the consumers choice the providers they prefer. Several matchmaking solutions have been proposed in the literature to do so [KH95, LH04, PKPS02]. Nevertheless, given the great number and diversity of providers (services), the selection of the right provider becomes a hard task for consumers. This is why, in addition to find the set of relevant providers, a mediator must be able to narrow down the set of relevant providers, or to directly allocate the query to some of them, according to a given criteria. Much work in this context has focused on distributing the query load among the providers in a way that maximizes overall performance (typically high throughput and short response times), i.e. *query load balancing* (*qlb*) [ABKU99, GBGM04, MTS90, RM95, SKS92].

Nevertheless, participants usually have certain expectations with respect to the mediator, which are not only performance-related (see Example 1). Such expectations mainly reflect their *preferences* to allocate and perform queries in the long run. Consumers' preferences may represent e.g. their interests towards providers (based on reputation for example) or their interests in quality of service. Providers' preference may represent, for example, their topics of interests, relationships with other participants, or strategies. These participants' preferences (expectations) are clearly illustrated by Google AdWords [goo], which proposes relevant commercial providers to consumers and relevant consumers to commercial providers according to some keywords of their interest.

Example 1 (Participants' Preferences).
Consider a provider that represents a courier company. During the promotion of its new international shipping service, the provider is more interested in treating queries related to international shipments rather than national ones. Once the advertising campaign is over, the provider's preferences may change. Similarly, consumers expect the system to provide them with information that best fits their preferences.

In this context, because of participants' autonomy, *dissatisfaction* may lead participants to leave the

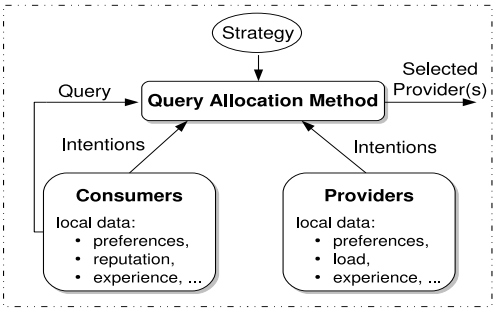


Figure 9 – Overview of Query Allocation in Distributed Systems with Autonomous Participants.

mediator, which in turn may cause some loss of system capacity to perform queries as well as some loss of system functionalities. If a participant's departure is not justified, a domino effect could lead all participants to quit the mediator. In the case of a single mediator, leaving the mediator is equivalent to depart from the system, but it could be that, in a multi-mediator system, a participant registers to another competing mediator. Thus, it is quite important to have a query allocation strategy that balances queries such that participants are satisfied. Participant's *satisfaction* means that the query allocation method meets its expectations. To make this possible, a natural solution could be to take the participants' preferences into consideration when allocating queries. However, preferences are usually considered as private data by participants (e.g. in an e-commerce scenario, enterprises do not reveal their business strategies). In addition, preferences are quite static data, i.e. long-term, while the desire of a participant to allocate and perform queries may depend not only on its preferences, but also on its context and thus is more dynamic, i.e. short-term. For instance, in Example 1, even if the provider (the courier company) prefers to perform queries related to international shipments during its advertising campaign, it is possible that, at some time, it may not desire to perform such queries because of other local reasons, e.g. by *overload*. Thus, participants are required to express their desire to allocate and perform queries via their *intention*, which may stem e.g. from combining their preferences and other local consideration such as *load* and reputation (see Figure 9).

Considerable effort has focused on the semantic description of provider so that those providers having the highest semantic score be selected [MSZ01, PSK03]. However, this does not always fit the participants' intentions. Economic solutions [FNSY96] can claim to take participants' intentions into account by integrating them into a *utility function* [Kre90], which is usually money-related. Moreover, unlike several economical models [FNSY96, FYN88, SAL$^+$96], queries must be always treated whenever possible (if there exits at least one provider to perform it) even if providers do not desire to deal with them. This is because consumers that do not get results may become dissatisfied and thus simply leave the system, which may hurt providers as well. Thus, adequate techniques for query allocation (or dynamic provider selection) are still needed.

In such distributed information systems, query allocation is a challenge for several reasons.

- There is no definition of satisfaction to reflect how well the system meets the participants' intentions in the long-run.
- Participants' expectations may be contradictory among them as well as with respect to the system performance.
- The query allocation process should be adaptable to applications and self-adaptable to changes in the participants' expectations because such expectations usually change in the course of time.
- Participants' departures may have consequences on the functionalities provided by the system.

The providers' departure may mean the loss of important system capabilities and the consumers' departure is a loss of queries for providers.

To the best of our knowledge, this problem has never been addressed before in its whole generality. Thus, our main objective in this thesis is to provide a complete solution to this problem.

Motivation

Let us illustrate distributed information systems with autonomous participants by means of a general *e-commerce* example. Consider a public e-marketplace where thousands of companies can share information and do business (such as ebay-business [eba] and freightquote [fre]). Here, business is understood in a very general sense, not necessarily involving money. Each site, which represents a company, preserves its *preferences* to allocate and perform queries. To scale up and be attractive over time, in [FFS98] it was stated that an e-marketplace should :

- protect, in the long-run, the participants' *intentions* for doing business,
- allow consumers to quickly obtain results, and
- allocate queries so that providers should have the same possibilities for doing business, i.e. to avoid *query starvation*.

Consider a simple scenario where a company ($eWine$), which desires to ship wine from France to USA, requests the mediator for companies providing international shipping services, such as freightquote [fre]. Here, a query is a call for proposals that providers have to answer in order to provide their services. Suppose that $eWine$, to make its final choice, desires to receive proposals from the two best providers that meet its *intentions*. Similarly, providers desire to participate only in those negotiations that involve queries meeting their *intentions*. The entire treatment of this scenario encompasses different aspects.

First, query planning processes may be required. This problem is addressed in different ways in the literature [ÖV99]. Thus, we do not consider this problem in this thesis and we can indifferently assume that it is done by the consumer or any other site.

Second, it needs to identify the sites that are able to deal with $eWine$'s query, i.e. to find the relevant providers. There is a large body of work on matchmaking, see e.g. [KH95, LH04], so we do not focus on this problem in this thesis.

Third, the mediator should obtain $eWine$'s *intentions* to deal with such providers and the providers' *intention* to deal with $eWine$'s query, which can be done following the architecture proposed in [LCLV07]. For simplicity, we assume in this example that the *intentions* values are binary. Assume that the resulting list contains, for simplicity, only 5 providers : $p_1, \ldots, p_5$. Table 6 shows these providers with their *intention* to perform the query and $eWine$'s *intention* to deal with each of them. To better illustrate the query allocation problem in these environments, we also show in Table 6 the providers' *available capacity*. However, it is not always possible to know this information since providers may consider it as private.

Suppose, then, that p_5 is *overloaded*, i.e. has no more resources for doing business, and that p_2 and p_4 do not intend to deal with $eWine$'s query (notice that this does not means they can refuse it) because e.g. p_2 is more interested in its new shipping service to the Asian continent (such as in Example 1) and p_3 has bad experience with $eWine$. Also, assume that $eWine$ does not intend to deal with p_1 nor p_3 since it does not trust them e.g. because of their reputation or its past experiences.

Finally, the mediator needs to select the two most available providers, such that $eWine$'s and providers' *intentions* be respected. To the best of our knowledge, no existing e-marketplace is able to do so. In fact, current *qlb* methods, whose aim is to select the most available providers, also fail in such scenarios since neither p_2 intends to deal with the query nor p_1 is of $eWine$'s interest. Thus, allocating the query to

Providers	Provider's Intention	Consumer's Intention	Available Capacity
p_1	Yes	No	0.85
p_2	No	Yes	0.57
p_3	Yes	No	0.22
p_4	No	Yes	0.15
p_5	Yes	Yes	0

Table 6 – Providers for $eWine$'s query.

these providers dissatisfies p_2 and $eWine$ in such a query allocation. And, whether this occurs several times may cause their departure from the system. The only satisfactory option, regarding the participants' *intention*, is p_5. But, allocating the query to p_5 may considerably hurt response time by overloading it and the mediator may desire to avoid such phenomena. Again, whether this occurs several times may also penalize consumers with long response times, which may cause their departure from the system. Furthermore, $eWine$ desires to receive two different proposals.

So, *what should the mediator do in the above scenario ? Should it consider the consumer's intention ? the providers' intention ? the providers' available capacity ? all three ? Which importance should the mediator pay to each of them ?* And, *how can one know that the mediator is meeting, or not, the participants expectations in the long-run ?* In this thesis, we focus on given an answer to these questions so that one can evaluate or design query allocation methods for autonomous environments, i.e. systems with autonomous participants.

Contributions

We carried out the work presented in this thesis in the context of *Atlas Peer-to-Peer Architecture* (APPA) [AM07] and of several joint projects including : the Grid4All European STREP project [gri], the ANR Massive Data Projects MDP2P [mdp], and Respire project [res]. Generally speaking, the objective of this thesis is to provide a complete query allocation framework for distributed information systems with autonomous participants. Especially, we focus on characterizing participants' intentions, allocating queries by considering participants' intentions, scaling up query allocation, and creating backup queries to deal with participants' failures. In particular, our main contributions are the following.

Our first set of contributions is the following,

- We propose a new model to characterize the participants' intentions in the long-run, which allows evaluating a system from a satisfaction point of view. Also, we formally define the utilization of a provider and make precise the query starvation notion in distributed information systems with autonomous providers. A particularity of this model is that it allows comparing query allocation methods having different approaches to regulate the system, such as the economical and *qlb* methods. Moreover, this model facilitates the design and evaluation of new query allocation methods for distributed systems that are confronted to autonomous participants (Section 1.3).
- We finally define the properties that allow evaluating the quality of query allocation methods and propose measures to do so (Section 1.4).

Then, our second main contribution is a query allocation framework that considers participants' intentions besides *qlb*. In particular,

- We propose *Satisfaction-based Query Load Balancing* (S_bQA, in short), a flexible framework with *self-adapting* algorithms to allocate queries while considering both *qlb* and participants' intentions.

Salient features of S_bQA are that :
- it affords consumers the flexibility to trade their preferences for the providers' reputation (Section 2.2),
- it affords providers the flexibility to trade their preferences for their utilization (Section 2.3),
- it allows a mediator to trade consumers' intentions for providers' intentions (Section 2.4.1), and
- it affords the mediator the flexibility to adapt the query allocation process to the application by varying several parameters (Section 2.4.2).

- We demonstrate, through experimental validation, that S_bQA significantly outperforms baseline methods, the $Capacity\ based$ and *Mariposa-like* methods, and yields significant performance benefits. We demonstrate the self-adaptability of S_bQA to participants' expectations and its adaptability to different kinds of application. We also show that applying the proposed measures over the provided model allows the prediction of possible departures of participants (Section 2.6).

After, we aim at scaling query allocation up to several mediators while ensuring as good system performance as in systems with a single mediator. Our third main set of contributions is the following.

- We discuss the challenges of using virtual money as a means of regulation in the query allocation process and make precise how the virtual money circulates within the system.
- We propose *Economic Satisfaction-based Query Allocation* method ($\$_bQA$, for short). Generally speaking, $\$_bQA$ is S_bQA using virtual money. In particular,
  - We define a way in which a provider computes its bid by considering its preferences, its satisfaction, its current utilization, and its current virtual money balance. Also, we propose three strategies that allows a provider to bid for queries in the presence of several mediators.
  - We define how a mediator allocates queries by considering both consumers' intentions and providers' bids. And, we define how a mediator should invoice providers.
  - We state the communication cost of $\$_bQA$ and demonstrate that its additional cost with respect to S_bQA is not high.
- We analytically demonstrate that $\$_bQA$ allows scaling up to several mediators with no additional network cost with respect to a single mediator.
- Finally, from a methodological point of view, it is important to compare three microeconomic methods (included $\$_bQA$) with a non-microeconomic method using satisfaction as a "money independent" measure.

Finally, the fourth main contribution is the study of an interesting variation of the fault-tolerance problem that captures the participants' satisfaction. In particular,

- We propose a satisfaction model that considers participants' failures and define the expected satisfaction of participants concerning the allocation of a given query, which we call the the global satisfaction. The global satisfaction definition takes into consideration participants' failure probability as well as their intentions.
- We propose S_bQR (for *Satisfaction-based Query Replication*), a query replication technique to compute the backup queries rate in accordance to participants' satisfaction.
- We experimentally demonstrate that S_bQR better performs, from a satisfaction and performance point of view, than replicating all incoming query. We also demonstrate that by replicating each incoming query the system suffer serious problems of performance for high workloads, but worse it loses more query results than when one does not apply a fault-tolerant technique.

Thesis Outline

This thesis is structured as follows. We propose in Chapter 1 a model that characterizes participants' intentions in the long-run, which allows us, among others, to know if a mediator is meeting the participants' intentions. In Chapter 2, we propose a complete, flexible, and self-adapting query allocation framework that considers both *qlb* and participants' intentions. The proposed framework can be used in many environments since it can be adapted to applications. We propose in Chapter 3 a query allocation method that allows handling several mediators while ensuring both good system performance and participants' satisfaction. In Chapter 4, we propose a non systematic query replication method to deal with participants' failures in distributed information systems with autonomous participants. Finally, we conclude this thesis and discuss future directions of research.

CHAPTER 1

Participants Characterization and Measures

As said in the introduction of this thesis, we consider open distributed information systems where participants (consumers and providers) are free to join and leave the mediator at will. Entrance may be motivated by some expected benefits while exit may result from disappointment, which is in general due to *dissatisfaction*. In this context, dissatisfaction means a degree of penalty to participants' expectations. This is why it is crucial, to the good operation of the system, to preserve the most possible diversity at both levels, to avoid having participants leave the system. However, to the best of our knowledge, there is no work that characterizes how well a mediator meets the participants' intentions in the long-run. Economical models consider *utility* [Kre90, MCWG95], which may be related to *satisfaction* but does not exactly fit, and *individual rationality* [San99], which is not a long-run notion. Thus, the characterizing properties must be defined in a new model, so that one can evaluate in the long-run if a mediator is *fair* or not with respect to participants. Therefore, in this chapter, our goal is to propose a model that defines such long-run notions of participants and that allows us to know if a mediator is meeting the participants' expectations. The content of this chapter is based on our material published in [QRLV07b, QRLV07c]. Our main contributions are the following :

- We propose a new model to characterize the participants' intentions in the long-run, which allows evaluating a system from a satisfaction point of view. In this model, we formally define the query starvation notion in distributed information systems with autonomous providers. Also, we define a measure to evaluate the way in which a query allocation method performs from a participants' satisfaction point of view. A particularity of this model is that it allows comparing query allocation methods having different approaches to regulate the system, such as the economical and *qlb* methods. Moreover, this model facilitates the design and evaluation of new query allocation methods for distributed systems that are confronted to autonomous participants.
- We finally define the properties that allow evaluating the quality of query allocation methods and propose measures to do so.

This chapter is structured as follows. We define the problem we address in Section 1.1. We present in Section 1.2 a traditional characterization of providers. In Section 1.3, we propose a model to characterize participants' intentions in the long-run. In Section 1.4, we define some measures that allow the evaluation of the system performance. Then, we present in Section 1.5 related work. Finally, we conclude this chapter in Section 1.6.

1.1 Problem Statement

The fact that queries and resources come from autonomous participants requires special attention. One obvious consequence is that participants are not homogeneous. The heterogeneity may be with respect to capacities, but a new heterogeneity notion comes from the fact that participants usually have different goals. For example, two participants from the same organization may have different objectives and even contradictory. Thus, the main concern of a system is to satisfy participants when allocating them queries. The common quantitative considerations (e.g. response time and throughput) for query allocation are no longer enough to evaluate this kind of systems. The objectives and relationships of participants are as much important as these common quantitative considerations since dissatisfied participants may use their autonomy to leave the system. This possibility is also part of the specificity of Internet-based systems where computational resources are not captive as in a cluster of PCs. In this chapter, we aim at proposing a model that characterizes participants' intentions, in particular, we aim at mesuring the *satisfaction* of participants and the *efficiency* of query allocation methods to satisfy providers.

Formally, we wish to modelize a distributed system that consists of a set C of consumers and a set P of providers. Let P_q denote the set of providers that are able to perform a query q, where $N_q = ||P_q||$ and $P_q \subseteq P$. A consumer $c \in C$ is free to express its intention for allocating its query q to each provider $p \in P_q$, which are stored in vector $\overrightarrow{CI}_q$. Similarly, a provider $p \in P_q$ is free to express its intention for performing a query q. A provider $p \in P$ tracks its expressed intentions for performing the k last proposed queries (allocated to it or not) into vector $\overrightarrow{PPI}_p$. We denote the k last proposed queries to p by set PQ_p^k. The expressed intentions of a participant in the k last interactions with the system denote somehow its expectations. The values of participants' intentions are between the interval $[-1..1]$. A positive value means that a provider (resp. a consumer) intends to perform (allocate) a query, while a negative value means that a provider (a consumer) does not intend to perform (allocate) a query. It is worth remembering that this does not means it can refuse to perform (resp. allocate) the query. A null value, i.e. a 0 value, denotes a participant's indifference.

Intuitively, a mediator satisfies participants if it meets their intentions. Nevertheless, the problem is to which extent the mediator should meet these intentions. An extreme point of view is that, at each interaction, i.e. query allocation, all participants' intentions are met. This is not always possible, in particular when no provider wants to perform a query and if we want each query to be treated. A more realistic view of satisfaction is that each participant benefits in the "long-run". Thus, in this chapater, we restrict ourselves to the following problem.

Problem Statement Develop a model that characterizes, in the long-run, the participants' intentions for allocating and performing queries so that one can evaluate if the system is meeting, or not, their intentions when it allocates them queries.

1.2 A Usual Characterization of Providers

Generally, providers are characterized according to their capacities to perform queries by defining the providers' load notion. Providers' load is usually defined in terms of number of queries that providers have in their run queue [ABKU99], but this approach assumes that queries and providers are homogeneous. That is, it assumes that providers have the same capacities to perform queries and that queries require the same computational resources to be treated by providers. Recently, *Roussopoulos and Baker* [RB06] defines providers' load in terms of their maximum capacity. The maximum capacity is a contract each provider advertises indicating the maximum number of queries a provider claims to

handle per time unit. The advantage of this proposal is that it is not affected by changes in the workload. However, this approach inherently assume that queries are homogeneous.

Unlike above approaches, we consider providers' and queries' heterogeneity by defining providers' load in terms of their current utilization. This is also known as the *capacity-based* approach. We formalize such a heterogeneity as follows. A provider has a finite *capacity* to perform queries, denoted by function cap whose values are greater than zero. The capacity of a provider denotes the number of computational units that it can have. Similarly, a query has a *cost*, $cost > 0$, that represents the computational units that the query consumes at a given provider. Let Q_p denote the set of queries that have been allocated to a provider p and that it has not already treated, i.e. the pending queries at p. The load of a provider p is defined as the cost sum of all queries in Q_p, formally $\sum_{q \in Q_p} cost_p(q)$. Thus, generally speaking, the *utilization* of a provider, $\mathcal{U}$, denotes its load with respect to its capacity. Formally, we define the utilization of p at time t as the computational units that set Q_p consumes at p (see Definition 12).

Definition 12. *Provider's Utilization*

$$\mathcal{U}_t(p) = \frac{\sum_{q \in Q_p} cost_p(q)}{cap(p)}$$

The provider's utilization values are in the interval $[0..\infty]$ because its load theoretically increases up to ∞. We say that a provider p is *overutilized* at time t if the cost sum of all queries in Q_p crosses its own capacity $cap(p)$, i.e. if $\mathcal{U}_t(p) > 1$.

1.3 Satisfaction Model

In this section, we go further in the characterization of a participant. We are interested in two more characteristics of participants that show how they perceive the system in which they interact : *adequation* and *satisfaction*. Such a characterization needs to use the memory of participants. While a consumer $c \in C$ tracks its k last issued queries in set IQ_c^k, recall that a provider $p \in P$ tracks the k last proposed queries to it in set PQ_p^k. It is worth noting that, because of autonomy, preserving the participants' intentions is quite important so that they stay in the system. At first glance, the system should satisfy participants in each interaction with them. However, this is simply not possible in reality, considering that a query is generally not allocated to all relevant providers. Furthermore, it is not because a single query allocation penalizes a participant's intention that it decides to leave the system. A participant generally considers the last queries to measure its happiness in the system and to evaluate if it should leave the system. A way to achieve this is to make a regular assessment over all their past interactions with the system, but participants have a limited memory capacity. Thus, they regularly assess only their k last interactions with the system. This is why we define the characteristics of participants over the k last interactions. Clearly, the k value may be different on each participant depending on its memory capacity. For simplicity, we assume they all use the same value of k. Also, we are interested in a third characteristic of providers : the *intention-based profit*. The intention-based profit allows a provider to evaluate if it is getting enough interesting queries to survive in the system. Besides these three characteristics (adequation, satisfaction, profit), we are interested in two characteristics of a query allocation mehtod : the *Allocation Efficiency with respect to a Consumer* and the *Allocation Efficiency with respect to a Provider*. Notice that these characteristics are only observable by the mediator.

In the following, we define above participants' characteristics in Sections 1.3.1, 1.3.2 and 1.3.3, and the query allocation method's characteristics in Section 1.3.4. Then, we conclude this section by given

some final remarks of the proposed model. Before presenting this, let us make two general remarks. First, the participant's characteristics may evolve with time, but for the sake of simplicity we do not introduce time in our notations. Second, the following presentation can be expressed with respect to participants' intentions (context-dependent and more dynamic data) or with respect to their preferences (context-independent and quite static data). However, applying the following characterization to intentions and preferences yields to different results, because the intentions of participants consider their context (such as their strategy and utilization) and their preferences do not. While in almost all information systems preferences tend to be private information, intentions tend to be public. Since we only intend to observe the system behavior, we develop the following definitions for intentions.

1.3.1 Participants' Adequation

From a general point of view, two kinds of adequation could be considered :

- The system adequation to a participant, δ_{sa}, e.g. a system where a provider (respectively consumer) cannot find any query (resp. provider) it desires is considered inadequate to such a participant.
- The participant's adequation to the system, δ_a, e.g. a provider (respectively consumer) that no consumer wants to deal with (resp. issuing queries that no provider intends to treat) is considered inadequate to the system.

While the first adequation notion only considers the information that a participant can obtain from the system, the adequation of a participant allows it to evaluate if other participants are in general interested in it. Let us illustrate both adequations via an example. Consider the case of the courier company of the Example 1, which is interested in its new international shipping service. A market place may be adequate to such a courier company because many consumers are interested in sending products abroad. But the courier company may be not adequate to the market place because its services are expensive and hence many consumers do not want to deal with it. Both adequation notions are needed to evaluate if it is possible for a participant to reach its goals in the system. A participant cannot know what the other participants think about it, except if it has a global knowledge of the system. Therefore, we consider the participant's adequation to the system as a global characteristic.

1.3.1.1 Consumer

The two kinds of adequation are intuitively useful to answer the following questions :

- "How well do the intentions of a consumer correspond to the providers that were able to deal with its last queries ?" – *System Adequation w.r.t. Consumer* – , and
- "How well do the last queries of a consumer correspond to the intentions of the providers that were able to deal with ?" – *Consumer Adequation.*

Let us first introduce the system's adequation concerning a consumer. The system *adequation* to a consumer characterizes the perception that the consumer has from the system. For example, in our motivating example given in the introduction of this document, $eWine$ considers the mediator as interesting (i.e. adequate), in such a query allocation, because it advertises providers that $eWine$ considers interesting : p_2, p_4, and p_5. Formally, we define the system adequation regarding a consumer $c \in C$ and concerning a query q, denoted by $\delta_{sa}(c, q)$, as the average of c's intentions towards set P_q (Equation 1.1). Its values are in the interval $[0..1]$.

$$\delta_{sa}(c, q) = \frac{1}{N_q} \cdot \sum_{p \in P_q} \left(\left(\overrightarrow{CI}_q[p] + 1 \right) \Big/ 2 \right) \tag{1.1}$$

We thus define the system adequation to a consumer as the average over the adequation values concerning its k last queries (see Definition 13). Recall that set IQ_c^k denotes the k last queries issued by consumer c. Its values are between 0 and 1, and the closer the value to 1, the more a consumer considers the system as adequate.

Definition 13. *System Adequation w.r.t. a Consumer*

$$\delta_{sa}(c) = \frac{1}{||IQ_c^k||} \cdot \sum_{q \in IQ_c^k} \delta_{sa}(c, q)$$

Conversely to Definition 13 that evaluates how much a consumer is interested in providers that can deal with its queries, the consumer's adequation to the system evaluates how much providers are interested in the queries of this consumer. Going back to our motivating example, we can say that $eWine$ is adequate to the system regarding query q since great part of providers desire to treat its query. According to this intuition, the adequation of a consumer c to the system concerning its interaction with the system for allocating its query q, noted $\delta_a(c, q)$, is defined as the average of the intentions shown by set P_q towards its query q (Equation 1.2). Its values are between 0 and 1. Vector $\overrightarrow{PI}_q$ denotes the P_q's intentions to perform q.

$$\delta_a(c, q) = \frac{1}{Nq} \cdot \sum_{p \in P_q} \left(\left(\overrightarrow{PI}_q[p] \ + 1 \right) \Big/ 2 \right) \tag{1.2}$$

Thus, we define the consumer's adequation to the system as the average over the δ_a values obtained in its k last queries. Its values are between 0 and 1. The closer the value to 1, the greater the *adequation* of a consumer to the system.

Definition 14. *Consumer Adequation*

$$\delta_a(c) = \frac{1}{||IQ_c^k||} \cdot \sum_{q \in IQ_c^k} \delta_a(c, q)$$

1.3.1.2 Provider

The two kinds of adequation concerning a provider are useful to answer the following questions :

- "How well do the intentions of a provider correspond to the last queries that the mediator has proposed to it ?" – *System Adequation w.r.t. Provider* – , and
- "How well does a provider correspond to the consumer's intentions ?" – *Provider Adequation*.

As for a previous section, in this section, we start by introducing the system adequation concerning a provider and then we define the provider's adequation. The system adequation w.r.t. a provider evaluates if the system corresponds to the intentions of a provider. Intuitively, this corresponds to a market survey, that is, what any enterprise does for evaluate the market where it desires to launch a product. Considering our motivating example, one can consider the mediator as adequate to p_1, p_3, and p_5, because $eWine$'s query is of their interest. However, it is difficult to conclude by considering only one query. An average over the k last interactions is more informative. Thus, we define the adequation of the system w.r.t. a provider $p \in P$, $\delta_{sa}(p)$, as the average of p's shown intentions towards set PQ_p^k. Remember that PQ_p^k is the set of k last proposed queries to p.

Definition 15. *System Adequation w.r.t. a Provider*

$$\delta_{sa}(p) = \begin{cases} \frac{1}{||PQ_p^k||} \cdot \sum_{q \in PQ_p^k} \left((\overrightarrow{PPI}_p[q] + 1) \Big/ 2 \right) \\ 0 \qquad\qquad if\ PQ_p^k = \emptyset \end{cases}$$

The values that this adequation can take are in the interval $[0..1]$. The closer the value is to 1, the greater the adequation of the system to a provider is. Now, the adequation of a provider to the system allows to evaluate if consumers are interested in interacting with it. To illustrate the *Provider Adequation*, we use again our motivating example. One may consider p_1 and p_3 as inadequate to the system (with regards to what they can perceive) since $eWine$ does not want to deal with. Nevertheless, the most important is to evaluate that interaction over set PQ_p^k of queries. So, we formally define the adequation of a provider $p \in P$ to the system over the last k proposed queries as follows.

Definition 16. *Provider Adequation*

$$\delta_a(p) = \begin{cases} \frac{1}{||PQ_p^k||} \cdot \sum_{q \in PQ_p^k} \left((\overrightarrow{CI}_q[p] + 1) \Big/ 2 \right) \\ 0 \qquad\qquad if\ PQ_p^k = \emptyset \end{cases}$$

Its values are in the interval $[0..1]$. The closer the value to 1, the greater the adequation of a provider to the system.

1.3.2 Participants' Satisfaction

This section is devoted to the characterization of the provider's happiness with the things it is doing in and receiving from the system. As for adequation, two kinds of satisfaction could be considered :

- The satisfaction of a participant with what it gets from the system, δ_s, e.g. a provider (respectively consumer) that receives queries (resp. results from the providers) it does not want is not satisfied.
- The participant's satisfaction with the job that the query allocation method does, δ_{as}, e.g. a provider (respectively consumer) that performs queries (resp. that gets providers) it does not want is not satisfied with the query allocation method whether there exist queries (resp. providers) of its interests that it does not get.

To illustrate both satisfactions, consider again the case of the courier company of the Example 1. This courier company may be dissatisfied, in a market place, because consumers are rarely interested in doing their shipments abroad and thus almost all queries it performs are requests for national shipments. Nevertheless, it is possible that this courier company is satisfied with the query allocation method because all the incoming queries requesting for international shipments are allocated to it. Both satisfaction notions may have a deep impact on the system, because participants may decide whether to stay or to leave the system based on them. While the first kind of satisfaction depends on the participants, the second one may be the result of the query allocation method design. Before presenting these notions, let us say that our satisfaction definitions do not directly consider either response times (for consumers) nor the number of performed queries (for providers). Instead, it is up to a participant to consider such aspects, if it is interested in, when computing their intentions.

1.3.2.1 Consumer

The characteristics we present here are useful to answer the following questions :

- "How far do the providers that have dealt with the last queries of a consumer meet its intentions ?" – *Consumer Satisfaction* – , and
- "Does the query allocation method propose the best providers (regarding consumers' intentions) to a consumer ?" – *Consumer Allocation Satisfaction*.

The *satisfaction* of a consumer allows this consumer to evaluate if a mediator is allocating its queries to the providers from which it desires to get results. To define the notion over its k last issued queries, we first define the satisfaction of a consumer concerning the allocation of a given query. The average of intentions expressed by a consumer to the providers that performed its query is an intuitive technique to define such a notion. Nevertheless, a simple average does not take into account the fact that a consumer may desire different results. Indeed, a consumer may desire to receive results from n different providers. Let us illustrate this using our motivating example. Assume that the mediator allocates $eWine$'s query only to p_2, to which $eWine$ has an intention of 1, but it was requiring two providers. A simple average would not take this into account. This is why the following equation takes this point into account using n instead of $||\widehat{P_q}||$, where $\widehat{P_q}$ denotes the set of providers that performed q.

$$\delta_s(c,q) = \frac{1}{n} \cdot \sum_{p \in \widehat{P_q}} \left((\overrightarrow{CI}_q[p] \ + 1) \Big/ 2 \right) \tag{1.3}$$

In above equation, parameter n stands for the number of required results by consumer c. The $\delta_s(c,q)$ values are in the interval $[0..1]$. The satisfaction of a consumer is then defined as the average over its obtained satisfactions concerning its k last queries. Its values are between 0 and 1. The closer the satisfaction to 1, the more the consumer is satisfied.

Definition 17. *Consumer Satisfaction*

$$\delta_s(c) = \frac{1}{||IQ_c^k||} \cdot \sum_{q \in IQ_c^k} \delta_s(c,q)$$

Since this notion of satisfaction does not consider the context, it does not allow to evaluate the efforts made by the query allocation method to satisfy a consumer. Let us illustrate this by means of our motivating example. Assume that $eWine$ has an intention of 1, 0.9, and 0.7 for allocating its query to p_2, p_4, and p_5, respectively. Now, suppose that the mediator allocates the query to p_4. Such a query allocation corresponds to $eWine$'s high intentions, so $eWine$ is satisfied. However, there is still a provider to which its *intention* is higher (p_2). The *Consumer Allocation Satisfaction* notion, denoted by $\delta_{as}(c)$, allows to evaluate how well the query allocation method works for a consumer. Its values are in the interval $[0..\infty]$.

Definition 18. *Consumer Allocation Satisfaction*

$$\delta_{as}(c) = \frac{1}{||IQ_c^k||} \cdot \sum_{q \in IQ_c^k} \frac{\delta_s(c,q)}{\delta_{sa}(c,q)}$$

If the obtained value is greater than 1, the consumer can conclude that the query allocation method acts to its favor. Conversely, if the value is smaller than 1, the query allocation method dissatisfies the consumer. Finally, a value equal to 1 means that the query allocation method is neutral.

1.3.2.2 Provider

Intuitively, it is through the characteristics we present in this section tha a provider may answer the following two questions :

- "How well do the last queries that a provider has treated meet its intentions ?" – *Provider Satisfaction* – , and
- "Does the query allocation method give the best queries (regarding providers' intentions) to a provider ?" – *Provider Allocation Satisfaction*.

Conversely to the adequation of a provider, its satisfaction only depends on the queries that it performs and is independent of the other queries that have been proposed to it. Let us exemplify the satisfaction notion using our motivating example. Suppose that in our example, the mediator allocates $eWine$' query to p_2. In such a query allocation, p_2 is not satisfied since it did not intend to perform the query. Thus, one can say that p_2 may quit the system by dissatisfaction, but it is possible that it does not do so because it receives in general interesting queries from the system. Indeed, what is more important for a provider is to be globally satisfied with the queries it performs. Let SQ_p^k (with $SQ_p^k \subseteq PQ_p^k$) denote the set of queries that provider p performed among the set of proposed queries (PQ_p^k). We define the satisfaction of a provider $p \in P$ as its intention average over set SQ_p^k (see Definition 19). The $\delta_s(p)$ values are between 0 and 1. The closer the value to 1, the greater the *satisfaction* of a provider.

Definition 19. *Provider Satisfaction*

$$\delta_s(p) = \begin{cases} \frac{1}{||SQ_p^k||} \cdot \sum_{q \in SQ_p^k} \left(\left(\overrightarrow{PPI}_p[q] + 1 \right) \Big/ 2 \right) & \\ 0 & if\ SQ_p^k = \emptyset \end{cases}$$

The satisfaction notion evaluates whether the system is giving queries to a provider according to its (those of the provider) intentions so that it fulfills its objectives. So, as for consumers, a provider is simply not satisfied when it does not get what it expects. Here again, there are different reasons for this. First, it may be because the system does not have interesting resources, i.e. the system has a low adequation w.r.t. the provider. Second, the query allocation method may go against the provider's intention. The latter is measured by the *allocation satisfaction* notion. In other words, by means of this notion a provider can evaluate how well the query allocation method works for it. Conversely to a consumer that always receives results at each interaction, a provider is not allocated all the proposed queries. So the formal definition is a little different. We formally define the allocation satisfaction notion of a provider $p \in P$, denoted by $\delta_{as}(p)$, as the ratio of its satisfaction to the adequation that the system has towards it. Resulting values are between 0 and ∞.

Definition 20. *Provider Allocation Satisfaction*

$$\delta_{as}(p) = \frac{\delta_s(p)}{\delta_a(p)}$$

If the allocation satisfaction of a provider p is greater than 1, the query allocation method works well for p (from the point of view of p). If the value is smaller than 1, the closer it is to zero, the more p is dissatisfied with the query allocation method. Finally, a value equal to 1 means the query allocation method is neutral.

1.3.3 Provider Intention-based Profit

We strongly believe that in addition to evenly distribute query load among provider at some time t, a query allocation method should also be fair (regarding the providers' utilization) in the long-run. That is, with all other parameters being equal, providers should have, in average, almost the same utilization in

some discrete time interval. Furthermore, we believe that an autonomous provider considers its intentions towards the queries it performed in such a discrete time interval. This is clearly illustrated by an e-commerce application where a provider may be in starvation, whatever the number of queries it received, because it simply does not obtain much benefits from the queries it obtained in a given time interval. This is why we make precise, in this section, what *query starvation* means in distributed information systems with autonomus providers. To this end, we introduce the *Intention-based Profit*, π, definition. The intention-based benefit of a provider denotes the sum of the intentions it expressed towards a set of queries that it performed in a given discrete time interval. Let $date_p(q)$ denote the date in which a query q has been performed by provider p, we formally define the Intention-based Profit of a provider in a time interval $[t', t]$, with $t' < t$, as follows.

Definition 21. *Provider's Intention-based Profit*

$$\pi_p(t', t) = \sum_{\substack{q \in SQ_p^k \\ date(q) \in [t', t]}} \overrightarrow{PPI}_p[q]$$

Its values are in the interval of $-\infty$ and $+\infty$. Recall that set SQ_p^k denotes the set of queries that provider p performed among the set of k last queries that the mediator proposed to it. Then, let St_p be the minimal intention-based profit that a provider p can support, we say that p is in starvation in a discrete time interval $[t', t]$ if and only if,

$$starv(p) = \pi_p(t', t) < St_p \tag{1.4}$$

Notice that a consumer may also suffer from starvation : in those cases that it does not receive the number of answers it requires, i.e. when $\widehat{P_q} < n$. We do not introduce a starvation notion for a consumer because we already considered this in the definition of its satisfaction (see Definition 17).

1.3.4 Query Allocation Method Efficiency

Having formally defined the participants' characteristics (adequation, satisfaction, and profit), we proceed to introduce the efficiency notion, which is a characteristic of the query allocation method. Intuitively, this characteristic allows to answer the following two questions :

- "How well does the query allocation method perform regarding a consumer ?" – *Allocation Efficiency w.r.t. a Consumer*, and
- "How well does the query allocation method perform concerning a provider ?" – *Allocation Efficiency w.r.t. a Provider*.

1.3.4.1 Consumer

The *query allocation efficiency w.r.t. a consumer* $c \in C$, $\delta_{ae}(c)$, is then defined as in Definition 22. Its values are between 0 and ∞. As for the *allocation satisfaction* notion, the *query allocation efficiency w.r.t. a consumer* allows to evaluate the job done by the query allocation method for a consumer. But, this evaluation is objective since it considers the consumer's adequation to the system in addition to the system's adequation to the consumer.

Definition 22. *Allocation Efficiency w.r.t. a Consumer*

$$\delta_{ae}(c) = \frac{1}{||IQ_c^k||} \cdot \sum_{q \in IQ_c^k} \frac{\delta_s(c, q)}{\delta_{sa}(c, q) \cdot \delta_a(c, q)}$$

If the efficiency value of the query allocation regarding a consumer is greater than 1, the query allocation method does a good job for it. In contrast, if the value is smaller than 1, the query allocation method does not do a good job for it.

1.3.4.2 Provider

We then define the efficiency of the query allocation regarding a provider $p \in P$, denoted by the function $\delta_{ae}(p)$, as the ratio of its satisfaction to the product of the adequation that the system has towards it by its adequation. Its values are in $[0..\infty]$.

Definition 23. *Allocation Efficiency w.r.t. a Provider*

$$\delta_{ae}(p) = \frac{\delta_s(p)}{\delta_{sa}(p) \cdot \delta_a(p)}$$

As for a consumer, if the efficiency value of the query allocation with regards to a provider is greater than 1, the query allocation method does a good job for it. If the value is smaller than 1, the efficiency of the query allocation is not good. And, in the case the value is 1, the query allocation method is neutral to the provider.

1.3.5 Discussion

The model we presented in this section can be applied with different purposes. First, to evaluate how well a query allocation method satisfies the participants' intentions. Second, to try to explain the reasons of the participants' departures from the system. For example, to know if they are leaving the system because *(i)* they are dissatisfied with the queries they perform, *(ii)* they are dissatisfied with the mediator's job, or *(iii)* the system is simply inadequate to them. To do so, one has to apply the system measures, which reflect a global behavior, over all concepts of the model : *adequation*, *satisfaction*, and *allocation efficiency* (see Section 1.4). Third, to design new self-adaptable query allocation methods that meet the participants' intentions in the long-run (see Chapter 2).

As noted earlier, even if the model can be applied to the *preferences* and *intentions* of participants, the interpretation of results is not the same. Thus, two different levels of *satisfaction* exist : at the *preferences*' and *intentions*' level. On the one hand, the *satisfaction* at the *preferences*' level reflects the happiness of a participant with what it is doing in the system. On the other hand, it is with the *satisfaction* at the *intentions*' level that a participant evaluates if the mediator generally gives to it the queries it asks for. Thus, a participant can know if it is properly computing its *intentions* by evaluating both *satisfactions*. For instance, a participant can observe that its expressed *intentions* do not allow it to be satisfied at its *preferences*' level even if the mediator does a good job for it and then it is satisfied at its *intentions*' level.

As final remark, reputation does not directly appear, but it is clear that it has a major role to play in the manner that participants work out their *intentions*. Thus, it is taken into account as much as participants consider it important. Moreover, notice that several possibilities to compute participants' satisfaction may exist. For example, participants' satisfaction may decrease with the time or consider the number of received queries. However, to explore, explain, and compare all the possibilities to compute participants' satisfaction is well beyond the scope of this thesis. In fact, such a study is an open problem and could be the topic of a new doctoral dissertation.

1.4 System Measures

The system measures we use are the same for consumers and providers, and can be used to evaluate the δ_{sa}, δ_a, δ_s, δ_{as}, δ_{ae}, and $\mathcal{U}_t$ values of a participant. Thus, for simplicity, the g function denotes one of these functions and S denotes either a set of consumers or providers, i.e. $S \subseteq C$ or $S \subseteq P$. To better evaluate the quality of a query allocation method for balancing queries, one should reflect :

- the effort that a query allocation method does for either maximizing or minimizing a set S of g values – *efficiency* – ,
- any change in a set S of g values – *sensitivity* – , and
- the distance from the minimal value to the maximal one in a set S of g values – *balance* – .

A well-known measure that reflects the *efficiency* of a query allocation method is the *mean* μ function. Because participants' characteristics (see Section 1.3) are additive values and may take zero values, we utilize the arithmetic mean to obtain this representative number (Equation 1.5).

$$\mu(g, S) = \frac{1}{||S||} \sum_{s \in S} g(s) \tag{1.5}$$

However, the *mean* measure might be severely affected by extreme values. Thus, we must reflect the g values' fluctuations in S, i.e. the *sensitivity* of a query allocation method. In other words, we evaluate how *fair* a query allocation method is w.r.t. a set S of g values. An appropriate measure to do so is the *fairness index* f proposed in [JCH84] (defined in Equation 1.6). Its values are between 0 and 1.

$$f(g, S) = \frac{\left(\sum_{s \in S} g(s)\right)^2}{||S|| \sum_{s \in S} g(s)^2} \tag{1.6}$$

Intuitively, the greater the *fairness* value of a set S of g values, the fairer the query allocation process with respect to such values. To illustrate the *sensitivity* property, suppose that there exist two competitive mediators m and m' in our motivating example. Assume, then, that the set of providers registered to m and m' are $P = \{p_1, p_2, p_3\}$ and $P' = \{p'_1, p'_2, p'_3\}$, respectively. Now, consider that the *satisfaction* of such providers are $\delta_s(p_1) = 0.2$, $\delta_s(p_2) = 1$, $\delta_s(p_3) = 0.6$, $\delta_s(p'_1) = 1$, $\delta_s(p'_2) = 0.7$, and $\delta_s(p'_3) = 0.9$. Reflecting the *sensitivity* of both mediators w.r.t. *satisfaction* (0.77 and 0.97 for m and m' respectively), we can observe that companies have almost the same chances of doing business in m', which is not the case in m.

Finally, a traditional measure that reflects the ensured *balance* by a query allocation method is the *Min-Max* ratio. The *Min-Max* ratio σ is defined in Equation 1.7 (where $c_0 > 0$ is some fixed constant). Its values are between 0 and 1. The greater the *balance* value of a set S of g values, the better the *balance* of such values. The *Min-Max* ratio is useful to know whether there exists a great different between the most satisfied entity $s \in S$ and the less satisfied entity $s' \in S$ (with $s \neq s'$), and then, one can evaluate if this is because of the query allocation method or the entity's *adequation*.

$$\sigma(g, S) = \frac{\min_{s \in S} g(s) + c_0}{\max_{s' \in S} g(s') + c_0} \tag{1.7}$$

The above three measures are complementary to evaluate the global behavior of the system, and the use of only one of them may cause the loss of some important information.

1.5 Related Work

To the best of our knowledge, economic models are the only ones that are related to the model we proposed in this chapter. Economics is a social science that studies how individuals, firms, governments, and organizations make choices ; and how these choices determine the way wealth is produced and distributed. It is subdivided into macroeconomics and microeconomics. Macroeconomics studies aggregated indicators to understand how the whole economy functions [Bla85]. In other words, it deals with the performance, structure, and behavior of a national or regional economy as a whole. Macroeconomic models and their forecasts are used by both governments and large corporations to assist in the development and evaluation of business strategy. Microeconomics [MCWG95, Kre90], which examines how individual decisions and behaviors affect the supply and demand of goods and services, which determines prices. In other words, it studies how individuals make decisions to allocate limited resources. Indeed, in this work we are interested in the participants themselves and for this reason we focus in microeconomics.

In microeconomics, one describes participants' preferences by means of a *utility* function. A utility function assigns a numerical value to each element of a set of choices, ranking such elements according to the participants' preferences [MCWG95]. That is, for each query (good or service) a participant computes its *marginal utility* of participating in the allocation of such a query. Notice that, in our case, the participants' intentions represent somehow their marginal utility. Then, a participant computes its *total utility* gained in a given set of queries by adding its marginal utility gained in each query. For simplicity, in the remainder of this section, we only use the term utility to denote total utility. In other words, as the satisfaction notion, the utility is an abstract concept that measures the happiness or gratification of participants by consuming or performing queries. Furthermore, the utility as well as the satisfaction makes no assumption about the way in which participants compute their marginal utility function and intention function, respectively. This is because both marginal utility and intention functions depend on applications and participants. We go beyond this by proposing a way in which participants can compute their intentions. For all this, utility is clearly related to the notion of satisfaction we presented in this chapter, but the satisfaction notion differs from the total utility in three ways. First, the satisfaction is bounded by 0 and 1 and normalized while the utility is neither bounded nor normalized. Therefore, one can easily compare the satisfaction of participants. Second, while utility generally considers all the queries that a participant consumed or performed, satisfaction only considers the k last queries. This is very useful when participants have a limited capacity. Finally, utility is generally reduced to monetary concerns only, which is not the case for the satisfaction notion.

We now introduce two well-known economic properties : Pareto-optimality and Nash equilibrium. First, Pareto-optimality is a situation which exists when resources have been allocated in such a way that no-participant can be made better off without sacrificing the well-being of at least one participant. Otherwise, we say that there exists a Pareto improvement. Thus, a query allocation is said to be Pareto optimal when no further Pareto improvements can be made. However, it is not obvious to satisfy participants since several Pareto solutions may exist. Second, in game theory [vNM44] the Nash-equilibrium [Nas51] is a condition in which no participant would want to change its strategy given the strategies adopted by other participants. Participants are said to be in equilibrium if a change in strategies by any one of them would lead that participant to earn less than if it remained with its current strategy. As for the Pareto-optimality property, several Nash-equilibriums may exist, but also it may not exist a Nash-equilibrium in some cases. Furthermore, as most of the economic properties, both Pareto-optimality and Nash-equilibrium properties focus on only one interaction. As a result, most of the economic approaches [FNSY96], which are based on one of these economic properties, look for the happiness of participants in solely one query allocation and not in the long-run. In contrast, the satisfaction notion we proposed represents the happi-

ness of participants in several interactions, i.e. in the long-run.

In the field of distributed rational decision making [San99], participants are assumed to be *individually rational* : the utility of any participant in the process is no less than the utility it would have by not participating. This is not relevant in environments where participants may have the interest that the system be efficient and hence, in some query allocations, they may be interested in participating in some query allocations even if this means to lose sometimes. Furthermore, it is not relevant in cooperative contexts where some participants may be imposed, which implies having a lower utility in participating. Therefore, the satisfaction notion is still relevant because it is a long-run notion.

Qu et al. [QLM06] propose a definition of consumer's satisfaction, called the *User Satisfaction Metric* (USM), that is quite related to ours. They define USM of a given consumer as the sum average of the consumer's satisfaction in each query it has issued. Nevertheless, unlike our consumer's satisfaction definition, their USM definition assumes that consumers are only interested in response times and information freshness. Indeed, this is very important for consumers in some applications, such as in web-database systems, but in other applications consumers may be interested in some other criteria, such as providers' reputation. Our satisfaction definitions are more general by computing participants' satisfaction with respect to their intentions, which are individually computed by participants considering their own preferences.

1.6 Chapter Summary

In this chapter, we addressed the problem of modeling autonomous participants with special interests towards queries. We proposed a model that defines long-run notions to know if the system is adequate to participants and if it is meeting their intentions. To the best of our knowledge, this model is the first effort to characterize participants in their generality. Summarizing, our main contributions in this chapter are the following.

- We characterized the participants' intentions in a new model, which allows to evaluate a system from a satisfaction point of view. The definitions that we proposed are original, considering the long-run notions of *adequation* and *satisfaction*. They are independent of the way participants compute their intentions and how the mediator considers them. This model facilitates the evaluation and the design of query allocation methods for these environments. The proposed model is general, and thus, can be used for any distributed system architecture.
- We defined the provider's *intention-based profit* notion, which allows a provider to evaluate if it performed the required providers to survive. In particular, we made precise the query starvation notion in environments where providers have some preferences towards queries.
- We proposed three different measures to evaluate the quality of query allocation methods : the *mean* measure reflects the effort that a query allocation method does for equally either maximizing or minimizing a given set of values ; the *fairness* measure evaluates how fair a query allocation method is ; the *balance* measure measures the Min-Max values.

Future Work In this chapter, we proposed a model that defines many notions to characterize, in the long-run, participants' intentions. One of these notions is participants' satisfaction that, generally speaking, denotes the happiness of a participant with the things it obtains from the system. We presented a way to compute participants' satisfaction, but several possibilities to do so may exist. This requires a depth study to explore different satisfaction definitions, which is out of the scope of this thesis. Thus, in a future work, we plan to explore a large number of possibilities to compute participants' satisfaction so that we

can understand the advantages and disadvantages of each of them. The satisfaction notion we presented in this chapter is somehow linked to the notions of trust [AD01, AG07] and reputation [KSGM03] from distributed systems. This study is well beyond the scope of this thesis, thus we report it to a future work. Recently, we observed that, from a general point of view, sociology [Mac04] focus on a similar problem to that we focused on, i.e. we model the satisfaction of participants. We are interested in exploring this discipline in order to study the possible links that could exist between its properties and the properties we presented in this chapter.

CHAPTER 2

Satisfaction-based Query Allocation

One of the problems that has been thoroughly investigated in the area of query allocation is *query load balancing* (*qlb*) [ABKU99, GBGM04, MTS90, RM95, SKS92], which main objective is to maximize overall system performance (specifically throughput and response times) by balancing query load among providers. Nevertheless, as seen so far (Example 1), in distributed information systems where participants are autonomous the participants' intentions are not only performance related. In such environments, when a participant is no longer satisfied with the mediator, the only way to express its dissatisfaction is to leave it, which may have consequences on the capacity and functionalities provided by the system. On the one hand, a provider departure decreases the system's *capacity* (which denotes the aggregate of all providers' capacity) and, as a result, may significantly hurt the system performance. On the other hand, providers' departure can result in losing in system's functionalities, but also consumers' departure is a loss of queries for providers. Therefore, to preserve full system capacity and functionalities in these environments, it is quite important to take into account participants' intentions in addition to *qlb*. This is particularly timely with the potential profusion of software based on web services in particular and on services oriented to architecture in general.

In this chapter, we propose a query allocation framework that considers participants' intentions besides *qlb*. The content of this chapter is based on our material published in [LQRV07, QRLV06, QRLV07a, QRLV07b]. Our main contributions are the following :

- We propose *Satisfaction-based Query Load Balancing* (S_bQA, in short), a flexible framework with *self-adapting* algorithms to allocate queries while considering both *qlb* and participants' intentions. Salient features of S_bQA are that :
  - it affords consumers the flexibility to trade their preferences for the providers' reputation,
  - it affords providers the flexibility to trade their preferences for their utilization,
  - it allows a mediator to trade consumers' intentions for providers' intentions, and
  - it affords the mediator the flexibility to adapt the query allocation process to the application by varying several parameters.
- We demonstrate, through experimental validation, that S_bQA significantly outperforms baseline methods, the $Capacity\ based$ and *Mariposa-like* methods, and yields significant performance benefits. We demonstrate the self-adaptability of S_bQA to participants' intentions and its adaptability to different kinds of application. We also show that applying the proposed measures over the provided model allows the prediction of possible departures of participants.

The remainder of this chapter is organized as follows. In Section 2.1, we formally define the query allocation problem in systems with autonomous participants having special interests towards queries. As part of the S_bQA framework, we define a way to compute consumers' and providers' intentions in Sections 2.2 and 2.3, respectively. Then, in Section 2.4, we define a way to allocate queries by considering

both consumers' and providers' intentions and define a strategy to adapt query allocation to different kinds of application. We give some final remarks on S_bQA in Section 2.5. In Section 2.6, we validate S_bQA performance to allocate queries and demonstrate its adaptability to participants' intentions and applications. We then survey related work in Section 2.7. Finally, we conclude this chapter in Section 2.8.

2.1 Problem Definition

We consider a system consisting of a mediator m, of a set of consumers C and of a set of providers P. These sets are not necessary disjoint, an entity may play more than one role. Provider are heterogeneous : *(i)* they have different processing capabilities and *(ii)* they may provide different results, e.g. because they have different private data. We assume that providers compute their utilization $\mathcal{U}$ as defined in Section 1.2.

Queries are formulated in a format abstracted as a triple $q =< c, d, n >$ such that $q.c \in C$ is the identifier of the consumer that has issued the query, $q.d$ is the description of the task to be done (e.g. a SQL statement), and $q.n \in \mathbb{N}^*$ is the number of providers to which the consumer wishes to allocate its query. Indeed, a consumer may want to query different providers, in particular in the case they can provide different answers. Parameter $q.d$ is intended to be used within a matchmaking procedure to find the set of providers that are able to treat q, denoted by set P_q. As noted so far, such techniques are out of the scope of this paper and thus we assume there exists one in the system, e.g. [KH95, LH04], that is sound and complete : it does not return false positive nor false negatives. We use N_q for denoting $||P_q||$, or simply N when there is no ambiguity on q.

Consumers send their queries to mediator m that allocates each incoming query q to $q.n$ providers in P_q. And, a consumer poses a query to a mediator when it cannot locally perform the query or just because it has certain benefits by outsourcing the query. We only consider the arrival of *feasible queries*, that is those queries in which there exists at least one provider, which is able to perform them, in the system. For the sake of simplicity we only use, throughout this paper, the term "query" to denote a feasible query. Query allocation of some query q among the providers in P_q is a vector $All\overrightarrow{oc}_q$, or simply $All\overrightarrow{oc}$ when there is no ambiguity on q,

$$\forall p \in P_q,\ All\overrightarrow{oc}_q[p] = \begin{cases} 1 & \text{if } p \text{ gets } q \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

We assume that each incoming query q must be treated, even if no provider desires to perform it. This leads to $\sum_{p \in P_q} All\overrightarrow{oc}[p] = min(q.n,\ N)$. In other words, in the case that $q.n > N_q$, consumer $q.c$ gets N_q queries instead of $q.n$. In the following, the set of providers such that $All\overrightarrow{oc}_q[p] = 1$ is noted $\widehat{P_q^r}$, where, given a predefined scoring function, r denotes the worst provider's score in set $\widehat{P_q^r}$. For simplicity, when the knowledge of r is not required, we only use $\widehat{P_q}$ to denote $\widehat{P_q^r}$. Notice that, without any loss of generality, in some cases, e.g. when consumers pay services with real money, query allocation just means that providers are selected for participating in a negotiation process with consumers.

Participants are free to express their intentions to allocate and perform queries. The way in which participants compute their intentions is considered as private and hence it is not revealed to others, which is the case of several current applications (e.g. in an *e-commerce* scenario, enterprises do not reveal their business strategies). However, even if this information is private, the way in which participant compute their intentions has an indirect impact on the system's behavior. For instance, if participants are interested in short response times, as a result the system will ensure low response times. But, if they are not interested in, the resulting system will have a poor performance.

In these environments, where participants are autonomous, it is crucial to consider their intentions when allocating queries to avoid massive participants' departure from the system and hence to preserve the total system capacity, i.e. the aggregate capacity of all providers (e.g. in terms of computational or physical resources). To summarize, we can state the problem as follows.

Query Allocation Problem Given a mediator m confronted to autonomous participants, m should allocate each incoming query q to a set $\widehat{P_q}$ such that $||\widehat{P_q}|| = min(q.n, N_q)$, short response times, system capacity, and participants' satisfaction are ensured in the long long-run.

2.2 Consumer's Side

When a consumer is required by the mediator to give its intention for allocating its query q to a given provider p, it computes its intention based on both its preferences towards p and p's reputation. The idea is that a consumer makes a balance between its preferences for allocating queries and the providers' *reputation*, in accordance to its past experience with providers. For example, if a consumer does not have any past experience with a provider p, it pays more attention to the reputation of p. A consumer may base its preferences on different criterias, such as quality of service, response times or price of services. Hence, several ways to compute preferences exist. Dealing with the way in which a consumer obtains its preferences is beyond the scope of this thesis.

We formally define the intention of a consumer $c \in C$ to allocate its query q to a given provider $p \in P_q$ as in Definition 24. Function $prf_c(q,p)$ gives c's preference (which may denote e.g. some interest to *quality of service* or *response time*) for allocating q to p, and function $rep(p)$ gives the reputation of p. Values of both functions (prf and rep) are in the interval $[-1..1]$.

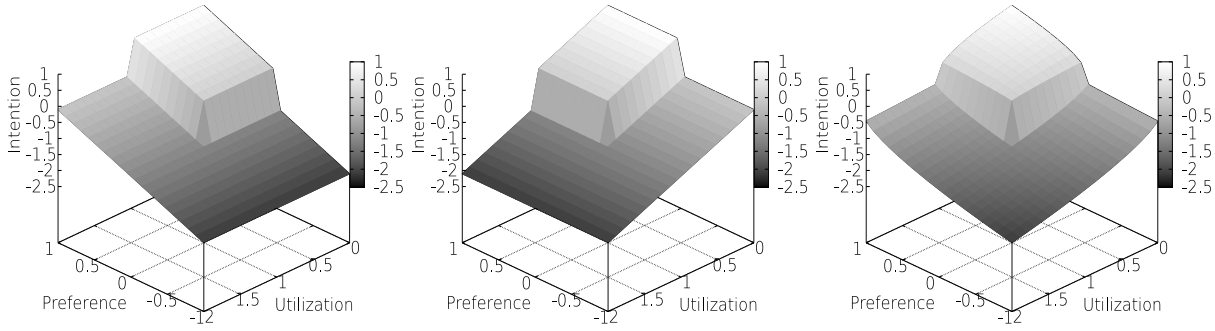
Definition 24. *Consumer's Intention*

$$ci_c(q,p) = \begin{cases} prf_c(q,p)^{\upsilon} \times rep(p)^{1-\upsilon} & if\ prf_c(q,p) \geq 0 \wedge\ rep(p) \geq 0 \\ -\Big(\big(1-((prf_c(q,p)+1)/2)\big)^{\upsilon} \times \big(1-((rep(p)+1)/2)\big)^{1-\upsilon}\Big) & elseif\ prf_c(q,p) < 1 \wedge \\ & \wedge\ rep(p) < 1 \\ -\Big(\big(1-((prf_c(q,p)+1)/2)+\epsilon\big)^{\upsilon} \times \big(1-((rep(p)+1)/2)+\epsilon\big)^{1-\upsilon}\Big) & else \end{cases}$$

Parameter $\epsilon > 0$, usually set to 0.01, prevents the consumer's *intention* from taking zero values when the consumer's preference or provider's reputation values is equal to 1. Parameter $\upsilon \in [0..1]$ ensures a balance between the consumer's preferences and the providers' reputation. In particular, if $\upsilon = 1$ (resp. 0) the consumer only takes into account its preferences (resp. the provider's reputation) to allocate its query. So, if a consumer has enough experience with a given provider p, it sets $\upsilon > 0.5$, or else it sets $\upsilon < 0.5$. When $\upsilon = 0.5$, it means that a consumer gives the same importance to its preferences and the provider's reputation.

2.3 Provider's Side

The provider's intention to perform a given query is based on its preferences for performing such a query and its current utilization. Nonetheless, the question that arises is : *what is more important for a provider, its preferences or its utilization ?* We propose to balance, on the fly, the preferences and utilization of a provider according to its satisfaction. Intuitively, on the one hand, if a provider is satisfied, it can then accept sometimes queries that do not meet its intentions. On the other hand, if a provider is



(a) when provider's *satisfaction* is 0 (b) when provider's *satisfaction* is 1 (c) when provider's *satisfaction* is 0.5

Figure 2.1 – Tradeoff between *preference* and *utilization* for getting providers' *intention*.

dissatisfied, it does not pay so much attention to its utilization and focuses on its preferences so as to obtain queries that meet its intentions. To do so, the satisfaction it uses to make the balance has to be based on its preferences and not on its intentions. Thus, the satisfaction definition of Section 1.3.2.2 has to be adapted to the preferences of a provider by using its *preferences* instead of its intentions. As for a consumer, a provider may compute its preferences either by considering its context or independently of its context. For example, a provider may no longer desire to perform some kind of queries when it is overutilized and another provider may always have the same preferences for queries no matter its utilization. In fact, several strategies can be adopted by a provider to compute its preferences. However, how a provider implements its preference's function, prf, is out of scope of this paper. We just assume that providers' preferences are in the interval $[-1..1]$.

We thus define the intention of a provider $p \in P_q$ to deal with a given query q as in Definition 25. Function $prf_p(q) \in [-1..1]$ gives p's preference to perform q.

Definition 25. *Provider's Intention*

$$pi_p(q) = \begin{cases} prf_p(q)^{1-\delta_s(p)} \times (1-\mathcal{U}_t(p))^{\delta_s(p)}, & if\ prf_p(q) \geq 0 \wedge \mathcal{U}_t(p) \leq 1 \\ -\Big(\big(1-((prf_p(q)+1)/2)\big)^{1-\delta_s(p)} \times \big(\mathcal{U}_t(p)\big)^{\delta_s(p)}\Big) & else\,if\ prf_p(q) < 1 \wedge \mathcal{U}_t(p) > 0 \\ -\Big(\big(1-((prf_p(q)+1)/2)+\epsilon\big)^{1-\delta_s(p)} \times \big(\mathcal{U}_t(p)+\epsilon\big)^{\delta_s(p)}\Big) & else \end{cases}$$

Parameter $\epsilon > 0$, usually set to 0.01, prevents the intention of a provider from taking 0 values when its preference or its utilization is equal to 0. Figure 2.1 illustrates the behavior that function pi takes for different provider's satisfaction values. We can observe in Figure 2.1(a) that when a provider is not satisfied at all, its utilization has no importance for it and its preferences denote its intentions. In contrast, when a provider is completely satisfied, its utilization denotes its intentions (see Figure 2.1(b)). In the case that a provider has a satisfaction of 0.5 (Figure 2.1(c)), we observe that its preferences and utilization have the same importance for it. Moreover, we can observe in Figure 2.1 that a provider shows positive intentions, whatever its satisfaction is, only when it is not overutilized and queries are of its interests. This helps satisfying providers while keeping good response times in the system.

2.4 Mediator's Side

So far, we assumed that a matchmaking technique has found the set of providers that are able to deal with a query q, denoted by set P_q. Therefore, we only focus on the allocation of q among set P_q.

Given a query q, S_bQA allows the mediator to trade consumers' intentions for providers' intentions according to their satisfaction (Section 2.4.1). Furthermore, S_bQA affords the mediator the flexibility to regulate the system w.r.t. some predefined function and adapt the query allocation process to the application by varying its parameters (Section 2.4.2). In Section 2.4.3, we describe the way in which S_bQA allocates queries among providers and analyze, in Section 2.4.4, the number of messages that the mediator transfers over the network to allocate an incoming query q.

2.4.1 Scoring and Ranking Providers

A natural way to perform query allocation is to allocate queries in a consumer-centric fashion, such as several e-commerce applications do. This leads to take into account the consumers' intentions only, which may seems correct at first glance. However, doing so may severely penalize providers' intentions and hence it may cause their departure from the mediator, which implies a loss of capacity and functionality of the system but also a loss of revenues for the mediator when it is paid by providers after each transaction (e.g. in ebay sellers pay a percent of the transactions they conclude). Respectively, if a mediator only considers the providers' intentions when allocating queries, consumers may quit the mediator by dissatisfaction, which in turn may cause the departure of providers. This is why we decide to balance consumers' and providers' intentions with the aim that both of them be satisfied.

Thus, given a query q, a provider is scored by considering both its intention for performing q and $q.c$'s intention for allocating q to it. That is, the *score* of a provider $p \in P_q$ regarding a given query q is defined as the balance between the $q.c$'s and p's intentions (see Definition 26).

Definition 26. *Provider's Score*

$$scr_q(p) = \begin{cases} \left(\overrightarrow{PI}_q[p]\right)^{\omega}\left(\overrightarrow{CI}_q[p]\right)^{1-\omega} & if\ \overrightarrow{PI}_q[p] > 0 \wedge \overrightarrow{CI}_q[p] > 0 \\ -\Big(\left(1-\overrightarrow{PI}_q[p]+\epsilon\right)^{\omega}\left(1-\overrightarrow{CI}_q[p]+\epsilon\right)^{1-\omega}\Big) & else \end{cases}$$

Vector $\overrightarrow{PI}_q[p]$ denotes P_q's *intentions* to perform q. Parameter $\epsilon > 0$, usually set to 1, prevents the provider's *score* from taking 0 values when the consumer or provider's *intention* is equal to 1. Parameter $\omega \in [0..1]$ ensures a balance between the consumer's *intention* for allocating its query and the provider's *intention* for performing such a query. In other words, it reflects the importance that the query allocation method gives to the consumer and providers' *intentions*. To guarantee equity at all levels, such a balance should be done in accordance to the consumer and providers' satisfaction. That is, if the consumer is more satisfied than the provider, then the query allocation method should pay more attention to the provider's intentions. Thus, we compute the ω value as in Equation 2.2. Conversely to provider's intention, the query allocation module has not access to private information. Thus, the satisfaction it uses must be based on the intentions.

$$\omega = \Big(\big(\delta_s(c) - \delta_s(p)\big) + 1\Big) \Big/ 2 \tag{2.2}$$

Figure 2.2 illustrates the tradeoff between the consumer and provider' intention for obtaining the ω value. One can also set ω's value according to the kind of application. For instance, if providers are cooperative (i.e. not *selfish*) and the most important is to ensure the quality of results, one can set ω near or equal to 0. Finally, providers are ranked from the best to the worst scored, the $\overrightarrow{R}_q$ vector. Intuitively, $\overrightarrow{R}_q[1]$ is the best scored provider to deal with q, $\overrightarrow{R}_q[2]$ the second, and so on up to $\overrightarrow{R}_q[N]$ which is the worst. As a result, if $q.n <= N$ the $q.n$ best ranked providers are selected, or else all the N providers are selected.

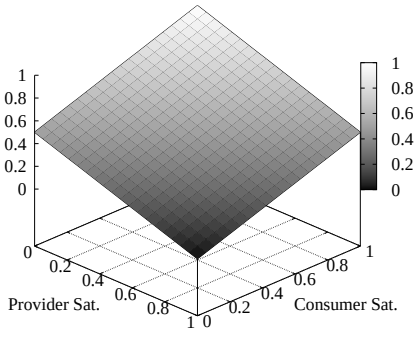


Figure 2.2 – The values that ω can take.

2.4.2 Regulating the System

The mediator can proceed to allocate queries by considering only the providers' ranking based on their score ($\overrightarrow{R}$), which affords participants to take the control of the query allocation process. However, the mediator may have certain objectives or goals that it aims to achieve. It is possible that the mediator wants to regulate the system regarding some predefined function τ, e.g. to ensure good response times to consumers. To allow this, we propose the K_nBest strategy and assume that the mediator applies it to allocate queries. K_nBest is inspired by the *two random choices (TRC)* paradigm [Mit01, ABKU99]. The idea is that, given a query q, the mediator selects a set K_n of k_n providers that either maximize or minimize function τ from set K, where set K is a random selection of k' providers from set P_q of providers. We can indifferently assume that k' and k_n values are predefined by the administrator or defined on the fly by the mediator. Then, it allocates q to the $q.n$ best ranked providers among set K_n of providers. We explain further the query allocation principle in Section 2.4.3. We assume, without any loss of generality, that function τ denotes function $\mathcal{U}$, which means that the mediator strives to regulate the system with respect to providers' utilization (i.e. to perform *qlb*). The following theorem summarizes the K_nBest's properties that bound its behavior.

Theorem 2. *Given a query q, the behavior of a query allocation method using K_nBest is bounded by the following properties,*

(i) if $k' = 2q.n \wedge k_n = q.n$, K_nBest has a TRC behavior.
(ii) if $k' = N_q \wedge k_n = q.n$, K_nBest has a $Capacity\ based$ behavior.
(iii) if $k' = N_q \wedge k_n = k'$, K_nBest has an $Intention\ based$ behavior.

Proof. Say a query allocation method qa implements the K_nBest strategy. The following is the same for any value that parameter $q.n$ can take.

Consider that qa sets $k' = 2q.n \wedge k_n = q.n$. In this case, qa allocates a query q to the less utilized provider $p \in P$ among a set of $2q.n$ random selected providers from P_q. This leads to satisfy the below equation,

$$\forall p \in \widehat{P_q}, \nexists p' \in K \backslash \widehat{P_q} : \mathcal{U}_{(p')} < \mathcal{U}_{(p)}$$

which is also ensured by a query allocation method using a TRC process. This proves property *(i)*.

Now, consider that qa sets $k' = N_q \wedge k_n = q.n$. In this case, qa allocates an incoming query q to the less utilized providers in set P_q, which is also the objective of a $Capacity\ based$ method. Thus, both

qa and $Capacity\ based$ ensure the following equation,

$$\forall p \in \widehat{P_q}, \nexists p' \in P_q \backslash \widehat{P_q} : \mathcal{U}_{(p')} < \mathcal{U}_{(p)}$$

which proves property *(ii)*.

Finally, consider that qa sets $k' = N_q \wedge k_n = k'$. Doing so, an incoming query q is allocated by qa to a set $\widehat{P_q}$ such that,

$$\forall p \in \widehat{P_q}, \nexists p' \in P_q \backslash \widehat{P_q} : scr_q(p') > scr_q(p)$$

Thus, the only thing that is considered by qa is the participants' intentions and thus it will have an $Intention\ based$ behavior. In other words, the mediator has no control to regulate the system. We call this way to operate the *intention based* approach. This proves property *(iii)*. □

The great advantage of using K_nBest is that it allows the mediator to adapt the query allocation process to the application by varying its parameters. To illustrate this, consider the following examples. First, if providers and incoming queries are homogeneous, the mediator can take a *TRC* behavior (which has been proved to operate well in homogeneous distributed systems [Mit01]) when allocating queries by setting parameters of K_nBest as in property *(i)*. Second example, consider that providers and incoming queries are heterogeneous and that the most important is to perform *qlb* with no consideration for participants' intentions. In this case, the mediator can allocate queries following a $Capacity\ based$ behavior, by setting parameters of K_nBest as in property *(ii)*. Finally, consider that participants are autonomous and there is no other objective in the system than satisfying participants, the mediator can then allocate queries based only on the participants' intentions by setting parameters of K_nBest as in property *(iii)*.

As we focus on heterogeneous distributed information systems in this thesis, we assume that k' is always equal to N in the rest of this chapter (i.e. we discard the random selection phase).

2.4.3 Query Allocation Principle

We now describe how the mediator allocates queries. Figure 3.2(a) illustrates the general S_bQA system architecture and Algorithm 3 shows the main steps of the query allocation process. Given a query q and a set P_q of providers that are able to perform q, the mediator first asks for $q.c$'s *intention* for allocating q to each provider $p \in P_q$ (line 2 of Algorithm 3). In parallel, it also asks for P_q's *utilization* (with the assumption that function τ denotes function $\mathcal{U}$) and *intention* for performing q (lines 3 and 4). Then, it waits for this information from both $q.c$ and set P_q or for a given *timeout* (line 5). Once such vectors $\overrightarrow{CI}_q$, $\overrightarrow{U}$, and $\overrightarrow{PI}_q$ are computed (where $\overrightarrow{U}$ stores the *utilization* of each provider in P_q), the mediator selects the k_n less utilized providers, denoted by set K_n, from set P_q (line 6). This selection phase can be solved using a sorting algorithm, so, in the worst case, its complexity is $O\big(N\ \log_2(N)\big)$. Next, the mediator computes the score of each provider $p \in K_n$ by making a balance between $q.c$'s and p's *intentions* (line 7 and 8) and computes the ranking of providers in K_n (line 9), whose complexity is $O\big(k_n\ \log_2(k_n)\big)$ in the worst case. Finally, the mediator allocates q to the $q.n$ best *scored* providers in set K_n and sends the mediation result to all P_q providers (lines 10 and 11). Notice that in the case that $q.n \geq k_n$, the mediator thus allocates q to all k_n providers. Indeed, Algorithm 3 can be optimized, but our goal is to show the steps involved in the query allocation process.

2.4.4 Communication Cost

We analyze the communication cost of S_bQA in terms of number of messages that the mediator should transfer over the network to perform a query. The communication cost of S_bQA is given by the

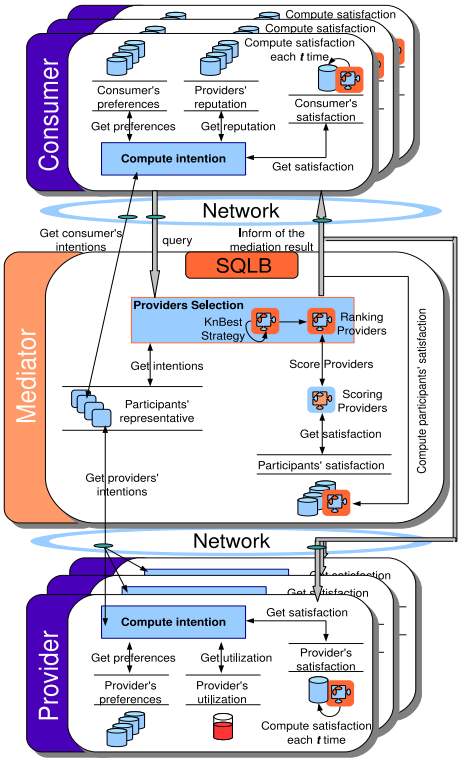


Figure 2.3 – SQLB system architecture.

following theorem.

Theorem 3. *The total number of transferred messages by S_bQA to perform a query is* $3(N+1)+n$.

Proof. As we saw in the previous section, given any incoming query q, the mediator transfers $mssg_0 = 2N+2$ messages over the network to ask the consumer's *intentions* and the *utilization* and *intention* of providers in set P_q. Then, it selects the k_n least utilized providers in set P_q and allocates q to the $q.n$ best scored providers in set K_n. After this, the mediator informs all providers in set P_q of the mediation result and waits for results from the $q.n$ selected providers. This implies to exchange $mssg_1 = N+n$ messages among the mediator and participants, where n stands for $q.n$. Finally, the mediator transfers $mssg_2 = 1$ messages to give results to $q.c$. Thus, the total number of messages transferred over the network by the mediator to perform a query is $mssg_0 + mssg_1 + mssg_2 = 3(N+1)+n$. □

This is not a high cost considering that microeconomic-based query allocation methods transfers $3N+n+1$ messages to perform a query : N messages to ask for providers' bid, N messages for receiving providers' bid, N messages to inform providers of the mediation result, n messages to get results from selected providers, and 1 message to return results to the consumer. We can further reduce the number of exchanged messages by using participants' *representatives* [LCLV07] or by introducing again the random selection phase (see Section 2.4.2). However, the problem of reducing communication cost is orthogonal to the problem we address in this thesis.

Algorithm 3: QueryAllocation

Input : q, k_n, P_q
Output: $All\overrightarrow{oc}_q$

begin
  `// Consumer's intentions`
  fork ask for $q.c$'s intentions towards each provider in P_q;
  `// Providers' intention`
  foreach $p \in P_q$ **do**
    fork ask for the utilization and intention of provider p with regards to q ;
  waituntil $\overrightarrow{CI}_q$, $\overrightarrow{U}$, and $\overrightarrow{PI}_q$ be calculated or a given *timeout*;
  `// qlb regulation`
  $K_n \leftarrow$ select k_n less utilized providers from set P_q ;
  `// Scoring and ranking providers`
  foreach $p \in K_n$ **do**
    compute p's score concerning $\overrightarrow{CI}_q[p]$ & $\overrightarrow{PI}_q[p]$;
  rank set K_n of providers regarding $scr_p(q)$, $\overrightarrow{R}_q$;
  `// Query Allocation`
  for $i = 1$ **to** $min(n, k_n)$ **do** $All\overrightarrow{oc}[\overrightarrow{R}_q[i]] \leftarrow 1$;
  for $j = min(n, k_n) + 1$ **to** N **do** $All\overrightarrow{oc}[\overrightarrow{R}_q[j]] \leftarrow 0$;
end

2.5 Discussion

We pointed out in Sections 2.2 and 2.3 that there exist several ways a participant can compute its preferences. To the best of our knowledge, there is no work that proposes a comparison study of these different preference functions and hence it is still an open problem. We believe that such a study may be quite interesting to allow a participant knowing which strategy it can adopt to compute its preferences. Similarly, several manners to compute the consumers' and providers' intentions exist. This is also an open problem that should be explored so as to identify the best ways for a participant to adapt their intentions to their context and application. Improving on these functions is not the focus of our work. Instead, our framework is designed so it can leverage any existing preference and intention function.

Moreover, the score function of a query allocation method is usually based on specific demands, which are given by the application challenges that one wants to solve. Thus, a large number of specific query allocation methods with different behaviors may exist. For example, the score function of a *qlb* method is designed for those applications whose goal is to ensure good system performance. However, when the behavior of a query allocation method is specific to an application, it cannot be applied elsewhere, and worse, it cannot perform in environments where participants change their interests on the fly. Therefore, we proposed a score function that makes no assumption about either the kind of application nor the way in which a participant obtains its preferences. It just allocates queries based on the participants' intentions. But, we are aware that sometimes a mediator, or even the system administrator, is required to satisfy some constraint, e.g. to ensure a specific *Quality of Service*, no matter what participants prefer. This is why we also proposed a strategy that allows the query allocation method to regulate the system with regards to a given function. As a result, conversely to specific query allocation methods,

S_bQA is quite general, self-adaptable to the interests of participants, and adaptable to the application. This allows S_bQA to perform in many kinds of environments and to perform as well as any specific query allocation method by tuning its parameters or if participants desire so.

2.6 Experimental Validation

Our experimental validation, in this section, has three main objectives :

- To evaluate how well different query allocation methods operate in distributed systems with autonomous participants.
- To analyze if S_bQA satisfies participants while ensures good *qlb* because it is not obvious that when adding new criteria a query allocation method still gives good results for an initial criteria.
- To study how well our measures capture query allocation methods' operation.

To do so, we carry out four kinds of evaluations. First, we evaluate the general query allocation process by applying the satisfaction model and measures we present in Chapter 1. Second, we evaluate the impact of participants' *autonomy* on performance. Third, we evaluate the self-adaptability of S_bQA to participants' intentions. Finally, we analyze the effects of varying the values of k_n parameter, i.e. we evaluate the S_bQA's adaptability to different kinds of applications.

2.6.1 Setup

We built a Java-based simulator that simulates a *mono-mediator* distributed information system, which follows the mediation system architecture presented in [LCLV07]. For all the query allocation methods we tested, the following configuration (Table 2.1) is the same and the only change is the way in which each method allocates the queries to providers. Before defining our experimental setup let us say that the definition of a synthetic workload for environments where participants are autonomous and have special interests towards queries is an open problem. *Pieper et al.* [PPS07] discuss the need of benchmarks for scenario-oriented cases, which are similar to the case we consider, but this remains an open problem. Another possibility to validate our results is to consider real-world data over long periods of time. However, even if we had (we don't) the resources to obtain real-world data, the validation would get biased towards the specific applications. Therefore, in our experiments, we decided to generate a very general workload that can be applied for different applications and environments in order to thoroughly validate our results.

Participants work out their *satisfaction*, *adequation*, and *allocation satisfaction* as presented in Section 1.3. We initialize them with a satisfaction value of 0.5, which evolves with their last 200 issued queries and 500 queries that have passed through providers. That is, the size of k is 200 for consumers and 500 for providers. The number of consumers and providers is 200 and 400 respectively, with only one mediator allocating all the incoming queries. We assign sufficient resources to the mediator so that it does not cause bottlenecks in the system. We assume that all providers in the system are able to perform each incoming query to the system, i.e. for any incoming query q the size of set P_q is 400. We also assume that consumers and providers compute their intentions as defined in Sections 2.2 and 2.3, respectively. For simplicity, we set $\upsilon = 1$, i.e. the consumers' preferences denote their intentions.

To simulate high heterogeneity of the consumers' preferences for allocating their queries to providers, we divide the set of providers into three classes according to the interest of consumers : to those that consumers have *high interest* (60% of providers), *medium interest* (30% of providers), and *low interest* (10% of providers). Consumers randomly obtain their preferences between $.34$ and 1 for high-interest

Parameter	Definition	Value
nbConsumers	Number of consumers	200
nbProviders	Number of providers	400
nbMediators	Number of mediators	1
qDistribution	Query arrival distribution	Poisson
iniSatisfaction	Initial satisfaction	0.5
conSatSize	k last issued queries	200
proSatSize	k last treated queries	500
nbRepeat	Repetition of simulations	10

Table 2.1 – Simulation parameters.

providers, between $-.54$ and $.34$ for medium-interest providers, and between -1 and $-.54$ for low-interest providers. On the other side, to simulate high heterogeneity of the providers' preferences towards the incoming queries, we also create three classes of providers : those that have *high adaptation* (35% of providers), *medium adaptation* (60% of providers), and *low adaptation* (5% of providers). Here, adaptation stands for the *system adequation w.r.t. a provider* notion we defined in Section 1.3.1.2. Providers randomly obtain their preferences between $-.2$ and 1 (high-adaptation), between $-.6$ and $.6$ (medium-adaptation) or between -1 and $.2$ low-adaptation). More sophisticated mechanisms for obtaining such preferences can be applied (for example using the *Rush* language [SBD94]), but this is well beyond the scope of this thesis and orthogonal to the problem we address in this chapter. Without any loss of generality, the participants' intentions, in the long run, are static in our simulations. We assume this to evaluate the query allocation methods in a long-term trend, but our satisfaction model allows intentions to be dynamic.

We set the providers' capacity heterogeneity following the results presented in [SGG02]. We generate around 10% of providers with low-capacity, 60% with medium-capacity, and 30% with high-capacity. The high-capacity providers are 3 times more powerful than medium-capacity providers and still 7 times more powerful than low-capacity providers. We generate two classes of queries that consume, respectively, 130 and 150 treatment units at the high-capacity providers. High-capacity providers perform both classes of queries in almost 1.3 and 1.5 seconds, respectively. We assume that providers compute their utilization as in Definition 12.

We do not consider the random selection phase because we consider heterogeneous distributed systems. In other words, we assume in all our experimentations that k' is equal to N. We assume that queries arrive to the system in a *Poisson* distribution, as found in dynamic autonomous environments [Mar02]. Since our main focus is to study the way in which queries are allocated, we do not consider in this thesis the bandwidth problem and assume that all participants have the same network capacities. Finally, for the sake of simplicity, we assume that consumers only ask for one informational answer (i.e. $n = 1$) and all the providers in the system are able to perform all the incoming queries.

2.6.2 Baseline Methods

2.6.2.1 Capacity based

In distributed information systems, there are two well-known approaches to balance queries across providers : $Load\ Based$ and $Capacity\ based$ methods. We discard $Load\ Based$ [GBGM04, ABKU99] methods since, unlike $Capacity\ based$, they inherently assume that providers and queries are homo-

geneous. In $Capacity\ based$ [MTS90, RM95, SKS92] methods, one common approach is to allocate each query q to providers that have the highest available capacity (i.e. the least utilized) among set P_q of providers. $Capacity\ based$ has been shown to be better than $Load\ Based$ in heterogeneous distributed information systems. Thus, we use $Capacity\ based$ in our simulations. Note that $Capacity\ based$ does not take into account the consumers nor providers' intentions.

2.6.2.2 Mariposa-like

Economical models have been shown to provide efficient query allocation in heterogeneous systems [FNSY96, FYN88, SAL$^+$96]. Mariposa [SAL$^+$96] is one of the most important approaches to allocate queries in autonomous environments. In this approach, all the incoming queries are processed by a *broker* site that requests providers for *bids*. Providers bid for obtaining queries based on a local bulletin board and then the broker selects the set of bids that has an aggregate price and delay under a *bid* curve provided by the consumer. In Mariposa, providers modify their bids with their current load (i.e. *bid* $\times$ *load*) in order to ensure *qlb*. Since Mariposa has shown good results, we implemented a *Mariposa-like* method to compare it with S_bQA. In our *Mariposa-like* implementation we assume that consumers are only interested in the price for getting results. Note that different economical methods may lead to different performance results than those presented here.

2.6.3 Results

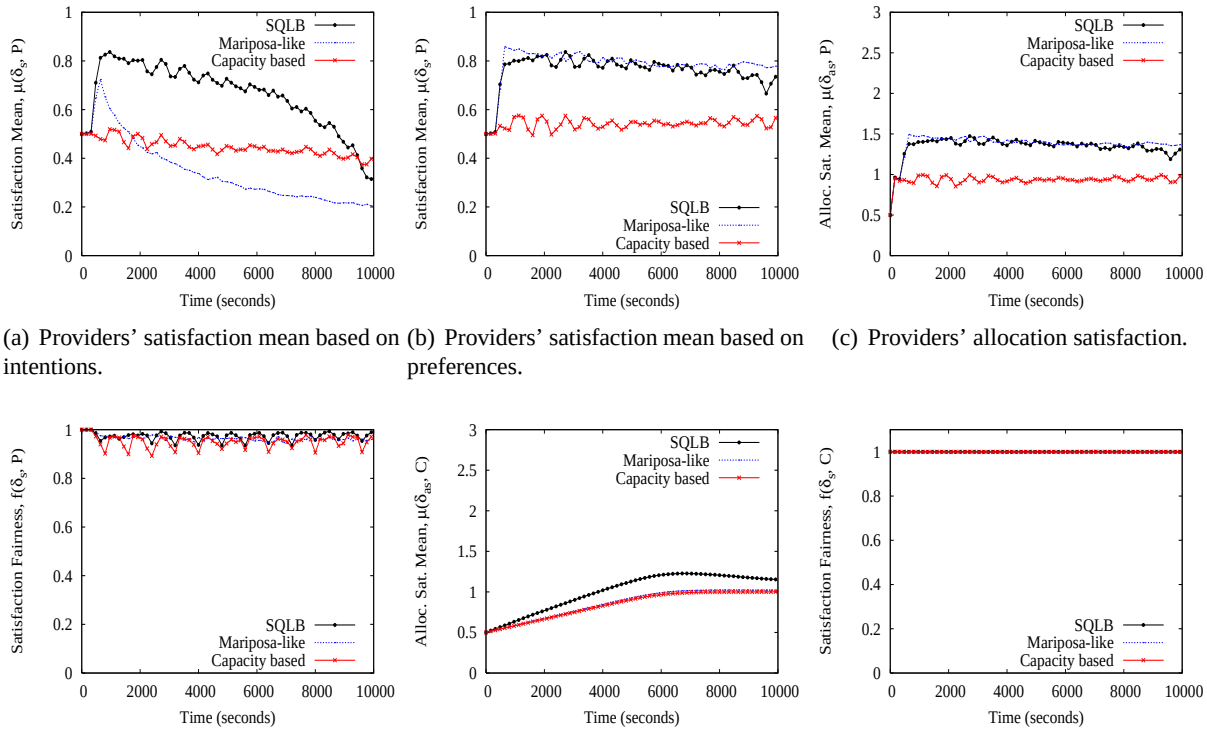
We start, in Section 2.6.3.1, by evaluating the quality of the three query allocation methods, with regards to satisfaction and *qlb*, in environments where participatns are not allowed to leave the system (i.e. with captive participants). In Section 2.6.3.2, we evaluate how well these methods deal with the possible participants' departure by *dissatisfaction*, *query starvation*, or *overutilization*. Then, in Section 2.6.3.3, we show the self-adaptability of S_bQA to participants' intentions. In these three first sections, we assume that $k_n = k'$, i.e. set K_n denotes set P_q considering that $k' = N$. Finally, in Section 2.6.3.4, we study the adaptability of S_bQA to the kind of application by varying parameter k_n.

2.6.3.1 Quality Results with Captive Participants

If participants are autonomous, they may leave the system by *dissatisfaction*, *query starvation*, or *overutilization*. Nevertheless, the choice of such departure's thresholds is very subjective and may depend on several external factors. Thus, for these first experiments, we consider *captive* participants, i.e. they are not allowed to leave the system. To measure the quality of the three methods, we apply the measures defined in Section 1.4. We ran a series of experiments where each one starts with a workload of 30% that uniformly increases up to 100% of the total system capacity.

We first analyze the providers results. Figure 2.4(a) shows the satisfaction mean ensured by the three methods. The satisfaction used in this measurement is based on the providers' intentions, i.e. what the mediator can see. We observe in these results that providers are more satisfied with S_bQA than with the two others. As the workload increases, providers' satisfaction decreases because their intentions decrease as they are loaded (just because utilization becomes the most important for them). Thus, S_bQA cannot satisfy the providers' intentions for high workloads since their adequation (based on intentions) is low. $Capacity\ based$ and *Mariposa-like* do not satisfy the providers' intentions from the beginning, simply because they allocate queries based on other criteria, which do not exactly meet intention.

Nonetheless, this does not reflect what providers really feel with respect to their preferences. To show this, we need to measure the mean ensured by the three methods concerning the providers' satisfaction



(a) Providers' satisfaction mean based on intentions.

(b) Providers' satisfaction mean based on preferences.

(c) Providers' allocation satisfaction.

(d) Provider satisfaction fairness.

(e) Consumers' allocation satisfaction.

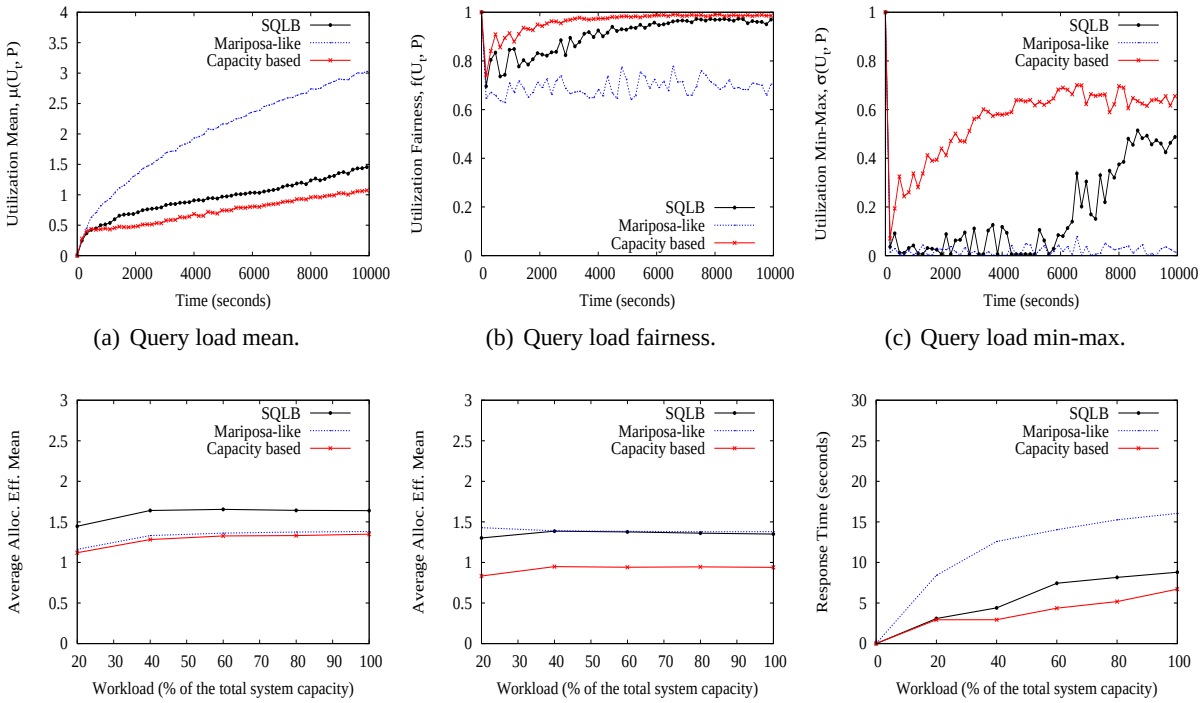
(f) Consumer satisfaction fairness.

Figure 2.4 – Participants' satisfaction results for a workload range from 30 to 100% of the total system capacity when participants are captive.

based on their preferences. Although we can measure such a satisfaction in our simulations, this is not always possible since such preferences are usually considered as private. Figure 2.4(b) shows the results of these measurements. We observe that S_bQA has the same performance as *Mariposa-like* even if it considers the consumers' intentions. When the workload is close to 100%, the providers' satisfaction slightly decreases with S_bQA. As noted earlier, this is because providers pay more attention to their utilization for obtaining their intentions, thus their preferences are less considered by the S_bQA method.

It is worth noting that, as expected, $Capacity\ based$ is the only one among these three methods that penalizes the providers. This is clear in Figure 2.4(c), which illustrates the mean ensured by these three methods with respect to the providers' allocation satisfaction. We observe that providers are not satisfied with $Capacity\ based$ having, in general, allocation satisfaction values under 1. Then, based on these results, we can predict that when providers will be free to leave the system, $Capacity\ based$ will suffer from serious problems with providers' departures by dissatisfaction reasons. Figure 2.4(d) illustrates the satisfaction fairness ensured by the three methods. We see that they guarantee almost the same satisfaction fairness. However, as seen in the previous results, this does not mean that providers are satisfied with all three methods.

Now, let us analyze the consumer results. Figure 2.4(e) illustrates the allocation satisfaction mean concerning the consumers' intentions. We observe that while S_bQA is the only one to satisfy consumers, the two others are neutral to consumers (mean values equal to 1). These results allows us to predict that $Capacity\ based$ and *Mariposa-like* may suffer from consumer's departures while S_bQA does not. The S_bQA's mean decreases for high workloads because of providers. Remember that providers' satisfac-



(a) Query load mean. (b) Query load fairness. (c) Query load min-max.

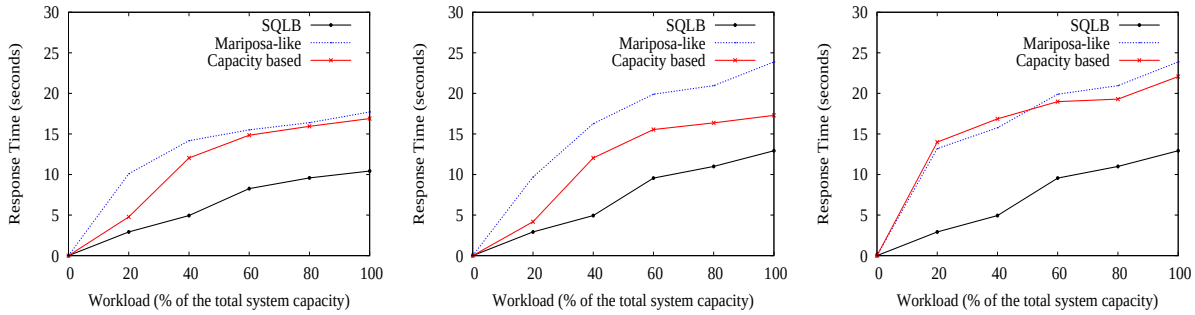
(d) Allocation efficiency w.r.t. consumers. (e) Allocation efficiency w.r.t. providers. (f) Response times

Figure 2.5 – (a) and (c) : query load balancing results for a workload range from 30 to 100% of the total system capacity when participants are captive, (d) and (e) : allocation efficiency results for different workloads, and (f) : ensured response times. All these results are with captive participants.

tion decrease because they take care of their utilization. So, S_bQA pays more attention to providers' satisfaction than to consumers' satisfaction. Nonetheless, consumers are never penalized ! Conversely to providers, we can observe in Figure 2.4(f) that consumers' satisfaction fairness has less variations because they are not in direct competition to allocate queries.

Concerning *qlb*, as expected, $Capacity\ based$ better balances the queries among providers than S_bQA and *Mariposa-like* (see Figure 2.5(a)). We can observe that S_bQA performs well, while *Mariposa-like* has serious problems to balance queries. Thus, *Mariposa-like* may lose providers by query starvation or overutilization reasons. Figure 2.5(b) shows that S_bQA has some difficulties to be fair (w.r.t. *qlb*) for workloads under 40%. In contrast, when the workload increases, S_bQA pays more attention to *qlb* and becomes fairer. This is clearly illustrated in Figure 2.5(c), which shows the results about the utilization Min-Max. The reason that S_bQA performs better for high workloads is that providers become overutilized and thus they take much more care with their utilization, which is not the case for low workloads. These *qlb* results demonstrate the high adaptability of S_bQA to the variations in the workloads.

Figures 2.5(d) and 2.5(e) illustrate the allocation efficiency with respect to consumers and providers for different workloads. These results clearly illustrate the superiority of S_bQA over $Capacity\ based$ and *Mariposa-like* since we can observe, *(i)* on the one hand, that S_bQA significantly outperforms $Capacity\ based$ in both cases ; and *(ii)* on the other hand, that S_bQA and *Mariposa-like* have the same allocation efficiency w.r.t. providers, but S_bQA significantly outperforms *Mariposa-like* in the consumers' case, which demonstrates the equity at both levels of S_bQA.



(a) Providers may leave by *dissatisfaction*.

(b) Providers may leave by *dissatisfaction* or *starvation*.

(c) Providers may leave by *dissatisfaction*, *starvation*, or *overutilization*.

Figure 2.6 – Impact on performance of providers' departure.

Finally, Figure 2.5(f) shows the ensured response times in these environments (with captive participants). As is conventional, response time is defined as the elapsed time from the moment that a query q is issued to the moment that $q.c$ receives the response of q. As expected, the $Capacity\ based$ method outperforms the two others. However, even if S_bQA takes into account the participants' intentions, it only degrades performance by a factor of 1.4 in average while *Mariposa-like* does so by a factor of 3 !

All above results show that $Capacity\ based$ may severely suffer from providers' departures by dissatisfaction, while *Mariposa-like* may also suffer from providers' departures by query starvation or overutilization. Furthermore, above results demonstrate the S_bQA's self-adaptability to changes in the participants' satisfaction and to the workload. This feature makes our proposal highly suitable for autonomous environments. Furthermore, as concluding remark, we can say that even if not designed for environments where participants are captive, S_bQA ensures quite good response times and pays attention to the quality of results and queries that consumers and providers get from the system, respectively.

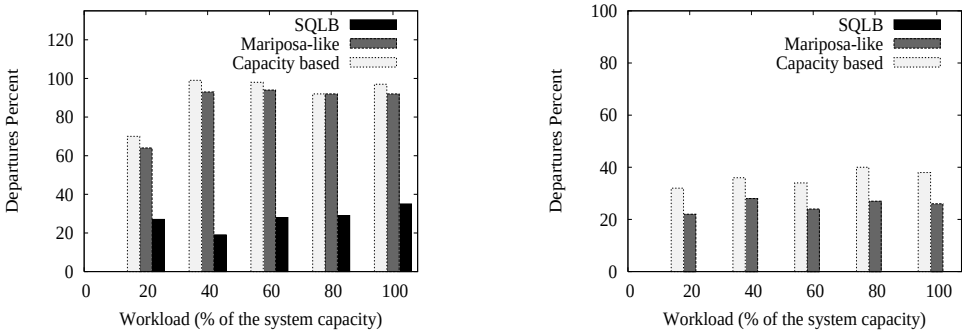
2.6.3.2 Dealing with Autonomy

To validate our measurements and intuitions of previous section, we also ran several experimental simulations where participants are given the *autonomy* to leave the system. Our main goal, in this section, is to study the reasons by which providers leave the system and evaluate the impact on performance that such departures may have. We evaluate the ensured response times by the three methods in autonomous environments and compare it with those of the captive environments (see Figure 2.5(f)).

To do so, we have to set the thresholds under, or over, which a participant decides to leave the system. To avoid any suspicion on the choice of such thresholds and to be fair with baseline methods, we assume that participants support high degrees of dissatisfaction, query starvation, and overutilization. Thus, a consumer leaves the system, by dissatisfaction, if its satisfaction is smaller than its adequation, i.e. the allocation method penalizes it. A provider leaves the system by

- dissatisfaction, if its satisfaction value is 0.15 smaller than its adequation,
- query starvation, if its intention-based profit, π, is smaller than 0.2 in a period of 2 minutes, i.e. the minimal intention-based profit $starv$ of a provider is 0.2 (Equation 1.4), and
- overutilization, if its utilization is greater than 220% of its optimal utilization, where the optimal utilization of a provider is 0.8 when the workload is 80% of the total system capacity.

We ran a first series of experiments with different workloads where providers are allowed to leave the system by dissatisfaction only (see Figure 2.6(a)). We can see that our approach outperforms both



(a) Number of providers' departures. (b) Number of consumers' departures.

Figure 2.7 – Participants' departures.

$Capacity\ based$ and *Mariposa-like* because it better satisfies providers than $Capacity\ based$, and better ensures *qlb* in the system than *Mariposa-like*. Recall that in previous section we note that *Mariposa-like* tends to overutilize some providers (those that are the most adapted to the incoming queries). This is why, even if *Mariposa-like* better satisfies providers than $Capacity\ based$ (see Figure 2.4(b)), it ensures higher response times than $Capacity\ based$.

A second series of experiments allows providers to leave the system by dissatisfaction or starvation. A provider might quit the system by starvation e.g. when it simply does not obtain the queries that it needs to survive. Figure 2.6(b) illustrates these results. We observe again that S_bQA significantly outperforms the other two methods for all workloads and that its performance is almost the same than last series of experiments, which means that S_bQA generally does not suffer from starvation departures. Furthermore, we can see that $Capacity\ based$ better performs than *Mariposa-like* because it better balances the query load than *Mariposa-like*. As previous series of experiments, this is because $Capacity\ based$ ensures a better *qlb* in the system.

Also, we run a series of experiments where providers are allowed to leave the system by dissatisfaction, starvation, or overutilization. A provider may quit the system by overutilization if this implies for example a loss of business for it, e.g. when overutilization deteriorates the quality of service provided by a provider and consumers are interested in good quality of services. This results are illustrated by Figure 2.6(c). We observe that while S_bQA and *Mariposa-like* degrade their performance only by a factor of 1.4 in average (w.r.t. Figure 2.5(f)), $Capacity\ based$ does it by a factor of 3.5 ! Figure 2.7(a) shows the number of provider's departures with the three methods. We observe that, except for a workload of 20%, $Capacity\ based$ and *Mariposa-like* lose almost all the providers for all workloads. Note that S_bQA only loses 28% of providers in average ! This demonstrates the high efficiency of S_bQA in autonomous environments.

We show, in Table 2.2, an analysis of providers' reasons to leave the system when the workload is 80%. We observe that, as predicted in Section 2.6.3.1, providers leave the system with $Capacity\ based$ because of dissatisfaction, while they do so because of overutilization with *Mariposa-like*. Furthermore, the providers that decide to leave in both methods are mainly those that are the most adapted to incoming queries and that consumers desire the most. With S_bQA, providers leave the system by dissatisfaction, but such providers are mainly those that are low-capacity. In fact, we can see that S_bQA mainly maintains the high-interest, high-adaptation, and high-capacity providers in the system.

Finally, Figure 2.7(b) shows the consumers' departure by dissatisfaction with these three methods. Again, S_bQA is a clear winner with no consumer's departures. Note that, the consumer's departures

		S_bQA				$Capacity\ based$				*Mariposa-like*			
		low	med	high	total	low	med	high	total	low	med	high	total
	C.I.P.	1%	5%	13%		5%	16%	31%		1%	7%	11%	
D	P.A.	2%	9%	8%	19%	3%	34%	15%	52%	0%	15%	4%	19%
	P.C.	13%	6%	0%		13%	30%	9%		5%	12%	2%	
		low	med	high	total	low	med	high	total	low	med	high	total
	C.I.P.	0%	0%	4%		0%	0%	0%		0%	2%	6%	
S	P.A.	4%	0%	0%	4%	0%	0%	0%	0%	3%	3%	2%	8%
	P.C.	2%	2%	0%		0%	0%	0%		3%	5%	0%	
		low	med	high	total	low	med	high	total	low	med	high	total
	C.I.P.	0%	0%	6%		0%	0%	38%		0%	0%	65%	
O	P.A.	0%	3%	3%	6%	3%	8%	27%	38%	1%	15%	49%	65%
	P.C.	1%	4%	1%		0%	18%	20%		0%	30%	35%	

Table 2.2 – Reasons of the provider's departures for a workload of 80% of the total system capacity. [**C.I.P.** stands for Consumer Interest to Providers, **P.A.** stands for Provider's Adequation, and **P.C.** stands for Provider's Capacity. And, the providers' departure by dissatisfaction, query starvation, and overutilization are given by rows **D**, **S**, and **O**, respectively.]

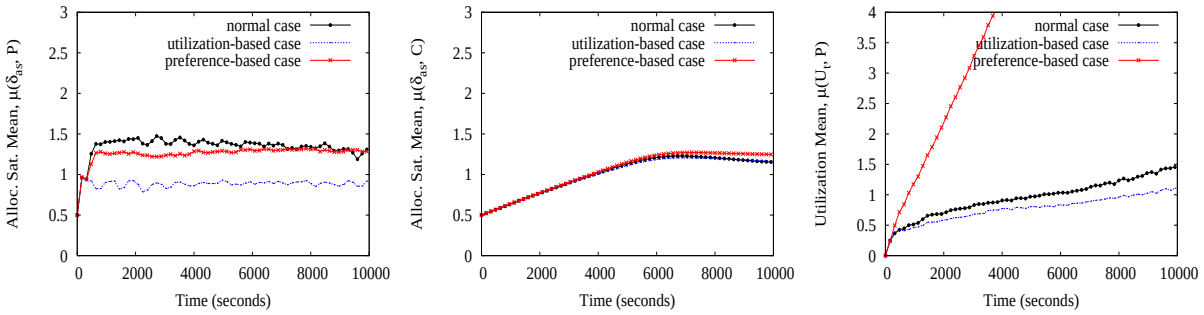
have also a direct impact on performance since the less the incoming queries, the less the chances for satisfying providers.

2.6.3.3 Adaptability to Participants' Interests

Our objective in this section is to study how well S_bQA adapts to different participants' *intentions*. With this in mind, we consider again captive environments such as in Section 2.6.3.1. For simplicity, we evaluate providers with two different intentions : those that are only interested in their preferences (the *preference-based case*), i.e. the providers' preferences denote their intentions, and those that are only interested in their load (the *utilization-based case*), i.e. providers compute their intentions based on their utilization. Consumers work out their intentions regarding the providers' capacity to perform queries, such as in previous sections. We compare results of S_bQA in both cases with those obtained in the normal case, i.e. when providers make a balance between their preferences and utilization to compute their intentions, such as in Section 2.6.3.1.

Figure 2.8 shows the results of these experiments with a workload range from 30 to 100% of the total system capacity. We can observe in Figures 2.8(a) and 2.8(b) that the results are strongly related to the participants' intentions. We can observe in Figure 2.8(a) that, as expected, providers are more satisfied in the preference-based case than in the utilization-based case. But, contrary to the expected, providers are less satisfied in the preference-based case than in the normal case. During our experimentations, we observed that those providers with high-adaptation tend to monopolize the queries, which causes dissatisfaction to the medium and low- adaptation providers. This phenomenon does not occur in the normal case because S_bQA also considers the providers' utilization. This is why providers are in average less satisfied in the preference-based case than in the normal case. However, since in the normal case providers pay more attention to their utilization as the workload increases, providers have the same degree of satisfaction, for high workloads, in both preference-based and normal cases.

In Figure 2.8(b), we observe that consumers have the same degree of satisfaction in the three cases,



(a) Providers' allocation satisfaction. (b) Consumers' allocation satisfaction. (c) Query load mean.

Figure 2.8 – Quality results for a workload range from 30 to 100% of the total system capacity when participants are captive and for three kinds of providers : *(i)* when they are interested only in their preferences (the *preference-based case*), *(ii)* when they are just interested in their utilization (the *utilization-based case*), and *(iii)* when their utilization is as important as their preferences (the *normal case*).

but we can observe, in the preference-based case, a very small gain for high workloads. This is because for high workloads, providers give more importance to their utilization in both utilization-based and normal cases. Hence, for high workloads, the query allocation pays more attention to providers and thus the consumers' satisfaction decreases in these both cases.

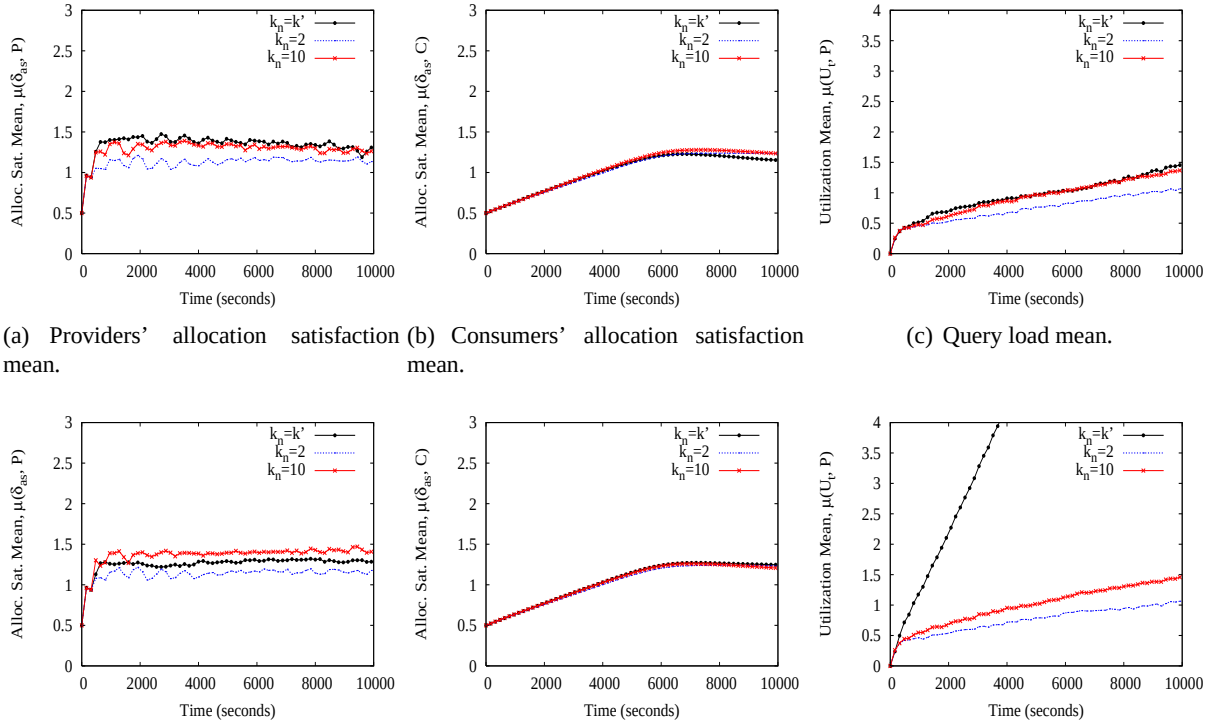
Now, concerning *qlb*, S_bQA performs well in the utilization-based and normal cases while, in the preference-based case, S_bQA significantly degrades the providers' utilization because providers have no consideration for *qlb*. On the other side, observe that, in the utilization-based case, S_bQA follows the behavior of the $Capacity\ based$ approach (see Figures 2.8(a) and 2.8(c)) with regards to the providers' results, but it is much better from a consumer point of view.

All above results allow us to conclude that S_bQA allows participants to obtain from the system what they want and not what the system considers relevant for them. In other words, our results demonstrate that S_bQA ensures good levels of satisfaction as far as the system is adequate to participants and *vice versa*. Thus, if the participants correctly work out their intentions, S_bQA allows them to reach their intentions in the system.

2.6.3.4 Adaptability to Applications

We finally discuss how to adapt S_bQA to different applications by varying the parameter of K_nBest. Without any loss of generality, we assume in this thesis that the mediator wants to regulate the system concerning *qlb* so as to ensure good response times. To better illustrate the effects of varying parameter k_n (i.e. the regulation of the system concerning *qlb*), we consider two kinds of providers : those that do not have any consideration for their utilization when they compute their intentions (the preference-based case), and those that make a balance of their preferences and their utilization to compute their intentions (the normal case).

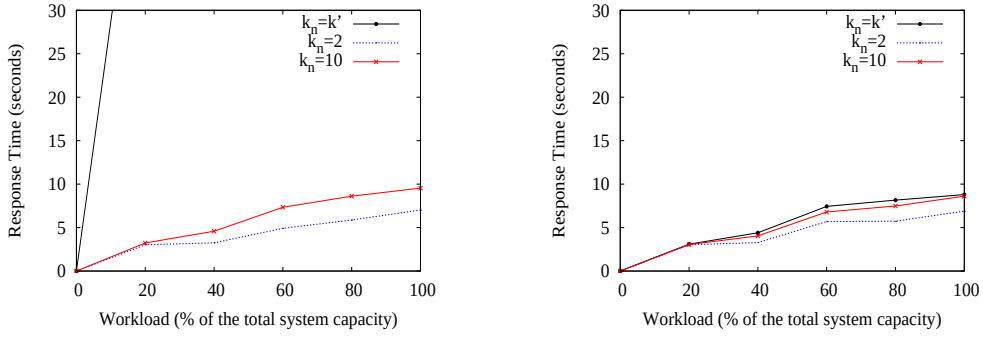
For simplicity, we consider only two different applications in this work : *(i)* one where ensuring the performance of the system is mandatory such as in distributed databases and *(ii)* other where participants' satisfaction is mandatory and some level of system's performance is desired such as in *e-commerce* scenarios. For the first kind of application, the mediator should perform *qlb* while guaranteeing interesting results and queries to participants because of their autonomy. For the second kind of application, the mediator's priority is to satisfy providers while ensuring an acceptable system performance. To do so,



(a) Providers' allocation satisfaction mean. (b) Consumers' allocation satisfaction mean. (c) Query load mean.

(d) Providers' allocation satisfaction mean. (e) Consumers' allocation satisfaction mean. (f) Query load mean.

Figure 2.9 – Quality results for a *workload* range from 30 to 100% of the total system capacity when participants are *captive* and : (a)-(c) providers compute their intentions based on their preferences and utilization (the *normal case*) and (d)-(f) providers compute their intentions based on their preferences (the *preference-based case*).



(a) when providers' preferences denote their intentions (the *preference-based case*).

(b) when providers make a balance between their preferences and utilization to compute their intentions (the *normal case*).

Figure 2.10 – Performance results with captive participants.

the mediator sets parameter $k_n = 2$ and it sets $k_n = 10$ for the first kind of application and the second one, respectively.

To clearly see the impact of parameter k_n, we compare both results (i.e. when $k_n = 2$ and $k_n = 10$) with the case where the mediator has no control to regulate the system (i.e. when $k_n = k'$). Notice that the previous sections assumed that $k_n = k'$, thus the results of S_bQA in the normal and preference-based cases that we present in this section are the same as those we presented in Sections 2.6.3.1 and 2.6.3.3, respectively. We present them again as references for both two other k_n sizes ($k_n = 2$ and $k_n = 10$).

We can see in Figures 2.9(a) and 2.9(d) that for all three different k_n values and both normal and preference-based cases, providers are generally satisfied with the job done by S_bQA, which is not obvious in applications when *qlb* is the most important (e.g. the $k_n = 2$ case). Notice that providers are more satisfied in the normal case as the k_n value increases (see Figure 2.9(a)), but this is not the case for providers in the preference-based case when $k_n = k'$ and $k_n = 10$ (see Figure 2.9(d)). This is because, as noted in the previous section, the high-adaptation providers tends to monopolize the queries when they compute their intentions based only on their preferences, i.e. the preference-based case. But, when the mediator regulates the system with respect to *qlb*, it better distributes queries among providers and thus avoids, in the preference-based case, the query starvation in the less adapted providers (i.e. in the providers with medium and low-adaptation). Of course, when k_n takes small values the providers are less satisfied (which is the case of $k_n = 2$) because the mediator pays less attention to the providers' intentions. In these cases, however, even if the objective is the same for both, S_bQA performs much better than the $Capacity\ based$ approach because it satisfies both consumers and providers (see Figures 2.4(c) and 2.4(e) for $Capacity\ based$). In fact, we can observe in Figures 2.9(b) and 2.9(e) that the regulation of the system has almost no impact on the consumers, which are equally satisfied for all k_n values.

Concerning *qlb*, we can see in Figures 2.9(c) and 2.9(f) that the mediator can ensure good *qlb* even if providers do not have any consideration to their utilization. Obviously, the smaller the k_n value, the better the ensured *qlb* in the system. In these results, it is worth noting that, even when ensuring participants' satisfaction is the most important in an application (when $k_n = k'$), the way in which S_bQA computes the providers' score allows it to ensure an acceptable *qlb* in the system as far as providers take care of their load, e.g. in the normal case (see Figure 2.9(c)). This is not the case for the preference-based case, when $k_n = k'$, even if providers' preferences are the same (see Figure 2.9(f)). But, by setting small k_n values, S_bQA can ensure good response times for consumers in both cases, no matter how providers compute their intentions.

The ensured response times with different k_n values are shown by Figures 2.10(a) and 2.10(b). We can observe that, as expected, the mediator can ensure good response times, even if providers are not interested in, by playing with parameter k_n (the $k_n = 2$ and $k_n = 10$ results). This is not the case when the mediator does not regulate the system and providers do not care about the system performance (the $k_n = k'$ results for the preference-based case).

The results in this section demonstrate that with small k_n values, one can adapt S_bQA to applications where the mediator needs to regulate the system regarding a given predefined function (*qlb* in this work) without mattering how participants compute their intentions. With high k_n values, one can adapt S_bQA to applications where the mediator has to meet the participants' intentions.

2.7 Related Work

The query allocation problem, which appears as a subproblem of query processing [Kos00], is very general and is addressed in many domains such as distributed databases, networking systems, grid systems, and multi-agent systems. The assumptions and techniques to allocate queries often differ depending on the context and the applications' goals. To the best of our knowledge, the problem of allocating

queries by considering *qlb* and the participants' intentions has not received much attention and is still an open field. In the following, we discuss five main domain related to our query allocation framework : data mediators, multi-agent, web services, load balancing, and economic approaches.

2.7.1 Data Mediator Systems

Over the last years, data mediator systems [Wie92] have been accepted as a viable approach for integrating heterogeneous and distributed providers. Data mediators allow consumers to query different providers that are typically wrapped to provide an uniform interface to a mediator. Two of the most prominent approaches are TSIMMIS [GMPQ$^+$97] and Information Manifold [LRO96]. In data mediator systems, the mediator allocates queries to providers and integrates results for consumers, much like distributed database systems [ÖV99]. Nevertheless, data mediators require some global information such as global schemas [TRV98], which is difficult to maintain in dynamic systems because source schemas change frequently. S_bQA does not require any global knowledge, but it does not address the data integration problem.

2.7.2 Multi-Agents

Multi-agent systems (MAS) focuses on systems in which many intelligent and autonomous agents interact with each other [Syc98]. Agents can share a common goal or can pursue their own interests, i.e. they can interact in a cooperative or selfish way. MASs have been used in recent years for creating applications in dynamic, very large distributed environments, such as the Internet. In particular, MASs are frequently used if there are different organizations with different goals and proprietary information want to interact with each other. In MASs the query allocation (known as task allocation) is the problem of assigning responsibility and problem-solving resources to an agent. In this context, the MAS designer can make the assignment of a set of queries, but this approach is inflexible and inadequate for environments with a high degree of dynamism and openness.

Davis and Smith [DS83] focused on the issue of flexible query allocation to multiple agents, whose work resulted in the well-known *Contract Net Protocol* (CNP) [Smi81]. Given a query to allocate, this protocol consists in four interaction phases, involving two roles : *contractor* and *bidder*. First, the contractor agent announces the query (or set of queries), to be performed, to its neighbors (the bidders). Second, the bidder agent replies its intention, via a bid, to perform the query (or each query). Third, the contractor agent collects all bids from bidder agents, compares the collected bids, selects the best of them according to its own criteria, and allocate the query accordingly. Finally, the selected bidder agent confirms its intention to get such a query. This protocol has three great limitations : *(i)* it cannot detect or resolve conflicts, *(ii)* it is network communication intensive, and *(iii)* contractors do not inform bidder agents of the query allocation result (only the selected bidder agent is informed).

Improvements to CNP have been proposed and we briefly review some of these here. *Sandholm* [San93] proposes the TRACONET system, which uses a variant of the classical CNP to allow negotiation over the exchange of bundles of resources. *Sandholm and Lesser* [SL95] present ways of varying the stage of commitment and how to implement varying levels of commitment. This allows a more flexible local deliberation a a wider variety of negotiation risk by allowing agents to back out of contracts. *Sycara* [Syc97] presents a model based on the financial option pricing theory to achieve flexible contracting schemes in uncertain environments. This model allows studying contingent contracts involving multiple contractors and bidders in an uncertain environment. *Souza et al.* [SRN03] present a new version of the CNP where bidders first propagate constraints between them so as to guarantee the coherence of different operations

related to the same task (query). *Aknine et al.* [APS04] point out that when many contractors negotiate simultaneously with many contractors, CNP can lead to unsatisfactory results. Thus, they introduce the pre-bidding and pre-assignment phases before the bidding and assignment phases, respectively, of the CNP.

Even the above works, CNP remains a simple protocol and thus there is no control to regulate the system. Furthermore, it is generally assumed a rather small number of agents and a detailed description of the conditions of execution. To overcome this difficulty, several approaches of middle-agents have been defined in the literature [DSW97, GKD97, NBN99, NFK+00]. A classical task of a middle-agent is to locate and bind bidders (providers) with contractors (consumers) in dynamic environments. The basic mediation process done by middle-agents has the following form. First, providers register their capabilities to a middle-agent, which stores these advertisements in a local register. Languages to advertise capabilities have been defined, e.g. [**?**]. Second, consumers send their queries to a middle-agent, which match it with its local register. Finally, the middle-agent returns the set of relevant providers or the result of the query treatment. Most of the work in this context have focused on the matchmaking problem [AEK+00, KH95, **?**]. A survey can be found in [KS01]. All these works are efficient but the number of selected providers may remain too large.

Therefore, some works have investigated the possibility of reducing the list of selected providers. *Zhang and Zhang* [ZZ02] propose to perform classical matchmaking and then refine the result list of relevant providers by considering the providers' quality. In *Ono et al.* [ONK+03], the middle-agent collects and maintains private "word-of-mouth" trust information as well as capabilities from each agent and uses this information for personalized trust-based mediation for each agent. This mediation is performed by the middle-agent through mediation protocols and a trust propagation mechanism. However, the participants' *intentions* are not considered by these works, which does not allow a participant to have an active participation in the selection process.

D. Bernstein *et al.* [BFLZ03] propose an adaptive approach to allocate queries, in file sharing-systems, based on the machine learning methodology. In this approach, a consumer can perform partial downloads from providers before finally settling on one. This approach allows the consumers to improve response times by aborting bad download attempts until an acceptable provider is discovered. However, the authors inherently assume that consumers are only interested in response times and providers have no interests to perform queries. *Gorobets and al.* [GN04] propose uses an economic approach to model dynamic systems of interacting agents. In this approach, a consumer sends queries to providers according to its most preferences, i.e. the most preferred providers are first contacted by a consumer. A provider is free to accept or reject a query in accordance to its preferences. If one query is rejected, the consumer sends it to less preferred providers according to its ordering. This is repeated until the query is accepted or the tolerance threshold is reached. In this latter case, the consumer carries out the query itself. After this matching phase, the selected provider performs the query and delivers results to the consumer.

Nevertheless, [GN04] cannot adequately consider both consumer's and provider's preferences. It may lead to two opposite cases : a consumer-centric or a provider-centric case. On the one hand, the consumer-centric case occurs when the first consumer's preferred provider accepts to deal with the query. In this case, the proposed approach may discard more interested providers than the the selected and thus leading to their dissatisfaction. On the other hand, the provider-centric case occurs when no provider wants to deal with the query, which may lead to the consumer dissatisfaction.

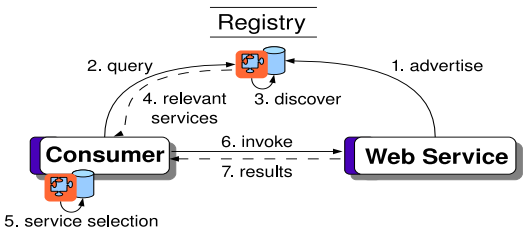


Figure 2.11 – Cycle of a Web service invocation.

2.7.3 Web Services

Web services [web] are Internet-based, distributed modular applications that provide standard interfaces and communication protocols aiming at efficient and effective service integration. They are rapidly becoming a standard for sharing data and functionality among loosely-coupled, heterogeneous systems and started to show their usefulness in wide variety of domains such as business-to-business integration, e-sourcing, and business process management. Thus, many enterprises are moving towards a *service oriented architecture* by putting their databases behind Web services, thereby providing a well-documented, inter-operable method of interacting with their data.

Figure 2.11 illustrates the way in which a Web service is typically invoked by a consumer. Web services (providers) advertise their capabilities to a registry, such as the *Universal Description Discovery and Integration* (UDDI) [udd], to make itself known and available to consumers. UDDI specification is a standard for service discovery and is designed to function in a way similar to yellow pages, where business and services can be looked up by name or by a standard service taxonomy. The advertisement of Web services is a an essential precondition for a transaction to take place. The registry directory can be hosted and managed by a trusted entity (centralized approach) or by several Web services (in a peer-to-peer fashion) [ACKM04]. To locate a Web service, a consumer send its query to the registry, which has to find the Web services advertisement that match the query. As a result, the consumer receives the set of Web services that are able to deal with its query, among which it has to select the service that it prefers. Then, it invokes the selected Web service to process its query. To support this interaction, Web services must declare, using e.g. *Web Services Description Language* (WSDL) [CCMW], what information it needs from the consumer, in what order, and in which format it expects this information.

As Web services may change frequently or consumers requirements and preferences may also change, Web services selection is a key challenge. A typical way to do so is that consumers analyze each service description and select the service that more closely fits its needs. Nevertheless, given the great number and diversity of services, the selection of the right service becomes a hard task for consumers. Considerable effort has focused on the semantic description of Web services and then a simple solution may be to select those Web services having the highest semantic score [MSZ01, PSK03, wsm]. However, this way to select Web services does not always fit the consumers' preferences. Thus, adequate techniques for dynamic Web service selection is still needed.

To address this problem, several Web service discovery mechanisms have been proposed with the aim of narrowing down the Web service selection based on *quality of service* (QoS for short). *Maximilien and Singh* [MS04a] propose an agent framework coupled with QoS for dynamic Web services selection. QoS is collaboratively determined by participants via the agent framework. Ran [Ran03] extends the traditional Web service discovery model by adding a new role called a *Certifier*, which verifies the advertised QoS of a Web service before its registration. Consumers verify the advertised QoS with the Certifier before invoking a Web service. *Liu et al.* [LNZ04] present an extensible and flexible QoS-based selection

model that takes into account the feedback from users as well as other business related criteria. All these works help consumers to have satisfactory transactions with the Web services.

Reputation mechanisms have been also proposed to narrow down the Web service selection. *Majithia et al.* [MARW04] propose a framework for reputation-based semantic Web service discovery. This framework supports different contexts that either refer to particular application or particular types of users. A weight is attached to each particular context, which reflects its importance to a particular set of users. *Maximilien and Singh* [MS04b] propose a concrete framework for Web service selection that considers the consumers' preferences and Web services' reputation. Their approach is based on an architecture and programming model in which applications and services are represented by agents. Works by [DD04, MP05] propose a Web service selection mechanism based on the consumers' past-experiences. A consumer report their experiences with Web services to global site, which is consulted by other consumers before selecting a Web service. *Jurca et al.* [JFB07] propose a reputation manager based on incentives for the clients to report honestly.

Moreover, Web service selection mechanisms based on other criteria than QoS and reputation have been proposed by several groups. For example, *Bonatti and Festa* [BF05] formalize three kinds of optimal service selection problems based on cost minimization and on two different quality maximization criteria. They also study the complexity of and propose suboptimal solutions for these three problems. *Balke and Wagner* [BW03] propose an algorithm to reduce the set of discovered Web services by considering the consumers' profile (preferences). Given a consumer's query and set A of discovered services (those that can deal with the query), this algorithm proceeds in four phases. First, it groups all discovered services by signature parameters and discards those that do not allows querying with all consumer's query terms (resulting set A'). Second, it obtains the service parameters that are not covered by the query. If existent, it gets the preferred values for these obtained parameters. Third, it expands the query with the preferred values and queries Web services in set A'. Finally, it collects results of all services in A' and orders by their importance.

Nevertheless, all of the aforementioned approaches consider providers as captive values and thus providers cannot express their preferences (or intentions) to perform queries. *Lamparter et al.* [LASG07] present a selection model that considers both consumers and providers preferences. This model is based on service configurations and associated preferences, which are both modeled in a formal way by attaching price information to property values. In this model, consumers and Web services show their preferences to perform queries via a *request* and *offer*, respectively. Given a query, the Web service selection mechanism obtains the utility of a Web service by computing the difference between the consumer's request and the Web service's offer. Then, it selects the Web service that maximizes such an utility. However, by adding offers and requests to work out the utility, a request (resp. an offer) may neutralize the offer (the request) whereby the selection mechanism does not correctly consider the offer (the request), which may lead to the dissatisfaction of the consumer (the Web service). Furthermore, Web services cannot express their preferences towards consumers.

2.7.4 Load Balancing Approaches

Query Load balancing is the act of distributing query load among a set of providers as evenly as possible. Generally speaking, a query load balancing method is composed of two main parts, a *load metric* and a *load policy*. The former is an estimator of the level of load or utilization of a given provider and the latter is the uses the load metric to make query load balancing decisions. For example, the load metric would be the number of queries in the run queue of providers and the load policy would be to send incoming queries to those providers with the smallest load. Since the load balancing literature is quite

extensive, we only describe the most relevant to our work.

According to their load metric, we can classify qlb methods into two approaches : *load based* and *capacity based*. Generally, in load based methods [ABKU99, GBGM04], load is defined as the number of queries that providers have in their run queue. These methods usually allocate queries to those providers with the highest inverse probability of their reported load. However, load based methods are not adequate for heterogeneous systems because they inherently assume that providers and queries are homogeneous. That is, unlike capacity based methods, they assume that providers have the same capacities to perform queries and that queries require the same computational resources to be treated by providers. Capacity based methods [MTS90, RM95, RB06, SKS92] already take into account such heterogeneity by defining providers' load (a.k.a. utilization) as the maximum query rate that a provider can treat. Then, a common approach is to allocate each incoming query to providers that have the highest available capacity, i.e. the least utilized, among a set of relevant providers. All these works mainly model and address the problem of minimizing the providers' load or utilization for improving system performance, such as short response times and high throughput.

We can also classify qlb methods regarding their load policy. In this classification, two approaches are well known : the shortest expected delay (*sed*) and the greedy throughput (*gt*) policies [KK92, WS, Zho88]. On the one hand, the *sed* policy attempts to minimize response times by always selecting those providers that faster perform queries at a given time. On the other hand, the *gt* policy strive to maximize the expected number of queries. However, unlike S_bQA, all above approaches do not consider participants' intentions, which drastically penalize participants' autonomy as seen in Chapter 2.6.

Several methods have been proposed for providers selection [BFLZ03, QL07, RB03] with the assumption that consumers selfishly want to choose providers that allow them to get results with short response times. But, they do not consider providers' intentions. Recently, many systems have been built on top of *distributed hash tables* (DHTs) [RFH+01, SMLN+03, RD01] and several solutions have been proposed to address load balancing problem [AHKV03, AMZ03, DW01]. Nevertheless, they do not consider participants' intentions.

2.7.5 Economic Approaches

Economic approaches can claim to take into account the participants' intentions and have been shown to provide efficient query allocation in heterogeneous systems [FYN88, SAL+96]. A survey of economic models for various aspects of distributed system is presented in [FNSY96].

Mariposa [SAL+96] is one of the first systems to deal with the query allocation problem in distributed information systems using a *bidding* process. In Mariposa, all the incoming queries are processed by a *broker site* that requests providers for *bids*. Providers bid for acquiring queries based on a local bulletin board. Then, the broker site selects a set of bids that has an aggregate price and delay under a bid curve provided by the consumer. Mariposa ensures a crude form of load balancing by modifying the providers' bid with the providers' load. Nevertheless, our experimentations show that, in some cases, providers suffer from overutilization. Besides, queries may not be treated even if providers exist in the system. This leads to a certain domination of the providers' intentions over the consumers' intentions.

In [PI06], the authors focus on the optimization algorithms for *buying* and *selling* query answers, and the negotiation strategy. Their query trading algorithm runs iteratively, progressively selecting the best execution plan. At each iteration, the buyer sends requests for bids, for a set of queries, and sellers reply with offers (bids) for dealing with them. Then, the buyer finds the best possible execution plan based on the offers it received. These actions are iterated until either the found execution plan is not better than the plan found in the previous iteration or the set of queries has not been modified (i.e. there

is no new subqueries). This approach uses some kind of bargaining between the buyer and the sellers, but with different queries at each iteration. However, this way of dealing with subqueries optimization is orthogonal to our proposal and one may combine them to improve performance. In [LCLV07], the authors propose an economic *flexible mediation* approach that allocates queries by taking into account the providers' *quality* (given by consumers) and the providers' bids. In contrast to our approach, the authors inherently assume that participants are captive. In addition, their proposed economic model is complementary to our proposal and one can combine them to obtain an economic version of S_bQA.

2.8 Chapter Summary

We considered large-scale distributed information systems where participants are free to leave the system at any time and have special interests towards queries. In this context, it is crucial to consider the participants' *intentions* to allocate and perform queries so that their intentions, response times, and system capacity are ensured. We proposed, in this chapter, a general and complete framework, called S_bQA, to allocate queries among providers by considering the participants' intentions in addition to *query load balancing* (*qlb*). The originality of S_bQA is to perform all query demand while satisfying participants' intentions. In summary, our main contributions are the following.

- We proposed a manner to compute consumers' intentions that considers their preferences and providers' reputations. The particularity of this approach is that it affords consumers the flexibility to trade their preferences for providers' reputation in accordance to their experience with providers.
- We proposed a manner to compute providers' intentions, which allows providers to trade their preferences for their utilization while keeping their strategic information private. The main idea behind this approach is that providers be sensitive to workload variations so that they pay more attention to their utilization when they becomes overutilized.
- We proposed a query mediation mechanism that considers both consumers' and providers' intentions. The four strong points of this proposal is that :
  - It allows a mediator to trade consumers' intentions for providers' intentions according to their satisfaction.
  - It strives to balance queries at runtime via the participants' satisfaction, thus reducing *starvation*.
  - It affords a mediator the flexibility to regulate the system with respect to some predefined function and to adapt the query allocation process to the kinds of application.
  - It can ensure good levels of satisfaction as far as the system is adequate to participants and *vice versa*, which allows participants to reach their intentions in the system whether they correctly work out their intentions and preferences.
- We evaluated and compared S_bQA with two baseline query allocation methods ($Capacity\ based$ and *Mariposa-like*), in two kinds of environments : *captive* and *autonomous*. We showed through experimentation that, by considering together the *qlb* and satisfaction of participants, S_bQA significantly outperforms both baseline methods. We observed that participants are, in general, very satisfied with S_bQA and *Mariposa-like*, which is not the case for $Capacity\ based$ that suffers from several providers' departures due to dissatisfaction. However, *Mariposa-like* has serious problems for balancing queries correctly. On the one hand, we showed that, unlike the baseline methods, S_bQA maintains the *high*-interest, *high*-adaptation, and *high*-capacity providers in the system. On the other hand, the results show that while baseline methods lose more than 20% of consumers (for all workloads), S_bQA has no consumer's departures ! We showed the self-adaptability of S_bQA to the intentions and satisfaction of participants. We also discussed its adaptability to different kinds

of applications. All these results demonstrate that S_bQA can scale up with autonomous participants, while $Capacity\ based$ and *Mariposa-like* cannot.

Future Work In this chapter, we stressed the importance of studying many different ways in which participants can compute their preferences and intentions. We desire to study this in a future work so as to understand which can be the best strategy to adopt by a participant given its context and the application.

CHAPTER 3

Scaling Up Query Allocation

In large-scale, heterogeneous information systems, mediators are widely used to perform query allocation [ÖV99]. A set of participants (consumers and providers) with at least one of them playing the role of mediator form a *Virtual Organization* (*VO*). The main function of a mediator is to allocate each incoming query to providers that can answer it. As noted so far, we consider that participants may leave a mediator at will and may express their intentions to allocate and perform queries. In these environments, it is important to consider participants' intentions to avoid they leave a mediator by dissatisfaction. In previous chapter we presented a query allocation framework, called S_bQA, which considers intentions and current satisfaction of participants. We experimentally demonstrated that S_bQA has very good system performance when performing the query allocation task in distributed systems with a single mediator (mono-mediator VO). However, a mediator may quickly become a single point of failure for its VO as well as a potential performance and scalability bottleneck. This is why it is crucial to have more than one site that cooperatively play the role of mediator. In this case, S_bQA does not scale well because it considers current participants' satisfaction, which a mediator can no longer compute itself as it also depends on the query allocations made by other mediators. Thus, when allocating a query, a mediator should keep informed all other mediators of the mediation results to update participants' satisfaction. This tends to increase significantly the network traffic.

A way to avoid such a traffic overhead between mediators is that providers express their interest for queries through "monetary" bids. Thus, the mediators no longer consider the providers' satisfaction but only their bids. This requires introducing some "virtual" money to be used by providers and mediators. In this case, virtual money is totally disconnected from the real money we use in current life. As S_bQA is not designed to deal with bids, this also requires to consider other methods able to consider bids and possibly other elements to allocate queries. Several works use microeconomic methods to allocate queries or resources in distributed systems [DVR$^+$07, LCLV07, PI07, SAL$^+$96]. But, to our knowledge, no microeconomic method has ever been evaluated through a measure that is outside the microeconomic scope like satisfaction.

Therefore, in this chapter, our goal is twofold. First, with the aim of scaling query allocation up to several mediators, we want to adapt S_bQA to systems with several mediators (multi-mediator) so that it ensures as good system performance as in mono-mediator systems, i.e. as previous chapter. Second, evaluate by the first time microeconomic methods from a satisfaction point of view. The content of this chapter is based on our material published in [QRLCV07a, QRLCV07b, QRLCV08]. Our main contributions are the following :

- We discuss the challenges of using virtual money as a means of regulation in the query allocation process and make precise how the virtual money circulates within the system.
- We propose *Economic Satisfaction-based Query Allocation* method ($\$_bQA$, for short). Generally speaking, $\$_bQA$ is S_bQA using virtual money. In particular,

  - We define a way in which a provider computes its bid by considering its preferences, its satisfaction, its current utilization, and its current virtual money balance. Also, we propose three strategies that allows a provider to bid for queries in the presence of several mediators.
  - We define how a mediator allocates queries by considering both consumers' intentions and providers' bids. And, we define how a mediator should invoice providers.
  - We state the communication cost of $\$_bQA$ and demonstrate that its additional cost with respect to S_bQA is not high.
- We analytically demonstrate that $\$_bQA$ allows a VO to scale up to several mediators with no additional network cost with respect to a VO with a single mediator.
- Finally, from a methodological point of view, it is important to compare different microeconomic methods (included $\$_bQA$) with a non-microeconomic method using satisfaction as a "money independent" measure.

The rest of this chapter is organized as follows. We state in Section 3.1 the problem we address. In Section 3.2, we stress the challenges of using virtual money as a means of regulation in the query allocation process and make precise the flow of virtual money. As part of the $\$_bQA$ method, in Section 3.3, we define a way to compute providers' bids and propose three strategies that allow a provider to bid in the presence of several mediators. And, in Section 3.4, we present a mediation mechanism to allocate queries by considering both consumers' intentions and providers' bids, and present a way to invoice providers. In Section 3.5, we analytically demonstrate that $\$_bQA$ can easily scale up to several mediators. In Section 3.6, we compare $\$_bQA$ with two microeconomic methods and one non-microeconomic method and validate $\$_bQA$'s performance in multi-mediator systems. Finally, we present related work in Section 3.7 and conclude in Section 3.8.

3.1 Problem Statement

We assume a distributed system to be a set $\mathcal{I}$ of participants which form a *Virtual Organization* (VO). Each participant of a VO can play one or more of the following roles : *consumers* which send queries ; *providers* which answer queries ; and *mediators* which allocate consumers' queries to providers. The set of participants playing the role of consumer, resp. provider and mediator, is noted C, resp. P and M. We assume that a VO may be *x-redundant*, there are several mediators acting as a single one by behaving cooperatively. The introduction of redundancy into the assignment of mediators has been proved to have gains in performance [YGM03]. We formally define a x-redundant VO in Definition 27 and illustrate a x-redundant VO with 3 participants playing the role of mediator in Figure 3.1.

Definition 27. x-redundant VO *A VO is said to be $x - redundant$ if and only if there is a set M ($||M|| > 1$) of participants playing the role of mediator and each provider in P is connected to each mediator in M.*

Mediators are responsible to allocate consumers' queries to providers and hence it is up to them to make everything work well. Intuitively, a mediator should allocate queries so that good system performance is ensured. If performance is linked to clear notions in distributed systems (such as load distribution and answering time), intentions and satisfaction are less usual. However, participants' satisfaction has a deep impact on a system general behavior, particularly when the participants are autonomous. Indeed, they can decide on their own either to enter a VO with the hope of improving their lot or to leave it because of dissatisfaction. But also, they have special interests towards queries.

Consumers formulate queries as in previous chapter, i.e. in a format abstracted as a triple $q =< c, d, n >$. Recall that c denotes the consumer identifier that issued q, d the task to be done, and n the

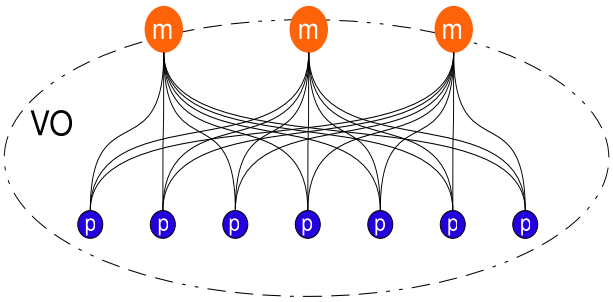


Figure 3.1 – A x-redundant VO with 3 mediators.

number of answers that c wishes to obtain. Because of autonomy, a consumer may be interested in the way its query is treated. So, it should have some intention of how the system allocates its query q among providers. Recall that those intentions are denoted by vector $\overrightarrow{CI}_q$. By convention, values are in $[-1..1]$.

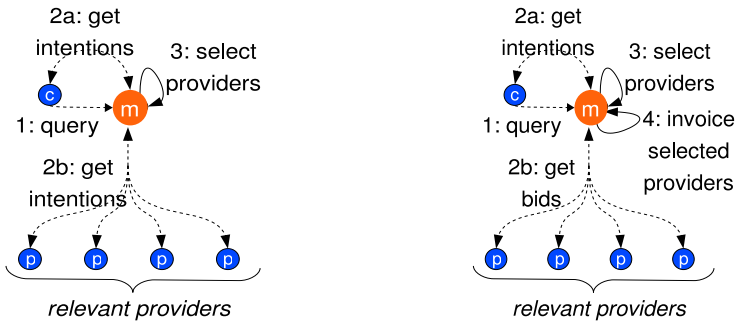
Providers are heterogeneous : *(i)* they have different processing capabilities and *(ii)* they may provide different results, for example because they have different private data. The former point means that some providers can treat more queries per time unit than others. The *utilization* of a provider $p \in P$ at a given time t, $\mathcal{U}_t(p)$, is defined as p's load with respect to its capacity. In other words, function $\mathcal{U}_t(p)$ denotes the total cost of the queries that have been allocated to p and have not already been treated at time t. Because of autonomy, a provider may prefer to perform some queries than others so that it fulfills its objectives. Thus, as for a consumer, a provider is simply not satisfied when it does not get what it expects.

We demonstrate in previous Chapter that, in these environments where participants are autonomous, it is crucial to consider participants' intentions and satisfaction when allocating queries to avoid massive participants' departure from the system and hence to preserve the total system capacity. To do consider providers' satisfaction in x-redundant VO is challenging since a provider receives queries from different mediators transparently and hence providers' satisfaction depends on the query allocations made by all mediators in the x-redundant VO. In this case, S_bQA cannot perform as well as in a VO with a single mediator because the participants' satisfaction computed by each mediator is local and hence it is different. A simple solution is that consumers send their current satisfaction with their queries and that the x mediators in a x-redundant VO frequently exchange messages to update providers' satisfaction. Nevertheless, these up-to-date messages considerably increase the network traffic and may hurt system performance. Furthermore, there will be always a time interval where satisfaction is not the same at all mediators because of network latency. Thus, we define the query allocation problem we address in this chapter as follows.

Problem Statement Let P_q denote the set of providers that can deal with a query q. Given a x-redundant VO with autonomous participants, a mediator $m \in M$ should allocate each incoming query q to a set $\widehat{P_q} \subset P_q$ such that $||\widehat{P_q}|| = min(q.n, N)$ as well as good system performance and participants' satisfaction are ensured in the long long-run with a low network cost.

3.2 Use of Virtual Money

A way to avoid the traffic overhead between mediators due to providers' satisfaction updates is that providers express their intentions for queries through "monetary" bids. Thus, a mediator no longer considers the providers' satisfaction but only their bids. This requires introducing some "virtual" money to



(a) SQLB-based (b) virtual money-based

Figure 3.2 – General system architecture. Site m denotes the mediator, c a consumer, and p a provider.

be used by providers and mediators. In this case, virtual money is purely virtual and is totally disconnected from the real money we use in current life. We could speak of tokens or jetons as well. This point has to be stressed upon for two main reasons. First, we do not focus on any particular business model : we only use the virtual money as a means to regulate the query allocation in the system. Indeed, after a consumer has decided which providers it chooses, it might give *real money* to them because it uses their services. This point is far beyond the focus of this thesis. Second, when using real money, one can assume that consumers and providers get money from elsewhere. For example, when designing an auction mechanism for e-commerce one can assume that people spend the money they have earned by working (in real life). When dealing with virtual money, one can no longer make such assumptions. In fact, the general architecture of the virtual money-based system is almost the same as when one does not use virtual money (see Figure 3.2), but we must deal with the four following additional points :

1. We must make precise the way in which virtual money circulates within the system since the regulation of the system depends on it. This is a difficult task since it is a macroeconomic concern and hence one must have a clear idea of the global system behavior. Besides, the policy used to regulate the virtual money flow also depends on the query allocation method.

2. Providers no longer express their intentions directly to the mediator. Instead, they express their intentions through *bids*, which also consider their current virtual money balance as well as their strategy to bid. This is challenging because a provider should compute its bid so that it generally obtains the queries it prefers and do not become overloaded, which may degrade its offered services. But also, a provider should not spend all its current virtual-money balance in a given query allocation so as to have chances to get interesting queries in the future.

3. A mediator must compare and select providers based on their bids instead of their intentions. But, in our context, this selection process is challenging because of consumers' intentions, which do not allow a mediator to select providers based only on their bids. Furthermore, the mediator must satisfy both consumers and providers in the long-run.

4. As we consider systems where providers have to "pay" for performing or receiving queries, a mediator must invoice providers after each query allocation. This is in some way similar to an invoicing such as that of Google AdWords [goo], except we consider virtual money. Invoicing providers in our context is challenging because some providers may be imposed a query that they do not desire to perform.

In the next section, we make precise the way in which the virtual money circulates within the system (point 1) and, in the remainder of this chapter, we discuss how we address the other 3 points so as to adapt S_bQA to use virtual money as a means of regulation.

3.2.1 Flow of Virtual Money

The way in which virtual money circulates within a system is a macroeconomic concern and hence we adopt a simple solution. First of all, only the providers and mediators deal with virtual money, i.e. the consumers still show their intentions. Providers spend and earn virtual money through a mediator only. On the one hand, they spend money by bidding on queries and to compensate other providers that have been imposed a query. On the other hand, they earn money when they are imposed a query that they do not desire to perform. Every time a provider has been allocated a query, has been imposed a query, or has been required to compensate an imposed provider, it is informed, by the concerned mediator, of the amount of virtual money it payed or won (in the case of imposition). Of course, a provider is completely responsible of its virtual money balance and hence no provider can spend the virtual money of another one. Therefore, a provider always has an exact mirror of its virtual money balance in local.

In contrast to providers, in our mediation process, mediators never looses money, but tends to accumulate money coming from the providers in the course of time, thus making the providers poorer and poorer. Indeed, this could distort the mediation process or even block the system when the providers no longer have money. A simple solution has been adopted : in case a mediator has earned an amount of virtual money above a defined threshold, it distributes such an amount of virtual money to providers in an equitable way. From the providers' point of view, this is another, regular, way of earning money. We assume that there exists a *trusted third-party* in the system that plays the role of bank, which is in charge of the flow of virtual money. Several ways to implement the bank exist (using a DHT [SMLN+03] for example), but this is well beyond the scope of this thesis. Without any loss of generality, we only assume that there is a bank entity that allows mediators to control the flow of money in the system. For clarity, we omit the bank in the remainder of this paper when we talk about the virtual money balance of a provider. Indeed, the flow of virtual money requires some network messages. We state this cost in the following proposition.

Proposition 1. *Only* 3 *messages per query are required to control the flow of virtual money.*

Proof (Sketch). First of all, we assume a *vickrey* auction to allocate queries and hence no message is required by a provider to discover the bids of other providers. Similarly, at first glance, a provider may require a network message to know its current virtual money balance so as to bid for queries. However, a mediator informs a provider of any change in its virtual money balance, thereby allowing a provider to always know its current virtual money balance. Thus, no network message is required by a provider to know its virtual money balance. Now, before a query mediation, two network messages are exchanged between a mediator and the bank in order to validate the bids of providers, i.e. to verify if they have enough virtual money that support their bids. After a query allocation, the concerned mediator sends another network message to the bank, which is in charge of invoicing providers. Notice that the bank requires no network message to invoice providers since it has both providers' and mediators' virtual money balance in local. Finally, every time that the amount of virtual money earned by a mediator exceeds a given threshold, the bank should inform the mediator, which replies with a virtual money distribution table. However, this depends on several external factors, such as the kind of incoming queries and the strategies of providers to bid, which we cannot predict a priori. Furthermore, these messages are expected to occur after a large number of incoming queries and hence they do not impact the system

Table 3.1 – Virtual money balance along a sequence of mediations.

		p1	p2	p3	p4	p5	m
init	CI	0.9	0.8	0.8	0.6	0.6	
	σ	20.00	20.00	20.00	20.00	20.00	0.00
q1	B	2.00	2.00	2.00	2.00	2.00	
		*	*				
	σ	**18.16**	**18.00**	20.00	20.00	20.00	**3.84**
q2	B	0.73	0.72	2.00	2.40	3.20	
					*	*	
	σ	18.16	18.00	20.00	**17.63**	**17.63**	**8.59**
q3	B	-4.00	0.72	-18.00	-6.00	0.35	
			*			*	
	σ	18.16	18.00	20.00	17.63	17.63	8.59
q4	B	-18.00	-12.00	-8.00	0.35	-2.00	
					*	*	
	σ	**16.05**	**15.89**	**17.89**	**15.51**	**22.51**	**12.15**
q5	B	1.60	1.59	1.79	1.55	2.25	
				*		*	
	σ	16.05	15.89	**16.14**	15.51	**20.42**	**15.99**
Distr	σ	**19.24**	**19.09**	**19.34**	**18.71**	**23.62**	**0.00**

$\omega = 0.5\,;\, n = 2$

performance. Thus, we neglect these messages in this analysis. To summarize, providers do not generate messages while mediators need 2 messages to validate bids and 1 message to invoice providers. □

Table 3.1 illustrates the flow of money along a sequence of five mediations followed by redistribution of money by a mediator, where parameter ω ensured the balance between consumers' and providers' interests and n stands for the required answers by the consumers. At the initiation step, we quote the providers' and mediator's initial money balance (σ), and the consumer's intentions with regards to the providers (which we assume are constant across these five mediations). Then, for each provider and each query, we quote the bid (B), those which are allocated the query ($*$) and the new money balance (σ). Each time there is a change in provider's money balance (respectively of the mediator), the new value is in bold face. Notice that the allocation of q_3 is neither a competition nor an imposition, thus there is no change in the money balances. After the five mediations, the mediator distributes the money it has piled up (15.99) among the five providers.

3.3 Provider's Side

In this section, we discuss how providers express their interests towards queries in x-redundant VOs. In particular, we define in Section 3.3.1 the way in which a provider computes its bids to perform queries and propose, in Section 3.3.2, three heuristics to manage providers' virtual money balance.

3.3.1 Computing Bids

The way in which a provider compute its bid is independent of the query allocation method. This amounts to consider truth-revealing providers, which rationally bid according to their preferences and load. A simple way to obtain providers' *bid*, is that each provider maintains a local bulletin board, which contains a billing rate for its resources based on its preferences to perform queries (denoted by function $prf \in [-1..1]$). Then, a provider's bid for getting a query may be the product of its current utilization by the billing rate, such as in [SAL+96]. However, in our case, the context of a provider is more complex

because we have to consider its current satisfaction and current virtual money balance (denoted by bal_p) in addition to its preferences and load.

Thus, a provider first works out its intention to perform a given query q as defined in Definition 25 and then it proceeds to work out its bid to perform such a query. Intuitively, the bid of a provider may be the product of its intention by its current virtual money balance. The current virtual money balance of a provider p is denoted by bal_p. Nonetheless, such a procedure may lead a provider to spend all, or almost all, its money on only one query. To avoid this, a provider offers at most only a defined percent of its current virtual money balance, denoted by the constant c_0 whose values are in $]0..1]$. Having said all this, we formally define the bid of a provider as follows.

Definition 28. Provider's Bid *Given an incoming query q, a provider $p \in P_q$ computes its bid to perform q, $bid_p(q)$, as follows,*

$$bid_p(q) = \begin{cases} (prf_p(q)^{1-\delta_s(p)}) \times (1 - \mathcal{U}_p(t))^{\delta_s(p)} \times (bal_p \cdot c_0) & if (prf_p(q) > 0) \wedge \\ & \wedge (\mathcal{U}_p(t) < 1) \\ -\big((1 - prf_p(q) + 1)^{1-\delta_s(p)} \times (\mathcal{U}_p(t) + 1)^{\delta_s(p)}\big) \times c_1 & otherwise \end{cases}$$

As a provider may be paid by others if it is imposed a query, constant c_1, is set to the initial virtual money balance of a provider so that, in the worst case, a provider obtains what it got when it joined the system. The idea behind the above definition is that a provider always sets a positive bid when it desires to perform queries and it is not overutilized, otherwise it sets a negative bid. In traditional microeconomic-based methods providers do not bid (or give a null bid) when they do not desire to perform a query. However, this does not allow them to express how much unpleasant it is for them to perform a query and how much overloaded they are. This is why we allow a provider to make negative bids. At first glance, there is no difference between providers' bids and intentions, but by showing bids providers can keep private their real intentions, which is crucial in competitive environments.

3.3.2 Bidding in the Presence of Several Mediators

When a provider receives queries from different mediators, it should pay special attention to the way in which it computes its bid so that it never bids more than its current virtual money balance. To overcome this difficulty, we propose 3 heuristics that allow a provider to manage its virtual money balance in x-redundant VOs. Before going to present these heuristics, let us say that, after the bidding phase of a query q (i.e. the moment at which providers bid for q), a provider p locally stores in vector $\overrightarrow{CB}_p$ its bid $bid_p(q)$ and removes such a bid from $\overrightarrow{CB}_p$ when it receives the invoice for such a query (how providers' invoice are computed by a mediator is the focus of Section 3.4.2).

Optimistic An optimistic provider assumes that it gets all those queries to which it bids positively and that it does not get those to which it expressed a negative bid. Thus, an optimistic provider p modifies its current virtual money balance after the bidding phase of a given query q as follows,

$$bal_p = \begin{cases} bal_p - bid_p(q), & if\ bid_p(q) > 0 \\ bal_p & else \end{cases} \tag{3.1}$$

Then, when provider p receives the final invoice $bill_q(p)$ concerning query q from the respective mediator, it sets its virtual money balance as below equation.

$$bal_p = \begin{cases} bal_p + \overrightarrow{CB}_p[q] - bill_q(p), & if\ bid_p(q) > 0 \\ bal_p - bill_q(p) & otherwise \end{cases} \tag{3.2}$$

Preventive A preventive provider assumes that it gets all those queries to which it bids, independently of its bid value. In other words, conversely to an optimistic provider, it also assumes that it gets those queries to which it made a negative bid. Thus, a preventive provider p modifies its current virtual money balance after the bidding phase of a given query q as follows,

$$bal_p = bal_p - bid_p(q) \tag{3.3}$$

and when p receives the final invoice of query q, it sets its virtual money balance as follows.

$$bal_p = bal_p + \overrightarrow{CB}_p[q] - bill_q(p) \tag{3.4}$$

Pessimistic A pessimistic provider assumes, conversely to an optimistic and a preventive provider, that it never gets the queries to which it bids. Thus, a pessimistic provider p does not modifies its virtual money balance after bidding for queries. It therefore modifies its current virtual money balance when it receives the final invoice from the concerned mediator as follows.

$$bal_p = bal_p - bill_q(p) \tag{3.5}$$

3.4 Mediator's Side

Let us consider the allocation of some query q initiated by some consumer $c \in C$. The providers in P_q bid on q. Providers' bid are only public to the mediator and other participants cannot know such values. Bids are represented by a vector $\overrightarrow{B}$, with $\overrightarrow{B}[p] \in \overrightarrow{R}$ for all $p \in P_q$. If a bid is positive, the higher it is the more p wants to be allocated q. If it is negative, the lower it is the less p wants to treat q. Intuitively, the bid of a provider p reflects its intention to perform q. Thus, this should lead to the providers' satisfaction. However, if only bids are considered as several other approaches [FNSY96, FYN88, PI07], a consumer may be dissatisfied either because its intentions with respect to providers are not considered (when it gets answers from providers it doesn't desire to deal with) or because some its queries are not performed (because no provider wants to treat them). Hence, to satisfy consumers, a mediator : *(i)* directly considers the consumer's intentions ($\overrightarrow{CI}$) ; and *(ii) imposes* the query when not enough providers desire to perform it, as in [ST01]. We detail the way in which a $\$_bQA$ allocates queries among providers in Section 3.4.1 and define the way in which it invoices providers in Section 3.4.2.

3.4.1 Computing Providers' Level

As S_bQA, a mediator using $\$_bQA$ allocates a query q to the $min(n, N_q)$ "best" providers, which are given by vector of ranking $\overrightarrow{R}$. Where $\overrightarrow{R}_q[1] = p$ denotes the best ranked provider and $\overrightarrow{R}[min(n, N_q)]$ stands for the worst ranked provider. Hence, $All\overrightarrow{oc}_q[p] = 1$ iff $\exists i, \overrightarrow{R}[i] = p$ and $i \leq min(n, N_q)$. Vector $\overrightarrow{R}$ is computed by a mediator regarding the providers' *level*, denoted by vector $\overrightarrow{L}$, which means that those providers having the highest levels are allocated queries. Thus, given an incoming query q, the level of each $p \in P_q$ is defined as the balance between the consumer's intentions and the providers' bid with regards to q (Definition 29).

Table 3.2 – $\$_bQA$ in a (a) competition and an (b) imposition case, with $\omega = 0.5$ and $n = 2$.

(a)

		p1	p2	p3	p4	p5	m
init	CI	0,9	0,8	0,8	0,6	0,6	
	bal	16,05	15,89	17,89	15,51	22,51	0,00
q5	B	1,60	1,59	1,79	1,55	2,25	
	Level	2,22	2,16	2,24	2,02	2,28	
	Rank	3	4	2	5	1	
	Alloc			*		*	
	Trans			**1,75**		**2,09**	
	bal	16,05	15,89	**16,14**	15,51	**20,42**	**3,84**

(b)

		p1	p2	p3	p4	p5	m
init	CI	0,9	0,8	0,8	0,6	0,6	
	bal	18,16	18,00	20,00	17,63	17,63	0,00
q4	B	-18,00	-12,00	-8,00	0,35	-2,00	
	Level	-3,16	-2,69	-2,24	1,47	-1,37	
	Rank	5	4	3	1	2	
	Alloc				*	*	
	Trans	**2,11**	**2,11**	**2,11**	**2,11**	**-4,89**	
	bal	**16,05**	**15,89**	**17,89**	**15,51**	**22,51**	**3,56**

Definition 29. *Provider's Level*

$$\overrightarrow{L}[p] = \begin{cases} (\overrightarrow{B}[p]+1)^{\omega} \times (\overrightarrow{CI}_q[p]+1)^{1-\omega} & if\ \overrightarrow{B}[p] \geq 0 \\ -(-\overrightarrow{B}[p]+1)^{\omega} \times (\overrightarrow{CI}_q[p]+1)^{\omega-1} & otherwise \end{cases}$$

It is worth noting that, conversely to the provider's score (Definition 26 of previous chapter), a mediator, using $\$_bQA$, does not set ω value with respect to participants' satisfaction. Using $\$_bQA$, ω value is static, i.e. it does not change in every query allocation. Values of ω are in the interval $[0..1]$. If $\omega = 0$, only the consumer's intentions are considered by the mediator, thus leading to providers dissatisfaction. Conversely, if $\omega = 1$, the mediator only considers bids, leading to consumers dissatisfaction. This is why a mediator should set parameter ω according to the importance that it wants to pay to the consumers' intentions and providers' bid. Indeed, given the level definition, we can note that a mediator might allocate q to a provider that does not desire to deal with q. We call this an imposition case, otherwise, we have a competition case.

Table 3.2(a) shows the case of a competition. The consumer asks for two providers, and more than two of them bid positively. Providers p_5 and p_3 are allocated the query because they get the two highest levels, respectively 2.28 and 2.24. Notice that the consumer's intention with respect to p_5 is lower than its intention with respect to p_3. Thus, p_5 only got the query because of its bid (2.25) which is higher than p_3's bid (1.79), meaning that it wanted the query more than p_3. Table 3.2(b) shows an imposition case where no provider but p_4 wants to treat the query, whereas the consumer asks for two providers. Provider p_5 is imposed the query because of both its bid (which is the highest negative bid) and the consumer's intention with respect to it, which leads to the value 1.47 of its level. In both tables, the tuple "Trans" (for money "Transfer") denotes the amount of virtual money providers must pay for the query allocation.

3.4.2 Invoicing Providers

As start point, a natural strategy to invoice a provider that have been allocated a given query is that it pays what it bids. This kind of invoicing process is also know as *first-price*. However, it is well know that following a *first-price* invoicing, providers (or bidders) are incited to shade their bids below their true value [MCWG95]. This may distort the whole system by having providers revealing false bids in order to maximize their satisfaction (e.g. revenues) while other providers reveal true bids and hence may suffer from dissatisfaction or query starvation. Therefore, we adopt a *second-price* invoice mechanism (a.k.a. *vickrey*), which has been proved to give providers an incentive to bid their true value [Vic61]. However, we cannot directly compare providers' bids because of consumers' intentions. To overcome this difficulty, we introduce the *theoretical bid* (proposed in [LCLV07]), which corresponds to the amount

that a provider should bid for reaching a level. With $\omega \neq 0$ and $\alpha = 1$ if $l \geq 0$ or $\alpha = -1$ otherwise, the theoretical bid of a provider p to reach a level l, denoted by function $b^{th}(p, l)$, is given by,

$$b^{th}(p,l) = \alpha \cdot max(((\alpha \times l)^{\frac{1}{\omega}}(\overrightarrow{CI}_q[p] + 1)^{\frac{\alpha(\omega-1)}{\omega}} - 1), 0) \tag{3.6}$$

For example, in Table 3.2(a), we have already noticed that provider p_5 gets a level slightly higher than p_3's, because of its higher bid and despite the lower consumer's intention. In fact, to come exactly to p_3's level, p_5 should bid 2.136 (theoretical bid). A mediator invoices providers based on above equation. Remember that a query may be allocate to several providers, hence the mediator also needs to invoice a provider with respect to the selection of another provider. As noted earlier, two cases could exist when allocating queries : competition and imposition.

In a competition case, i.e. when all selected provider expressed a positive bid, a provider allocated a query owes the amount of its theoretical bid to reach the level of the best provider that has not been allocated the query. And, it does not pay by the selection of other selected providers. Formally, we define the *partial bill* of a provider p w.r.t. the selection of a provider p', with $\overrightarrow{B}[p'] \geq 0$, as follows.

Definition 30. *Provider's Partial Invoice in a Competition Case*

$$bill_q(p,p') = \left| \begin{array}{ll} b^{th}(p, \overrightarrow{L}[\overrightarrow{R}_q[q.n+1]]) & if\ \ p = p',\ \ \overrightarrow{B}[\overrightarrow{R}_q[q.n+1]] \geq 0,\ \ and\ \ q.n < N_q \\ 0 & otherwise \end{array} \right.$$

An imposition case occurs when at least one provider is imposed the query, i.e. when one provider that does not desired to perform a query is allocated the query. Obviously, being imposed does not meet at all the intention of an imposed provider. Thus, it would not be fair if a provider pays for an imposed query. Hence, to keep it satisfied in the long run, the idea is to distribute the cost of the imposition of query q on *all* the providers in P_q (in the spirit of [ST01], but also considering the consumer's intentions). Then, having obtained a reward, the an imposed provider is more likely, in the future, to obtain the queries it expects (because it has more money) so leading to its satisfaction. We formally define the *partial bill* of a provider p w.r.t. the selection of a provider p', with $\overrightarrow{B}[p'] < 0$, as follows.

Definition 31. *Provider's Partial Invoice in an Imposition Case*

$$bill_q(p,p') = \left| \begin{array}{ll} \dfrac{-b^{th}(p, \overrightarrow{L}[\overrightarrow{R}_q[q.n+2]])}{N_q} & if\ \ p \neq p' \\ b^{th}(p, \overrightarrow{L}[\overrightarrow{R}_q[q.n+1]]) - \dfrac{b^{th}(p, \overrightarrow{L}[\overrightarrow{R}_q[q.n+2]])}{N_q} & else \end{array} \right.$$

Having defined the partial bill for both competition and imposition cases, the *bill* that a provider must pay for having obtained a given query is then defined as the sum of all its partial bills (Equation 3.7). Formally, a mediator invoices each provider $p \in P_q$ as follows,

$$bill_q(p) = \sum_{p' \in \widehat{P_q}} bill_q(p,p') \tag{3.7}$$

Overall, a selected provider never pays more than its own bid and only pays for the selection of other provider when the latter has been imposed the query. Moreover, the invoicing process we presented here never requires a mediator from a financial point of view.

3.4.3 Communication Cost

As for S_bQA, we analyze the communication cost of $\$_bQA$ in terms of number of messages that should be transferred over the network to perform a query. This is given by the following theorem.

Theorem 4. *The total number of transferred messages by $\$_bQA$ to perform a query is $3(N+2)+n$.*

Proof. Given a query q and set P_q of providers, the mediator first asks for $q.c$'s intention and P_q's bids, which return such an information to the mediator. The number of exchanged messages at this phase is $mssg_0 = 2N + 2$. Once received the participants' interests, the mediator verifies if providers have enough virtual money to support their bids. This requires $mssg_1 = 2$ messages between the mediator and the bank. Next, it computes the level of each provider in P_q as defined in Section 3.4.1 and ranks them according to their level. Having done this, the mediator invoices providers, informs all P_q providers of the mediation result, and waits for results from the n selected providers. The number of transferred messages at this phase is $mssg_2 = 1$ to invoice providers, $mssg_3 = N$ to inform providers, and $mssg_4 = n$ to receive results from selected providers. Finally, the mediator sends the results to consumer $q.c$, which implies one more network message, $mssg_5 = 1$. Thus, $Mssg = mssg_0 + mssg_1 + mssg_2 + mssg_3 + mssg_4 + mssg_5 = 3(N+2) + n$ total messages are exchanged by a mediator, using $\$_bQA$, to perform a query. □

Corollary 1. *The additional cost of performing $\$_bQA$ w.r.t. S_bQA is 3 network messages per query.*

Proof. Implied by Theorems 3 and 4. □

3.5 Cost of Federating Mediators

We refer to mediators federation as several mediators operating, from the query allocation point of view, as a single mediator. That is, several mediators operate in a cooperative way for a shared purpose. Of course, mediators federation comes at a cost. Independently of the query allocation method used by mediators in an x-redundant VO, mediators must have an updated providers list of the VO. This needs $||M|| - 1$ messages every time a provider enters or leaves a VO. We do not consider this network cost in the following because such messages are required by any query allocation method. Now, when dealing with a x-redundant VO, a mediator using S_bQA can no longer calculate the providers' satisfaction itself. This is because a provider uses several mediators and hence its satisfaction results from the queries obtained with all of them. Thus, each mediator must exchange information about providers' satisfaction after each query allocation, which significantly increases network traffic. We formally state this network cost in the following proposition.

Proposition 2. *Let Q denote a set of queries and N_Q denote $||Q||$. Given a set Q of queries arriving into a mediator m in M of a x-redundant VO, using S_bQA, mediator m must exchange $(||M|| - 1) \cdot N_Q$ messages to update providers' satisfaction.*

Proof. Since after the allocation of a query q the satisfaction of any provider $p \in P_q$ changes, a mediator $m \in M$ must send a message containing the new satisfaction values of providers in P_q to all $M \backslash \{m\}$ mediators in the x-redundant VO. Thus, given a set Q of incoming queries, a mediator must exchange $||M|| - 1$ messages N_Q times. □

This is a cost that makes S_bQA unsuitable for performing in x-redundant VOs where a large number of mediators act as a single mediator to achieve a shared purpose. In contrast, $\$_bQA$ has no network cost

when dealing with several mediators and continues to perform, from a satisfaction point of view, as in a mono-mediator VO. We state this in the following theorem.

Theorem 5. *$\$_bQA$ always satisfies (i) consumers and (ii) providers in a x-redundant VO as well as in a mono-mediator VO with no additional network cost.*

Proof. Consider a x-redundant VO, denoted by S_{vo} and a mono-mediator VO, denoted by S_m, consisting of the same set of participants P. Consider also that the incoming queries in S_{vo} are the same to those arriving in S_m. We prove both (i) and (ii) by contradiction.

(i) Assume to the contrary that, for the allocation of some query q, consumer $q.c$ is not equally satisfied by S_{vo} and S_m. If this is the case, we can know, by Definition 17, that S_{vo} allocated q to a set $\widehat{P_q}$ such that there exists at least a provider $p \in \widehat{P_q}' : p \notin \widehat{P_q}$, where $\widehat{P_q}'$ is the set of providers selected by S_m. Hence, we can know that the set of relevant providers found by S_{vo} is different to the set found by S_m. This implies that provider p is not connected to the mediator that allocated q in S_{vo}, which contradicts the definition of a x-redundant VO.

(ii) Assume to the contrary that a provider $p \in P$ is not equally satisfied by S_{vo} and S_m. Then, by Definition 19, we can know that p did not perform the same set of queries in S_{vo} as in S_m. This means that p is not connected to all mediators in S_{vo} so as to receive all queries it can perform, which contradicts the definition of a x-redundant VO.

Finally, given the provider's level definition (Definition 29), a mediator does not directly deal with providers' satisfaction because it is up to a provider to manage its virtual money balance so as to be satisfied in the long-run (Definition 28). Thus, the mediator has no message to exchange among mediators to update providers' satisfaction. Clearly, the invoice and bidding processes require computational resources of mediators and providers, respectively, but these costs are negligible because of capacities that current computers have. □

Above theorem shows that even if $\$_bQA$ generates 3 more messages per query than S_bQA (Corollary 1), it allows a VO, and hence a system, to scale up to as many mediators as the VO desires with no loss in system performance. To discuss how queries may be forwarded to other VOs (inter-VO query allocation) and the way in which a VO is created is well beyond the scope of this thesis.

3.6 Experimental Validation

Our three main objectives in this experimental validation are :

- To evaluate how well $\$_bQA$ selects and invoices providers and to analyze the impact of using virtual money as a means of regulation when performing query allocation.
- To analyze if $\$_bQA$ satisfies participants as well as S_bQA.
- To evaluate the performance of $\$_bQA$ and S_bQA when dealing with x-redundant VOs.

With this in mind, we carry out two kinds of evaluations. A first serie of experiments to compare $\$_bQA$ with some baseline methods in mono-mediator VOs so as to validate its performance. Then, we vary the number of mediators and participants to evaluate the performance and scalability of $\$_bQA$.

3.6.1 Setup

We implemented our prototype in java and simulate x-redundant VOs, with different number of mediators, following the system architecture presented in [LCLV07]. We run our experiments in a computer

running Linux Ubuntu $4.0.3$ with a Petium IV processor of 3 GHz and 1 GB in RAM. The system consists of 200 consumers, 400 providers. Participants compute their satisfaction as defined in Chapter 1. They initialize their satisfaction with a value of 0.5, which evolves with their last 200 issued queries and 500 queries that have passed through providers (i.e. $k = 200$ for a consumer and $k = 500$ for a provider). Providers implement an optimistc strategy to bid for queries. We generate around 10% of providers with *low*-capacity, 60% with *medium*, and 30% with *high*. The *high*-capacity providers are 3 times more powerful than *medium*-capacity providers and still 7 times more powerful than *low*-capacity providers [SGG02]. We generate two classes of queries that *high*-capacity providers perform in 1.3 and 1.5 seconds, respectively, and assume that they arrive in a *Poisson* distribution, as found in dynamic autonomous environments [Mar02].

Concerning participants' departure, we assume on the one hand that a consumer leaves a mediator by dissatisfaction if its satisfaction is smaller than 0.7. On the other hand, we assume that a provider leaves a mediator : by dissatisfaction if its satisfaction value is smaller than 0.5 ; by query starvation if, in an interval of 2 minutes, it does not perform a set of queries towards which it has a preference of at least 0.2 in average, and ; by overutilization if its utilization is greater than 2. Finally, we repeat each serie of experiments we run 10 times and present the average results of all these experimentations.

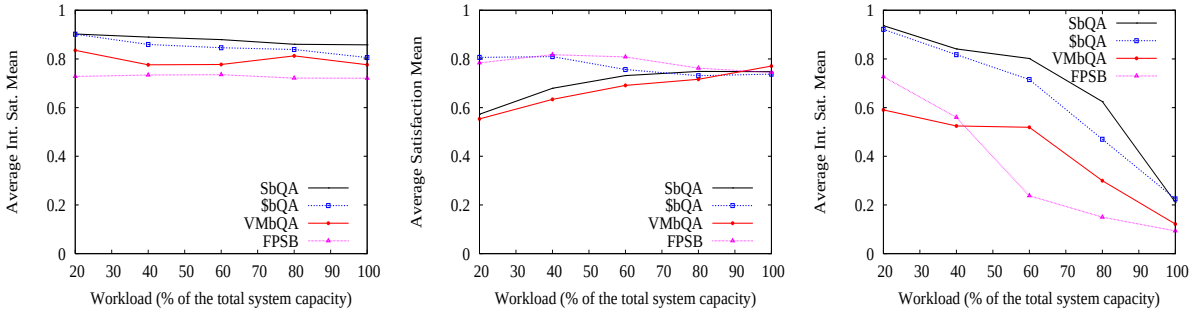
3.6.2 Results

In Section 3.6.2.1, we start by evaluating the quality of $\$_bQA$ and three other baseline methods (S_bQA included), with regards to participants' satisfaction and response times. Then, in Section 3.6.2.2, we study the scalability of both both S_bQA and $\$_bQA$ in x-redundant VOs.

3.6.2.1 Quality Results in Mono-Mediator VOs

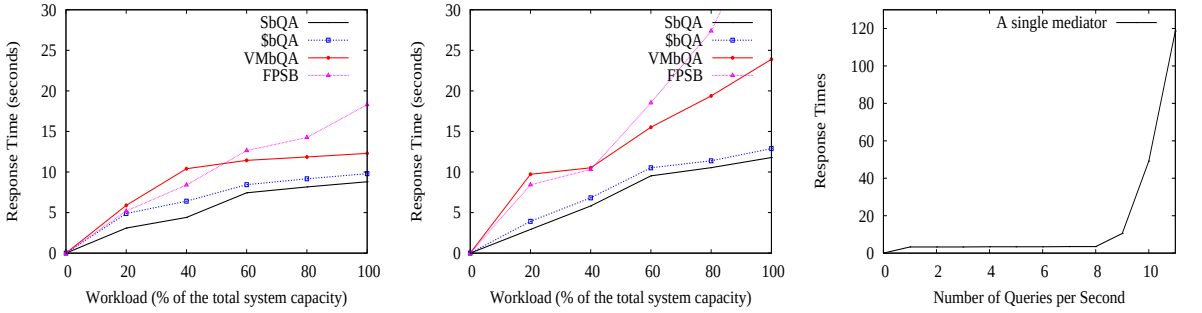
In this series of experiments we proceed as follows. First, to see the possible loss of performance that $\$_bQA$ may have, from the provider's point of view, we compare it with a *first-price sealed-bid* method ($FPSB$). $FPSB$ allocates queries to those providers having made the highest bids and invoices providers the amount of virtual money that they offered for the query. Second, to study the efficiency of the way in which $\$_bQA$ invoices providers, we compare it with a query allocation method that selects providers as $\$_bQA$, but invoices them as $FPSB$. We call this new query allocation method as *Virtual Money-based Query Allocation* ($VMbQA$). Finally, to validate $\$_bQA$, from a satisfaction point of view, we compare it with S_bQA. In these experiments, we assume that the mediator has enough resources so that it does not cause a performance bottleneck. Moreover, to avoid that the fact of having several mediators impacts these results, we run these experiments with a single mediator. We discard such an assumption in the next section.

Figure 3.3 illustrate how these methods satisfy participants for different workloads. We observe in Figure 3.3(a) that, as expected, $FPSB$ is completely neutral to consumers because it does not take into account their intentions. This is not the case for $VMbQA$, $\$_bQA$ and S_bQA, which consider consumers' intentions to allocate queries. But, we observe that $\$_bQA$ is the only one to ensure almost the same performance as S_bQA. Regarding the providers' satisfaction in Figure 3.3(b), we can observe that $\$_bQA$ has again almost the same performance as S_bQA, but also, we observe that $FPSB$ better performs than $\$_bQA$. Indeed, this is because $FPSB$ only considers providers' bids while $\$_bQA$ also considers consumers' intentions. However, we observed during our experiments that $FPSB$ and $VMbQA$ methods have some problems to balance queries because most adequate and preferred (by consumers) providers tend to monopolize incoming queries. $\$_bQA$ does not suffer from this phenomenon by establishing a



(a) Consumers' satisfaction mean based on intentions. (b) Providers' satisfaction mean based on preferences. (c) Providers' satisfaction mean based on intentions.

Figure 3.3 – Quality results in mono-mediator VOs for different workloads and with captive participants.



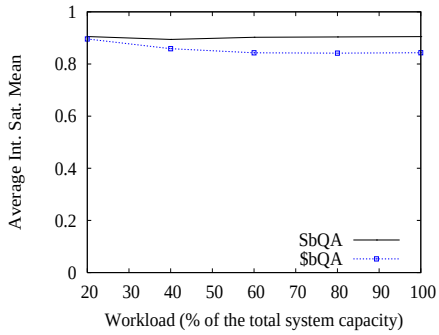
(a) Provider are not allowed to leave the system. (b) Providers may leave by dissatisfaction, starvation or overutilization. (c) Scale up.

Figure 3.4 – Impact on performance of providers' departure.

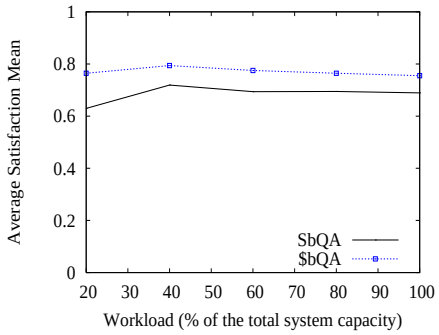
more sophisticated invoice mechanism. This is why $\$_bQA$ method satisfies providers' intentions as well as S_bQA and much better than $FPSB$ and $VMbQA$ (see Figure 3.3(c)). Remember that providers' intentions is the merge of providers' preferences with providers' utilization.

We now analyze the performance of all four methods to ensure short response times. To this end, we proceed to measure their ensured response times when providers are captive, i.e. they are not allowed to quit the system, as well as when they may leave the system. Figure 3.4(a) illustrates the response times ensured by these four methods when providers are captive, i.e. they are not allowed to quit the system. We observe that $\$_bQA$ significantly outperforms both $FPSB$ and $VMbQA$ by ensuring almost the same response times as S_bQA. The low performance of $FPSB$ and $VMbQA$ methods are mainly due to fact that some providers (the most preferred and adequate providers) monopolize queries. Thus, when providers are allowed to leave the system, these providers quit by overutilization while the least adequate and preferred leave by starvation or dissatisfaction. We illustrate these results in Figure 3.4(b). As expected, we can observe in these results that, conversely to $FPSB$ and $VMbQA$, $\$_bQA$ ensures short response times because it balances well queries while satisfies participants.

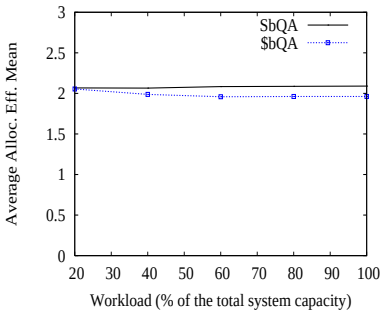
Given these results, we can conclude that we can introduce virtual money, without any loss of system's performance, to regulate a system as long as we care about the way in which providers are selected and invoiced. Nevertheless, as noted so far, a single mediator is a performance bottleneck that may not allow a VO to scale up. To demonstrate this, we discard the assumption that a mediator has enough



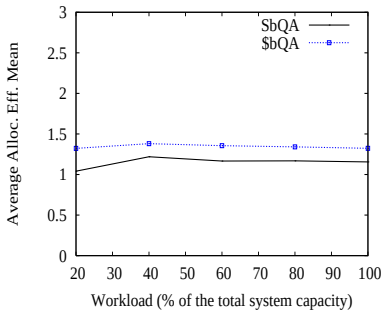
(a) Consumers' satisfaction mean based on intentions.



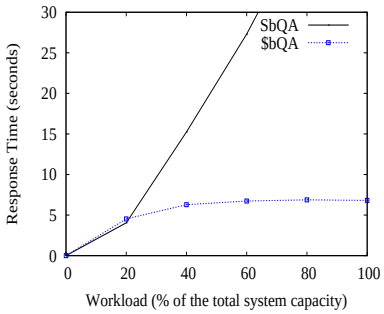
(b) Providers' satisfaction mean based on intentions.



(c) Allocation efficiency w.r.t. consumers.



(d) Allocation efficiency w.r.t. providers.



(e) Response times.

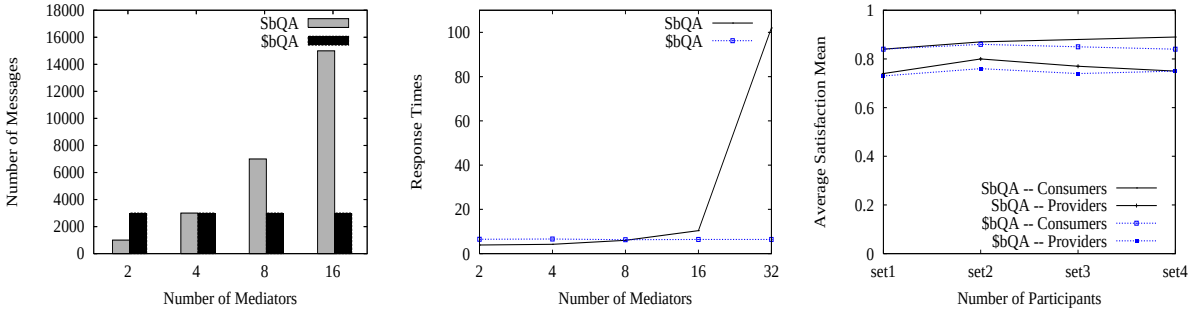
Figure 3.5 – Quality results with captive participants for different workloads in a x-redundant VO with 8 mediators.

resources to perform queries and run several experiments with different rates of incoming queries per second. We illustrate these results in Figure 3.4(c). We can observe that from 9 queries per second the mediator becomes performance bottleneck for the VO. This is why it is quite important to have several mediators performing the query allocation task in a VO. We validate this in the following section.

3.6.2.2 Dealing with x-Redundant VOs

In previous section, we experimentally demonstrated that $\$_bQA$ significantly outperforms both $FPSB$ and $VMbQA$, and demonstrated that it ensures almost the same performance as S_bQA. This is why we only evaluate in this section the efficiency of $\$_bQA$ and S_bQA to scale query allocation up to several mediators in x-redundant VOs. Moreover, we demonstrated in previous section that as S_bQA as $\$_bQA$ deal well with providers' departure, thus we consider in these experimentations captive participants to better study the scalability of both methods.

We start by evaluating the possible impact that, from a satisfaction and performance point of view, the fact of having several mediator allocating queries could have. We run a series of experiments for different workloads in a x-redundant VO with 8 mediators. We observe in Figure 3.5(a) that S_bQA still better satisfies consumers than $\$_bQA$, but this is no more the case for providers (see Figure 3.5(b)). This is also illustrated in Figures 3.5(c) and 3.5(d) where we can observe that S_bQA has a better allocation efficiency regarding consumers than $\$_bQA$, but $\$_bQA$'s allocation efficiency is better with respect to providers.



(a) Exchanged messages among mediators.

(b) Speed up.

(c) Dealing with several participants.

Figure 3.6 – Performance results (a) and (b) for different workloads in a x-redundant VO with 8 mediators and captive participants, and ; (c) for a workload of 60% of the total system capacity with 50 consumers and 100 providers (set1), 100 consumers and 200 providers (set2), 200 consumers and 400 providers (set3) and 400 consumers and 800 providers (set4).

S_bQA has this deterioration regarding providers because of the time it takes to update providers' satisfaction at all mediators. Furthermore, the network messages generated by S_bQA consume computational resources at the mediators' side, which degrades the response times ensured by S_bQA (see Figure 3.5(e)). This is not the case for $\$_bQA$, which exchanges no network message to update providers' satisfaction and hence it significantly outperforms S_bQA.

Now, we run a series of experiments with the aim of analyzing the impact, from a performance point of view, of having several mediators allocating queries. In Figure 3.6(a) we plot the number of network messages exchanged by S_bQA and $\$_bQA$ for every 1000 incoming queries and for different number of mediators. Notice that S_bQA generates network messages to update providers' satisfaction while $\$_bQA$ generates network messages to validate providers' bids and invoice providers. In these results, we can observe that $\$_bQA$ always generates 3 network messages per query while the number of network messages generated by S_bQA depends on the number mediators in a x-redundant VO. Observe that from 4 mediators S_bQA already generates the same number of messages as $\$_bQA$. Figure 3.6(b) illustrates the response times ensured by these two methods with different number of mediators. We can see, on the one hand, that $\$_bQA$'s performance does not depend on the number of mediators in a x-redundant VO and hence its performance is constant. On the other hand, we observe that S_bQA cannot performs well for a high number of mediators because of the number of messages it generates. These results show that $\$_bQA$ better deals with large numbers of incoming queries and mediators than S_bQA.

Finally, we analyze how well S_bQA and $\$_bQA$ satisfy participants when the number of participants in a VO varies. With this end, we run several experiments with a single mediator with a workload of 60% of the total system capacity, but with different number of participants. Figure 3.6(c) illustrates these results. We can observe that both methods have the same performance no matter the number of participants in a VO. This means that both methods can scale up, in terms of number of participants, without a loss of performance in satisfaction.

All above results demonstrate that, in contrast to S_bQA, $\$_bQA$ can easily scale up in terms of number of participants, mediators, and incoming queries while satisfying participants as in monomediator VOs.

3.7 Related Work

In this chapter, we addressed the problem of scaling to several mediators and thus to a large number of participants in distributed information systems. To solve this problem, we introduced the use of virtual money to the query allocation process. Thus, related work can be divided into two parts : economic-based query allocation approaches and design of scalable architectures. Since in Chapter 2 we already surveyed the most important economic-based query allocation methods, in this section, we focus on scalability. Scalability determines a key metric of distributed systems to describe in which sense this system is able to cope with many occurrences of an event. Several definitions of scalability exist in the literature. For example, [Hwa93] argues that strict scalability of a system demands that its efficiency asymptotically remain constant as the system grows to large scale. In this chapter, we especially focused on the relationship between computational needs of participants and the population size by increasing the number of mediators in the system. Thus, we assumed that a system scales up when it supports the joins of new participants without suffer from a decrease in system performance. Different network topologies can allow a system to scale up in this context. In the following, we discuss the most important network topologies that allow a system to scale up in number of participants.

3.7.1 Peer-to-Peer Networks

Peer-to-peer (P2P) networks are built on top of the physical network (typically the Internet), and thus referred to as overlay network. The degree of centralization and the topology of the overlay network strongly impact the nonfunctional properties of a P2P system, such as fault-tolerance, self-maintainability, performance, scalability, and security. In the following, we present the three main classes of P2P networks : unstructured, structured, and hybrid.

3.7.1.1 Unstructured

In unstructured P2P networks, the overlay network is created in a nondeterministic (ad hoc) manner and data placement is completely unrelated to the overlay topology. Each peer knows its neighbors, but does not know the resources they have. Examples of these protocols are the Freenet [CMH+02] and Gnutella. To analyze the properties, possibilities and limitations of pure Peer-to-Peer networks, we describe the basic Gnutella protocol in this section. Gnutella consists of a large number of nodes which may be distributed throughout the world, without any central element. A node becomes part of the Gnutella network by establishing some TCP-connections to other active Gnutella nodes, whose IP addresses it may receive from a bootstrap server [SH03]. New nodes, to which the node can connect if an active connection breaks, are explored by broadcasting PING messages in the virtual overlay network. These PING messages are also used as keep alive pattern and are broadcasted in regular time intervals.

For routing Gnutella employs simple flooding of the request messages, i.e. queries and PING messages. Every new incoming PING or query, which has not been received before, is forwarded to all neighbors except the one it received the message from, if the *time-to-live* (TTL) value is at least one. If a node receives the same message more than once, these messages are not further flooded. Response messages, like PONG or query response messages, are routed back on the same path the request message used, which is called backward routing. In Gnutella the virtual Peer-to-Peer layer is not matched to the physical layer, which leads to zigzag routes, as described in [SK03]. Only enhancements, as described by the approach of geo-sensitive Gnutella [SK03], provide means to adapt the virtual network to the physical network. Fault-tolerance is very high since all peers provide equal functionality and are able to replicate data. However, one of the main problems of unstructured networks is scalability.

Query routing is typically done by flooding the query to the peers that are in limited hop distance from the query originator. This mechanism does not scale up to a large number of peers because of the huge amount of load which they incur on the network. Furthermore, the incompleteness of the results can be high since some peers containing relevant data may not be reached because they are too far away from the query originator. More sophisticated and efficient query routing techniques in unstructured systems can be found e.g. in [KGZY02, YGM02].

3.7.1.2 Structured

Structured networks have emerged to solve the scalability problem of unstructured networks. They achieve this goal by tightly controlling the overlay topology and data placement. Data are placed at precisely specified locations and mappings between data and their locations (e.g. a file identifier is mapped to a participant address) are provided in the form of a distributed routing table. *Distributed hash table* (DHT) is the main representative of structured P2P networks. A DHT provides a hash table interface with primitives $put(key, value)$ and $get(key)$, where key is an object identifier, and each participant is responsible for storing the values (object contents) corresponding to a certain range of keys. Each participant also knows a certain number of other participant, called neighbors, and holds a routing table that associates its neighbors' identifiers to the corresponding addresses. Most DHT data access operations consist of a lookup, for finding the address of the provider p that holds the requested data, followed by direct communication with p. In the lookup step, several hops may be performed according to participant' neighborhoods.

Queries can be efficiently routed since the routing scheme allows one to find a participant responsible for a key in $O(\log N)$ routing hops, where N is the number of participants in the network. Since a participant is responsible for storing the values corresponding to its range of keys, autonomy is then limited. Furthermore, DHT queries are typically limited to exact match keyword search. Active research is on-going to extend the DHT capabilities to deal with more complex queries such as range queries [GS04] and join queries [HHL+03]. Examples of P2P systems supported by structured networks include Chord [SMLN+03], CAN [RFH+01], Tapestry [ZHS+04], Pastry [RD01], PIER [HHL+03], P-Grid [ACMD+03], among others. P-Grid is not supported by a DHT, instead, it is based on a virtual distributed search tree.

Because limited autonomy, DHT networks is unlikely to support systems as we consider in this thesis : where providers usually desire to preserve their data in local. Nevertheless, one can imagine a DHT-based system where providers publish in the network only their offered services. In this case, a possibility to implement $\$_bQA$ in these systems is by designing responsible participants of query mediations such as [APV07] does for designing responsible participants for timestamping data versioning. Thus, consumers send their queries to one of these responsible participants in the network, which return participants providing relevant services to their queries. Thus, this possibility requires that providers register their functionalities and capabilities at these responsible participants. We would like to explore this possibility in a future work.

3.7.1.3 Hybrid

Unstructured and structured P2P networks are considered "pure" because all their providers provide the same functionality. In contrast, hybrid networks (a.k.a. super-peer networks) are a merge between client-server systems and pure P2P networks. Like client-server systems, some participants (called super-peers), act as dedicated servers for some other participants and can perform complex functions such

as indexing, query processing, access control, and meta-data management. Using only one super-peer reduces to client-server with all the problems associated with a single server. Like pure P2P networks, super-peers can be organized in a P2P fashion and communicate with one another in sophisticated ways, thereby allowing the partitioning or replication of global information across all super-peers. Super-peers can be dynamically elected (e.g. based on bandwidth and processing power) and replaced in the presence of failures. In a super-peer network, a requesting peer simply sends the request, which can be expressed in a high-level language, to its responsible super-peer. The super-peer can then find the relevant peers either directly through its index or indirectly using its neighbor super-peers. In fact, this hybrid network topology is quite similar to that we considered in this chapter. Hence, one can, easily and in a transparent way, implement $\$_bQA$ in this kind of networks.

The main advantages of super-peer networks are efficiency and quality of service (i.e. the user-perceived efficiency, e.g. completeness of query results, query response time, etc.). The time needed to find data by directly accessing indices in a super-peer is very small compared with flooding. In addition, super-peer networks exploit and take advantage of providers' different capabilities in terms of CPU power, bandwidth, or storage capacity as super-peers take on a large portion of the entire network load. This significantly reduces the high message load, which can be observed in a Gnutella network. Thus, to keep the advantages of Gnutella, i.e. the complete self organization and decentralization, super-peers are introduced in [SR02]. In contrast, in pure P2P networks, all nodes are equally loaded regardless of their capabilities. Access control can also be better enforced since directory and security information can be maintained at the super-peers.

By introducing such enhancements, the load on the network can be reduced without introducing preconfigured, centralized servers. The network is still scalable, but one super-peer should not have more than 50 to 100 registered participants, depending on the processing power and the connection of the super-peer. Thus it is necessary, that the number of super-peers increases according to the total number of participants in the network. $\$_bQA$ allows such a scalability because the creation of new mediators (super-peers) does not incur additional network cost. Examples of super-peer networks include Napster [nap], Publius [WRC00], Edutella [NWQ+02], and JXTA [jxt].

3.7.2 Grid-based Networks

The term "Grid" was coined in the mid90s to denote a proposed distributed computing infrastructure for advanced science and engineering [Fe99]. In a regular grid topology, each node in the network is connected with two neighbors along one or more dimensions. If the network is one-dimensional, and the chain of nodes is connected to form a circular loop, the resulting topology is known as a ring. Grid networks enables aggregation and sharing of these resources through by bringing together communities with common objectives and creating virtual organizations (VOs) [Fos01].

Considerable progress has since been made on the construction of such an infrastructure (e.g. [BJB+00, JGN99, SWDC97]), but the term "Grid" has also been conflated, at least in popular perception, to embrace everything from advanced networking to artificial intelligence. The real and specific problem that underlies the Grid concept is coordinated computational resource sharing and problem solving in dynamic, multi-institutional VOs. This sharing is, necessarily, highly controlled, with participants defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs. As noted so far, a set of participants defined by such sharing rules form what we call a VO.

The organization of participants in a Grid system determines its scalability. Figure 3.7 shows the possible organization that participants can have. The organization describes how the participants involved in resource management make scheduling decisions, the communication organization between these par-

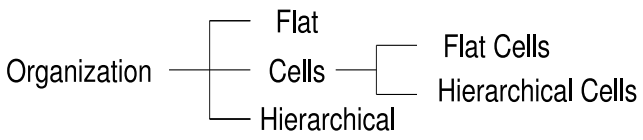


Figure 3.7 – Grid Organization.

ticipants, and the different roles the participants play in the scheduling decision. In a flat organization all participants can directly communicate with each other without going through an intermediary (e.g. a mediator). In a hierarchical organization participants in the same level can directly communicate with the participants directly above them or below them, or peer to them in the hierarchy. The fan out below a participant in the hierarchy is not relevant to the classification. Most current Grid systems use this organization since it has proven scalability. In a cell topology, the participants within the cell communicate between themselves using flat organization. Designated participants within the cell function acts as boundary elements that are responsible for all communication outside the cell. The internal topology of a cell is not visible from another cell, only the boundary participants are. Cells can be further organized in flat or hierarchical topologies. A Grid that has a flat cell topology has only one level of cells whereas a hierarchical cell topology can have cells that contain other cells. The major difference between a cell topology and hierarchical topology is that a cell topology has a designated boundary with a hidden internal organization whereas in the hierarchical topology the organization is visible to all participants in the Grid.

There are many different approaches and models for developing Grid resource-management systems. For example, Globus [FK97] system enables modular deployment of Grid systems by providing the required basic services and capabilities in the Globus Metacomputing Toolkit (GMT). This toolkit consists of a set of components that implement basic services, such as security, resource location, resource management, data management, resource reservation, and communications. Most grid systems have for the most part focused on either a computational Grid or a service Grid. The other category of system is the Grid scheduler such as Nimrod/G [BAG00] and AppLeS [BW97] that is integrated with another Grid RMS such as Globus [CFK+98, FK97] or Legion [CKKG]. These combinations are then used to create application oriented computational Grids that provide certain levels of QoS.

However, the harnessing the power of grids remains to be a challenging problem for users due to the complexity involved in the creation and composition of applications and their deployment on distributed resources. Resource brokers or mediators hide the complexity of grids by transforming consumer requirements into a set of queries that are scheduled on the appropriate computational resources, managing them and collecting results when they are finished. A broker/mediator must have the capability to locate relevant providers to queries and must also have the ability to select the best providers [AG00, COBW00, VBW04].

Summarizing, Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed "autonomous" resources dynamically at runtime depending on their availability, capability, performance, cost, and consumers' requirements.

3.7.3 Multi-Agent Networks

Multi-agent systems (MAS) is the emerging subfield of artificial intelligence that aims to provide both principles for construction of complex systems involving multiple agents and mechanisms for co-

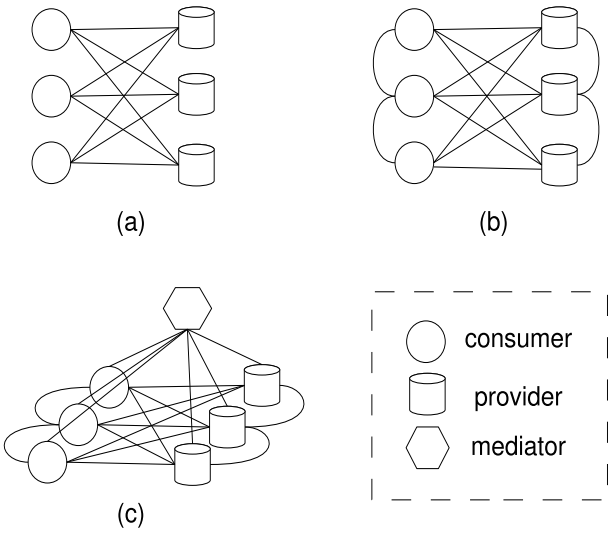


Figure 3.8 – Acquaintance topology forms.

ordination of independent agents' behaviors. Most of the on multi-agent systems deals with systems in which agents are peers of each other. However, as seen in Section 3.7.1.1, it seems unlikely that such structures are the most appriopiate when hundreds or thousands of agents are required. For this reason, MAS designers started to use metaphores from human social and economic ornganizations [Fox88]. For example, human organizations operate by enforcing avenues of communication and control between individuals in order for the overall grouping to achieve its goals. Of rough equivalance, MASs use acquaintance topologies to perform the same function of defining and constraining interaction. In fact, these topologies may be the same as for P2P networks (see provious section) and the relationships between agents may be of master, slave, or peer. Generally speaking, the relationships among agents (the participants) can be distinguished by the constraints within which participants interact with each other. In Figure 3.8, we illustrate the three main organization forms of participants in MASs.

Figure 3.8(a) illustrates the most simple organization form, where each consumer can communicate with each provider and vice versa. However, this organization does not allows participants to cooperate because neither consumers nor providers are aware of the existent of other consumers and providers, respectively. The organization form illustrated in Figure 3.8(b) allows consumers and providers to communicate with each other consumer and provider, respectively. Therefore, participants can form groups with other participnats, allowing them to cooperate. Topologically, this organization is fully connected network that represents a fully connected peer MAS. The third organization form of MASs (see Figure 3.8(c)) is identical to the second one, with the exception that intermediary participant that facilates intermediary functions, such as matchmaking, brokering, query optimization, query planning, etc. [KH95]. *Turner and Jennings* [TJ00] discuss how a MAS can dynamically adapt its structure for various population size. In fact, one of the benefits of MAS is their scalability. Since they are inherently modular, it should be easier to add new agents to a multiagent system than it is to add new capabilities to a monolithic system. Moreover, systems whose capabilities and parameters are likely to need to change over time or across agents can also benefit from this advantage of MAS.

3.7.4 Small-World Networks

The notion of small world phenomenon originates from social science research [Mil67]. It has developed to become a very active current research topic in physics, computer science, and mathematics. It has been observed that the small-world phenomenon is pervasive in a wide range of settings such as social networks, biological environments, data/communication networks, the connectivity of the Internet, and gene networks. More recent studies using the Internet have come to the same conclusion, see [DMW03]. For example, recent studies (e.g., [ZGG04]) have shown that peer-to-peer networks such as Freenet may exhibit small world properties. Informally, a small world network can be viewed as a connected graph in which two randomly chosen nodes are connected by just about six degrees of separation. In other words, the average shortest distance between two randomly chosen nodes is approximately six hops. This property implies that one can locate information stored at any random node of a small world network by only a small number of link traversals.

Despite the excitement that followed the Milgram experiments [Mil67] there was no convincing network model generating a network that is locally highly clustered and at the same time has a small diameter until 1998. Then, *Watts and Strogatz* [WS98] analyzed three different kinds of real networks and noted that graphs could be classified according to two independent structural features, namely the clustering coefficient and average node-to-node distance, the latter also known as average shortest path length. They measured that in fact many real-world networks have a small average shortest path length, but also a clustering coefficient significantly higher than expected by random chance. They then proposed a novel graph model, now currently named the Watts and Strogatz model, with (i) a small average shortest path length, and (ii) a large clustering coefficient. Viewed from another perspective, such a model indicates that a small number of random edges decreases the average path length significantly since they can be viewed as "short-cuts" spanning the regular graph. With this model a part of the riddle regarding real networks was solved.

However, it was not until *Kleinberg*'s work in 2000 [Kle00] that a mathematical model was developed for how efficient routing can take place in such networks. Kleinberg showed that the possibility of efficient routing depends on a balance between the proportion of shortcut edges of different lengths with respect to coordinates in the base grid. Under a specific distribution, where the frequency of edges of different lengths decreases inverse proportionally to the length, simple greedy routing (always walking towards the destination) can find routes in $O(\log^2 N)$ steps on average, where N is the size of the graph. Recently, *Hui et al.* [HLY06] proposed protocols to create and manage a small-world structured P2P network. They have demonstrated how a low average hop distance between nodes can reduce the number of link traversals in object lookup.

3.7.5 Summary

In this section, we surveyed the most important distributed system architectures that allows a system to deal with a great number of participants. We studied the P2P, Grid, multi-agent, and small-world networks. Summarizing, we saw that the architectures that are close to that we considered in this chapter are : the hybrid (or super-peer) topology from P2P networks, the cells topology from Grid networks, and the multi-agent topology that uses intermediary agents. In these three topologies, one can implement $\$_bQA$ in a transparent way. For example, in a super-peer topology, super-peers can be used as mediators for their cluster of peers. In fact, we discussed this in [QRLCV08]. In the cells topology, one can use the boundary participants of cells, which are responsible to communicate with other cells, as mediators for their cells. And, in a intermediary-based mutli-agent topology, intermediaries can play the role of

mediators for their cluster of agents.

3.8 Chapter Summary

We considered large-scale distributed information systems where several mediators may cooperate to allocate queries. And, we assumed that participants are free to leave the system at any time and have special interests towards queries. In particular, we addressed the problem of scaling query allocation up to several mediators so that it does not suffer from performance bottlenecks due to a single mediator. The challenge in these environments is to perform query allocation so that participants' satisfaction be the same as in systems with a single mediator while good system performance be also ensured. To overcome this problem, we proposed in this chapter an economic version of S_bQA, called $\$_bQA$, that uses virtual money, instead of participants' satisfaction, as a means of regulation. In summary, our main contributions are the following.

- We discussed the challenges of using virtual money as a means of regulation and made precise a way in which virtual money should circulate within a system. Then, we stated the number of network messages needed by a mediator to control the flow of virtual money. Conversely to the expected, we demonstrated that only a few number of network messages (3 per query) are required to control the flow of virtual money.
- We defined a new way in which a provider computes its bids to get queries, which allows it to consider its preferences, its satisfaction, its current utilization, and its current virtual money balance. Moreover, we defined 3 strategies to bid for queries in the presence of several mediators.
- We proposed an *Economic Satisfaction-based Query Allocation* method ($\$_bQA$, for short) that considers both consumers' intentions and providers' bid while ensuring good system performance. Generally speaking, $\$_bQA$ is S_bQA using virtual money as a means of regulation. The two strong points of $\$_bQA$ is that : it allows a mediator to balance consumers' intentions and providers' bids according to the importance it desires to pay to both of them, and ; it allows a mediator to indemnify imposed providers (i.e. those providers that do not desire a query and perform the query) so that they get interesting queries in the future. We analytically demonstrated that $\$_bQA$ allows a VO to scale up to several mediators with no additional network cost.
- We compared three microeconomic methods (included $\$_bQA$) with a non-microeconomic one using satisfaction as a "money independent" measure. This is interesting from a methodological point of view since it allows knowing the possible loss or gain of using virtual money as a means of regulation. To our knowledge, besides query load and response time, no microeconomic method has ever been evaluated through measures that are outside the microeconomic scope. A key result of such a study is that, conversely to several proposals, one must care about the selection, invoicing, and bidding phases when designing a microeconomic query allocation method.
- We validated $\$_bQA$ in x-redundant VOs, i.e. in VOs with several mediators. Results show that $\$_bQA$ can easily scale up in terms of number of : mediators, participants, and incoming queries. In fact, a key result is that $\$_bQA$ allows a VO to scale up while ensuring good system performance and the same participants' satisfaction as in VO with a single mediator.

Future Work We discussed in the first chapter that participants may be satisfied at two levels : concerning their preferences and concerning their intentions. But, as we now introduced the use of virtual money, we desire to integrate the virtual money into the provider's satisfaction notion so that we can see the differences with the other two levels of satisfaction. We desire to do this in a future work. Fur-

thermore, we want to study, via the model we proposed, different query allocation methods based on microeconomics [MCWG95] and game theory [vNM44], which are generally studied with a utility function and the Pareto optimal property, respectively. With this study, we aim at seeing the efficiency of such methods to meet participants' intentions and as well as the gains of using satisfaction as an evaluation measure.

CHAPTER 4

Dealing with Participants' Failures

Nowadays Internet offers many opportunities in large-scale distributed systems. One of the most recent solutions is the use of thousands or even millions of unreliable, autonomous personal computers (the providers) connected to the Internet to share information or computational resources with each other. Providers put their computational functionalities or resources at the service of others (the consumers) for collaborative reasons or for their own benefits. On the one hand, Web services [web] are a clear example of massive distributed competitive computation when their invocation incurs a monetary cost. On the other hand, some examples of massive distributed cooperative computation are the SETI@home [set] and distributed.net [dis] projects. Indeed, in these environments participants have special interests towards queries and may enter and join the system at will. For example, a participant, donating its computational resources to several research projects, may desire to perform in average more queries of some specific projects than of others.

The fact of considering large-scale, open (for autonomous participants) distributed systems has another consequence : the possibility of participants' *failure*, or more generally dysfunction of participants. In fact, as the scale of a distributed system is increased in number of participants, the possibility that one of them is subjected to failure also increases. Studies of participants' availability in widely deployed distributed systems such as Overnet [BSV03], Napster and Gnutella [SGG03] show that there is a significant churn due to failure. Hence, in this context, the utility of distributed applications is increasingly limited by availability rather than performance. This problem of dealing with participants' failures has been extensively studied by several works in distributed systems [BBMS08, JBH+05, HBR+05a, HXcZ07, LML01]. Because of autonomy, however, a provider may act maliciously, i.e. may be Byzantine [LSP82], and hence it may return erroneous or incomplete results, or simply may return no result, for a query. This is why some distributed systems replicate the same query (i.e. it creates backup queries for a query) on several providers to compare their results. It is, for example, the policy of SETI@home [set]. Therefore, query replication may meet two objectives : to compare query results of different providers and to support possible providers' failures. In this chapter, we focus on the latter and report the former to future work. Indeed, query replication cost should not be overlooked because it may quickly utilize all the computational resources in the system. From the system point of view, query replication requires either more powerful providers or additional providers. From the participants point of view, it is not obvious that a participant has the same intention, and thus the same satisfaction, to be utilized as primary source than as backup source. To the best of our knowledge, no fault-tolerant model has dealt with participants' intentions and satisfaction, hence no query replication technique is appropriate for distributed information systems with autonomous participants that may have special interests towards queries.

In this chapter, we propose a query replication technique that aims at bearing providers' failures while increasing participants' satisfaction. In particular, our main contributions are the following :

- We propose a satisfaction model that considers participants' failures. In particular, we characterize the fact that *(i)* queries have different importance ; *(ii)* a consumer may receive less results that it expects because of providers' failures ; and *(iii)* a provider may perform queries for nothing because of backup queries and consumers' failures.
- We define the global satisfaction, that is, the expected satisfaction of participants concerning the allocation of a given query. A particularity of this definition is that it takes into consideration both participants' intentions and participants' failure probability.
- We propose *Satisfaction-based Query Replication* (S_bQR for short), a query replication technique to compute the rate of backup queries according to the global satisfaction. The S_bQR's goal is to replicate those queries that allows to increase the global satisfaction.
- We experimentally demonstrate that S_bQR better performs, from a satisfaction and performance point of view, than replicating all incoming query. We also demonstrate that by replicating each incoming query the system suffer serious problems of performance for high workloads, but worse it loses more query results than when one does not apply a fault-tolerant technique.

The remainder of this chapter is organized as follows. We formally state the problem we address in Section 4.1. Then, in Section 4.2, we propose definitions of participants' satisfaction that consider both participants' failures and queries importance for consumers. In the same section, we define the expected satisfaction of participants regarding a given query allocation. We propose S_bQR in Section 4.3. We present in Section 4.4 the $\$_bQA$'s experimental results. We survey related work in Section 4.5 and we finally conclude this chapter in Section 4.6.

4.1 Problem Definition

The distributed system we consider consists of a set $\mathcal{I}$ of autonomous participants. As in previous chapters, autonomy means that a participant may enter and leave the system at any time because of their own desire. Participants may play two different roles : consumer and provider. Moreover, we assume that the system has at least one mediator m, which is in charge of allocating queries so that everything works well, from a satisfaction and performance point of view. A consumer $c \in C$ (with $C \subseteq \mathcal{I}$) poses a query to a mediator when it cannot locally perform the query or just because it has certain gains by outsourcing such a query. For example, a consumer may query the system to perform a given application because *(i)* it has not enough resources to run the application, or *(ii)* other participant (a provider) performs the application faster. We assume that a consumer informs the mediator of how much critical a query is for it. A query is considered as critical by a consumer when it is crucial for it to get all the answers it requires. We refer to this query importance as *criticity*. For example, going back to our motivating example, $eWine$'s query could have a high criticity because $eWine$ wants to compare prices and some other properties (such as delivery time) so as to receive a good service for its query. Thus, in order for a mediator to consider this, a consumer formulates queries by including their criticity, that is, in a format abstracted as a 4-tuple $q = < c, d, n, \gamma >$. Where the first three parameters are the same as in previous chapters (consumer identifier, query description, and the number required answers, respectively) and γ denotes how the query criticity, with $\gamma \in [0..1]$. The greater parameter γ is, the more critical the query is. In the following, we simply use c, d, n, or γ when there is no ambiguity on q.

Until now, we discussed throughout this thesis the importance of allocating queries while satisfying participants and ensuring good system performance (such as short response times and system functionalities). However, we did not consider the fact that in large-scale distributed information systems any participant $i \in \mathcal{I}$ has a probability f_i to fail, which may also impact on system performance as well as on

participants' satisfaction. We consider participants' *fail-stop* failures of participants and report Byzantine and partial faults to future work (see [BBJ+08, BT98, FLSG06, NDMR08] for related work on this). Fail-stop failures assumes that the only way a participant can fail is by simply not functioning at all during a no-short time interval. Because of this, having allocated a query q to a set $\widehat{P_q^r}$ of providers, it is possible that only a set $P_q^{ok} \subseteq P_q$ of providers be available after the treatment of a given query q. Hence, it is possible that only the results of a set $\widehat{\widehat{P_q}} \subset \widehat{P_q^r}$ of providers are returned to consumer $q.c$. Indeed, the returned results may be less than the required by the consumer, i.e. $||\widehat{\widehat{P_q}}|| < q.n$, which may impact on the consumer's satisfaction. Thus, in this chapter, we assume that a mediator m creates backup queries (i.e. replicates queries) to support participants' failures and hence preserving both good consumer's satisfaction and good system performance.

Because of backup queries, when a mediator asks a provider for its intentions, besides its intention, the provider also replies with the cost of performing a query in the case its results are not returned to the consumer. Vector $\overrightarrow{PC}_q$, whose values are in the interval $[0..1]$, contains the cost to perform a query q of each provider $p \in P_q$. Indeed, allocating backup queries means that those providers allocated backup queries utilize their computational resources to produce results that may not be returned to the consumer. This could significantly dissatisfy (depending on their cost $\overrightarrow{PC}_q$) and overload providers, which may cause their departure from the system. On the other side, consumers receiving no result for their critical queries may leave the system by dissatisfaction. Thus, given all this, we formally state the problem we address in this chapter as follows.

Problem Statement Given a set $\mathcal{I}$ of autonomous participants, each $i \in \mathcal{I}$ with a failure probability f_i, mediator m must allocate each incoming query q to a set $\widehat{P_q^r}$ of $\min(q.n, N)$ providers so that results for critical queries, participants' satisfaction, and short response times are ensured.

4.2 Satisfaction Model for Faulty Participants

Remember that in Chapter 1 we proposed a model to characterize the participants' intentions in the long run, which already defines some definitions of participants' satisfaction. However, we discard in that model that *(i)* queries may have different criticity for a consumer, *(ii)* the results produced by a provider may not be returned to a consumer because of consumer's or provider's failure. This last point implies, in other words, the model we proposed in Chapter 1 inherently assumes that, given a query q, the results of each provider in $\widehat{P_q^r}$ is returned to the consumer. In this section, as noted early, we release such an assumption and propose participants' satisfaction definitions that consider the possibility that only the results of a set $\widehat{\widehat{P_q}}$ of providers are returned to the consumer (Sections 4.2.1 and 4.2.2). Moreover, we propose a global satisfaction definition, which considers the failure probability of participants, to characterize the expected satisfaction of the participants concerned by the allocation of a given query (Section 4.2.3).

4.2.1 Consumer Satisfaction

As defined in Chapter 1, it is by means of its satisfaction that a consumer can evaluate if it gets, or not, the results it expects from the mediator. Considering that a consumer may desire different results for a query, we defined the consumer's satisfaction so that the more results it gets the more satisfied it is. However, this is not always the case when providers may fail and queries may have different criticity values. For example, a consumer, requiring two results for a given low critical query, may be more

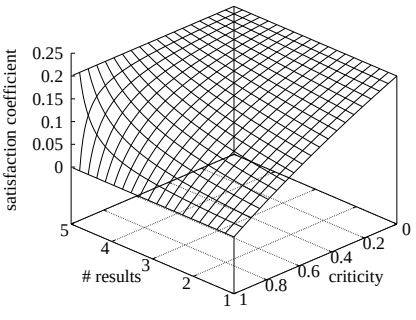


Figure 4.1 – Number of results vs query's criticity when a consumer requires five results.

satisfied of receiving only one result from a provider towards which it has an intention of 1 than receiving results from two providers towards which it has an intention of 1 and -1, respectively. This depends on the query criticity for a consumer to receive as many results as it requires. Intuitively, if an incoming query has a criticity $\gamma = 1$ (respectively $\gamma = 0$) means that the consumer would not be satisfied at all if it did not receive all the results it requires (resp. means that the satisfaction of the consumer strongly depends on the number of results it receives). To reflect this, let $\widehat{\widehat{P_q}}$ denote the set of providers whose results are returned to the consumer, we modify the satisfaction coefficient $\frac{1}{n}$ of Equation 1.3 as follows,

$$\frac{1-\gamma}{n-\gamma\cdot||\widehat{\widehat{P_q}}||} \tag{4.1}$$

We illustrate the behavior of this above satisfaction coefficient in Figure 4.1. Observe that as more critical a query is and the number of received results decreases, the satisfaction coefficient decreases, which also leads to a decrease of satisfaction. It is worth noting that when the criticity of a query takes the value of 1, the satisfaction coefficient always takes zero values if the number of results is not the required by the consumer. Then, considering Equation 4.1 and the fact that providers may fail, we define the satisfaction of a consumer as follows.

Definition 32. *Consumer Satisfaction Concerning a Single Query Allocation (revisited)*

$$\delta_s(c,\widehat{\widehat{P_q}}) = \begin{cases} \dfrac{1-\gamma}{n-\gamma\cdot||\widehat{\widehat{P_q}}||}\cdot\Big(\sum\limits_{p\in\widehat{\widehat{P_q}}}(\overrightarrow{CI}_q[p]+1)/2\Big) & if\ \gamma<1 \\ \dfrac{1}{n}\cdot\Big(\sum\limits_{p\in\widehat{\widehat{P_q}}}(\overrightarrow{CI}_q[p]+1)/2\Big) & otherwise \end{cases}$$

4.2.2 Provider Satisfaction

As noted so far, a provider can evaluate, by means of its satisfaction, if the mediator allocates it those queries that meet its intentions. Conversely to a consumer, the fact that a query has a high criticity, or not, does not influence the satisfaction of a provider. In turn, the fact that a provider performs a query and its results are not returned to the consumer may significantly impact on its satisfaction (depending

on its cost). This is because a provider is usually selfish and hence the fact of spending computational resources to perform queries from which it obtains no benefit does not meet their intentions at all. Thus, given this, we define again the satisfaction of a provider $p \in P_q^{ok} \cap P_q$ as follows.

Definition 33. *Provider Satisfaction Concerning a Single Query Allocation (revisited)*

$$\delta_s(p, \widehat{P_q^r}, \widehat{\widehat{P_q}}) = \left| \begin{array}{ll} \big(\overrightarrow{PPI}_p[q] + 1\big)/2 & if\ p \in \widehat{\widehat{P_q}} \\ \big(-\overrightarrow{PPI}_p[q] + 1\big)/2 & if\ p \in (P_q \backslash \widehat{P_q^r}) \cap P_q^{ok} \\ \big(1 - \overrightarrow{PC}_q[p]\big)/2 & if\ p \in \big(\widehat{P_q^r} \backslash \widehat{\widehat{P_q}}\big) \cap P_q^{ok} \end{array} \right.$$

Remember that vector $\overrightarrow{PPI}_p$ contains the intentions expressed by p towards the k last proposed queries. The idea behind the above definition is that if a provider performs a query and its produced result is returned to the consumer, its satisfaction concerning such an allocation is then based on its intention (line 1 of above equation). Otherwise, if a provider does not perform a query, its satisfaction concerning such a query allocation is based on its negative intention (line 2). This means that if a provider expresses a negative intention to perform a given query and it is not allocated the query, it is satisfied with the mediator job because it does not spend computational resources to perform a disgusting query. In the above definition, we also consider the case where a provider performs a query and its produced result is not returned to the consumer (line 3). In this case, we assume that a provider is not satisfied of performing queries for nothing. Thus, we define the provider's satisfaction based on its cost to perform a query. We translate the cost values into the interval $[0..0.5]$, which means that a provider always has a low satisfaction in such cases.

4.2.3 Global Satisfaction

We make precise in this section the global satisfaction regarding a given query allocation. One of the main goals when dealing with providers' unavailability in query allocations is to create backup queries so that answers with short response times are ensured to consumers. In autonomous distributed systems, allocating backup queries is not an easy task because of participants' autonomy. So far, we defined the query allocation problem in dynamic and autonomous distributed systems as a global satisfaction maximization. However, according to the definitions of Sections 4.2.1 and 4.2.2, participants' satisfaction are contradictory, that is, when creating backup queries may improve consumers' satisfaction (by ensuring their required answers), it may decrease providers' satisfaction (by not returning their results to consumers). We define the global satisfaction by considering this contradictory point. With this aim, we consider the failure probability of participants.

First of all, it is worth noting that we assume that faults are not correlated. Thus, the probability that a participant i does not fail in a time unit is $1 - f_i$. Let t_q denote the required time by a provider p (which does not appear in the notation for clarity reasons) to perform a query q. Consequently, the probability $\mathcal{A}_i^{t_q}$ that i does not fail in a discrete time interval t_q (i.e. that be always available during time interval t_q) is given by below equation.

$$\mathcal{A}_i^{t_q} = (1 - f_i)^{t_q} \tag{4.2}$$

Given this, let us first characterize the probability that a query be successfully treated by at most h providers among which the worst ranked provider has a ranking r, that is, the probability that at most h providers until ranking r in vector $\overrightarrow{R}_q$ do not fail before returning results of a given query.

Lemma 3. *The successful probability $\mathcal{S}_q^h(\widehat{P_q^r})$ that a given query q has to be performed by at most h providers in $\widehat{P_q^r}$ is given by,*

$$\mathcal{S}_q^h(\widehat{P_q^r}) = \sum_{\substack{P_q^{ok} \subseteq \widehat{P_q^r} \\ ||P_q^{ok}|| \leq h}} \Big(\prod_{p \in P_q^{ok}} \mathcal{A}_p^{t_q} \prod_{p \in \widehat{P_q^r} \setminus P_q^{ok}} (1 - \mathcal{A}_p^{t_q}) \Big)$$

Proof. The probability that a set of providers successfully performs a query is given by its available probability. By Equation 4.2, the available probability of a set P_q^{ok} of providers in set $\widehat{P_q^r}$ is $\prod_{p \in P_q^{ok}} \mathcal{A}_p^{t_q} \cdot \prod_{p \in \widehat{P_q^r} \setminus P_q^{ok}} (1 - \mathcal{A}_p^{t_q})$. Consequently, the successful probability of a query q to be performed by at most h providers in $\widehat{P_q^r}$ is given by the available probability sum of all different sets P_q^{ok} in $\widehat{P_q^r}$ that satisfy the constraint $||P_q^{ok}|| \leq h$, hence $\mathcal{S}_q^h(\widehat{P_q^r}) = \sum_{\substack{P_q^{ok} \subseteq \widehat{P_q^r} \\ ||P_q^{ok}|| \leq h}} \Big(\prod_{p \in P_q^{ok}} \mathcal{A}_p^{t_q} \prod_{p \in \widehat{P_q^r} \setminus P_q^{ok}} (1 - \mathcal{A}_p^{t_q}) \Big)$. □

Let us now characterize the probability that the results produced by a specific provider and a given set of providers are returned to the consumer.

Lemma 4. *Let x denote $||\widehat{\widehat{P_q}}||$, given a query q, the probability $\mathcal{S}_q^a(\widehat{P_q^r}, x)$ that the results produced by provider $\overrightarrow{R}_q[a]$ and $x-1$ other providers in $\widehat{P_q^r}$ be returned to consumer $q.c$ is given by,*

$$\mathcal{S}_q^a(\widehat{P_q^r}, x) = \begin{cases} \displaystyle\sum_{\substack{\widehat{\widehat{P_q}} \subseteq \widehat{P_q^r} \\ ||\widehat{\widehat{P_q}}|| < x \\ \overrightarrow{R}_q[a] \in \widehat{\widehat{P_q}}}} \Big(\prod_{p \in \widehat{\widehat{P_q}}} \mathcal{A}_p^{t_q} \prod_{p \in \widehat{P_q^r} \setminus \widehat{\widehat{P_q}}} (1 - \mathcal{A}_p^{t_q}) \Big) \; if \; x < q.n \\ \displaystyle\sum_{\substack{\widehat{\widehat{P_q}} \subseteq \widehat{P_q^r} \\ ||\widehat{\widehat{P_q}}|| = x \\ \overrightarrow{R}_q[a] \in \widehat{\widehat{P_q}}}} \Big(\prod_{p \in \widehat{\widehat{P_q}}} \mathcal{A}_p^{t_q} \prod_{\substack{p = \overrightarrow{R}_q[j] \\ j \leq \max(k) \\ \overrightarrow{R}_q[k] \in \widehat{\widehat{P_q}} \\ p \notin \widehat{\widehat{P_q}}}} (1 - \mathcal{A}_p^{t_q}) \Big) \quad else \end{cases}$$

Proof. We use a reasoning close to Lemma 3. Overall, given a query q, two cases can exist in order for provider $\overrightarrow{R}_q[a]$ returns its produced result to consumer $q.c$: (i) that less than $q.n$ providers in $\widehat{P_q^r}$ be available during discrete time interval t_q, and (ii) that the same or more than $q.n$ providers in $\widehat{P_q^r}$ be available during discrete time interval t_q, but at most $x-1$ providers have a higher score than $\overrightarrow{R}_q[a]$.

In the first case, provider $\overrightarrow{R}_q[a]$ must only be available during the discrete time interval t_q to be in $\widehat{\widehat{P_q}}$. Thus, the probability $\mathcal{S}_q^a(\widehat{P_q^r}, x)$ that $\overrightarrow{R}_q[a]$'s results be returned to $q.c$, when $x < q.n$, is given by the available probability sum of all different sets $\widehat{\widehat{P_q}}$ in $\widehat{P_q^r}$ that satisfy the constraint $\overrightarrow{R}_q[a] \in \widehat{\widehat{P_q}}$. Hence, by Equation 4.2 and for the $x < q.n$ case, $\mathcal{S}_q^a(\widehat{P_q^r}, x) = \sum_{\substack{\widehat{\widehat{P_q}} \subseteq \widehat{P_q^r} \\ ||\widehat{\widehat{P_q}}|| < x \\ \overrightarrow{R}_q[a] \in \widehat{\widehat{P_q}}}} \Big(\prod_{p \in \widehat{\widehat{P_q}}} \mathcal{A}_p^{t_q} \prod_{p \in \widehat{P_q^r} \setminus \widehat{\widehat{P_q}}} (1 - \mathcal{A}_p^{t_q}) \Big)$.

In the second case, conversely to the first case, provider $\overrightarrow{R}_q[a]$ must be available during the discrete time interval t_q, but also must have at least the $q.n$ worst ranking in $\overrightarrow{R}_q$ to be in $\widehat{\widehat{P_q}}$. Thus, the probability $\mathcal{S}_q^a(\widehat{P_q}, x)$ that $\overrightarrow{R}_q[a]$'s results be returned to $q.c$, when $x = q.n$, is given by the available probability sum of all different sets $\widehat{\widehat{P_q}}$ in $\widehat{P_q^r}$ that satisfy the constraints $\overrightarrow{R}_q[a] \in \widehat{\widehat{P_q}}$ and $\nexists \overrightarrow{R}_q[j] \in \widehat{P_q^r} : j < a$. Hence, by Equation 4.2 and for the $x = q.n$ case, $\mathcal{S}_q^a(\widehat{P_q^r}, x) = \sum_{\substack{\widehat{\widehat{P_q}} \subseteq \widehat{P_q^r} \\ ||\widehat{\widehat{P_q}}|| = x \\ \overrightarrow{R}_q[a] \in \widehat{\widehat{P_q}}}} \Big(\prod_{p \in \widehat{\widehat{P_q}}} \mathcal{A}_p^{t_q} \prod_{\substack{p = \overrightarrow{R}_q[j] \\ j \leq \max(k) \\ \overrightarrow{R}_q[k] \in \widehat{\widehat{P_q}} \\ p \notin \widehat{\widehat{P_q}}}} (1 - \mathcal{A}_p^{t_q}) \Big)$. □

Then, we formally state the global satisfaction with respect to the allocation of a given query in Theorem 6. Since the global satisfaction is computed by the mediator, given a query q, we consider in the following vector $\overrightarrow{PI}_q$ to represent the intentions of providers in P_q, but notice that $\overrightarrow{PI}_q[p] = \overrightarrow{PPI}_p[q]$. The complexity of the global satisfaction computation is $\theta(n \cdot r^2)$. Of course, this computation is not optimal, but, in this thesis, we only focus on studying the possible impact of considering the intentions and failures probabilities of participants when replicating queries. We report a possible optimization of this computation to future work.

Theorem 6. *The global satisfaction $\Theta(\widehat{P_q^r})$ of allocating a given query q to a set $\widehat{P_q^r}$ as,*

$$
\begin{aligned}
\Theta(\widehat{P_q^r}) \;=\; & \sum_{j=1}^{r} \Bigg(\mathcal{A}^{t_q}_{\overrightarrow{R}_q[j]} \cdot \Big(\mathcal{A}_c^{t_q} \cdot \mathcal{S}_q^{n-1}(\widehat{P_q^{j-1}}) \cdot \overrightarrow{PI}_q[\overrightarrow{R}_q[j]] \;+ \\
& \qquad \big(1 - \mathcal{A}_c^{t_q}\big) \cdot \mathcal{S}_q^{n-1}(\widehat{P_q^{j-1}}) \cdot \overrightarrow{PC}_q[\overrightarrow{R}_q[j]] \;+ \\
& \qquad \big(1 - \mathcal{S}_q^{n-1}(\widehat{P_q^{j-1}})\big) \cdot \overrightarrow{PC}_q[\overrightarrow{R}_q[j]] \Big) \Bigg) \;+ \\
& \sum_{j=r+1}^{||P_q||} \mathcal{A}^{t_q}_{\overrightarrow{R}_q[j]} \cdot -\overrightarrow{PI}_q[\overrightarrow{R}_q[j]] \;+ \\
& \mathcal{A}_c^{t_q} \cdot \sum_{j=0}^{n} \left(\frac{1-\gamma}{n - \gamma \cdot j} \cdot \sum_{a=1}^{r} \Big(\mathcal{S}_q^a(\widehat{P_q^r}, j) \;\cdot \overrightarrow{CI}_q[\overrightarrow{R}_q[a]] \Big) \right)
\end{aligned}
$$

Proof. For clarity, we proceed to demonstrate above equation line per line. The global satisfaction of allocating a given query q is the sum of the expected satisfaction of providers in P_q and the expected satisfaction consumer $q.c$. We first focus on the providers side. Given Definition 33, three cases may occur : (i) when a provider is in set $\widehat{\widehat{P_q}}$, (ii) when a provider is in set $(\widehat{P_q^r} \backslash \widehat{\widehat{P_q}}) \cap P_q^{ok}$, and (iii) when a provider is in set $(P_q \backslash \widehat{P_q^r}) \cap P_q^{ok}$. Indeed, in all these three cases, a provider in P_q must be in P_q^{ok} to compute its satisfaction. This, probability is given by Equation 4.2, $\mathcal{A}^{t_q}_{\overrightarrow{R}_q[j]}$.

In order for a provider $\overrightarrow{R}_q[j]$ in $\widehat{P_q^r} \cap P_q^{ok}$ to be in set $\widehat{\widehat{P_q}}$, consumer $q.c$ must be available during the discrete time interval required by $\overrightarrow{R}_q[j]$ to perform q, which is given by Equation 4.2, $\mathcal{A}_c^{t_q}$, and that at most other $q.n - 1$ providers with a ranking smaller than j also be in set $\widehat{P_q^r} \cap P_q^{ok}$, which is given by Lemma 3, $\mathcal{S}_q^{n-1}(\widehat{P_q^{j-1}})$. Thus, the probability that the results produced by providers in $\widehat{P_q^r}$ be returned

to $q.c$ is,

$$\sum_{j=1}^{r}\left(\mathcal{A}^{t_q}_{\overrightarrow{R}_q[j]}\cdot\mathcal{A}^{t_q}_c\cdot\mathcal{S}^{n-1}_q(\widehat{P^{j-1}_q})\cdot\overrightarrow{PI}_q[\overrightarrow{R}_q[j]]\right)$$

which is multiplied by $\overrightarrow{R}_q[j]$'s intention since its results are returned to $q.c$. This proves the first line of the global satisfaction equation.

Now, a provider $\overrightarrow{R}_q[j]$ in $\widehat{P^r_q}\cap P^{ok}_q$ may not be in set $\widehat{\widehat{P_q}}$ for two main reasons : first, because consumer $q.c$ fails in the discrete time interval t_q, and ; second, because at least $q.n$ other providers with a ranking smaller than j be in $\widehat{P^r_q}\cap P^{ok}_q$. By Equation 4.2 and Lemma 3, we have that the probability that the first possibility occurs is,

$$\sum_{j=1}^{r}\left(\mathcal{A}^{t_q}_{\overrightarrow{R}_q[j]}\cdot\left(1-\mathcal{A}^{t_q}_c\right)\cdot\mathcal{S}^{n-1}_q(\widehat{P^{j-1}_q})\cdot\overrightarrow{PC}_q[\overrightarrow{R}_q[j]]\right)$$

and that the second possibility occurs is,

$$\sum_{j=1}^{r}\left(\mathcal{A}^{t_q}_{\overrightarrow{R}_q[j]}\cdot\left(1-\mathcal{S}^{n-1}_q(\widehat{P^{j-1}_q})\right)\cdot\overrightarrow{PC}_q[\overrightarrow{R}_q[j]]\right)$$

which are multiplied by $\overrightarrow{R}_q[j]$'s cost since its results are not returned to $q.c$ in both of two possibilities. This proves the second and third lines of the global satisfaction equation.

To finalize with the provider' side, we now consider the case that a provider not be allocated a query. By Equation 4.2, the probability that a set of providers with ranking $j > r$ be in $(P_q\backslash\widehat{P^r_q})\cap P^{ok}_q$ is,

$$\sum_{j=r+1}^{||P_q||}\mathcal{A}^{t_q}_{\overrightarrow{R}_q[j]}\cdot-\overrightarrow{PI}_q[\overrightarrow{R}_q[j]]$$

which is multiplied by $\overrightarrow{R}_q[j]$'s negative intention since its not allocated q. This proves the fourth line of the global satisfaction equation.

Concerning the consumer's side, as for a provider, to compute its satisfaction, a consumer must be available in a discrete time interval t_q, $\mathcal{A}^{t_q}_c$. The expected satisfaction of consumer $q.c$ w.r.t. a given provider $\overrightarrow{R}_q[a]\in\widehat{\widehat{P_q}}$ is given by the multiplication of the probability that provider $\overrightarrow{R}_q[a]$ and other $j-1$ providers in $\widehat{P^r_q}$ be in $\widehat{\widehat{P_q}}$ and $q.c$'s intentions towards $\overrightarrow{R}_q[a]$. Hence, by Lemma 4, the expected satisfaction of consumer $q.c$ concerning a set $\widehat{\widehat{P_q}}$ is given by,

$$\mathcal{A}^{t_q}_c\cdot\sum_{a=1}^{r}\left(\mathcal{S}^a_q(\widehat{P^r_q},j)\cdot\overrightarrow{CI}_q[\overrightarrow{R}_q[a]]\right.$$

Consequently, by Definition 32, the expected satisfaction of consumer $q.c$ concerning all possible sets $\widehat{\widehat{P_q}}$ in $\widehat{P^r_q}$ is,

$$\mathcal{A}^{t_q}_c\cdot\sum_{j=0}^{n}\left(\frac{1-\gamma}{n-\gamma\cdot j}\cdot\sum_{a=1}^{r}\left(\mathcal{S}^a_q(\widehat{P^r_q},j)\cdot\overrightarrow{CI}_q[\overrightarrow{R}_q[a]]\right)\right)$$

which finally proves the last (fifth) line of the global satisfaction equation. □

4.3 Non Systematic Query Replication Based on Satisfaction

We present in this section *Satisfaction-based Query Replication* (S_bQR for short), a new method to replicate queries so as to handle with participants' failures. Conversely to several works that create a backup provider per query ($n_b = 1$), S_bQR replicates incoming queries with the aim of increasing participants' satisfaction. Thus, it only replicates a query when this implies an increase of the global satisfaction (see Theorem 6). Algorithm 4 shows the main steps of the query replication process. S_bQR receives as input the query q to be allocated, ranking vector $\overrightarrow{R}_q$, vector $\overrightarrow{CI}_q$ of consumers' intentions, vector $\overrightarrow{PI}_q$ of providers' intentions, and vector $\overrightarrow{PC}_q$ of providers' cost. We assume that $\overrightarrow{R}_q$ is generated by using providers' score as ranking function (see Definition 26), but, without any loss of generality, vector $\overrightarrow{R}_q$ could be generated by using any other ranking function (such as the utilization $\mathcal{U}_t$ function). First of all, to set the number n_b of backup providers, S_bQR initializes n_b to zero and r to the number of required answers by consumer $q.c$ (lines 2 and 3, respectively, of Algorithm 4). By setting r to n, it means that the n_b backup providers are to be considered from the $\overrightarrow{R}_q[n+1]$ to $\overrightarrow{R}_q[||P_q||]$ providers. As second step, it builds the sets $\widehat{P_q^r}$ and $\widehat{P_q^{r+1}}$ (lines 4-7). Finally, S_bQR verifies if the global satisfaction concerning set $\widehat{P_q^{r+1}}$ is greater than that concerning set $\widehat{P_q^r}$ (line 8). This computation is given by Theorem 7. If so, it increments the number of backup providers and add the next best ranked providers to sets $\widehat{P_q^r}$ and $\widehat{P_q^{r+1}}$. Then, it restarts from line 8 until $\Theta(\widehat{P_q^r}) \geq \Theta(\widehat{P_q^{r+1}})$ or there is no more new providers in vector $\overrightarrow{R}_q$ (lines 9-15). Indeed, Algorithm 4 can be optimized, but our goal is to show the steps involved in the query replication process.

Theorem 7.

$$
\begin{aligned}
\Theta(\widehat{P_q^{r+1}}) - \Theta(\widehat{P_q^r}) \;=\; & \mathcal{A}^{t_q}_{\overrightarrow{R}_q[r+1]} \cdot \Bigg(\mathcal{A}_c^{t_q} \cdot \mathcal{S}_q^{n-1}(\widehat{P_q^r}) \cdot \overrightarrow{PI}_q[\overrightarrow{R}_q[r+1]] \;+ \\
& \qquad (1 - \mathcal{A}_c^{t_q}) \cdot \mathcal{S}_q^{n-1}(\widehat{P_q^r}) \cdot \overrightarrow{PC}_q[\overrightarrow{R}_q[r+1]] \;+ \\
& \qquad \big(1 - \mathcal{S}_q^{n-1}(\widehat{P_q^r})\big) \cdot \overrightarrow{PC}_q[\overrightarrow{R}_q[r+1]] \;+ \\
& \qquad \overrightarrow{PI}_q[\overrightarrow{R}_q[r+1]] \Bigg) \;+ \\
& \mathcal{A}_c^{t_q} \cdot \sum_{j=0}^{n} \Bigg(\frac{1-\gamma}{n - \gamma \cdot j} \cdot \Big(\sum_{a=1}^{r} \big(\mathcal{S}_q^a(\widehat{P_q^{r+1}}, j) - \mathcal{S}_q^a(\widehat{P_q^r}, j)\big) \cdot \overrightarrow{CI}_q[a] \;+ \\
& \qquad \mathcal{S}_q^{r+1}(\widehat{P_q^{r+1}}, j) \cdot \overrightarrow{CI}_q[\overrightarrow{R}_q[r+1]] \Big) \Bigg)
\end{aligned}
$$

Proof. Our demonstration is derived from algebraic reductions of Theorem 6 (the $\Theta(\widehat{P_q^{r+1}}) - \Theta(\widehat{P_q^r})$ case). For clarity reasons, we demonstrate equation of Theorem 6 line per line. First, in case that a provider is considered to get a query q and is also considered to be in set $\widehat{\widehat{P_q}}$, we have

$$
\sum_{j=1}^{r+1} \Big(\mathcal{A}^{t_q}_{\overrightarrow{R}_q[j]} \cdot \mathcal{A}_c^{t_q} \cdot \mathcal{S}_q^{n-1}(\widehat{P_q^{j-1}}) \cdot \overrightarrow{PI}_q[\overrightarrow{R}_q[j]] \Big) - \sum_{j=1}^{r} \Big(\mathcal{A}^{t_q}_{\overrightarrow{R}_q[j]} \cdot \mathcal{A}_c^{t_q} \cdot \mathcal{S}_q^{n-1}(\widehat{P_q^{j-1}}) \cdot \overrightarrow{PI}_q[\overrightarrow{R}_q[j]] \Big)
$$

and hence all values from 1 up to r are eliminated by the subtraction because of Lemma 3. Consequently, we only consider the $r+1$ value, that is,

$$
\mathcal{A}^{t_q}_{\overrightarrow{R}_q[r+1]} \cdot \mathcal{A}_c^{t_q} \cdot \mathcal{S}_q^{n-1}(\widehat{P_q^r}) \cdot \overrightarrow{PI}_q[\overrightarrow{R}_q[r+1]]
$$

Algorithm 4: Satisfaction-based Query Replication

Input : q, $\overrightarrow{R}_q$, $\overrightarrow{CI}_q$, $\overrightarrow{PI}_q$, $\overrightarrow{PC}_q$
Output: n_b

begin
  `// Variables setting`
  $n_b = 0$
  $r = n$
  `// Provider sets setting`
  for $i = 1$ **to** r **do**
    add provider $\overrightarrow{R}_q[i]$ to $\widehat{P_q^r}$
    add provider $\overrightarrow{R}_q[i]$ to $\widehat{P_q^{r+1}}$
  add provider $\overrightarrow{R}_q[r+1]$ to $\widehat{P_q^{r+1}}$
  `// Computing the number of backup providers`
  while $\Theta(\widehat{P_q^r}) < \Theta(\widehat{P_q^{r+1}})$ **do**
    increment n_b by one ($n_b = n_b + 1$)
    increment r by one ($r = r + 1$)
    if *there exists provider* $\overrightarrow{R}_q[r+1]$ **then**
      add provider $\overrightarrow{R}_q[r]$ to $\widehat{P_q^r}$
      add provider $\overrightarrow{R}_q[r+1]$ to $\widehat{P_q^{r+1}}$
    else
      break loop ;
end

which demonstrates the first line (of Theorem 7). When a provider is expected to get query q and not expected to be in set $\widehat{\widehat{P_q}}$, we have the case in which consumer $q.c$ is expected to fail,

$$\sum_{j=1}^{r+1}\left(\mathcal{A}_{\overrightarrow{R}_q[j]}^{t_q}\cdot\left(1-\mathcal{A}_c^{t_q}\right)\cdot\mathcal{S}_q^{n-1}(\widehat{P_q^{j-1}})\cdot\overrightarrow{PC}_q[\overrightarrow{R}_q[j]]\right)-\sum_{j=1}^{r}\left(\mathcal{A}_{\overrightarrow{R}_q[j]}^{t_q}\cdot\left(1-\mathcal{A}_c^{t_q}\right)\cdot\mathcal{S}_q^{n-1}(\widehat{P_q^{j-1}})\cdot\overrightarrow{PC}_q[\overrightarrow{R}_q[j]]\right)$$

and also have the case in which at least $q.n$ other providers with a ranking smaller than j be in $\widehat{P_q^r} \cap P_q^{ok}$,

$$\sum_{j=1}^{r+1}\left(\mathcal{A}_{\overrightarrow{R}_q[j]}^{t_q}\cdot\left(1-\mathcal{S}_q^{n-1}(\widehat{P_q^{j-1}})\right)\cdot\overrightarrow{PC}_q[\overrightarrow{R}_q[j]]\right)-\sum_{j=1}^{r}\left(\mathcal{A}_{\overrightarrow{R}_q[j]}^{t_q}\cdot\left(1-\mathcal{S}_q^{n-1}(\widehat{P_q^{j-1}})\right)\cdot\overrightarrow{PC}_q[\overrightarrow{R}_q[j]]\right)$$

In both cases, values from 1 up to r are eliminated by the subtraction and hence we only consider the $r+1$ value. Consequently, we have below equation for the first case,

$$\mathcal{A}_{\overrightarrow{R}_q[r+1]}^{t_q}\cdot\left(1-\mathcal{A}_c^{t_q}\right)\cdot\mathcal{S}_q^{n-1}(\widehat{P_q^r})\cdot\overrightarrow{PC}_q[\overrightarrow{R}_q[r+1]]$$

and,

$$\mathcal{A}_{\overrightarrow{R}_q[r+1]}^{t_q}\cdot\left(1-\mathcal{S}_q^{n-1}(\widehat{P_q^r})\right)\cdot\overrightarrow{PC}_q[\overrightarrow{R}_q[r+1]]$$

for the second case. The above two equations demonstrate lines 2 and 3, respectively. Now, for those providers that are expected to not get query q we have,

$$\sum_{j=r+2}^{||P_q||} \mathcal{A}^{t_q}_{\overrightarrow{R}_q[j]} \cdot -\overrightarrow{PI}_q[\overrightarrow{R}_q[j]] - \sum_{j=r+1}^{||P_q||} \mathcal{A}^{t_q}_{\overrightarrow{R}_q[j]} \cdot -\overrightarrow{PI}_q[\overrightarrow{R}_q[j]]$$

Notice that all values from $r+2$ up to $||P_q||$ are again eliminated by the subtraction. But, conversely to previous equations, the $r+1$ value remains in the right side and thus we take its negative value, which implies the following equation,

$$\mathcal{A}^{t_q}_{\overrightarrow{R}_q[r+1]} \cdot \overrightarrow{PI}_q[\overrightarrow{R}_q[r+1]]$$

This equation demonstrates the fourth line. Finally, to demonstrate the final line (i.e. the expected satisfaction concerning the consumer), we focus on the consumer's side of Theorem 6 and thus we have the following subtraction,

$$\mathcal{A}^{t_q}_c \cdot \sum_{j=0}^{n} \left(\frac{1-\gamma}{n-\gamma \cdot j} \cdot \sum_{a=1}^{r+1} \left(\mathcal{S}^a_q(\widehat{P^{r+1}_q}, j) \cdot \overrightarrow{CI}_q[\overrightarrow{R}_q[a]] \right) \right) -$$
$$\mathcal{A}^{t_q}_c \cdot \sum_{j=0}^{n} \left(\frac{1-\gamma}{n-\gamma \cdot j} \cdot \sum_{a=1}^{r} \left(\mathcal{S}^a_q(\widehat{P^r_q}, j) \cdot \overrightarrow{CI}_q[\overrightarrow{R}_q[a]] \right) \right)$$

Conversely to all above equations, even if we have repeated iterations (from 1 up to r), the subtraction cannot eliminate such values because of Lemma 4, which considers set $\widehat{\widehat{P_q}}$. In other words, $\mathcal{S}^a_q(\widehat{P^{r+1}_q}, j)$ is different to $\mathcal{S}^a_q(\widehat{P^r_q}, j)$ even for a same value j, which is not the case for $\mathcal{S}^{n-1}_q(\widehat{P^j_q})$. Therefore, we can only reduce above equation by grouping values from 1 up to r,

$$\mathcal{A}^{t_q}_c \cdot \sum_{j=0}^{n} \left(\frac{1-\gamma}{n-\gamma \cdot j} \cdot \left(\sum_{a=1}^{r} \left(\mathcal{S}^a_q(\widehat{P^{r+1}_q}, j) - \mathcal{S}^a_q(\widehat{P^r_q}, j) \right) \cdot \overrightarrow{CI}_q[a] \right) \right)$$

and separating the $r+1$ value,

$$\mathcal{A}^{t_q}_c \cdot \sum_{j=0}^{n} \left(\frac{1-\gamma}{n-\gamma \cdot j} \cdot \mathcal{S}^{r+1}_q(\widehat{P^{r+1}_q}, j) \cdot \overrightarrow{CI}_q[\overrightarrow{R}_q[r+1]] \right)$$

Thus, we have the following equation,

$$\mathcal{A}^{t_q}_c \cdot \sum_{j=0}^{n} \left(\frac{1-\gamma}{n-\gamma \cdot j} \cdot \left(\sum_{a=1}^{r} \left(\mathcal{S}^a_q(\widehat{P^{r+1}_q}, j) - \mathcal{S}^a_q(\widehat{P^r_q}, j) \right) \cdot \overrightarrow{CI}_q[a] + \right. \right.$$
$$\left. \left. \mathcal{S}^{r+1}_q(\widehat{P^{r+1}_q}, j) \cdot \overrightarrow{CI}_q[\overrightarrow{R}_q[r+1]] \right) \right)$$

which demonstrate the fifth line (of Theorem 7). □

Parameter	Definition	Value
nbConsumers	Number of consumers	150
nbProviders	Number of providers	300
nbMediators	Number of mediators	1
qDistribution	Query arrival distribution	Poisson
iniSatisfaction	Initial satisfaction	0.5
γ	Query criticity	from 0.3 to 1
fRate	Participant failure rate	0.03/second
nbRepeat	Repetition of simulations	10

Table 4.1 – Simulation parameters.

4.4 Experimental Validation

In this section, we validate S_bQR by comparing it with $replicateAll$, which is a traditional query replication method that only allocates each incoming query to one backup provider, no matter how many results a consumer desires [KLL97]. For our validations, we assume that S_bQA ranks providers and selects the best providers required by a consumer and that S_bQR and $replicateAll$ only select the n_b best ranked providers after the n first ranked providers. To clearly see the S_bQR's gains, we also compare it to S_bQA, that is, to the case when one never creates backup queries (for clarity, we call this case the $none$ case). We carry out our validations with three main objectives :

- To evaluate how well, from a satisfaction point of view, S_bQR operates distributed information systems where participants have special interests towards queries.
- To evaluate the impact on performance due to the backup queries generated by S_bQR.
- To analyze if S_bQR can adapt to different queries' criticity and to different probabilities of participants' failure.

With this in mind, we first evaluate S_bQR from a satisfaction and performance point of view (Section 4.4.2). Then, we study in Section 4.4.2.1 how well S_bQR performs when queries have a high criticity and finally study, in Section 4.4.2.2, how well it deals with high failures probabilities of providers.

4.4.1 Setup

We modify our java-based simulator, which we used for the validation of Chapter 2, so that it considers participants' failures and different values of queries criticity. We implemented S_bQR and $replicateAll$ methods on top of S_bQA, that is, S_bQA ranks providers and selects the n best ranked providers and as S_bQR as $replicateAll$ selects the n_b best ranked providers after n to perform backup queries. For both of them, the following configuration is the same and the only change is the way in which each method replicate queries. So far, we discussed that the definition of a synthetic workload for the environments we consider is an open problem. This is why we decide, in these experiments, to generate again a very general workload that can be applied for different applications. Since the way in which a mediator creates backup queries is independent of other mediators, we consider in these experiments only 1 mediator allocating queries to better study the impact of generating backup queries. Also, we assume that the mediator has enough computational resources so that it is not a performance bottleneck for the system.

We generated a network with 150 consumers and 300 providers who compute their satisfaction as presented in Sections 4.2.1 and 4.2.2, respectively. We initialize their satisfaction with a value of 0.5,

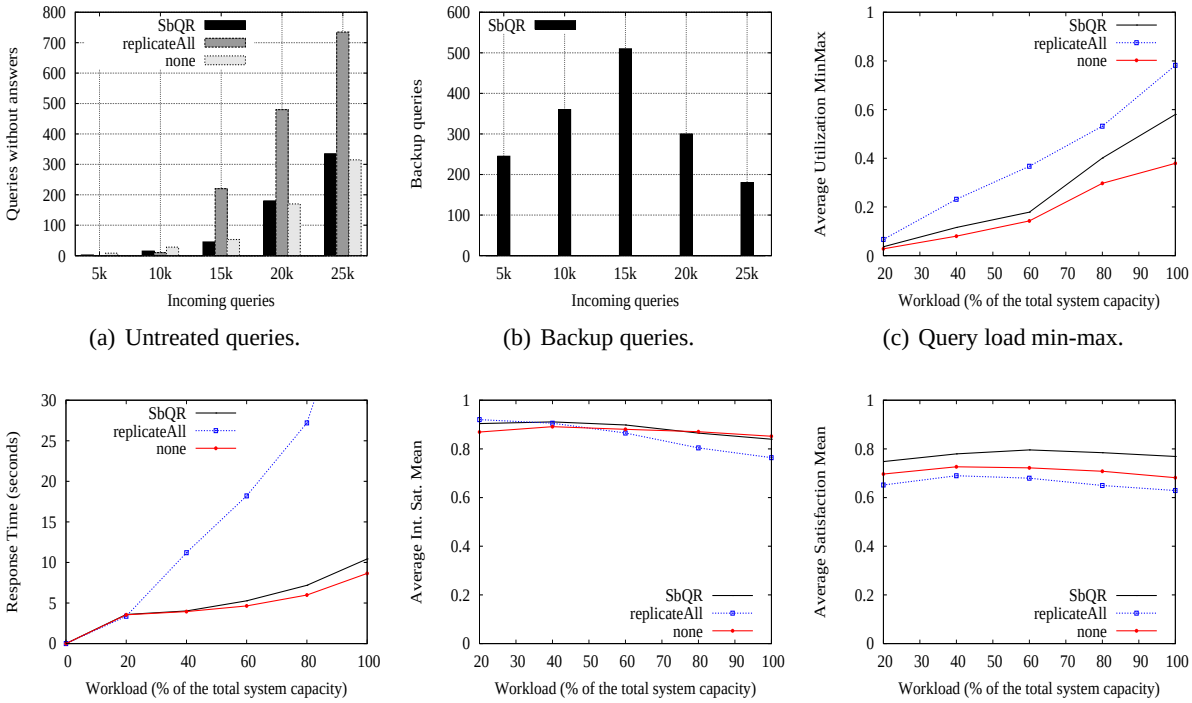
which evolves with their last 150 issued queries (for consumers) and 400 queries that have passed through providers. We consider that all 300 providers are able to perform any incoming query issued by consumers. We assume that a participant has a probability of 0.03 of failure per second. We generate around 10% of providers with *low*-capacity, 60% with *medium*, and 30% with *high*. The *high*-capacity providers are 3 times more powerful than *medium*-capacity providers and still 7 times more powerful than *low*-capacity providers [SGG02]. Concerning participants' preferences, to simulate high heterogeneity, we divide the set of providers into three classes according to the interest of consumers (as in Chapter 2.6) : to those that consumers have *high*-interest (60% of providers), *medium*-interest (30% of providers), and *low*-interest (10% of providers). Also, we create three classes of providers : those that have *high*-adaptation (35% of providers), *medium*-adaptation (60% of providers), and *low-adaptation* (5% of providers).

We run our experiments over 10000 seconds and repeat each series of experiments 10 times to present the average results of all these experimentations. We generate two classes of queries that *high*-capacity providers perform in 1.3 and 1.5 seconds, respectively, and assume that they arrive in a *Poisson* distribution, as found in high dynamic environments [Mar02]. Consumers issue queries with a criticity that they generate at random between 0.3 and 1. For the sake of simplicity, we assume that consumers only ask for one informational answer (i.e. $n = 1$). Finally, since our goal is to study how well S_bQR replicates queries, we assume captive participants in these experiments so that participants' departure (by dissatisfaction, overutilization, or starvation) does not impact on results.

4.4.2 Results

In Figure 4.4.2, we illustrate the results of a series of experiments with different workloads. We start by illustrating in Figure 4.2(a) the number of queries that remains without answers due to providers' failure. A query q without answers means that all providers in set $\widehat{P_q}$ failed before returning the result of q. We can observe that, for low workloads, both S_bQR and $replicateAll$ has less queries without answers than the $none$ case. However, conversely to the expected, this is not the case for high workloads. While S_bQR starts from workloads over 80% (i.e. 20k incoming queries) of the total system capacity to have a few more queries without answers than the $none$ case, $replicateAll$ starts from workloads over 60% (i.e. 15k incoming queries) of the total system capacity to have much more queries without answers than the $none$ case. This is because, by replicating each incoming query, $replicateAll$ significantly overutilizes those providers that are the most preferred by consumers and that the system is most adapted, which, given Equation 4.2, increases the probability of failure of a provider before returning the result of a given query. In contrast, S_bQR considers the failure probability and intentions of participants to decide if a query should be created or not. This is why S_bQR has, in average, 30 more queries without answers than the $none$ case for high workloads.

In Figure 4.2(b), we illustrate the number of backup queries created by S_bQR. As expected, we observe that the number of created backup queries increases as the number of incoming queries increases. But, the number of created backup queries decreases for high workloads because providers express negative intentions when they become overutilized and hence the fact of replicating queries implies to decrease the global satisfaction. As a result, S_bQR only replicates those queries that allow increasing the global satisfaction while ensuring more answered queries than $replicateAll$. Moreover, as seen in Section 2.6.3.1, S_bQA better balances queries in the system as the number of incoming queries increases. This is why, $replicateAll$ allows S_bQA to ensure a better qlb than with S_bQR and the $none$ case (see Figure 4.2(c))). Nevertheless, to create much more backup queries is reflected in the response times ensured by $replicateAll$ (see Figure 4.2(d)). We observe that while S_bQR has a performance quite close



(a) Untreated queries. (b) Backup queries. (c) Query load min-max.

(d) Response times. (e) Consumers' satisfaction mean based on preferences. (f) Providers' satisfaction mean based on preferences.

Figure 4.2 – Results with faulty participants and different workloads.

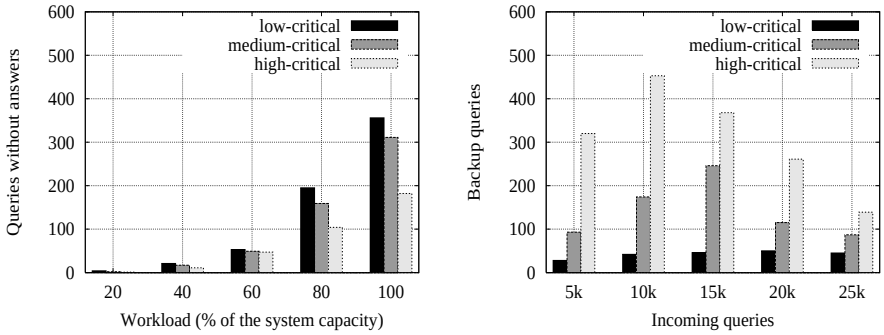
to the performance to the $none$ case, $replicateAll$ significantly increases response times.

Concerning participants' satisfaction, we observe in Figure 4.2(e) that, for a workload of 20% and 40% of the total system capacity, consumers have a satisfaction slightly higher, or equal, with $replicateAll$ than with S_bQR. But, this is no more the case for higher workloads where consumers are significantly more satisfied with S_bQR than with $replicateAll$ because, as seen in Figure 4.2(a), $replicateAll$ penalizes consumers with several queries without answers. We also observe that consumers are more satisfied, for workloads under 80% of the total system capacity, with S_bQR than the $none$ case. And, for workloads over 80% of the total system capacity, consumers are slightly less satisfied with S_bQR than the $none$ case. On the other side, we can observe in Figure 4.2(f) that providers are significantly more satisfied with S_bQR than with $replicateAll$ or the $none$ case. In particular, we can see that providers are less satisfied with $replicateAll$ than the $none$ case. This is because $replicateAll$ always generates backup queries and thus there exist several cases where providers work for nothing, which dissatisfy them.

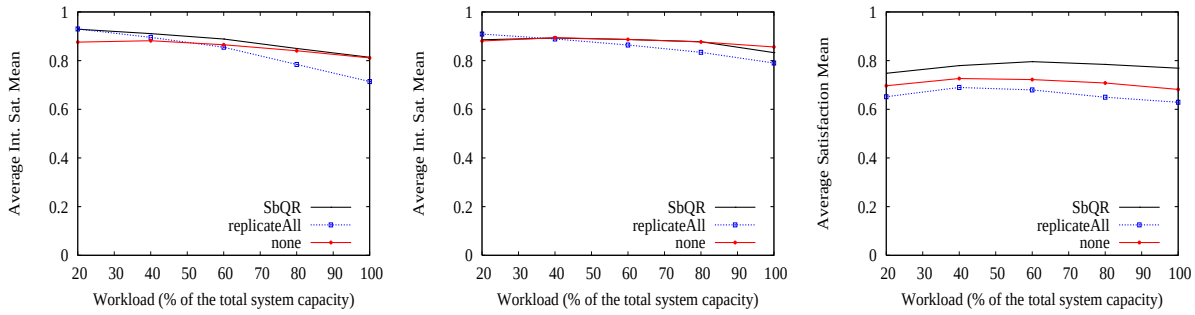
All above results demonstrate the efficiency of S_bQR to replicate queries so to increase participants' satisfaction while ensuring good system performance. In the following, we go further with our validations with the aim of evaluating how well S_bQR deals with different queries criticity and with high probabilities of participants' failure.

4.4.2.1 Varying Criticity of Queries

First of all, let us say that to better evaluate the impact of queries criticity, we assume in these experimentations that consumers require two results from different providers, i.e. $q.n = 2$. In this case, a



(a) Untreated queries. (b) Backup queries.



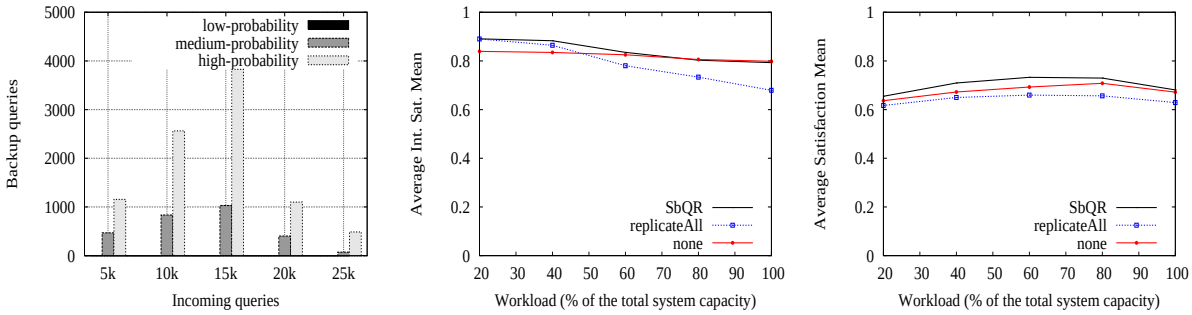
(c) Consumers' satisfaction with a query criticity of 1. (d) Consumers' satisfaction with a query criticity of 0. (e) Providers' satisfaction with a query criticity of 1.

Figure 4.3 – Results with different query criticity values and different workloads.

query q without answers denotes a result for less than the required. For example, a consumer, requiring 3 results for a given query, that receives 1 result, we say that there were 2 queries without answer.

Having stressed this, we start again by discussing the number of queries without answers and of the number of created backup queries. Notice that the criticity of queries does not impact on the performance of $replicateAll$ and the $none$ case because $replicateAll$ replicates all incoming queries independently of their criticity and in the $none$ case no query is replicated. This is why we only show the results for S_bQR. In Figures 4.3(a) and 4.3(b), we illustrate the number of queries that S_bQR has without answers and the number of created backup queries, respectively, for different workloads and for different values of criticity : with value of 0 (low-critical), with value of 0.5 (medium-critical), and with value of 1 (high-critical). We can observe in Figure 4.3(a) that, as expected, S_bQR has more queries without answers for low-critical queries than for medium and high-critical queries. This is because S_bQR tends to replicate much more queries for medium and high-critical queries than for low-critical queries (see Figure 4.3(b)). It is worth noting that providers quickly becomes overutilized for high-critical queries and hence S_bQR decreases the number of backup queries it creates from workloads of 60% of the total system capacity (15k) while it does so for medium-critical queries from 80% of the total system capacity (2k). This phenomenon does not occur for low-critical queries since the number of backup queries created by S_bQR does not overutilize providers.

Concerning consumers' satisfaction, we show in Figures 4.3(c) and 4.3(d) these results for criticity values of 1 and 0, respectively, and for different workloads. We can observe that all three S_bQR, $replicateAll$, and $none$ case, suffer for high workloads since it is in these cases that there are more



(a) Backup queries with a high probability of failure.

(b) Consumers' satisfaction with a high probability of failure.

(c) Providers' satisfaction with a high probability of failure.

Figure 4.4 – Results with a high-probability of failure and different workloads.

queries without answers, which, given the criticty, significantly impact consumers' satisfaction. This is why they less suffer for criticity values of 0 than for criticity values of 1. In fact, we can see that for a criticity value of 0, S_bQR has the same performance as the $none$ case except for a workload of 100% of the total system capacity. Notice that, given our Definition 33, the criticity of queries does not impact at all on providers' satisfaction. We show this in Figure 4.3(e) where we can observe that providers has the same satisfaction as in Figure 4.2(f) of previous section. In conclusion, we can say that, even for high-critical queries, S_bQR can ensure good consumers' satisfaction while providers are also satisfied.

4.4.2.2 Dealing with High Probabilities of Providers' Failure

We now validate S_bQR in systems where providers have different probabilities of failure per second : with a probability of 0.006 (low-probability), with a probability of 0.05 (medium-probability), and with a probability of 0.1 (high-probability). Moreover, we assume that queries arrive with a criticity of 1. As in previous section, we assume that consumers ask for two answers per query, i.e. $q.n = 2$.

In Figure 4.4(a), we can observe that the higher the failure probability of providers is, the more are the backup queries created by S_bQR so as to ensure that consumers get answers for their queries. However, we can observe again that when providers become overutilized, S_bQR starts to decrease the number of backup queries it creates. This is why S_bQR creates less backup queries from workloads of 80% of the total system capacity (i.e. from 20k incoming queries). In Figure 4.4(b), we can observe that in the worst case S_bQR satisfies consumers as well as the $none$ case. This not the case for $replicateAll$ that better satisfies consumers than the $none$ case only for workloads of 20% and 40% of the total system capacity. On the other side, we can see in Figure 4.4(c) that S_bQR always satisfies providers than $replicateAll$ and the $none$ case. In these results, it is worth noting all three S_bQR, $replicateAll$, and $none$ case, the providers' satisfaction is smaller than in previous section because of providers failures. Notice that in these two last results (Figures 4.4(b) and 4.4(c)) the performance of S_bQR is close to the $none$ case because participants fail several times (due to their high-probability of failure) and they compute their satisfaction only for those queries in which they do not fail.

4.5 Related Work

Most work on distributed query processing [Kos00] has been done to support access to multiple distributed, autonomous sources (providers), particularly addressing issues relating to heterogeneity, con-

sistency, and availability. However, systems have tended to gather data to a central site for some query processing issues such as query planning and inter-site joins. Furthermore, the emergence of new distributed systems such as grid computing [Fe99] provides support and motivation for the evolution of the more open query processing espoused in this thesis and in [BKK+01], where participants contribute not just data but also computational services. In these environments many widely distributed and autonomous providers may be utilized in the execution of a particular query and hence providers' failures may be not only likely but also costly. It is then better to tolerate possible faults of participants rather than throwing away the query or the work already done.

A possibility to deal with this is to duplicate logical resources such as data or queries at different physical locations. Replicating data near the point of its use makes communication both cheaper and faster. Data replication has been the focus of several works in different research areas [BE08, JQL06, MPV06, PGVA08]. Then, a simple solution to support failures is to re-allocate failed queries, after their failure detection, to those providers having a replica of the data or service. However, this solution inherently assumes that providers agree to share their data with others and that services are homogeneous, i.e. that providers produce quite similar results for a same query, which is not the case for the environments we consider. Moreover, this may significantly penalize consumers with long response times. Therefore, dealing with participants' failures in a preventive way so that a consumer gets, in short times, the number of results it requires for its queries is a key goal in large-scale distributed computing. Query replication allows this and may tremendously increases the performance of large distributed systems. In case of provider's failure the consumer still has access to the requested service or data at a different location. Notice that data replication and query replication are complementary approaches and one can use both two approaches together to have better performances when supporting providers' failures.

In this chapter, we addressed the problem of dealing with participants' failure by replicating queries when this improves the global satisfaction (i.e. the satisfaction of participants increases). Research in fault tolerance aims at making distributed systems more reliable by handling providers' failures in complex environments. Fault tolerance in distributed systems is a wide area with a significant body of literature that is vastly diverse in methodology and terminology. Thus, in Section 4.5.1, we discuss the most relevant works based on query replication only. But, since some of these works use rollback-recovery protocols to restore failed query processes, we discuss the most relevant protocols to recover query processes from providers' failures in Section 4.5.2. Finally, we make some concluding remarks in Section 4.5.3.

4.5.1 Query Replication

To ensure that a consumer get answers for their queries despite providers' failures, one can replicate a same query over a set of redundant, physically independent providers so that if some of these fail, the remaining ones provide the answer to the query. In other words, a query is allocated to some backup providers (which we refer to backup queries) besides the providers required by a consumer (which we refer to primary queries). We say that a set of redundant providers masks the failure of a provider in the redundant group whenever the required answers are returned to the consumer despite such a provider's failure. The output of a set of redundant providers is a function of the outputs of each provider in such a group. For example, a redundant group output can be the output generated by the fastest providers of the group, the output generated by some distinguished providers of the group, or the result of a majority vote on group providers' outputs. A group of backup providers able to mask from its consumer any l concurrent providers' failures will be termed l-fault tolerant ; when l is 1, the group will be called single-fault tolerant, and when l is greater than 1, the group will be called multiple-fault tolerant. The specific mechanisms needed for managing a redundant provider group in a way that masks member

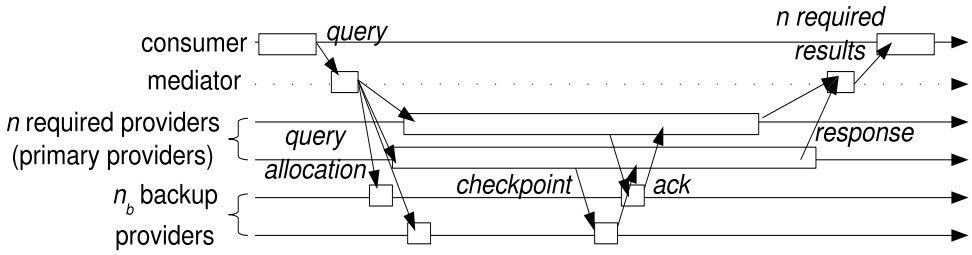


Figure 4.5 – Passive redundancy model.

providers' failures and at the same time makes the group behavior functionally indistinguishable from that of single provider strongly depend on the synchronization policies used. For any incoming query, the synchronization policy of a set of redundant providers prescribes the degree of local state synchronization that must exist among the redundant providers. Fault-tolerant techniques based on query replication can be classified into two categories : *passive redundancy* or *active redundancy* model. We discuss both models in the following two sections.

4.5.1.1 Passive Redundancy

In the passive replica model, as well know as the primary-backup model [BMST93], some of the replicas, called the primary providers, plays a special role : they actively performs queries and only them return their produced results to the mediator. The principle of this model, which we illustrate in Figure 4.5, is as follows. Given a query q issued by a consumer $q.c$, the mediator allocates q to the $q.n$ providers required by $q.c$ plus n_b backup providers. The $q.n$ providers perform q return their produced result to the mediator, while the n_b backup providers are in standby. In other words, only $q.n$ primary queries are running actively at any one time and those queries on the remaining providers (the backup queries) are non-active process. Primary providers regularly checkpoints their state to backup providers, which are either waiting for a checkpointing message or saving a checkpointing message. Finally, in case that a primary provider does not fail, it returns its results to the mediator, which finally sends the $q.n$ required results to consumer $q.c$. If a primary provider fails, a backup provider takes over the role of the primary provider by reading the last checkpointed state so as to recover a state that existed before the primary provider's failure. In this way, the failure can be masked to consumers, which only experience a delay in getting results.

Several commercial systems such as Delta-4 and Tandem [NF92, SB89, SS92] use the passive redundancy model to support fault-tolerance. The Paralex system [DGB+96], which also supports fault-tolerance by passive replication, dynamically balances queries among providers by the "late binding" of primary queries. Similarly, *Kim et al.* [KLL97] proposed a query allocation algorithm to dynamically balance queries among providers. They duplicate each incoming query and uses common techniques of checkpointing (see Section 4.5.2) to reflect primary queries' state in backup queries. [KLL97] consider the possibility of a single provider failure only. *Lee et al.* [LKH+95] duplicate incoming queries once, but re-generate backup providers as many times as primary providers fails. This allows to support several providers' failure. The only restriction, that they do to allow this, is that as primary queries as backup queries be allocated to different providers.

For achieving high availability, research in stream processing systems has focused also on passive replication by storing data and checkpointing [HBR+05b]. Commercial workflow systems [Cor03] also

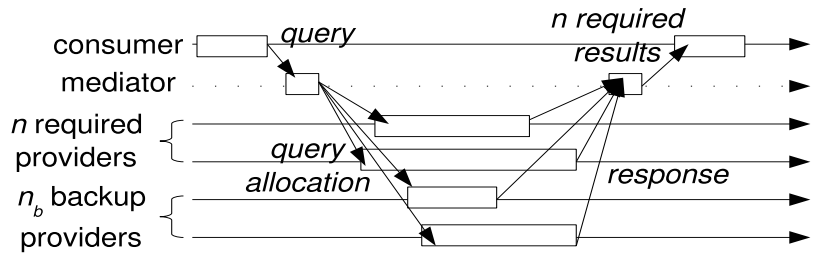


Figure 4.6 – Active redundancy model.

rely on this model to achieve high availability. Exotica workflow system [KAGM96], instead of backing up process states, logs changes to the workflow components, which store inter-query messages. Recently, *Bansal et al.* [BBJ+08] proposed a query replication algorithm based on the passive redundancy model for partial-fault tolerant applications, such as SDIMS [YD04] and PIER [HHL+03] that likely to be able to tolerate some missing objects while processing a query (e.g. AVG, MEDIAN, etc.) on a distributed database.

However, all above research works are inappropriate for the environments we focus on because it inherently assumes that providers are homogeneous from a functionality and data point of view such that they provide the same results for queries. Furthermore, to maintain replicas may incur significant overheads to provide strict consistency requirements [YV01] and such overheads can limit the benefits of checkpointing approaches.

4.5.1.2 Active Redundancy

In the active replication technique, also called *state-machine* approach [Sch93], all replicas play the same role : there is here no centralized control, as in the passive redundancy model. This model prescribes that local providers' states are closely synchronized to each other by letting all providers execute all queries in parallel and go through the same sequence of state transitions. In other words, as primary queries as backup queries are active processes running at any one time. The principle of the active redundancy model, which we illustrate in Figure 4.6, is as follows. Given a query q issued by a consumer $q.c$, the mediator allocates q to the $q.n$ providers required by $q.c$ plus n_b backup providers. The $q.n$ and n_b providers perform q in parallel and return their produced result to the mediator. Finally, the mediator returns the results from the $q.n$ fastest or best ranked provider to $q.c$. An advantage of this model is that it does not require checkpointing messages to maintain backup queries and that it is appropriate to environments where providers perform queries differently and provide different results.

Several proposals have been done based on this model. For example, *Oh and Son* [OS92] proposed and scheduling algorithm for multiprocessor systems based on the active redundant model. The reliability of a query is ensured by creating 1 backup query of each incoming query. Similarly, *Shatz et al.* [SWG92] proposed a query allocation algorithm that maximizes the reliability of heterogeneous systems. *Hashimoto et al.* [HTK02] proposed a scheduling algorithm to achieve fault tolerance in multiprocessor systems. Their algorithm first partitions a parallel program into subsets of queries, based on the notion of height of a query graph. For each subset, the algorithm then duplicates once and schedules the query in the subset successively. Thus, as for [OS92], their proposed algorithm can tolerate a single processor failure. *Girault et al.* [GKS03] proposed an algorithm consisting of a set of scheduling heuristics that actively replicates each incoming query a fixed number of times, say r, therefore producing sched-

ules that tolerate $d-1$ provider failures. Moreover, a number of rules for transforming non fault-tolerant services implemented by non-redundant application programs into fault-tolerant services implemented by active redundant model have been proposed in [Nie90].

A common assumption of all above research works is that a backup query has the same load as a primary query. Therefore, to replicate all incoming query implies having several active processes, which may quickly utilize all computational resources in the system. Recently, probabilistic approaches have been recently proposed to improve the reliability without increasing too much the number of redundant providers. In probabilistic approaches providers, hardware, or software components are characterized by a probability of failure, which can depend on e.g. the query duration and query complexity. Then, given a set of relevant provider to perform a given query a precise analysis can determine for each incoming query if duplication is required. An advantage of a probabilistic approach is that no assumption on the number of tolerated failures is made.

Assayad et al. [AGK04] proposed a bi-criteria scheduling heuristic for scheduling data-flow graphs of operations onto parallel heterogeneous architectures according to two criteria : the minimization of the schedule length and the maximization of the probability that an operation be successfully treated. Generally speaking, the proposed algorithm is a set of heuristics, based on a bi-criteria compromise function that introduces priority between the operations to be scheduled, and that chooses on what providers they should be scheduled. In this proposal each processor and communication link is associated with a failure rate. The authors then tackle the problem of improving reliability by using the active model query replication. If the system reliability or the schedule length requirements are not met, the a parameter of the compromise function can be changed and the algorithm re-executed. This process is iterated until both requirements are met.

Berten et al. [BGJ06], authors proposed two probabilistic algorithms to replicate queries in order to *(i)* given a maximum tolerated probability of provider's failure, minimize the number of required redundant providers such at least one of them terminates a query, and *(ii)* given the number of redundant providers find the best achievable reliability. Given a set of incoming queries Q, both algorithms increase the number of query replicas according to one of the following 5 heuristics. First, each incoming query $q \in Q$ is replicated. Second, only the query $q \in Q$ whose number of subqueries minimally increases the number of required providers is replicated. Third, a query $q \in Q$ is replicated if it is allocated to that provider having the highest failure probability. Fourth, the query $q \in Q$ having a high probability of failure and a great number of subqueries. Finally, this fifth heuristic combines the second and third heuristics, that is, it is replicated the query $q \in Q$ that requires a low the number of providers and that is allocated to that provider having a high failure probability. However, authors assume an identical multiprocessor platform.

4.5.2 Rollback-Recovery Protocols

The problem of rollback-recovery in distributed systems has been extensively studies (see [EAWJ02] for a survey in message-passing systems). Rollback recovery treats a distributed system as a collection of application processes that communicate through a network. It achieves fault tolerance by periodically saving the state of a process during failure-free execution, and restarting from a saved state upon a failure to reduce the amount of lost work. The saved computation state of a query is called a checkpoint, and the procedure of restarting from previously checkpointed state is called rollback-recovery. A checkpoint can be saved on either stable storage or the volatile storage of another process, depending on the failure scenarios to be tolerated. The providers have access to a stable storage device that survives all tolerated failures so that upon a failure, a failed provider uses the saved information to restart the computation

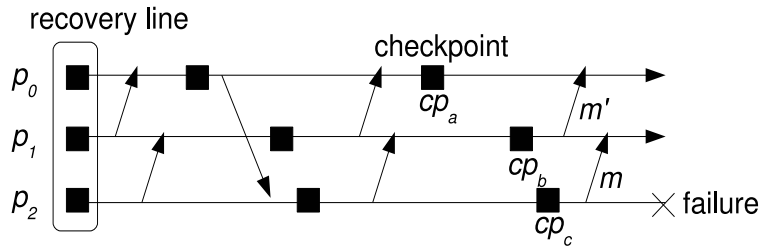


Figure 4.7 – Recovery line, rollback propagation, and domino effect.

from an intermediate state, thereby reducing the amount of lost computation. The recovery information includes at a minimum the states of the participating providers, called checkpoints. Other recovery protocols may require additional information, such as logs of the interactions with input and output devices, events that occur to each provider, and messages exchanged among the providers. We briefly discuss checkpointing protocols in Section 4.5.2.1 and logging protocols in Section 4.5.2.2.

4.5.2.1 Checkpointing Protocols

In checkpointing protocols, each provider periodically saves its state on stable storage. The state should contain sufficient information to restart a query execution. A consistent global checkpoint refers to a set of local checkpoints, one from each primary query, which forms a consistent system state. Any consistent global checkpoint can be used for system restoration upon a failure. To minimize the amount of lost work, the most recent consistent global checkpoint, called the recovery line [Ran75], is the best choice. Figure 4.7 gives an example where primary providers are allowed to take their checkpoints independently, without coordinating with each other. A black bar represents a checkpoint, and each provider is assumed to start its execution with an initial checkpoint. Suppose provider p_2 fails and rolls back to checkpoint cp_c. The rollback "unsends" message m and so provider p_1 is required to roll back to checkpoint cp_b to "unreceive" m. The rollback of p_2 thus propagates to p_1, therefore the term rollback propagation. p_1's rollback further "unsend" m' and forces p_0 to roll back as well. Such cascading rollback propagation can eventually lead to an unbounded rollback, called the domino effect [Ran75]. The recovery line for the single failure of p_2 consists of the initial checkpoints. Thus, the system has to roll back to the beginning of its execution and loses all useful work in spite of all the checkpoints that have been taken.

To avoid the domino effect, several techniques have been developed to prevent it. One such technique is to perform coordinated checkpointing in which primary providers coordinate their checkpoints in order to save a system-wide consistent state [CL85]. This consistent set of checkpoints can then be used to bound rollback propagation. Alternatively, communication-induced checkpointing forces each primary provider to take checkpoints based on information piggybacked on the application messages received from other processes [Rus80]. Checkpoints are taken such that a system-wide consistent state always exists on stable storage, thereby avoiding the domino effect. Therefore, checkpoint-based rollback-recovery relies solely on checkpointed states for system state restoration and depending on when checkpoints are taken, existing approaches can be divided into uncoordinated checkpointing, coordinated checkpointing and communication-induced checkpointing.

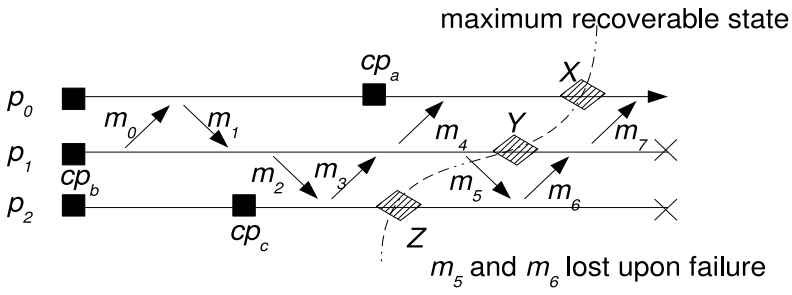


Figure 4.8 – Logging for deterministic replay.

4.5.2.2 Logging Protocols

Log-based rollback recovery uses checkpointing and logging to enable providers to replay the execution of a given query after a failure beyond the most recent checkpoint. This is useful when interactions with the outside world are frequent since it enables a provider to repeat its execution and be consistent with messages sent to other providers without having to take expensive checkpoints before sending such messages. Additionally, log-based recovery generally is not susceptible to the domino effect, thereby allowing primary providers to use uncoordinated checkpointing if desired. Log-based recovery relies on the piecewise deterministic (PWD) assumption [SY85]. Under this assumption, the rollback recovery protocol can identify all the nondeterministic events executed by each process, and for each such event, logs a determinant that contains all information necessary to replay the event should it be necessary during recovery. If the PWD assumption holds, log-based rollback-recovery protocols can recover a failed process and replay its execution as it occurred before the failure.

Log-based recovery relies on the assumptions underlined in a piecewise deterministic (PDW) execution model [SM94, SBY88] and employs an additional logging protocol. Under the PDW assumption, a query execution consists of a sequence of state intervals, each starting with a nondeterministic event such as a message receipt from another query process. The execution within each state interval is deterministic. Thus, by logging every nondeterministic event during failure-free execution and replaying the logged events in their original order during recovery, a provider can replay a query execution beyond the most recent checkpoint. A query state is recoverable if there is sufficient information to replay the execution up to that state despite any future failures in the system.

In Figure 4.8, suppose messages m_5 and m_6 are lost upon the failure affecting both providers p_1 and p_2, while all the other messages survive the failure. Message m_7 becomes an orphan message because provider p_2 cannot guarantee the regeneration of the same m_6 after the rollback, and p_1 cannot guarantee the regeneration of the same m_7 without the original m_6. As a result, the surviving provider p_0 becomes an orphan process and is forced to roll back as well. As indicated in Figure 4.8, providers states X, Y and Z then form the maximum recoverable state [JZ90], i.e., the most recent recoverable consistent system state. Providers p_0 (p_2) rolls back to checkpoint cp_a (cp_c) and replays message m_4 (m_2) to reach X (Z). Provider p_1 rolls back to checkpoint cp_b and replays m_1 and m_3 in their original order to reach Y.

4.5.3 Concluding Remark

As seen in this section, S_bQR is quite related to probabilistic approaches for query replication (e.g. [AGK04, BGJ06]) since both of them consider the failure probability of providers to dynamically set the replication rate of queries. However, S_bQR significantly differs from works on fault tolerance

in two main points. First, in addition to the failure probability of providers, S_bQR considers the failure probability of consumers. This consideration is quite important in environments where providers are autonomous because repeated consumers' failures may cause dissatisfaction departures from the system of those providers that perform their queries. This is because such providers waste their computational resources for producing results that are finally not returned to the consumer. Second, S_bQR goes further than a simple consideration of failures probabilities : it considers both participants' intentions and queries' criticity to set the replication rate of queries. This allows S_bQR to only replicate those queries that increase participants' satisfaction.

4.6 Chapter Summary

We addressed in this chapter the problem of dealing with participants' failures in distributed information systems where participants are autonomous and have special interests towards queries. In particular, we focused on query replication based on the active replica model. The addressed problem is challenging because replicating queries may decrease system performance and may also dissatisfy providers. But, if queries are not replicated, consumers may be dissatisfied because they may get no answer for their queries due to providers' failure. We proposed a *Satisfaction-based Query Replication* technique, S_bQR, that decides to replicate those queries that allows to increase participants' satisfaction (global satisfaction) while ensuring system performance. To our knowledge, this is the first work that addresses the problem of participants' failures with a satisfaction point of view and hence it opens a new issue in this field. In summary, our main contributions in this chapter are the following.

- We proposed a satisfaction model that considers participants' failures. In particular, we revisited the consumer's satisfaction definition so as to characterize the fact that queries have different criticity and that a consumer may receive less results than it expects because of providers' failures. We also revisited the provider's satisfaction definition in order to consider the fact that a provider may perform queries for nothing because of backup queries or consumers' failures.
- We defined the global satisfaction that denotes the expected participants' satisfaction concerning the allocation of a given query. This definition considers : the participants' intentions ; the participants' failure probability, and ; the probability that a query has to be successfully treated.
- We proposed S_bQR, a query replication technique to compute the rate of backup queries according to the global satisfaction. In other words, S_bQR replicates only those queries that allow to increase the satisfaction of participants. A particularity of S_bQR is that it makes no assumption on how many providers' failures can occur at any one time.
- We demonstrated that S_bQR significantly outperforms those techniques that replicate all incoming query (the $replicateAll$ technique). We also demonstrated that S_bQR dynamically adapts to the workload and ensures a good performance even for high probabilities of providers' failure. A key point of our validation is that $replicateAll$ suffers from serious problems of performance for high workloads, but worse it loses more query results than when one does not replicate queries at all.

Future Work As noted above, to deal with participants' failures, we opted to increase the number of providers that have to perform a query when the global satisfaction is increased only. Now, we plan to study, in a near future, the possibility of reducing the number of required providers to perform a query when this also implies to increase the global satisfaction. For example, given an incoming query with a low criticity and requiring results from 3 different providers, it could be better to allocate it to 1 provider because of its criticity and other two providers (which should get the query) do not desire to perform

it. Autonomy and intentions of participants introduce other problem : a participant may act maliciously, that is, it may be Byzantine [LSP82]. We also desire to address this, in a future work, so that consumers obtain high probability of correct acceptance of results with low additional computation, i.e. with a small number of required providers.

Moreover, as participants, in mediator-based distributed systems mediators can fail and can also be Byzantine. Indeed, when a mediator fails, one loses the queries that the mediator was mediating at that moment, but also one loses some information about participants (such as their satisfaction and current money balance). In a future work, we plan to implement a dynamic mechanism of mediators' replacement that allows ensuring the continuity of both consumers' queries and participants' information. Finally, notice that we validated S_bQR on the top of S_bQA. We now are interested in validating S_bQR on the top of $\$_bQA$ so as to handle with participants' failures in multi-mediator systems. We reported this to future work because our objective, in this chapter, was to only study the impact on system performance of considering participants' satisfaction when replicating queries.

Conclusion

We summarize in this chapter the main contributions of this thesis and discuss some future directions of research for query processing in large-scale distributed information systems where participants are autonomous and have special interests towards queries.

Summary

This work took place in the context of the *Atlas Peer-to-Peer Architecture* (APPA) [AM07] and of several joint projects including : the Grid4All European STREP project [gri], the ANR Massive Data Projects MDP2P [mdp], and Respire project [res]. In this thesis, we have addressed the query allocation problem in large-scale distributed systems where participants (consumers and providers) may join and leave the system at any time, but also they have special interests towards queries. This work was mainly motivated by the great interest of several enterprises, individuals, and research groups to collaborate, share, and do business in a previously impossible scale. For example, just to mention the most important, applications such as SETI@home, eBay, as well as Web 2.0 follow this common goal. Most of the works in this context has focused on distributing the query load among the providers in a way that maximizes system performance (typically high throughput and short response times), i.e. *query load balancing* (*qlb*) [ABKU99, GBGM04, MTS90, RM95, SKS92]. However, these work are not adequate for the environments we considered in this thesis because of interests of participants, which are not only performance related. In this thesis, we aim at providing a complete solution to the query allocation problem in this kind of distributed information systems. To achieve this goal, we proceed in fourth steps. First, we proposed a general model to characterize participants' interests in the long run and defined some properties that allow evaluating query allocation methods. Second, we proposed a set of algorithms that allows allocating queries while considering participants' satisfaction, participants' intentions, and the kinds of application. Third, we proposed a query allocation method based on virtual money that allows to scale up to several mediators with a small network cost. Finally, we proposed a query replication technique based on satisfaction and failure probability of participants in order to preserve participants' satisfaction even in the presence of faulty participants, i.e. participants that can fail by unexpected reasons such as network failures.

Main Contributions

Generally speaking, the main objective of this thesis has been to provide a complete query allocation framework for distributed information systems that satisfies participants by respecting, in the long-run, their intentions to allocate and perform queries. In particular, our main contributions are as follows.

Modeling We characterized, in the long-run, the participants' intentions in a new model that allows evaluating a system from a satisfaction point of view [QRLV07b, QRLV07c]. We also made precise what providers' utilization and query starvation means in distributed information systems with autonomous participants. We showed that this model can easily be used to design new query allocation methods for distributed systems that are confronted to autonomous participants. Similarly, we defined some system

properties that allow evaluating the quality of query allocation methods and propose measures to do so. Finally, we demonstrated that the proposed model can predict possible participants' departures from the system and allows comparing query allocation methods having different approaches to regulate the system.

Query Allocation We formally defined the query allocation problem in distributed information systems with autonomous participants. We proposed S_bQA (for *Satisfaction-based Query Load Balancing*) [LQRV07, QRLV06, QRLV07a, QRLV07b], a flexible framework with *self-adapting* algorithms to allocate queries while considering both *qlb* and participants' intentions and that afford : *(a)* consumers the flexibility to trade their preferences for the providers' reputation ; *(b)* providers the flexibility to trade their preferences for their utilization ; *(c)* a mediator to trade consumers' intentions for providers' intentions according to their satisfaction ; and *(d)* a mediator the flexibility to adapt the query allocation process to the application by varying several parameters. We analytically demonstrated that, to perform queries, S_bQA only requires 2 more network messages per query than baseline methods. We experimentally demonstrated that S_bQA significantly outperforms baseline methods and yields significant performance benefits. Similarly, we demonstrated the self-adaptability of S_bQA to participants' intentions and its adaptability to different kinds of application. Also, we demonstrated with our S_bQA validation that it can scale up in systems with autonomous participants, while baseline method cannot. Finally, using the *Berkeley Open Infrastructure for Network Computing* (BOINC) platform, we demonstrated the flexibility and efficiency of S_bQA to satisfy participants while allocating queries [QRLV08].

Scale Up We aimed at scaling query allocation up to several mediators [QRLCV07a, QRLCV07b, QRLCV08]. To this end, we first exposed the challenges of using virtual money as a means of regulation and made precise a way in which virtual money should circulate within a system. We formally stated the number of network messages required by a mediator to control the flow of virtual money. And, conversely to the expected, we demonstrated that only 3 network messages are required, per query, to control the flow of virtual money. We proposed $\$_bQA$ (for *Economic Satisfaction-based Query Allocation*) for distributed information systems with several mediators allocating queries cooperatively. In particular : *(a)* we defined how a provider may compute its bid by considering its preferences, its satisfaction, its current utilization, and its current virtual money balance. Similarly, we defined three strategies that allows a provider to bid for queries in the presence of several mediators ; *(b)* we defined how a mediator allocates queries by considering both consumers' intentions and providers' bids and how it should invoice providers even when one of them is imposed a query ; and *(c)* we formally demonstrated that $\$_bQA$ requires only 3 more network messages than S_bQA. Moreover, we analytically demonstrated that $\$_bQA$ allows a VO to scale up to several mediators with no additional network cost with respect to a VO with a single mediator, which makes $\$_bQA$ strong with respect to baseline methods. Finally, we experimentally demonstrated that $\$_bQA$ can easily scale up in terms of number of mediators, participants, and incoming queries, while ensuring good system performance and the same participants' satisfaction as in systems with a single mediator. A key result is that, conversely to several proposals, one must pay the same attention to the selection, invoicing, and bidding phases when designing a microeconomic query allocation method.

Query Replication We focused on query replication based on the active replica model so as to deal with participants' failures in environments where participants are autonomous and have special interests towards queries. In this context, we first proposed a satisfaction model that considers participants' failures. In particular, we characterized the fact that queries have different criticity and that a consumer may

receive less results than it expects because of providers' failures. We similarly characterized the fact that a provider may perform queries for nothing because of backup queries or consumers' failures. This leads to revisited consumer's and provider's satisfaction definitions that consider such new considerations. Also, we defined the expected participants' satisfaction (noted as global satisfaction) concerning the allocation of a given query, which considers the participants' intentions, the participants' failure probability, and the probability that a query has to be successfully treated. Then, we proposed S_bQR (for *Satisfaction-based Query Replication*), a query replication technique to compute the rate of backup queries according to the global satisfaction. The goal of S_bQR is to replicate only those incoming queries that allows to increase global satisfaction. A strong feature of S_bQR is that, conversely to most query replication approaches, it makes no assumption on how many providers' failures can occur at any one time. Finally, we experimentally demonstrated that S_bQR significantly outperforms those techniques that replicate all incoming query. We also demonstrated that S_bQR dynamically adapts to the workload and ensures a good performance even for high probabilities of providers' failure. A key result is that by replicating all incoming query causes serious problems of performance for high workloads, but worse originates more losses of query results than when one does not replicate queries at all, which is not the case for S_bQR.

Future Work

Although this thesis provide a complete query allocation framework for large-scale distributed systems with autonomous participants, there are still several open issues and important directions of future work. We discussed all these points at the end of each chapter (in the summary sections). Then, we only present here the research directions that we desire to pursue : *(i)* explore different ways to compute the preferences, intentions, and satisfaction of participants, *(ii)* study the links between the satisfaction notion we presented in this thesis and the notions of trust [AD01, AG07] and reputation [KSGM03] from distributed systems, *(iii)* analyze the satisfaction of providers at their bid level and evaluate from a satisfaction point of view different query allocation methods based on microeconomics, *(iv)* explore the sociology [Mac04] field to study the possible links between its properties and the properties of the model we proposed in this thesis, *(v)* analyze the possible gains or losses, in system performance and participants' satisfaction, of reducing the number of providers required by a consumer, *(vi)* study, from a satisfaction point of view, Byzantine [LSP82] faults of participants and mediators, and handle with *fail-stop* failures of mediators, and finally *(vii)* validate S_bQR in multi-mediator systems, that is, on the top of $\$_bQA$.

Appendixes

APPENDIX A

The S_bQA Prototype

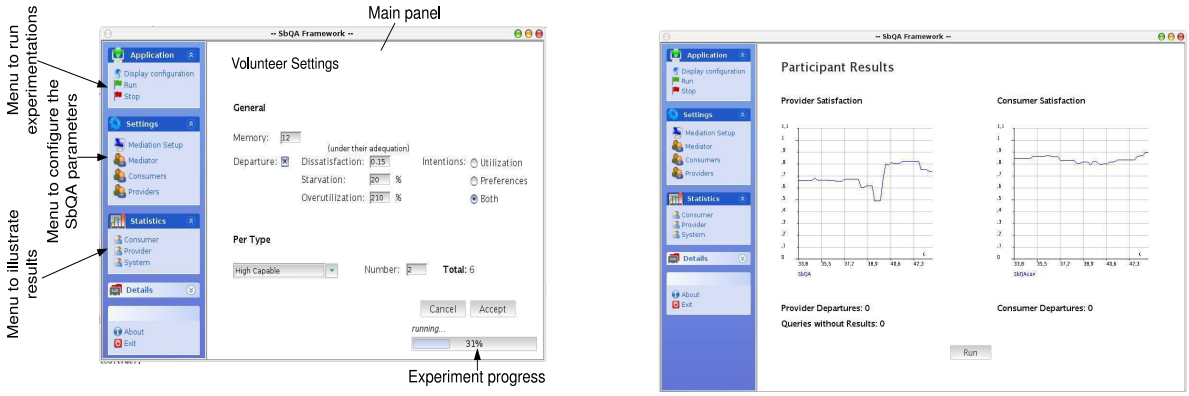
We implemented the S_bQA protype in Java and constructed a SimJava-based network to simulate network messages between participants and mediators. In this appendix, we present a S_bQA's demo we realized using the BOINC platform, a distributed platform for volunteering computing. The goal of such a demo is to show the great benefits of using S_bQA for allocating queries. Then, we discuss the S_bQA implementation within the STREP European Grid4All project [gri].

A.1 S_bQA's demo : A BOINC example

We present in [QRLV08] a demonstration session to show the flexibility and efficiency of S_bQA to allocate queries. In this demonstration, we use the *Berkeley Open Infrastructure for Network Computing* (BOINC) platform as an example of highly autonomous environments. BOINC is a middleware system for volunteer computing. In this context, the consumers are projects, which are usually from the academia, that require computational resources to perform queries and the providers are volunteers that donate computational resources to BOINC-based projects. Participants (i.e. both consumers and providers) in BOINC are autonomous as stated in Section . A query is an independent computational task, specified by a set of input files and an application program. Incoming queries are dispatched by a server (the mediator) to providers. As providers may be malicious, consumers may create several instances of a query so as to validate results returned by providers.

In BOINC, providers can express their intentions by specifying the fraction of computational resources devoted to each consumer. This allows providers to devote more resources to those consumers (projects) in which they are interested. However, this may waste idle computational resources of providers when their interesting consumers do not issue queries. For example, a provider may donate its computational resources to two consumers c_a and c_b in a fraction of 80% and 20%, respectively. In this case, c_b cannot use more than the assigned 20% of computational resources even if c_a is not generating queries. S_bQA could allow BOINC-providers to express their intentions in a more flexible way so that their donated computational resources be properly exploited while their intentions be also satisfied. On the other side, consumers cannot express their intentions with respect to providers in BOINC. Our framework may be used by BOINC designers to allow consumers to express intentions towards providers such as reputation-based preferences. To illustrate the benefits of S_bQA, we provide a set of GUIs that enable the user to setup the experimentations and S_bQA and that enable the display of all the relevant information (e.g. participants' satisfaction and response times) to illustrate how S_bQA performs. Figure A.1 shows some of these GUIs.

To demonstrate the S_bQA's benefits, we mainly focus on : the way in which queries are allocated by S_bQA ; how it adapts the query allocation process to the participants' intentions, and ; how it can be adapted to the kind of applications. With this in mind, we consider a system consisting, for simplicity, of three consumers, i.e. three different research projects. For clarity, we assume that those projects are



(a) Capturing volunteer settings

(b) Drawing results on-line

Figure A.1 – Some S$_b$QA GUIs.

the SETI@home [set], proteins@home [pro], and Einstein@home [ein]. We create a set of volunteers devoting their computational resources to all three projects in a way that : *(i)* SETI@home is popular, i.e. the majority of providers want to collaborate in this project, *(ii)* proteins@home is normal, i.e. great number, but not most, of providers want to collaborate in this project, *(iii)* Einstein@home is unpopular, i.e. most providers desire to collaborate, in this project, with a small fraction of computational resources. Then, we consider the following example scenarios.

Scenario 1. First of all, using the proposed satisfaction model, we compare, from a satisfaction point of view, the way in which BOINC allocates queries, which is equivalent to the $Capacity\ based$ method, with the *Mariposa-like* method. In this evaluation, we assume *captive environments*, that is, participants are not allowed to quit the BOINC platform. An example of these environments is when consumers use BOINC as platform for grid computing and they put in dedicated computers at their service [des]. This scenario demonstrates that our satisfaction model allows analyzing different query allocation techniques even if the way in which they allocate queries differs.

Scenario 2. We evaluate again baseline techniques, as in Scenario 1, but this time considering that BOINC is used as platform for volunteer computing, i.e. when participants are autonomous to leave the system. On the one hand, we assume that a provider leaves the BOINC platform if its satisfaction is smaller than 0.35. On the other hand, we assume that a consumer stops using BOINC if its satisfaction is smaller than 0.5. This scenario allows us to see that using our satisfaction model one can predict possible participant's departure by dissatisfaction.

Scenario 3. We evaluate S_bQA in an environment as in scenario 1 and compare its performance results (participants' *satisfaction* and response times) with those of baseline techniques. In such a comparison, we show that S_bQA's performance is not far from those of baseline techniques. This demonstrates that S_bQA is suitable for captive environments even if it was not designed for.

Scenario 4. We run again the evaluation of Scenario 3 but, now, in autonomous environments instead of captive ones. Our objective is to illustrate that S_bQA can significantly improve the performance of BOINC-based projects by preserving most volunteers online and hence more computational resources.

Scenario 5. We consider the same evaluation of Scenario 3, but we modify the manner in which participants compute their *intentions* so that projects be interested only in response times and volunteers be interested in their load. In this case, we show that S_bQA significantly improves response times and balances better queries among volunteers, which is what participants prefer. This proves that S_bQA

adapts to the participants' interests and thus can deal with heterogeneous participants (from their interests point of view), which may allow BOINC-based projects to have more volunteers.

Scenario 6. We consider an application whose goal is to ensure low response times to consumers and that is still composed by autonomous providers. We assume again that participants compute their *intentions* by considering their preferences. An example of this application is when the BOINC platform is used for grid computing, but the computational resources composing the grid are still donated by volunteers. In this context, besides ensuring low response times, BOINC should ensure some level of satisfaction at the providers' side so that they do not quit their resources from the grid. We demonstrate that S_bQA can be adapted to perform in such applications by varying parameter k_n of the K_nBest strategy and the manner in which the mediator scores providers, i.e. by varying parameter ω.

Scenario 7. We allow people attending the demo to play the role of a consumer or provider. The goal is to enable a person to set her own preferences and intentions, and observe how the different mediations react and which ones allow her to reach her objectives. Allowing this, people attending the demo could obtain a clear picture of the performance that the different mediations may have when they are confronted to human participants having different interests. In this scenario, we aim at demonstrating that the S_bQA mediation used by S_bQA is the only one that allows a participant to reach its objectives in all cases.

A.2 S_bQA within Grid4All

Grid4All embraces the vision of a democratic Grid as a ubiquitous utility whereby domestic users, small organizations and enterprises may share their resources and services, and use resources via the Internet without having to individually invest and manage computing and information technology (IT) resources. Generally speaking, Grid4All aims at bringing global computing to the broader society beyond that of academia and large enterprises by providing an opportunity to small organizations and individuals to reap the cost benefit of resource sharing without, however, the burdens of management, security and administration. Specifically, the objectives of the Grid4All project are :

- Alleviate administration and management of large scale distributed IT infrastructure,
- Provide self-management capabilities to provide scalability and resilience to failures and volatility,
- Widen the scope of Grid technologies by enabling on-demand creation and maintenance of dynamically evolving scalable virtual organizations even short lived,
- Capitalize on Grids as revenue generating sources to implement utility models of computing but using resources on the Internet.

In the context of Grid4All, there is the need for the discovery of available participants providing suitable resources and services for incoming queries. To achieve this, it is implemented a *Semantic Information Service* (SIS) that facilitates the discovery of both resources and services within the grid. SIS provides a matching and selection service between participants that provide or query resources and services within grid environments. SIS may be queried by software agents as well as by human users to select advertised resources and services. For resources as well as for services, queries are first matched to resources/services and a the list of found participants is ranked, or narrow down, according to participants' preferences. Thus, SIS is internally composed by two services : the *Matchmaking Service* (MS) and the *Selection Service* (SS). Generally speaking, SIS works as follows. Given an incoming query into SIS, MS first found the relevant participants providing the resources or services requested by the query. Then, it passes the found participants to SS, which ranks, or narrows down, the list of providers according to both (i) the participants' preferences, i.e. the preferences of the consumer and the found providers, and (ii) the query load of providers. To this end, participants declare, at any time, their preferences to SS

so as to get those providers and queries they prefer at the top of (or included in) the list of participants returned by SIS. S_bQA plays the role of SS within SIS, that is, it is used as the basis of ranking and selection of resources and services in Grid4All.

A.2.1 Grid4All Example Application

In this section, we discuss the importance of SS (i.e. of S_bQA) within Grid4All by means of one of the Grid4All example applications. We consider the market-oriented environment Grid application, which allows participants (e.g. home users, enterprises, and organizations) to share their computational resources and services to others. Resources and services are made available through markets initiated either by providers, consumers or third party entities. Markets are initiated by means of resource/service orders that participants issue in a distributed manner. Such an order can be either the request of a consumer, or an offer of a provider. Adopting such a distributed market model, resource consumers and providers negotiate over resources using auctions that run in markets.

The SIS provides a registry of the e-markets available. Consumers query the SIS for orders that match certain attributes and criteria. SIS uses MS to locate orders that concern matching resources, which returns the available providers (or e-markets) relevant to the query. Then, SIS may return the list of relevant providers to consumers or only the n most relevant providers. But, doing so has the following drawbacks. First, the list of relevant providers can be large, which can make difficult the selection task to the consumer. Second, this allows that some providers monopolize queries (or that some providers become overutilized) while some other providers suffer from query starvation (or that some computational resources are not exploited). Finally, if only the properties of resources and services are considered to select providers, as consumers as providers may become dissatisfied in the long-run because their preferences towards providers and queries, respectively, are not considered. It is here that SS plays an important role to the well operation of the system by ranking and selecting if necessary the relevant providers according to the preferences of participants. In other words, on the one hand, SS allows consumers to see their preferred providers in the top of the lists of providers returned by SIS. On the other hand, SS allows providers to generally get queries of their interests and to have almost the same chances as other providers of doing business.

A.2.2 Selection Service Specification

In this section, we present the interface exposed by SS to both MS and participants. In the following, we present the exposed SS's methods and indicate if a method is internal (invoked by MS) or external (invoked by participants).

Method :	informFinalSelection	
Parameters :	String queryId	The identifier of the query
	Collection selectedProvidersId	The identifier of the set of providers that the consumer finally chose.
Description :	This method allows to memorize the providers who get the queries so that SS updates participants' satisfaction and providers' load.	
Type :	internal	

Method :	selectProviders	
Parameters :	String queryId	The identifier of the query.
	Collection queryTypes	The set of leaf concepts (of the ontology) that concerns the query. This is a collection of Strings.
	String consumerId	The identifier of the query source that has initiated the query.
	Collection providersId	The set of relevant providers' identifier that can deal with the query.
	<int nbRequiredProviders>	The number of required providers by the consumer.
Description :	This method allows to narrow down the set of relevant providers found by MS so as to facilitate the final choice of consumers.	
Type :	internal	

Method :	subscribeConsumer	
Parameters :	String consumerId	The identifier of the consumer to create in SS.
Description :	This method allows MS to subscribe a consumer in SS so as to after considering its preferences when narrowing down a set of relevant providers.	
Type :	internal	

Method :	subscribeProvider	
Parameters :	String providerId	The identifier of the provider to create in SS.
Description :	This method allows MS to subscribe a provider in SS so as to after considering its preferences when narrowing down a set of relevant providers.	
Type :	internal	

Method :	unsubscribeConsumer	
Parameters :	String consumerId	The identifier of the consumer to delete from SS.
Description :	This method allows MS to unsubscribe a consumer from SS. By deleting a consumer, it can no more declare its preferences in SS.	
Type :	internal	

Method :	unsubscribeProvider	
Parameters :	String providerId	The identifier of the provider to delete from SS.
Description :	This method allows MS to unsubscribe a provider from SS. By deleting a provider, it can no more declare its preferences in SS.	
Type :	internal	

Method :	setQueryTypes	
Parameters :	Collection queryTypes	The set of leaf concepts (of the MS ontology) that a query could be concerned.
Description :	This method allows MS to declare types that a query can be concerned so that providers can after declare its preferences towards them.	
Type :	internal	

Method :	getQueryTypes	
Parameters :	– none –	
Description :	This method allows a provider to know the types that a query can be so as to after declare its preferences towards them.	
Type :	external	

Method :	setConsumerPreferences	
Parameters :	String consumerId	The identifier of the consumer whose preferences have to be updated.
	HashMap preferences	The new consumer's preferences
Description :	This method allows a consumer to set its preferences towards providers, which are abstracted as a tuple <providerId, preferenceValue>.	
Type :	external	

Method :	setProviderPreferences	
Parameters :	String providerId	The identifier of the provider whose preferences have to be updated.
	HashMap preferences	The new provider's preferences
Description :	This method allows a provider to set its preferences towards query, which are abstracted as a tuple <queryType, preferenceValue>.	
Type :	external	

Method :	setConsumerPreferencesByDefault	
Parameters :	String consumerId	The identifier of the consumer.
	double preferenceValue	The preference default value for unknown providers
Description :	This method allows a consumer to modify its preferences by default, which is utilized when it does not know a given provider.	
Type :	external	

Method :	setProviderPreferencesByDefault	
Parameters :	String providerId	The identifier of the provider.
	double preferenceValue	The preference default value for unknown query types
Description :	This method allows a provider to modify its preferences by default, which is utilized when it does not know a given query type.	
Type :	external	

The S_bQA PrototypeNotations

Notations

$\delta_a(i)$	Adequation of a given participant i.
$\delta_{sa}(i)$	System adequation with respect to a given participant i.
$\widehat{P_q}$	Set of providers that received a given query q to perform.
$\widehat{P_q^r}$	Set of providers that received a given query q to perform and where r is the rank of the worst ranked provider in such a set.
$\delta_{as}(i)$	Allocation Satisfaction of a given participant i.
$\overrightarrow{Allocs_q}$	Query allocation vector of a given query q to a given set of providers.
$\mathcal{A}_i^t$	The probability that a given participant i be available at time interval t.
P_q^{ok}	Set of providers that did not fail during the treatment of a given query q.
$\sigma(g, S)$	Ratio of the minimal and maximal g's values in a given set S.
$bid_p(q)$	The bid made by a given provider p for performing a given query q.
$\overrightarrow{B}$	Vector of bids shown by a given set of providers.
$bill_q(p)$	The bill that a given provider p must pay for the allocation of a given query q.
$cap(p)$	Computational capacity of a given provider p to perform queries.
$ci_c(q, p)$	The intention of a consumer c for allocating a given query q to a given provider p.
$\overrightarrow{CI}_q$	Vector of the intentions shown by a consumer to see its query q be performed by some providers.
C	Set of consumers in the system.
$\delta_{ae}(i)$	Query allocation efficacy function with respect to a given participant i.
f_i	The probability of failure of a given participant i.
$f(g, S)$	g's values fairness in a given set S.
k	The memory size of a participant to track previous queries.
$\overrightarrow{L}$	Vector of levels of the set of providers that are able to perform a given query.
$\mathcal{S}_q^h(P')$	The probability that a given query q has to be performed by at most h providers in a given set P'.
$\mu(g, S)$	g's values arithmetic mean in a given set S.
M	Set of mediators in the system.
IQ_c^k	Set of the k last issued queries by a given consumer c.
PQ_p^k	Set of the k last queries proposed by mediator(s) to a given provider p.
$\overrightarrow{CB}_p$	Vector of bids made by a given provider p to those queries the mediator is still mediating.
$\overrightarrow{PPI}_p$	Vector of the intentions shown by a given provider p to perform the k last queries proposed by a mediator.

$\mathcal{I}$	Set of participants in the system, i.e. the set of consumers, providers, and mediators together.
Q_p	Set of pending queries at a given provider p.
$pi_p(q)$	The intention of a given provider p for performing a given query q.
$\pi_p(t, t')$	Profit of a given provider p at time interval $[t..t']$.
$prf_p(q)$	Preference of a given provider p for performing a given query q.
$\overrightarrow{Prf}_c^q$	Vector of preferences of a given consuler consumer c for allocating its query q to the set of providers that are able to perform q.
$\overrightarrow{PC}_q$	Vector of query costs at some providers to perform a given query q.
P	Set of providers in the system.
$\overrightarrow{PI}_q$	Vector of the intentions shown by a given provider to perform a query q.
$q.c$	The identifier of the consumer that has issued query q.
$q.d$	The description of the task to be done to produce results for a query q.
$q.\gamma$	The importance for a consumer to receive the $q.n$ results for its query q.
$q.n$	The number of results required by a consumer for its query q.
Q	Set of incoming queries into the system.
$cost_p(q)$	Cost of a given query q at a given provider p.
$\overrightarrow{R}_q$	Rank vector of a set of providers concerning a given query q.
P_q	Set of providers that are able to perform a given query q.
$rep(p)$	Reputation of a given provider p.
$\widehat{\widehat{P_q}}$	Set of providers whose produced results for a given query q are returned to the consumer.
$\delta_s(i)$	Satisfaction of a given participant i.
$scr_q(p)$	Score of a given provider p for performing a given query q.
bal_p	The current balance of virtual money of a given provider p.
SQ_p^k	Set of the k treated queries by a given provider p among the set of k last proposed queries.
$\Theta(P')$	The global satisfaction of allocating a given query to a given set of providers P'.
$b^{th}(p, l)$	The theoretical bid that a given provider p should to make for reaching a given level l.
t_q	Time units that a given provider requires to perform a query q.
$\mathcal{U}_t(p)$	Utilization of a given provider p at time t.
$\overrightarrow{U}$	Utilization vector of a given set of providers.

Bibliography

[ABKU99] Yossi Azar, Andrei Broder, Anna Karlin, and Eli Upfal.
Balanced Allocations.
SIAM Journal on Computing, 29(1) :180–200, 1999.

[ACKM04] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju.
Web Services : Concepts, Architecture, and Applications.
Springer, 2004.

[ACMD+03] Karl Aberer, Philippe Cudré-Mauroux, Anwitaman Datta, Zoran Despotovic, Manfred Hauswirth, Magdalena Punceva, and Roman Schmidt.
P-Grid : a Self-Organizing Structured P2P System.
SIGMOD Record, 32(3) :29–33, 2003.

[AD01] Karl Aberer and Zoran Despotovic.
Managing Trust in a Peer-2-Peer Information System.
In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 310–317, 2001.

[AEK+00] Khaled Arisha, Thomas Eiter, Sarit Kraus, Fatma Ozcan, Robert Ross, and V. S. Subrahmanian.
IMPACT : Interactive Maryland Platform for Agents Collaborating Together.
IEEE Intelligent Systems, 14(2) :64–72, 2000.

[AG00] D. Abramson and J. Giddy.
High Performance Parametric Modeling with Nimrod/G : Killer Application for the Global Grid ?
In *Proceedings of the International Symposium on Parallel and Distributed Processing (IPDPS)*, page 520, 2000.

[AG07] Donovan Artz and Yolanda Gil.
A Survey of Trust in Computer Science and the Semantic Web.
Web Semantics, 5(2) :58–71, 2007.

[AGK04] Ismail Assayad, Alain Girault, and Hamoudi Kalla.
A Bi-Criteria Scheduling Heuristics for Distributed Embedded Systems Under Reliability and Real-Time Constraints.
In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, pages 347–356, June 2004.

[AHKV03] Micah Adler, Eran Halperin, Richard Karp, and Vijay Vazirani.
A Stochastic Process on the Hypercube with Applications to Peer-to-Peer Networks.
In *Proceedings of the International ACM Symposium on Theory of Computing (STOC)*, pages 575–584, 2003.

[AM07] Reza Akbarinia and Vidal Martins.
Data Management in the APPA System.
Grid Computing, 5(3) :303–317, 2007.

[AMZ03] Gagan Aggarwal, Rajeev Motwani, and An Zhu.

The Load Rebalancing Problem.
In *Proceedings of the International ACM Symposium on Parallel Algorithms and Architectures*, pages 258–265, 2003.

[APS04] Samir Aknine, Suzanne Pinson, and Melvin F. Shakun.
An Extended Multi-Agent Negotiation Protocol.
Autonomous Agents and Multi-Agent Systems, 8(1) :5–45, 2004.

[APV07] Reza Akbarinia, Esther Pacitti, and Patrick Valduriez.
Data Currency in Replicated DHTs.
In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 211–222, 2007.

[BAG00] Rajkumar Buyya, David Abramson, and Jonathan Giddy.
Nimrod/G : An Architecture for a Resource Management and Scheduling System in a Global Computational Grid.
In *Proceedings of the International Conference on High Performance Computing in Asia ?Pacific Region (HPC-Asia)*, 2000.

[BBJ+08] Nikhil Bansal, Ranjita Bhagwan, Navendu Jain, Yoonho Park, Deepak Turaga, and Chitra Venkatramani.
Towards Optimal Resource Allocation in Partial-Fault Tolerant Applications.
In *Proceedings of the International Conference on Computer Communications (INFOCOM)*, pages 1319–1327. IEEE, April 2008.

[BBMS08] Magdalena Balazinska, Hari Balakrishnan, Samuel Madden, and Michael Stonebraker.
Fault-Tolerance in the Borealis Distributed Stream Processing System.
ACM Transactions on Database Systems, 33(1) :1–44, 2008.

[BE08] André Brinkmann and Sascha Effert.
Data Replication in P2P Environments.
In *Proceedings of the International Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 191–193, 2008.

[BF05] Piero A. Bonatti and P. Festa.
On Optimal Service Selection.
In *Proceedings of the International World Wide Web Conference (WWW)*, pages 530–538. ACM, May 2005.

[BFLZ03] Daniel S. Bernstein, Zhengzhu Feng, Brian N. Levin, and Shlomo Zilberstein.
In *Proceedings of the International Workshop on Peer-to-Peer Systems (IPTPS)*, pages 237–246. Springer, February 2003.

[BGJ06] Vandy Berten, Joël Goossens, and Emmanuel Jeannot.
A Probabilistic Approach for Fault Tolerant Multiprocessor Real-Time Scheduling.
In *Proceedings of the International Conference on Parallel and Distributed Processing Symposium (IPDPS)*, 2006.

[BJB+00] Judy Beiriger, Wilbur Johnson, Hugh Bivens, Steven Humphreys, and Ronald Rhea.
Constructing the ASCI Computational Grid.
In *Proceedings of the International Symposium on High Performance Distributed Computing (HPDC)*, pages 193–200, 2000.

[BKK+01] Reinhard Braumandl, Markus Keidl, Alfons Kemper, Donald Kossmann, Alexander Kreutz, Stefan Pröls, Stefan Seltzsam, and Konrad Stocker.
Objectglobe : Ubiquitous Query Processing on the Internet.

The Very Large Data Bases Journal, 10(1) :48–71, 2001.

[Bla85] Mark Blaug.
Economic Theory in Retrospect.
Cambridge University Press, 1985.

[BMST93] Navin Budhiraja, Keith Marzullo, Fred Schneider, and Sam Toueg.
chapter The Primary-Backup Approach, pages 199–216.
ACM Press, 2nd edition, 1993.

[BSV03] R. Bhagwan, S. Savage, and G. M. Voelker.
Understanding Availability.
In *Proceedings of the International Workshop on Peer-to-Peer Systems (IPTPS)*, pages 256–267, February 2003.

[BT98] Philippe Bonnet and Anthony Tomasic.
Partial Answers for Unavailable Data Sources.
In *Proceedings of the International Conference on Flexible Query Answering Systems (FQAS)*, pages 43–54, 1998.

[BW97] Fran Berman and Rich Wolski.
The AppleS Project : A Status Report.
In *Proceeding of the International Symposium on NEC Research*, May 1997.

[BW03] Wolf-Tilo Balke and Matthias Wagner.
Towards Personalized Selection of Web Services.
In *Proceedings of the International World Wide Web Conference (WWW)*. ACM, May 2003.
–Alternate Paper Tracks–.

[CCMW] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana.
Web Services Description Language (wsdl) version 1.1, http ://www.w3.org/tr/wsdl.

[CFK+98] Karl Czajkowski, Ian Foster, Nicholas Karonis, Carl Kesselman, Stuart Martin, Warren Smith, and Steven Tuecke.
A Resource Management Architecture for Metacomputing Systems.
In *Proceedings of the International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP)*, pages 62–82, 1998.

[CKKG] Steve Chapin, Dimitrios Katramatos, John Karpovich, and Andrew Grimshaw.
The Legion Resource Management System.

[CL85] Mani Chandy and Leslie Lamport.
Distributed Snapshots : Determining Global States of Distributed Systems.
ACM Transactions on Computer Systems, 3(1) :63–75, 1985.

[CMH+02] Ian Clarke, Scott Miller, Theodore Hong, Oskar Sandberg, and Brandon Wiley.
Protecting Free Expression Online with Freenet.
IEEE Internet Computing, 6(1) :40–49, 2002.

[COBW00] Henri Casanova, Graziano Obertelli, Francine Berman, and Richard Wolski.
The Apples Parameter Sweep Template : User-Level Middleware for the Grid.
In *Proceedings of the International Conference on supercomputing (SC)*, 2000.

[Cor03] IBM Corporation.
IBM Websphere V5.0 : Performance, Scalability, and High Availability : Websphere Handbook Series.
IBM Redbook, July 2003.

[DD04] Julian Day and Ralph Deters.
Selecting the best web service.
In *Proceedings of the International Conference of the Centre for Adavanced Studies on Collaborative Research (CASCON)*, pages 1193–307. IBM Press, October 2004.

[des] The SZTAKI Project, http ://desktopgrid.hu.

[DGB+96] Renzo Davoli, Luigi-Alberto Giachini, Özalp Babaoglu, Alessandro Amoroso, and Lorenzo Alvisi.
Parallel Computing in Networks of Workstations with Paralex.
IEEE Transactions on Parallel Distributed Systems, 7(4) :371–384, 1996.

[dis] distributed.net project. http ://www.distributed.net/.

[DMW03] Peter Dodds, Roby Muhamad, and Duncan Watts.
An Experimental Study of Search in Global Social Networks.
Science, 301(5634) :827–829, August 2003.

[DS83] Randall Davis and Reid Smith.
Negotiation as a Metaphor for Distributed Problem Solving.
Artificial Intelligence, 20(1) :63–100, 1983.

[DSW97] Keith Decker, Katia P. Sycara, and Mike Williamson.
Middle-Agents for the Internet.
In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 578–583. Morgan Kaufmann, August 1997.

[DVR+07] Rajdeep Dash, Perukrishnen Vytelingum, Alex Rogers, Esther David, and Nicholas Jennings.
Market-Based Task Allocation Mechanisms for Limited Capacity Suppliers.
IEEE Transactions on Systems, 37(3) :391–405, 2007.

[DW01] John Douceur and Roger Wattenhofer.
Competitive Hill-Climbing Strategies for Replica Placement in a Distributed File System.
In *Proceedings of the International Conference on Distributed Computing*, pages 48–62, 2001.

[EAWJ02] E. N. (Mootaz) Elnozahy, Lorenzo Alvisi, Yi-Min Wang, and David Johnson.
A Survey of Rollback-Recovery Protocols in Message-Passing Systems.
ACM Computing Surveys, 34(3) :375–408, 2002.

[eba] The eBay System, http ://business.ebay.com.

[ein] The Einstein@home Project, http ://einstein.phys.uwm.edu.

[Fe99] Ian Foster and Carl Kesselman (editors).
The Grid : Blueprint for a New Computing Infrastructure.
Morgan Kaufmann, 1999.

[FFS98] Trevor Fong, Danielle Fowler, and Paula Swatman.
Success and Failure Factors for Implementing Effective Electronic Markets.
International Journal of Electronic Markets, 8(1) :45–47, 1998.

[FK97] Ian Foster and C. Kesselman.
Globus : A Metacomputing Infrastructure Toolkit.
Journal of Supercomputer Applications and High Performance Computing, 11(2) :115–128, Summer 1997.

[FLSG06] Antonio Fernandez, Luis Lopez, Agustin Santos, and Chryssis Georgiou.

Reliably Executing Tasks in the Presence of Untrusted Entities.
In *Proceedings of the International Symposium on Reliable Distributed Systems (SRDS)*, pages 39–50, 2006.

[FNSY96] Donald Ferguson, Christos Nikolaou, Jakka Sairamesh, and Yechiam Yemini.
Market-Based Control : A Paradigm for Distributed Resource Allocation, chapter Economic Models for Allocating Resources in Computer Systems, pages 156–183.
World Scientific, 1996.

[Fos01] Ian Foster.
The Anatomy of the Grid : Enabling Scalable Virtual Organizations.
In *Proceedings of the European Conference on Parallel Computing (Euro-Par)*, pages 1–4, 2001.

[Fox88] Mark Fox.
Distributed Artificial Intelligence, chapter An Organizational View of Distributed Systems, pages 140–150.
Morgan Kaufmann, 1988.

[fre] Freightquote.com, http ://www.freightquote.com.

[FYN88] Donald Ferguson, Yechiam Yemini, and Christos Nikolaou.
Microeconomic Algorithms for Load Balancing in Distributed computer systems.
In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, pages 491–499, June 1988.

[GBGM04] Prasanna Ganesan, Mayank Bawa, and Hector Garcia-Molina.
Online Balancing of Range-Partitioned Data with Applications to Peer-to-Peer Systems.
In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 444–455, September 2004.

[GKD97] Michael R. Genesereth, Arthur M. Keller, and Oliver M. Duschka.
Infomaster : An Information Integration System.
In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 539–542. ACM, May 1997.

[GKS03] Alain Girault, Hamoudi Kalla, and Yves Sorel.
An Active Replication Scheme that Tolerates Failures in Distributed Embedded Real-Time Systems.
In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, pages 159–168, June 2003.

[GMPQ+97] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. D. Ullman, V. Vassalos, and J. Widom.
The TSIMMIS Approach to Mediation : Data Models and Languages.
Journal of Intelligent Information Systems, 8(2) :117–132, 1997.

[GN04] Alexander Gorobets and Bart Nooteboom.
Agent Based Computational Model of Trust.
Technical report, Erasmus Research Institute of Management (ERIM), RSM Erasmus University, 2004.

[goo] Google adwords, http ://adwords.google.com.

[gri] Grid4all project, http ://www.grid4all.eu/.

[GS04] Jun Gao and Peter Steenkiste.

An Adaptive Protocol for Efficient Support of Range Queries in DHT-Based Systems.
In *Proceedings of the International Conference on Networks Protocols (ICNP)*, pages 239–250, 2004.

[HBR+05a] Jeong-Hyon Hwang, Magdalena Balazinska, Alexander Rasin, Ugur Çetintemel, Michael Stonebreaker, and Stan Zdonik.
High-Availability Algorithms for Distributed Stream Processing.
In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 779–790. IEEE, April 2005.

[HBR+05b] Jeong-Hyon Hwang, Magdalena Balazinska, Alexander Rasin, Ugur Cetintemel, Michael Stonebraker, and Stan Zdonik.
High-Availability Algorithms for Distributed Stream Processing.
In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 779–790. IEEE, 2005.

[HHL+03] Ryan Huebsch, Joseph Hellerstein, Nick Lanham, Boon-Thau Loo, Scott Shenker, and Ion Stoica.
Querying the Internet with PIER.
In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 321–332, 2003.

[HLY06] Ken Hui, John Lui, and David Yau.
Small-World Overlay P2P Networks : Construction, Management and Handling of Dynamic Flash Crowds.
Computer and Telecommunications Networkwing, 50(15) :2727–2746, 2006.

[HTK02] Koji Hashimoto, Tatsuhiro Tsuchiya, and Tohru Kikuno.
Effective Scheduling of Duplicated Tasks for Fault-Tolerance in Multiprocessor Systems.
IEICE Transactions on Information and Systems, E85-D(3) :525–534, 2002.

[Hwa93] K Hwang.
Advanced Computer Architecture.
Mc-Graw-Hill Series in Computer Science, 1993.

[HXcZ07] Jeong-Hyon Hwang, Ying Xing, Ugur Çetintemel, and Stanley B. Zdonik.
A Cooperative, Self-Configuring High-Availability Solution for Stream Processing.
In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 176–185, April 2007.

[JBH+05] Flavio Junqueira, Ranjita Bhagwan, Alejandro Hevia, Keith Marzullo, and Geoffrey Voelker.
Surviving Internet Catastrophes.
In *Proceedings of the International Annual Technical Conference on USENIX (ATEC)*, pages 45–60, 2005.

[JCH84] Raj Jain, Dah-Ming Chiu, and W. Hawe.
A Quantitive Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems.
Technical report, DEC-TR-301, Digital Equipment Corporation, 1984.

[JFB07] Radu Jurca, Boi Falting, and Walter Binder.
Reliable QoS Monitoring Based on Client Feedback.
In *Proceedings of the International World Wide Web Conference (WWW)*, pages 1003–1012. ACM, May 2007.

[JGN99] William Johnston, Dennis Gannon, and Bill Nitzberg.
Grids as Production Computing Environments : The Engineering Aspects of NASA's Information Power Grid.
In *Proceedings of the International Symposium on High Performance Distributed Computing (HPDC)*, 1999.

[JQL06] Yingwei Jin, Wenyu Qu, and Keqiu Li.
A Survey of Cache/Proxy for Transparent Data Replication.
In *Proceedings of the International Conference on Semantics, Knowledge, and Grid (SKG)*, page 35, 2006.

[jxt] Jxta, http ://www.jxta.org.

[JZ90] David B. Johnson and Willy Zwaenepoel.
Recovery in Distributed Systems Using Optimistic Message Logging and Checkpointing.
Algorithms, 11(3) :462–491, 1990.

[KAGM96] Mohan Kamath, Gustavo Alonso, Roger Guenthor, and C. Mohan.
Providing High Availability in Very Large Workflow Management Systems.
In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, pages 425–442, March 1996.

[KGZY02] Vana Kalogeraki, Dimitrios Gunopulos, and D. Zeinalipour-Yazti.
A Local Search Mechanism for Peer-to-Peer Networks.
In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 300–307, 2002.

[KH95] Daniel Kuokka and Larry Harada.
Matchmaking for Information Agents.
In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 672–678, August 1995.

[KK92] Orly Kremien and Jeff Kramer.
Methodical analysis of adaptive load sharing algorithms.
IEEE Transactions on Parallel and Distributed Systems, 3(6) :747–760, 1992.

[Kle00] Jon M. Kleinberg.
The Small-World Phenomenon : An Algorithm Pperspective.
In *Proceedings of the International Symposium on Theory of Computing (STOC)*, pages 163–170, 2000.

[KLL97] Jong Kim, Heejo Lee, and Sunggu Lee.
Replicated Process Allocation for Load Distribution in Fault-Tolerant Multicomputers.
IEEE Transactions on Computers, 46(4) :499–505, 1997.

[Kos00] Donald Kossmann.
The State of the Art in Distributed Query Processing.
ACM Computing Surveys, 32(4) :422–469, 2000.

[Kre90] David Kreps.
A Course in Microeconomic Theory.
Princeton University Press, February 1990.

[KS01] Matthias Klusch and Katia Sycara.
Coordination of Internet agents : models, technologies, and applications, chapter Brokering and Matchmaking for Coordination of Agent Societies : A Survey, pages 197–224.
Springer, 2001.

[KSGM03] Sepandar Kamvar, Mario Schlosser, and Hector Garcia-Molina.
The Eigentrust Algorithm for Reputation Management in P2P Networks.
In *Proceedings of the International World Wide Web Conference (WWW)*, pages 640–651, 2003.

[LASG07] Steffen Lamparter, Anupriya Ankolekar, Rudi Studer, and Stephan Grimm.
Preference-based Selection of Highly Configurable Web Services.
In *Proceedings of the International World Wide Web Conference (WWW)*, pages 1013–1022. ACM, May 2007.

[LCLV07] Philippe Lamarre, Sylvie Cazalens, Sandra Lemp, and Patrick Valduriez.
A Flexible Mediaton Process for Large Distributed Information Systems.
International Journal of Cooperative Information Systems, 16(2) :299–332, June 2007.

[LH04] Lei Li and Ian Horrocks.
A Software Framework for Matchmaking Based on Semantic Web Technology.
International Journal of Electronic Commerce, 8(4) :39–60, Summer 2004.

[LKH+95] Heejo Lee, Jong Kim, SungJe Hong, ByungHo Yae, and HaeSook Kim.
Fault-Tolerant Process Allocation with Load Balancing.
In *Proceedings of the International Symposium on Fault-Tolerant Computing (FTCS)*, pages 124–129, December 1995.

[LML01] Yibei Ling, Jie Mi, and Xiaola Lin.
A Variational Calculus Approach to Optimal Checkpoint Placement.
IEEE Transactions on Computers, 50(7) :699–708, 2001.

[LNZ04] Yutu Liu, Anne Ngu, and Liangzhao Zeng.
QoS Computation and Policing in Dynamic Web Service Selection.
In *Proceedings of the International World Wide Web Conference (WWW)*, pages 66–73. ACM, May 2004.

[LQRV07] Philippe Lamarre, Jorge-Arnulfo Quiané-Ruiz, and Patrick Valduriez.
Libra : Une Méthode de Médiation Auto-Adaptative en Fonction des Attentes des Participants.
In *Proceedings of the Journées Francophones sur les Systèmes Multi-Agents (JFSMA)*, pages 65–74, October 2007.

[LRO96] Alon Levy, Anand Rajaraman, and Joann Ordille.
Querying Heterogeneous Information Sources Using Source Descriptions.
In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 251–262, September 1996.

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease.
The Byzantine Generals Problem.
ACM Transactions on Programming Language Systems, 4(3) :382–401, 1982.

[Mac04] John Maclonis.
Sociology.
Prentice Hall, 10th edition, February 2004.

[Mar02] Evangelos Markatos.
Tracing a Large-Scale Peer to Peer System : An Hour in the Life of Gnutella.
In *Proceedings of the IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid)*, pages 65–74, May 2002.

[MARW04] Shalil Majithia, Ali Shaikh Ali, Omer Rana, and David Walker.
Reputation-based Semantic Service Discovery.
In *Proceedings of the International Workshops on Enabling Technologies ; Infrastructure for Collaborative Enterprises (WETICE)*, pages 297–302. IEEE Computer Society, June 2004.

[MCWG95] Andreu Mas-Colell, Michael Whinstonand, and Jerry Green.
Microeconomic Theory.
Oxford University Press, June 1995.

[mdp] Mdp2p project, http ://www.sciences.univ-nantes.fr/lina/atlas/mdp2p/.

[Mil67] Stanley Milgram.
The Small World Problem.
Psicology Today, 1(69) :60–67, 1967.

[Mil02] Renée miller, editor.
IEEE Data Engineering Bulletin : Special Issue on Integration Management, 25(3), September 2002.

[Mit01] Michael Mitzenmacher.
The Power of Two Choices in Randomized Load Balancing.
IEEE Transaction on Parallel and Distributed Systems, 12(10) :1094–1104, 2001.

[MP05] Umardand Manikrao and T. V. Prabhakar.
Dynamic Selection of Web Services with Recommendation System.
In *Proceedings of the International Conference on Next Generation Web Services Practices (NWESP)*, pages 117–121. IEEE Computer Society, August 2005.

[MPV06] Vidal Martins, Esther Pacitti, and Patrick Valduriez.
A Survey of Data Replication in P2P Systems.
Technical report, inria-00122282, INRIA, 2006.

[MS04a] Michael Maximilien and Munindar Singh.
A Framework and Ontology for Dynamic Web Services Selection.
IEEE Internet Computing, 8(5) :84–93, 2004.

[MS04b] Michael Maximilien and Munindar Singh.
Toward Autonomic Web Services Trust and Selection.
In *Proceedings of the International Conference on Service Oriented Computing (ICSOC)*, pages 212–221. ACM, November 2004.

[MSZ01] Sheila McIlraith, Tran Cao Son, and Honglei Zeng.
Semantic Web Services.
IEEE Intelligent Systems, 16(2) :46–53, 2001.

[MTS90] Ravi Mirchandaney, Don Towsley, and John Stankovic.
Adaptive Load Sharing in Heterogeneous Distributed Systems.
Journal of Parallel and Distributed Computing, 9(4) :331–346, August 1990.

[nap] Napster, http ://www.napster.com.

[Nas51] John Nash.
Non-cooperative games.
The Annals of Mathelatics, 54(2) :286–295, 1951.

[NBN99] Marian H. Nodine, William Bohrer, and Anne H. Ngu.
Semantic Brokering over Dynamic Heterogeneous Data Sources in InfoSleuth(tm).

In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 358–365. IEEE Computer Society, March 1999.

[NDMR08] Dushyanth Narayanan, Austin Donnelly, Richard Mortier, and Antony Rowstron.
Delay Aaware Querying with Seaweed.
The Very Large Data Bases Journal, 17(2) :315–331, 2008.

[NF92] A. Nanjia and David Finkel.
Transaction-Based Fault-Tolerant Computing in Distributed Systems.
In *Proceedings of the International Workshop on Fault-Tolerant Parallel and Distributed Systems*, pages 92–97, July 1992.

[NFK+00] Marian H. Nodine, Jerry Fowler, Tomasz Ksiezyk, Brad Perry, Malcolm Taylor, and Amy Unruh.
Active Information Gathering in Infosleuth.
International Journal of Cooperative Information Systems, 9(1-2) :3–28, March-June 2000.

[Nie90] Lambert Nieuwenhuis.
Static Allocation of Process Replicas in Fault-Tolerant Computing Systems.
In *Proceedings of the International Symposium on Fault-Tolerant Computing (FTCS)*, pages 298–306, June 1990.

[NWQ+02] Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjörn Naeve, Mikael Nilsson, Matthias Palmér, and Tore Risch.
EDUTELLA : a P2P Networking Infrastructure Based on RDF.
In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 604–615, 2002.

[ONK+03] Chihiro Ono, Satoshi Nishiyama, Keesoo Kim, Boyd C. Paulson, Mark Cutkosky, and Cutkosky J. Petrie.
Trust-Based Facilitator : Handling Word-of-Mouth Trust for Agent-Based E-Commerce.
Electronic Commerce Research, 3(3-4) :201–220, July-October 2003.

[OS92] Yingfend Oh and Sang Son.
An Algortihm for Real-Time Fault-Tolerant Scheduling in Multiprocessor Systems.
In *Proceedings of the EuroMicro Workshop on Real-Time Systems (ECRTS)*, pages 190–195, June 1992.

[ÖV99] Tamer Özsu and Patrick Valduriez.
Principles of Distributed Database Systems.
Prentice-Hall, 2nd edition, 1999.

[PGVA08] Prasanna Padmanabhan, Le Gruenwald, Anita Vallur, and Mohammed Atiquzzaman.
A Survey of Data Replication Techniques for Mobile Ad Hoc Network Databases.
The Very Large Data Bases Journal, 17(5) :1143–1164, 2008.

[PI06] F. Pentaris and Y. Ioannidis.
Query Optimization in Distributed Networks of Autonomous Database Systems.
ACM Transactions on Database Systems (TODS), 31(2) :537–583, 2006.

[PI07] Fragkiskos Pentaris and Yannis Ioannidis.
Autonomic Query Allocation Based on Microeconomics Principles.
In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 266–275. IEEE, April 2007.

[PKPS02] Massimo Paolucci, Takahiro Kawamura, Terry Payne, and Katia Sycara.
Semantic matching of web services.
In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 333–347, June 2002.

[PPS07] Sean Pieper, JoAnn Paul, and Michael Schulte.
A New Era of Performance Evaluation.
IEEE Computer, 40(9) :23–30, September 2007.

[pro] The proteins@home Project, http ://biology.polytechnique.fr/proteinsathome.

[PSK03] Massimo Paolucci, Katia Sycara, and Takahiro Kawamura.
Delivering Semantic Web Services.
In *Proceedings of the International World Wide Web Conference (WWW)*. ACM, May 2003.
–Alternate Paper Tracks–.

[QL07] Huiming Qu and Alexandros Labrinidis.
Preference-Aware Query and Update Scheduling in Web-Databases.
In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 356–365, 2007.

[QLM06] Huiming Qu, Alexandros Labrinidis, and Daniel Mosse.
Unit : User-centric transaction management in web-database systems.
In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 1–10, April 2006.

[QRLCV07a] Jorge-Arnulfo Quiané-Ruiz, Philippe Lamarre, Sylvie Cazalens, and Patrick Valduriez.
Satisfaction balanced mediation.
In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 947–950, November 2007.

[QRLCV07b] Jorge-Arnulfo Quiané-Ruiz, Philippe Lamarre, Sylvie Cazalens, and Patrick Valduriez.
A Satisfaction Balanced Query Allocation Process for Distributed Information Systems.
In *Proceedings of the Journées Francophones sur les Bases de Données Avancées (BDA)*, October 2007.

[QRLCV08] Jorge-Arnulfo Quiané-Ruiz, Philippe Lamarre, Sylvie Cazalens, and Patrick Valduriez.
Managing Virtual Money for Satisfaction and Scale Up in P2P Systems.
In *Proceedings of the International EDBT Workshop on Data Management in Peer-to-Peer Systems (DAMAP)*, March 2008.

[QRLV06] Jorge-Arnulfo Quiané-Ruiz, Philippe Lamarre, and Patrick Valduriez.
Satisfaction-based Query Load Balancing.
In *Proceedings of the International Conference on Cooperative Information Systems (CoopIS)*, pages 36–53, November 2006.

[QRLV07a] Jorge-Arnulfo Quiané-Ruiz, Philippe Lamarre, and Patrick Valduriez.
KnBest - A Balanced Request Allocation Method for Distributed Information Systems.
In *Proceedings of the International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 237–248, April 2007.

[QRLV07b] Jorge-Arnulfo Quiané-Ruiz, Philippe Lamarre, and Patrick Valduriez.
SQLB : A Query Allocation Framework for Autonomous Consumers and Providers.
In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 974–985, September 2007.

[QRLV07c] Jorge-Arnulfo Quiané-Ruiz, Philippe Lamarre, and Patrick Valduriez.
Un Modèle pour Characteriser des Participants Autonommes dans un Processus de Médiation.
In *Proceedings of the Journées Francophones sur les Modèles Formels de l'Interaction (MFI)*, volume 8, pages 389–396. Lamsade, May 2007.

[QRLV08] Jorge-Arnulfo Quiané-Ruiz, Philippe Lamarre, and Patrick Valduriez.
A Self-Adaptable Query Allocation Process.
In *Proceedings of the Journées Francophones sur les Bases de Données Avancées (BDA)*, October 2008.

[Ran75] Brian Randell.
System Structure for Software Fault Tolerance.
pages 437–449, 1975.

[Ran03] Shuping Ran.
A Model for Web Services Discovery with QoS.
SIGecom Exch., 4(1) :1–10, 2003.

[RB03] Mema Roussopoulos and Mary Baker.
Cup : Controlled update propagation in peer to peer networks.
In *Proceedings of the International USENIX Annual Technical Conference*, pages 167–1680, June 2003.

[RB06] Mema Roussopoulos and Mary Baker.
Practical load balancing for content requests in peer-to-peer networks.
Distributed Computing, 18(6) :421–434, 2006.

[RD01] Antony Rowstron and Peter Druschel.
Pastry : Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems.
In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms*, pages 329–350, 2001.

[res] Respire project, http ://respire.lip6.fr.

[RFH+01] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker.
A Scalable Content-Addressable Network.
In *SIGCOMM '01 : Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, 2001.

[RM95] Erhard Rahm and Robert Marek.
Dynamic Multi-Resource Load Balancing in Parallel Database Systems.
In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 395–406, September 1995.

[RS97] Mary Roth and Peter Schwarz.
Don't Scrap It ! Wrap It ! A Wrapper Architecture for Legacy Data Sources.
In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 266–275, August 1997.

[Rus80] David Russell.
State Restoration in Systems of Communicating Processes.
IEEE Transactions on Software Engineering, 6(2) :183–194, 1980.

[SAL+96] Michael Stonebraker, Paul Aoki, Witold Litwin, Avi Pfeffer, Adam Sah, Jeff Sidell, Carl Staelin, and Andrew Yu.

Mariposa : A Wide-Area Distributed Database System.
The International Journal on Very Large Data Bases, 5(1) :48–63, 1996.

[San93] Thuomas Sandholm.
An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations.
In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 256–262. The AAAI Press/The MIT Press, July 1993.

[San99] Tuomas Sandholm.
Multiagent Systems, a modern approach to Distributed Artificial Intelligence, chapter Distributed Rational Decision Making, pages 201–258.
The MIT Press, 1999.

[SB89] Neil Speirs and Peter Barret.
UsingPassive Replicates in Demlta-4 to Provide Dependable Distributed Computing.
In *Proceedings of the International Symposium on Fault-Tolerant Computing (FTCS)*, pages 184–190, June 1989.

[SBD94] Adam Sah, Jon Blow, and Brian Dennis.
An Introduction to the Rush Language.
In *Proceedings of the International Workshop on TCL*, June 1994.

[SBY88] Robert Strom, D. Bacon, and Shaula Yemini.
Volatile Logging in Fautl-Tolerant Distributed Systems.
In *Proceedings of the International Symposium on Fault-Tolerant Computing (FTCS)*, pages 44–49, 1988.

[Sch93] Fred Schneider.
Distributed Systems, chapter Replication Management Using the State-Machine Approach, pages 169–197.
ACM Press, 2nd edition, 1993.

[set] Seti@home project, http ://setiathome.berkeley.edu/.

[SGG02] Stefan Saroiu, Krishna Gummadi, and Steven Gribble.
A Measurement Study of Peer-to-Peer File Sharing Systems.
In *Proceedings of the International Conference on Multimedia Computing and Networking (MMCN)*, January 2002.

[SGG03] S. Saroiu, P. Krishna Gummadi, and S. D. Gribble.
Measuring and Analyzing the Characteristics of Napters and Gnutella Hosts.
Multimedia Systems, 9(2) :170–184, 2003.

[SH03] Rüdiger Schollmeier and Felix Hermann.
Topology-Analysis of Pure Peer-to-Peer Networks.
In *Proceedings of the Kommunikation in Verteilten Systemen (KiVS)*, pages 359–370, 2003.

[SK03] Rüdiger Schollmeier and Gerald Kunzmann.
GnuViz - Mapping the Gnutella Networks to its Geographical Locations.
Praxis der Informationsverarbeitung und Kommunication (PIK), 26(2) :74–79, 2003.

[SKS92] Niranjan Shivaratri, Phillip Krueger, and Mukesh Singhal.
Load Distributing for Locally Distributed Systems.
IEEE Computer, 25(12) :33–44, December 1992.

[SKWL99] Katia P. Sycara, Matthias Klusch, Seth Widoff, and Jianguo Lu.

Dynamic Service Matchmaking Among Agents in Open Information Environments.
SIGMOD Record, 28(1) :47–53, 1999.

[SL95] Thuomas Sandholm and Victor Lesser.
Issues in Automated Negotiation and Electronic Commerce : Extending the Contract Net Framework.
In *Proceedings of the International Conference on Multi-Agent systems (ICMAS)*, pages 328–335. The MIT Press, June 1995.

[SM94] Reinhard Schwarz and Friedemann Mattern.
Detecting Causal Relationships in Distributed Computations : In Search of the Holy Grail.
Distributed Computing, 7(3) :149–174, 1994.

[Smi81] Reid Smith.
The Contract Net Protocol : High-Level Communication and Control in a Distributed Problem Solver.
IEEE Transactions on Computers, C-29(12) :1104–1113, 1981.

[SMLN+03] Ion Stoica, Robert Morris, David Liben-Nowell, David Karger, Frans Kaashoek, Frank Dabek, and Hari Balakrishnan.
Chord : A Scalable Peer-to-Peer Lookup Protocol for Internet Applications.
IEEE/ACM Transactions on Networking, 11(1) :17–32, 2003.

[SR02] A Singla and Ch Rohrs.
Ultrapeers : Another Step Towards Gnutella Scalability.
Technical report, Lime Wire, November 2002.

[SRN03] P. Souza, C. Ramos, and J. Neves.
The Fabricare Scheduling Prototype Suite : Agent Interaction and Knowledge Base.
Journal of Intelligent Manufacturing, 14(5) :441–455, October 2003.

[SS92] Daniel Siewiorek and R. S. Swarz.
Reliable System Design : The Theory and Practice.
New York : Digital Press, 1992.

[ST01] Yoav Shoham and Moshe Tennenholtz.
Fair Imposition.
In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1083–1088, August 2001.

[SWDC97] Rick Stevens, Paul Woodward, Thomas DeFanti, and Charles Catlett.
From the I-WAY to the National Technology Grid.
Communications of the ACM, 40(11) :50–60, 1997.

[SWG92] Sol Shatz, Jia-Ping Wang, and Masanori Goto.
Task Allocation for Maximizing Reliability of Distributed Computer Systems.
IEEE Transactions on Computers, 41(9) :1156–1168, 1992.

[SY85] Robert Strom and Shaula Yemini.
Optimistic Recovery in Distributed Systems.
ACM Transactions Computer Systems, 3(3) :204–226, 1985.

[Syc97] Katia Sycara.
Using Option Pricing to Value Commitment Flexibility in Multiagent Systems.
Technical report, CMU-CS-97-193, School of Computer Science, Carnegie Mellon University, 1997.

[Syc98] Katia P. Sycara.
Multiagent Systems.
AI Magazine, 19(2) :79–92, Summer 1998.

[TJ00] Phillip J. Turner and Nicholas Jennings.
Improving the Scalability of Multi-Agent Systems.
In *Proceedings of the International Workshop on Infrastructure for Multi-Agent Systems*, pages 246–262, 2000.

[TRV98] Anthony Tomasic, Louiqa Raschid, and Patrick Valduriez.
Scaling Access to Heterogeneous Data Sources with DISCO.
IEEE Transanctions on Knowledge and Data Engineering (TKDE), 10(5) :808–823, 1998.

[udd] The UDDI Technical White Paper, http ://www.uddi.org.

[VBW04] Srikumar Venugopal, Rajkumar Buyya, and Lyle Winton.
A Grid Service Broker for Scheduling Distributed Data-oriented Applications on Global Grids.
In *Proceedings of the International Workshop on Middleware for Grid Computing (MGC)*, pages 75–80, 2004.

[Vic61] Willinam Vickrey.
Counterspeculation, auctions, and competitive sealed tenders.
International Journal of Finance, 16(1) :8–37, March 1961.

[vNM44] John von Neumann and Oskar Morgenstern.
Theory of Games and Economic Behavior.
Princeton University Press, 1944.

[web] Web Services, http ://www.w3.org/2002/ws/.

[Wie92] Gio Wiederhold.
Mediators in the Architecture of Future Information Systems.
IEEE Computer, 25(3) :38–49, March 1992.

[WRC00] Marc Waldman, Aviel Rubin, and Lorrie Faith Cranor.
Publius : A Robust, Tamper-Evident, Censorship-Resistant, Web Publishing System.
In *Proceedings of the USENIX Conference on Security Symposium*, pages 59–72, August 2000.

[WS] Abel Weinrib and Scott Shenker.
Greed is not enough : adaptive load sharing in large heterogeneous systems.
In *Proceedings of the International Conference on Computer Communications (INFOCOM)*, pages 986–994.

[WS98] Duncan Watts and S Strogatz.
Collective Dynamics of 'Small-World' Networks.
Nature, 393(6684) :440–442, June 1998.

[wsm] Web Service Modeling Ontology, http ://www.w3.org/submission/wsmo/.

[YD04] Praveen Yalagandula and Michael Dahlin.
A Scalable Distributed Information Management System.
In *Proceedings of the International Conference on Applciations, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 379–390, 2004.

[YGM02] Beverly Yang and Hector Garcia-Molina.
Improving Search in Peer-to-Peer Networks.
In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, pages 5–14, 2002.

[YGM03] Beverly Yang and Hector Garcia-Molina.
Designing a Super-Peer Network.
In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 49–60, 2003.

[YV01] Haifeng Yu and Amin Vahdat.
The Costs and Limits of Availability for Replicated Services.
In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, pages 29–42, 2001.

[ZGG04] Hui Zhang, Ashish Goel, and Ramesh Govindan.
Using the Small-World Model to Improve Freenet Performance.
Computer and Telecommunications Networkwing, 46(4) :555–574, 2004.

[Zho88] Songnian Zhou.
A trace-driven simulation study of dynamic load balancing.
IEEE Transactions on Software Engineering, 14(9) :1327–1341, 1988.

[ZHS+04] Ben Zhao, Ling Huang, Jeremy Stribling, Sean Rhea, Anthony Joseph, and John Kubiatowicz.
Tapestry : A Resilient Global-Scale Overlay for Service Deployment.
IEEE Journal on Selected Areas in Communications, 22(1) :41–53, 2004.

[ZZ02] Zili Zhang and Chengqi Zhang.
An Improvement to Matchmaking Algorithms for Middle Agents.
In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1340–1347. ACM, July 2002.

A Satisfaction-Based Query Allocation Framework for Distributed Information Systems

Abstract

In large-scale distributed information systems, where participants (consumers and providers) are autonomous and have special interests for some queries, query allocation is a challenge. Much work in this context has focused on distributing queries among providers in a way that maximizes overall performance (typically throughput and response time). However, participants usually have certain expectations with respect to the mediator, which are not only performance-related. Such expectations mainly reflect their *interests* to allocate and perform queries, e.g. their interests towards: providers (based on reputation for example), quality of service, topics of interests, and relationships with other participants. In this context, because of participants' autonomy, *dissatisfaction* is a problem since it may lead participants to leave the mediator. Participant's *satisfaction* means that the query allocation method meets its expectations. Thus, besides balancing query load, preserving the participants' interests so that they are satisfied is also important. In this thesis, we address the query allocation problem in these environments and make the following main contributions. First, we provide a model to characterize the participants' perception of the system regarding their interests and propose measures to evaluate the quality of query allocation methods. Second, we propose a framework for query allocation, called S_bQA, that dynamically trades consumers' interests for providers' interests based on their satisfaction. Third, we propose a query allocation approach, called $\$_bQA$, that allows a query allocation method (specifically S_bQA) to scale up in terms of the numbers of mediators, participants, and hence of performed queries. Fourth, we propose a query replication method, called S_bQR, which allows supporting participants' failures when allocating queries while preserving participants' satisfaction and good system performance. Last, but not least, we analytically and experimentally validate our proposals and demonstrate that they yield high efficiency while satisfying participants.

Keywords: distributed information systems, query allocation, mediation, autonomous participants, participants' satisfaction, scale up, participants' failure

Allocation de Requêtes dans des Systèmes d'Information Distribués avec des Participants Autonomes

Jorge-Arnulfo QUIANÉ-RUIZ

Résumé

Nous nous intéressons aux systèmes d'informations où les participants (clients et fournisseurs) sont autonomes, c.à.d. ils peuvent décider de quitter le système à n'importe quel moment, et qu'ils ont des intérêts particuliers pour certaines requêtes. Dans ces environnements, l'allocation de requêtes est un défi particulier car les attentes des participants ne sont pas seulement liées aux performances du système. Dans ce contexte, l'insatisfaction des participants est un problème car elle peut les conduire à quitter le système. Par conséquent, il est très important de répondre aux attentes des participants de sorte à ce qu'ils soient satisfaits. Dans cette thèse, nous abordons ce problème en apportant quatre contributions principales. Primo, nous fournissons un modèle pour caractériser la perception des participants par rapport au système et proposons des mesures qui permettent d'évaluer la qualité des méthodes d'allocation de requêtes. Secundo, nous proposons une méthode d'allocation de requêtes, S_bQA, qui permet d'équilibrer à la volée les intérêts tant des clients que des fournisseurs en se basant sur leur satisfaction. Tertio, nous proposons $\$_bQA$: une version économique de S_bQA qui permet de passer à l'échelle en nombre de médiateurs, de participants, et par conséquent, de requêtes traitées. Quarto, nous proposons S_bQR : une méthode de réplication de requêtes qui permet de supporter les pannes éventuelles des participants, tout en préservant leur satisfaction.

Mots-clés : systèmes d'information, allocation de requêtes, médiation, participants autonomes, satisfaction des participants, passage à l'échelle, panne des participants

ACM Classification

Categories and Subject Descriptors : H.2.4 [**Database Management**]: Systems—*Distributed databases, Query processing*; H.4.0 [**Information Systems Applications**]: General.

General Terms : Design, Information Systems, Management, Performance, Reliability.

Discipline : Informatique
Spcialit : Bases de Données

Laboratoire : LABORATOIRE D'INFORMATIQUE DE NANTES ATLANTIQUE.
UMR CNRS 6241. 2, rue de la Houssinire, BP 92 208 – 44 322 Nantes, CEDEX 3.