A Satisfaction Balanced Query Allocation Process for Distributed Information Systems

Jorge-Arnulfo Quiane-Ruiz, Philippe Lamarre, Sylvie Cazalens, Patrick Valduriez

A Satisfaction Balanced Query Allocation Process for Distributed Information Systems*

Jorge-Arnulfo Quiané-Ruiz⋆† Philippe Lamarre⋆ Sylvie Cazalens⋆
Patrick Valduriez‡

Atlas group, INRIA and LINA, Université de Nantes
2, rue de la Houssinière, 44322 Nantes - FRANCE
E-mail: `Firstname.Lastname@{univ-nantes.fr⋆, inria.fr‡}`

Abstract

We consider a distributed information system that allows autonomous consumers to query autonomous providers. We focus on the problem of query allocation from a new point of view, by considering consumers and providers' satisfaction in addition to query load. We define satisfaction as a long-run notion based on the consumers and providers' intentions. Intuitively, a participant should obtain good satisfaction as far as it (the participant) is adequate to the system. We propose and validate a mediation process, called SBMediation, which is compared to Capacity based query allocation. The experimental results show that SBMediation significantly outperforms Capacity based when confronted to autonomous participants.

Keywords

Autonomous participants, satisfaction, adequation, query allocation, imposition, payment

1. Introduction

We consider a distributed information system with a mediator that enables consumers to access distributed information providers through queries [10]. Participants[1] are autonomous in the sense that they are free to enter and leave the system at will and do not depend on anyone to do so. Then, the main function of the mediator is to allocate each incoming query to the providers that can answer it. Much work in this context has focused on *query load balancing (QLB)* [8, 15], i.e. distributing the query load among the providers in a way that maximizes overall performance (typically throughput and response time). This is obviously important for the efficiency of the system. However, participants may have certain expectations, w.r.t. the mediator, which are not only performance-related.

Providers' expectations reflect their *preferences* in performing some queries rather than others. For example, a provider p_c could represent a pharmaceutical company, which wants to promote a new insect repellent. Thus, it is more interested in treating the queries related to mosquitoes or insect bites than general queries. Once the advertising campaign is over, the provider's *preferences* may change. Consumers expect the mediator to

* Work partially funded by ARA "Massive Data" of the French ministry of research (Respire project) and the European Strep Grid4All project.

† This author is supported by the Mexican National Council for Science and Technology (CONACyT).

[1] We refer, throughout this paper, to both consumers and providers together as participants

provide them with information, which best fits their *preferences*. However, *preferences* are usually considered as private data by participants, since revealing them means revealing strategies. Thus, participants express their *preferences* via an *intention* notion, which can combine different criteria such as *preferences* and *load*. For instance, a provider may not intend to perform queries (even if it prefers them) because of local reasons, e.g. by *overload*. Since it is autonomous, a participant that is dissatisfied too long may just leave the mediator. In our context, this is equivalent to depart from the system, but it could be that the provider registers to another mediator. Therefore, it is quite important to satisfy participants while allocating queries in order to avoid having participants leave the system by dissatisfaction. Intuitively, the system satisfies the participants if the mediator meets their expectations.

In this context, query allocation is a challenge for several reasons. First, to our knowledge, there is no definition of *satisfaction* to characterize how well the system meets the participants' expectations in the long-run. Economical models consider utility and rationality [13], but this is not a long-run notion. Second, participants' expectations are usually contradictory. Third, the query demand should be satisfied even if sometimes consumers and providers do not desire to deal with providers and queries, respectively. Finally, 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. Our main contributions in this paper are the following.

- We define the notions of participants' *satisfaction* in a new model. Also, we define a notion of *system's adequation w.r.t. participants* in order to know whether queries correspond to participants' expectations. The proposed model eases the design of new mediation processes for query allocation that satisfy participants in allocating queries.

- We propose a query allocation mediation process, called $SBMediation$, with the objective of satisfying participants by finding not only relevant providers (i.e. interesting results) to consumers, but also finding interesting queries to providers.

- We implement $SBMediation$ and compare its behavior to a classic query allocation process, namely $Capacity\ based$ (e.g. [8]). We mainly study both processes from a *satisfaction* point of view, and analyze the impact on performance of the participants' autonomy.

The rest of the paper is organized as follows. After giving the motivation and focus of this paper at the remainder of this section, we give some preliminary concepts in Section 2. Section 3 presents the model that characterizes participants' expectations. In Section 4, we define a mediation process to allocate queries by considering the participants' *intentions* and providers' *satisfaction*, called $SBMediation$. We validate the proposed mediation in Section 5. In Section 6, we survey related work. Finally, Section 7 concludes.

1.1. Example Scenario

Consider a distributed information system gathering thousands of computer sites (such as schools, home users, students, and professors) with the aim of sharing information and computational resources, as in the Grid4All project [1]. Each site preserves its *preferences* for allocating and performing queries. For example, a family can offer its personal computational resources to be used by their children's college during class hours. However, even if they accept different class activities, they prefer those related to their children's class with a particular attention to the physics class project that needs a lot of computer resources to run simulations. Such contributors appreciate to know how their computer resources are used by who and to do what. It is up to the college to manage the different resources proposed by different persons with different *preferences* to run college's projects and make contributors globally satisfied with the use of their materials. Another example could be universities desiring to share their infor-

Table 1: Relevant providers to perform s_x's query.

Providers	Pro. Intention	Con. Intention
p_2	Yes	No
p_4	No	Yes
p_1	Yes	No
p_3	No	Yes
p_5	Yes	Yes

mation and resources preferably with some communities for particular purposes.

An example scenario in Grid4All may be a student s_x that desires to perform a specific application. She requests the system for 2 sites providing computational resources, e.g. CPU's time units, to deploy her distributed application. Since she needs some guidelines on how to present her report on this activity, she also requests the system for 3 answers from different sources to obtain some advice.

For both previous queries, the system must perform several tasks. First, it must find the sites that are able to deal with s_x's query (i.e. identify the relevant providers). There is a large body of work on matchmaking, see e.g. [7, 9] which we could simply reuse. Then, the system needs to select the required number of providers among relevant ones. If possible, it is best to select providers with least load and short answering time such that s_x's and providers' *preferences* are enforced. Let us assume that the mediator is able to obtain the load of the relevant providers. Assume, then, that p_2, p_4, p_1, p_3, and p_5 is the resulting list, ordered from the least to the most loaded (Table 1). Then, the mediator should obtain s_x's interests to deal with such providers and the providers' interests to deal with s_x's query. Consider that p_3 and p_4 do not desire to deal with s_x's query because, for instance, p_3 is already in competition with s_x (and does not want to help her) and p_4 is already loaded and prefers to treat other kinds of query. Similarly, consider that s_x does not desire to deal with p_1 and p_2 since her last experiences with them were not so good, and that p_5 is overloaded.

Given this context, there are several ways to allocate the query to the number of providers required by the consumer. One can allocate it from the providers' load point of view only, as QLB methods do. In this case, the participants' expectations may not be met. In the above scenario although p_2 and p_4 are the less loaded, allocating the query to these providers may dissatisfy them because neither s_x desires to allocate its query to p_2 nor p_4 intends to deal with s_x's query. Another way to allocate queries is to consider all the participants expectations and try to satisfy them as much as possible. However, as shown by the scenario, it may also happen that the allocation of some query does not satisfy some participants or even none of them. In our view, this is not a problem as long as *query allocation meets the participants' expectations in the long run*: they may be dissatisfied sometimes but still "globally" satisfied "in the long run".

1.2. Focus of the Paper

To our knowledge, no model enables the study of the *participants' long run satisfaction* in a system. In particular, QLB models fail to do so because they only consider load. Thus, the problem is not only to define query allocation mechanisms with the objective to ensure the participants satisfaction but also, especially, to define a new model that also considers the participants long run *satisfaction*. This model should enable to evaluate a given allocation process.

Hence, in this paper, the problem that we consider is twofold. As for the model definition, we focus on two main points. First, we want to investigate both notions of *satisfaction* and *adequation* of a participant w.r.t. the system. These notions are closely related, as for example, a provider of medical information does not fit in a system dealing with geology, and thus will obviously be dissatisfied by the queries it gets. Second, we aim at defining both local and global characterizations. Local characterization enables each participant to evaluate itself in the system and global characterization evaluates the query allocation process itself. As for a query allocation process, we focus on a mechanism in which the providers have to bid on queries, while the consumers directly show their *intentions* to the mediator. The problem here is not only to define the mechanism but also to propose a possible bidding strategy for the providers, as well as the flow of money (which is virtual) in the system. Last but not least, we focus on the validation of the proposed query allocation process through the proposed model.

2. Preliminary Concepts

The system consists of a mediator, m, of a set of consumers, C, and of a set of providers, P. These sets are not necessary disjoint, i.e. a participant may play different roles. Providers can be 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. In our context, data heterogeneity means that providers provide different data and thus produce different results for a same query.

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 query to be done, and $q.n \in \mathbb{N}^*$ is the number of providers to which the consumer wishes to allocate its query. Consumers send their queries to mediator m that allocates each incoming query q to $q.n$ providers. The set P_q denotes the set of providers that can treat the query q and are registered to a mediator m, which does not appear in the notation for simplicity. In the case where $q.n > ||P_q||$, the consumer $q.c$ gets $||P_q||$ results instead of $q.n$. In other words, if the number of providers that are able to deal with a query is smaller or equal than the number desired by the consumer, all these providers must perform the query.

The allocation of some query q is denoted by a vector $All\overrightarrow{oc}$ of length N, or $All\overrightarrow{oc}_q$ and N_q if there is an ambiguity on q, such that,

$$\forall p \in P_q,\ All\overrightarrow{oc}[p] = \left| \begin{array}{ll} 1 & \text{if } p \text{ gets the query} \\ 0 & \text{otherwise} \end{array} \right.$$

As we assume any incoming query should be treated if possible, this leads to $\sum_{p \in P_q} All\overrightarrow{oc}[p] = min(q.n,\, N_q)$. Notice that, without any loss of generality, in some cases (e.g. when consumers pay services with real money) the query allocation means that providers are selected for participating in a negotiation process with consumers.

Each provider $p \in P$ has a finite *capacity*, $cap_p > 0$, for performing queries. The *capacity* of a provider denotes the number of computational units that it can have. Similarly, each query q has a *cost*, $cost_p(q) > 0$, that represents the computational units that q consumes at p. Let Q_p denote the set of requests that have been allocated to p but have not already been treated at time t (i.e. the pending requests at p). We then say that the *utilization* of a provider $p \in P$ at time t, denoted by $U_p(t)$, is

$$U_p(t) = \frac{\displaystyle\sum_{q \in Q_p} cost_p(q)}{cap_p}$$

Participants express their *preferences* to allocate and perform queries via an *intention* value. The way in which participants compute their *intentions* is considered as private information (e.g. in an e-commerce scenario, enterprises do not reveal their business strategies). However, even if it is private, the way in which participants compute their *intentions* has a direct impact on the system's behavior. For instance, if participants are interested in system's performance, the end result is that the system has high performance.

3. Participants Characterization

We are interested in two characteristics of participants that show how they perceive the system in which they interact. The first one is *adequation.* From a general point of view, two kinds of adequation could be considered: *(i)* the system's adequation to a participant, e.g. a system where a provider cannot find any query it intends to perform is considered inadequate to such a provider (the *System-Provider Adequation*); and *(ii)* the participant's adequation to the system, e.g. a consumer issuing queries that no provider intends to treat is considered inadequate to the system (the *Consumer-System Adequation*). Through these notions, we can evaluate if it is possible for a participant to reach its goals in the system. A participant cannot know what 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 (see Section 3.3).

The second characteristic is *satisfaction*. As for *adequation*, two kinds of *satisfaction* could be considered: *(i)* the *satisfaction* of a participant with what it gets from the system, e.g. a consumer that receives results from the providers it wants to avoid is simply not satisfied (the *Consumer Satisfaction*); and *(ii)* the participant's *satisfaction* with the job that the query allocation method does for it, e.g. a provider that performs queries it does not want is not satisfied with the query allocation method if there exist queries of its interests that it does not get (the *Provider Allocation Satisfaction*). 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.

Besides the *adequation* and *satisfaction* of a participant, we are interested in two other global characteristics: *(i)* the query allocation efficiency w.r.t. a consumer and *(ii)* the query allocation efficiency w.r.t. a provider. These new global characteristics allow evaluating if the query allocation method really does a good job for a participant.

We assume that participants have a limited memory capacity and regularly assess only their k last interactions with the system. Notice that the k value may be different for each participant depending on its memory capacity, but also on its strategy. For simplicity, we assume that they all use the same value of k. Thus, we define the characteristics of the participants over their k last interactions.

Before going further, 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 is completely symmetrical for the participants' dynamic data (*intentions*) as well as for their static data (e.g. the *preferences*). However, applying the following characterization to *intentions* and *preferences* yields to different results, because *intentions* of participants consider their context (such as their strategy and *utilization*) and their *preferences* do not. While in almost all distributed information systems *preferences* tend to be private information, *intentions* tend to be public. For simplicity, we develop the following characteristics only for *preferences*, whose values are in $[-1..1]$.

3.1. Local Consumer Characterization

We characterize a consumer according to the information that it can obtain from the system. Intuitively, the consumer's characteristics are useful to answer the following questions: "How well the consumer's expectations correspond to the providers that were able to deal with its last queries?" – *System-Consumer Adequation* – ; "How far the providers that have dealt with the last queries of a consumer meet its expectations?" – *Consumer Satisfaction* – ; and "Is a consumer satisfied with the allocation process?" – *Consumer Allocation Satisfaction* –. All these notions are based on the memory of a consumer, which is denoted by IQ_c^k. The *preference* of a consumer c to allocate its query q to a provider p is given by function $Prf_c^q(p)$.

3.1.1. Adequation

The system's adequation to a consumer characterizes the perception that a consumer has from the system. In our example scenario of Section 1.1, we can say that the system is quite adequate to s_x since it contains providers that s_x considers interesting. Formally, the system's adequation w.r.t. a consumer $c \in C$ concerning a query q, denoted by $\delta_{sca}(c, q)$, is defined as the average of c's *preferences* towards the set P_q of providers (Equation 1). Its values are in the interval $[0..1]$.

$$\delta_{sca}(c,q) = \Big(\big(\frac{1}{||P_q||} \sum_{p \in P_q} Prf_c^q(p) \ \big) + 1 \Big) \Big/ 2 \quad (1)$$

We thus define the system's adequation to a consumer as the average over the adequations concerning its k last queries.

Definition 1 *System-Consumer Adequation*

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

Its values are between 0 and 1. The closer the δ_{sca} value to 1, the more a consumer considers the system as adequate.

3.1.2. Satisfaction

The satisfaction w.r.t. a consumer $c \in C$ concerning a query q, denoted by $\delta_s(c, q)$, is related to those providers that performed q (denoted by the set $\widehat{P_q}$). The average of *preferences* expressed by the providers in $\widehat{P_q}$ 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. Let us illustrate this using our example scenario presented in Section 1.1. If the system allocated s_x's query only to p_4, for which s_x has a *preference* of e.g. 1, then s_x would be considered as completely satisfied (i.e. with a satisfaction of 1) even if he did not receive the number of results she desired. The following equation takes into account of this point using n instead of $||\widehat{P_q}||$.

$$\delta_s(c, q) = \Big(\big(\frac{1}{n} \sum_{p \in \widehat{P_q}} Prf_c^q(p) \ \big) + 1 \Big) \Big/ 2 \qquad (2)$$

where n stands for $q.n$. 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.

Definition 2 *Consumer Satisfaction*

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

Its values are between 0 and 1. The closer the *satisfaction* to 1, the more a consumer is satisfied.

This notion of satisfaction does not consider the context. Then, it does not allow a consumer to evaluate the efforts made by the query allocation method to satisfy it. For example, assume that, in scenario of Section 1.1, s_x has a *preference* of 1, .9, and .7 for allocating her query to p_4, p_3, and p_5, respectively. Now, suppose the system allocates the query to p_3. This corresponds to s_x's high *preferences*, so s_x is quite satisfied. However, there is still a provider for which her *preference* is higher: p_4. The notion of *Consumer Allocation Satisfaction*, denoted by $\delta_{as}(c)$, allows a consumer to evaluate how well the query allocation method works for it. Its values are in the interval $[0..\infty]$.

Definition 3 *Consumer Allocation Satisfaction*

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

If the obtained value is greater than 1, the consumer can conclude that the query allocation method acts to its favor. However, 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.

3.2. Local Provider Characterization

This section is devoted to the characterization of a provider. Intuitively, we strive to answer the following questions: "How well the expectations of a provider correspond to the last queries that have been proposed to it?" – *System-Provider Adequation* – ; "How well the last queries that a provider has treated meet its expectations?" – *Provider Satisfaction* – ; and "Is a provider satisfied with the allocation process?" – *Provider Allocation Satisfaction* –. To define these characteristics, each provider tracks its shown *preferences* to perform the k last proposed queries into vector $\overrightarrow{Prf_p}$. The k last proposed queries to a provider p are denoted by vector PQ_p^k.

3.2.1. Adequation

The system's adequation to a provider helps this provider to evaluate if the system corresponds to its expectations. For example, in the scenario of Section 1.1, one can consider the system as adequate to p_2, p_1, and p_5, because s_x'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.

Definition 4 *System-Provider Adequation*

$$\delta_{spa}(p) = \begin{vmatrix} \Big((\frac{1}{||PQ_p^k||} \sum_{q \in PQ_p^k} \overrightarrow{Prf}_p[q]\ ) + 1 \Big) \Big/ 2 \\ 0 \qquad\qquad\qquad\qquad if\ PQ_p^k = \emptyset \end{vmatrix}$$

The values that this adequation can take are in the interval [0..1]. The closer the value to 1, the greater the *adequation* of the system to a provider.

3.2.2. Satisfaction

Conversely to the *adequation* notion, the *satisfaction* of a provider only depends on the queries that it performs and is independent of the other queries that the system has proposed to it. In the scenario of Section 1.1, assume that the system allocates s_x's query to p_4. In this query allocation, p_4 is not satisfied since it did not desire to deal with such a query. Nonetheless, considering a query allocation alone is not very meaningful for a provider. What is more important for a provider is to be globally satisfied with the queries it performs. Thus, we define the *Provider Satisfaction* of a provider $p \in P$, $\delta_s(p)$ in Definition 5. Set SQ_p^k ($SQ_p^k \subseteq PQ_p^k$) denotes the set of queries that provider p performed among set PQ_p^k. The $\delta_s(p)$ values are between 0 and 1. The closer the value to 1, the greater the *satisfaction* of a provider.

Definition 5 *Provider Satisfaction*

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

On the one hand, the provider's *satisfaction* evaluates whether the system is giving queries to a provider that enable it to fulfill its objectives. On the other hand, a provider may also be interested in the efforts that the query allocation method makes to satisfy it. Conversely to a consumer that always receives results at each interaction, a provider is not allocated all the proposed queries. Hence, a provider cannot evaluate the job that the query allocation method does for it at each interaction. We then formally define this notion w.r.t. a provider $p \in P$, denoted by $\delta_{as}(p)$, as the ratio of its *Satisfaction* to its *system-provider adequation.* Its values are between 0 and ∞.

Definition 6 *Provider Allocation Satisfaction*

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

If the *allocation satisfaction* of a provider p is greater than 1, the query allocation method works well for 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.

3.3. Global Characterization

Conversely to the local characteristics of a participant (Sections 3.1 and 3.2), the characteristics we define here can be only applied to public data. So, in almost all cases, these characteristics are just applied to the participants' *intentions.* However, to preserve an homogeneity of presentation with respect to local characteristics, we develop the following characteristics for the static data, i.e. for *preferences.* While the local characteristics evaluate the query allocation method regarding the perception of a participant, the global characteristics evaluate objectively the query allocation method. The goal is to answer the following questions: "How well the last queries of a consumer correspond to the expectations of the providers that were able to deal with?" – *Consumer-System Adequation* – ; "How well a provider correspond to the consumer's expectations?" – *Provider-System Adequation* – and; "How well the query allocation method performs with respect to a consumer or a provider?" – *Query Allocation Efficiency w.r.t. a Consumer* – *Query Allocation Efficiency w.r.t. a Provider* – , respectively. To do so, we have to know how much a participant is adequate to the system.

The adequation of a consumer to the system enables it to evaluate how much providers are interested in its queries. Going back to scenario of Section 1.1, we can say that s_x's query is adequate to the system since great part of providers desire to treat it. According to this intuition, the adequation of a consumer $c \in C$ to the system concerning its interaction with the system for allocating its query q, noted $\delta_{csa}(c, q)$, is defined as the average of the *preferences* shown by set P_q of providers towards its query q (Equation 3). Its values are between 0 and 1.

$$\delta_{csa}(c,q) = \Big(\big(\frac{1}{||P_q||} \sum_{p \in P_q} \overrightarrow{Prf}_p[q] \ \big) + 1 \Big) \Big/ 2 \quad (3)$$

Thus, we define the consumer's adequation to the system as the average over the δ_{csa} values obtained in its k last queries.

Definition 7 *Consumer-System Adequation*

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

Its values are between 0 and 1. The closer the δ_{csa} value to 1, the greater the *adequation* of a consumer to the system. Having defined δ_{csa}, the *query allocation efficiency w.r.t. a consumer* $c \in C$, $\delta_{ae}(c)$, is defined in Definition 8. Its values are between 0 and ∞.

Definition 8 *Query Allocation Efficiency w.r.t. a Consumer*

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

On the one hand, the *query allocation efficiency w.r.t. a consumer* allows to evaluate, as objectively as possible, how well the query allocation method works for a consumer. On the other hand, the *query allocation efficiency w.r.t. a provider* (Definition 10) allows to evaluate whether the query allocation method strives to satisfy a provider or not. To define this latter notion, as for a consumer, we have to know how much a provider is adequate to the system.

The adequation of a provider to the system allows it to evaluate if consumers are interested in interacting with it. To illustrate the *Provider-System Adequation*, we use the scenario of Section 1.1. One may consider p_1 and p_2 as inadequate to the system since s_x does not want to deal with. Nevertheless, the most important is to evaluate this over the set PQ_p^k of queries. So, we define the adequation of a provider $p \in P$ to the system, denoted by $\delta_{psa}(p)$, over the k last proposed queries (Definition 9). Its values are in [0..1]. The closer the $\delta_{psa}(p)$ value to 1, the greater the *adequation* of a provider to the system.

Definition 9 *Provider-System Adequation*

$$\delta_{psa}(p) = \left| \begin{array}{ll} \Big(\big(\frac{1}{||PQ_p^k||} \sum_{q \in PQ_p^k} \overrightarrow{Prf}_c^q[p] \ \big) + 1 \Big) \Big/ 2 & \\ 0 & if\ PQ_p^k = \emptyset \end{array} \right.$$

We then define the *query allocation efficiency* with respect to a provider $p \in P$, $\delta_{ae}(p)$, as the ratio of its *satisfaction* to the product of its *system-provider adequation* by its *provider-system adequation.* Its values are in $[0..\infty]$.

Definition 10 *Query Allocation Efficiency w.r.t. a Provider*

$$\delta_{ae}(p) = \frac{\delta_s(p)}{\delta_{spa}(p) \cdot \delta_{psa}(p)}$$

If the value of the *query allocation efficiency* with respect to a participant (consumer or provider) is greater than 1, the query allocation method does a good job for a given consumer, or else, if the value is smaller than 1, a given consumer is dissatisfied with the query allocation method. In the case the value is 1, the query allocation method is quite neutral to a given consumer.

4. Satisfaction Balanced Mediation

In this section, we define a particular query allocation process that takes into account both the consumers' and providers' *intentions.* Anticipating the validation section, we call it *Satisfaction Balanced Mediation* (*SBMediation*). The process assumes that the consumers in the system show their *intentions*, denoted by vector $\overrightarrow{CI}$, to the mediator, while the providers keep them private. Instead, providers bid on queries, which is a means to reflect their *intentions* while keeping them private. We focus on three main points: *(i)* the definition of the process itself, *(ii)* the providers' bidding strategies, and *(iii)* the money flow in the system.

4.1. Definition of the Process

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 $\overrightarrow{B}[p] \in \mathbb{R}$ for all $p \in P_q$. If a bid is positive, the higher the 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, provider p's bid reflects p's *intention* to perform q. This should lead to the providers' *satisfaction*. However, if only bids are considered, the consumer may be dissatisfied either because its *intentions* with respect to providers are not considered (when it gets answers from providers it doesn't want) or because some queries are not performed (because no provider wants to treat them). Hence, to satisfy the consumer, $SBMediation$: *(i)* directly considers the consumer's *intentions* ($\overrightarrow{CI}$); *(ii) imposes* the query when not enough providers want to perform it [16]. Processing $SBMediation$ for some query q, amounts to computing *(i)* $\overrightarrow{Alloc}_q$ (Section 2), and *(ii)* $\overrightarrow{Trans}_q$, which defines all the "monetary" transfers that occur among the providers in P_q. In both steps, consumer's *intentions* and bids have to be balanced to ensure both consumer's and providers' *satisfaction.*

4.1.1. Query allocation

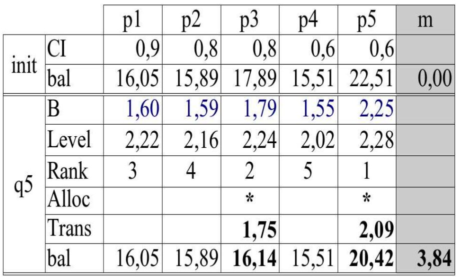
Query q is allocated to the $min(n, N_q)$ "best" providers, which are given by vector of ranking $\overrightarrow{R}$. Intuitively, $\overrightarrow{R}[1] = p$ iff p is the best ranked, $\overrightarrow{R}[2]$ stands for the second best ranked and so on. Hence, $\overrightarrow{Alloc}_q[p] = 1$ iff $\exists i, \overrightarrow{R}[i] = p$ and $i \leq min(N_q, n)$. Vector $\overrightarrow{R}$ is computed with the providers' levels $\overrightarrow{L}$ (Definition 11).

Definition 11 *Vector of providers' levels*
$\forall p \in P_q$, with $\omega \in [0..1]$,

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

Parameter ω reflects the relative importance the mediator gives to the consumer's *intentions* or the providers' bids (which themselves reflect the providers' intentions). If $\omega = 0$, only the consumer's *intentions* are considered, thus leading to providers dissatisfaction. Conversely, if $\omega = 1$, only bids are considered, leading to consumers dissatisfaction. Thus, the mediator should set parameter ω according to the balance between both consumers' and providers' *satisfaction* that it wants to reach. Table 2 shows the case of a competition. The consumer asks for two providers, and more than two 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. So, 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 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.

Table 2: $SBMediation$: a competition case



		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**

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

4.1.2. Monetary transfers

Bids cannot be directly compared, because of the provider's *intentions.* To overcome this difficulty, the *theoretical bid* ($\overrightarrow{B}^{Th}(p, l)$) corresponds to the amount that p should bid for reaching level l. With $\omega \neq 0$ and $\alpha = 1$ if $l \geq 0$, and $\alpha = -1$ otherwise, $\overrightarrow{B}^{Th}(p, l)$ is given by the following formula.

$$\overrightarrow{B}^{Th}(p,l) = \alpha max(((\alpha \times l)^{\frac{1}{\omega}} (\overrightarrow{CI}_q[p]+1)^{\frac{\alpha(\omega-1)}{\omega}} - 1), 0) \quad (4)$$

For example, in Table 2, 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).

Given a provider p', $\overrightarrow{PTr}[p,p']$ denotes what p' owes due to the allocation of q to p ($\overrightarrow{PTr}[p,p'] = 0$ if p is not allocated q). Then, the total amount paid by p' is defined by a sum (Formula 5).

$$\forall p' \in P_q, \quad \overrightarrow{Trans}[p'] = \textstyle\sum_{p \in P_q} \overrightarrow{PTr}[p,p'] \qquad (5)$$

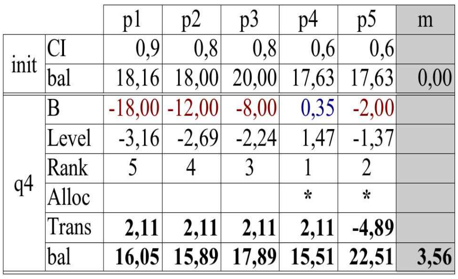
There is a competition when there are enough providers that want to be allocated the query. In that case, each of them pays the amount of its theoretical bid to reach the level of the best provider which has not been selected (in the spirit of a generalized Vickrey auction [18] except that the consumer's *intention* is considered). In Table 2, only providers p_5 and p_3 pay (respectively 2.09 and 1.75) to the mediator, thus decreasing their own money balance (*bal*). A requisition case occurs when at least one provider is imposed the query. Obviously, being imposed does not meet at all the expectations of the n_i imposed providers (n_i stands for the number of imposed providers). Hence, to keep them satisfied in the long run, the idea is then to distribute the cost of the imposition on *all* the providers in P_q (in the spirit of [16] considering the consumer's *intentions* too). Having obtained a reward, the n_i imposed providers are more likely, in the future, to obtain the queries they expect (because they have more money) so leading to their *satisfaction*. The formal definitions of the transfers in the imposition case are:

Definition 12 *Partial transfers in a requisition case*
If provider p is allocated q and $\overrightarrow{B}[p] < 0$, then for all $p' \in P_q$:

$$\overrightarrow{PTr}[p,p'] = \left| \begin{array}{ll} \dfrac{-\overrightarrow{B}^{Th}(p, \overrightarrow{L}[\overrightarrow{R}[min(n+2,N)]])}{N} & ifp \neq p' \\ \overrightarrow{B}^{Th}(p, \overrightarrow{L}[\overrightarrow{R}[min(n+1,N)]]) & else \\ \quad - \dfrac{\overrightarrow{B}^{Th}(p, \overrightarrow{L}[\overrightarrow{R}[min(n+2,N)]])}{N} & \end{array} \right.$$

In the example of Table 3, p_5 is imposed and thus gets a 4.89 reward. All the providers contribute to this reward. Notice also in both Table 2 and Table 3 that the mediator gets some money left, which is of no use for it. This point is discussed later, in section 4.3.

Table 3: $SBMediation$: an imposition case



		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**

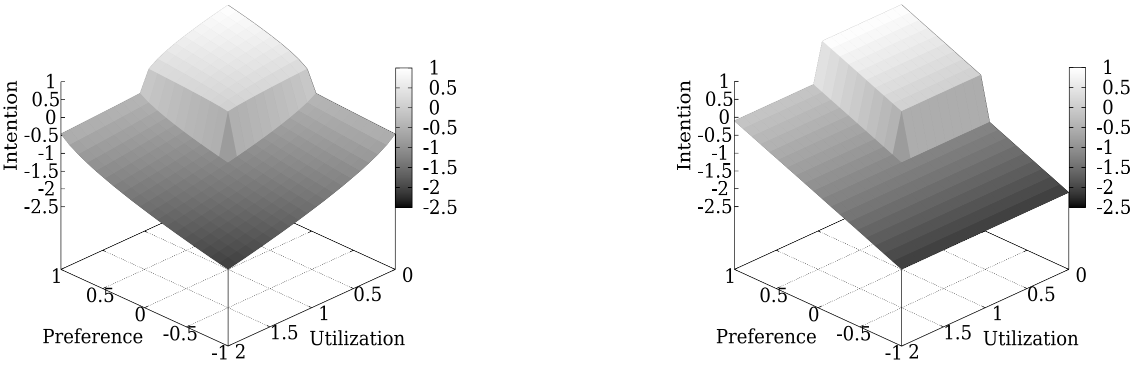
$\omega = 0.5$; $n = 2$

4.2. The Providers' Bidding Strategies

We now discuss the way in which a provider works out its bids to perform queries. A simple way to do so, is that each provider maintains a billing rate for its resources based on its *preferences* to perform queries. Then, a provider's bid to perform an incoming query may be the product of the estimated amount of resources required to perform such a query by the billing rate (such as in [17]). In our case, the context of a provider is more complex: we have to consider its *preferences*, load, current *satisfaction*, and current money balance. So, a provider first works out its *intention* to perform an given query q by considering its *preferences*, $Prf_p(q) \in [-1..1]$, its *utilization* at that time t, $U_p(t)$, and its current *satisfaction*. The *satisfaction* considered by a provider to compute its *intention* is based on its *preferences* as defined in Section 3.2.2. Intuitively, on the one hand, a provider can sometimes accept queries it does not want if it is satisfied. On the other hand, a provider does not pay so much consideration to its *utilization* and focuses on its *preferences*, to obtain desired queries, if it is not satisfied. Then, a provider computes its intention as follows [11].

$$PI_p(q) = \left| \begin{array}{ll} (Prf_p(q)^{1-\delta_s(p)})(1-U_p(t))^{\delta_s(p)}, & ifPrf_p(q) > 0 \\ & \wedge U_p(t) < 1 \\ -\big((1 - Prf_p(q) + 1)^{1-\delta_s(p)} \times & \\ \times (U_p(t) + 1)^{\delta_s(p)}\big) & otherwise \end{array} \right.$$

Figure 1 illustrates the behavior of function PI when providers have a satisfaction value of 0.5 and 0. On the one hand, we observe that when providers are satisfied of 0.5 (see Figure 1(a)) the providers' *preferences* and *utilization* have



(a) for a provider's *satisfaction* of 0.5

(b) for a provider's *satisfaction* of 0

Figure 1: Tradeoff between *preference* and *utilization* for getting intention.

the same importance. On the other hand, when providers are not satisfied at all (see Figure 1(b)) the providers' *utilization* has no importance for providers and their *intentions* are only defined by their *preferences*.

Once a provider obtains its *intention* w.r.t. a query, it then proceeds to work out its bid to perform such a query. Intuitively, the provider's bid is the product of its *intention* by its current money balance. The current 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. Thus, to avoid such a behavior, a provider offers at most only a pre-defined percent of its current money balance, denoted by the constant $0 < c_0 \leq 1$. We formally define the providers' bid in Definition 13, where constant c_1 is set to the initial money balance of a provider.

Definition 13 *Provider's Bid*

$$B_p(q) = \left| \begin{array}{ll} PI_p(q) \cdot bal_p \cdot c_0 & if\ PI_p(q) > 0 \\ PI_p(q) \cdot c_1 & otherwise \end{array} \right.$$

The idea behind the above definition is that a provider always sets positive bids when it desires to perform queries and it is not *overutilized*, otherwise it sets a negative bid. This allows a provider to preserve its *preferences* and computational resources while good response times are also ensured to consumers.

4.3. The Flow of Money

In the whole system, in particular in the mediation, the money used is *purely virtual*. We could speak of tokens or jetons as well. This point has to be stressed upon for two main reasons. First, it underlines the fact that 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. Later, after it has decided which providers it chooses, the consumer might give *real money* to them, because it uses their services. This point is far beyond the focus of this paper, which concern is the query allocation problem. Second, when using real money, one can assume that participants 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 the virtual money of a system, one can no longer make such assumptions. In our case, for the system to be correctly regulated, we have to make precise *the way money circulates* within it. There are several possibilities, but we describe the choices that have been implemented for the validation.

The system is composed of consumers, providers and the mediator which implements the mediation process. The consumers do not use the virtual money, as they directly show their intentions. The providers spend and earn money through the mediator only. They spend money by bidding on queries at the mediator's. They also

Table 4: 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**

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

spend money to compensate other providers that have been imposed by the mediation process. They earn money when they are imposed by the process. As for the mediator, it has been shown [6], that with the type of mediation process defined, the mediator never looses money. It even tends to accumulate money coming from the providers in the course of time, thus making the providers poorer and poorer. This could distort the mediation process or even block the system when the providers no longer have money. A simple solution has been adopted: the mediator regularly redistributes the money it piles up to the providers, in a equitable way. From the providers' point of view, it is another, regular, way of earning money.

Table 4 illustrates the flow of money along a sequence of five mediations followed by redistribution of money by the mediator. At the initiation step, we quote the providers' and mediator's initial money balance (σ), and the consumer's *intentions* w.r.t. 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.

4.4. Communication Cost

We analyze communication cost in terms of number of messages that should be transferred over the network to perform a query. The $SBMediation$'s communication cost is given by the following theorem where n_i is the number of imposed providers.

Theorem 1 *The total number of transferred messages, $Mssg$, by $SBMediation$ to perform a query is $Mssg = 3(N+1)+n$ for a competition case and $Mssg = 4N + n_i + 3$ for an imposition case.*

Proof 1 *Given a query q and the set P_q of providers, mediator m first asks for $q.c$'s intention and P_q's bids, which return such an information to m. The number of exchanged messages at this moment is $mssg_0 = 2N + 2$. Once m receives the participants' interests, it computes the P_q's level as defined in Section 4.1.1 and ranks them w.r.t. to their level. The complexity of evaluation phase is $O(N \log_2(N))$. Having done this, m informs all P_q providers of the mediation result and waits for results from the n selected providers. The number of transferred messages in this phase is $mssg_1 = N + n$. When an imposition case occurs, $mssg_1^* = N - n + n_i$ more messages are exchanged since all providers pay an amount of money to m, which gives an award to the n_i imposed providers. Finally, m sends the results to c, which implies $mssg_2 = 1$ exchanged messages. Therefore, for a competition case, $Mssg = mssg_0 + mssg_1 + mssg_2 = 3(N + 1) + n$, and for an imposition case, $Mssg = mssg_0 + mssg_1 + mssg_1^* + mssg_2 = 4N + n_i + 3$.*

We can reduce the number of transferred messages by using participants' *representatives* [6], but the problem of reducing communication cost is orthogonal to the problem we address in this paper.

5. Validation

Our main objective is to evaluate, from a satisfaction point of view, how well $SBMediation$ operates with autonomous providers.

5.1. Setup

We built a Java-based simulator which models a *mono-mediator* distributed information system following the mediation system architecture presented in [6]. We compare the $SBMediation$ process to $Capacity$ *based* one [8, 15], which is a well-known approach, in distributed information systems, to balance queries among providers. $Capacity$ *based* allocates queries to those providers that have the most available capacity amongst the set P_q of providers. For $SBMediation$ we set $\omega = 0.5$, which means that the consumers' and providers' interests are given the same importance for allocating queries. For both query allocation methods, the following configuration (Table 5) is the same and the only thing that changes is the way in which each method allocates the queries.

In all the experimentations, the number of consumer and provider sites is 200 and 400 respectively. Queries arrive to the system in a Poisson distribution. Consumers and providers are initialized with a satisfaction value of 0.5, and a satisfaction size[2] of 200 and 500 respectively. For each incoming query q, we randomly obtain $q.n$ between 1 and 3. Since our main focus is to validate the way in which queries are allocated, we do not consider the bandwidth problem in this work.

For our experiments, we set the heterogeneity of the providers' *capacity* in accordance to the results in [14]. We generate around 10% of low-capacity, 60% of medium-capacity, and 30% of high-capacity providers. The high-capacity providers are 3 times more capable than medium-capacity providers and still 7 times more capable than low-capacity ones. We generate two classes of query that consume, respectively, 130 and 150 treatment units at the high-capacity providers (i.e. 1.3 and 1.5 seconds respectively). In our experiments, the consumer's *preferences* denote their *intentions*, while the provider's *intentions* are computed as defined in Section 4.2. To simulate high autonomy in our experiments, we randomly obtain the consumers' *preferences* between 0 and 1, and the providers' *preferences* between -1 and 1. More sophisticated mechanisms for obtaining such *preferences* can be applied (e.g. using the *TCL* or *Rush* language), but it is beyond the scope of this paper. We assume that providers decide to leave the system if they are satisfied by 0.3 less than their *adequation* (given our simulation setup, the system's *adequation* w.r.t. providers is 0.5).

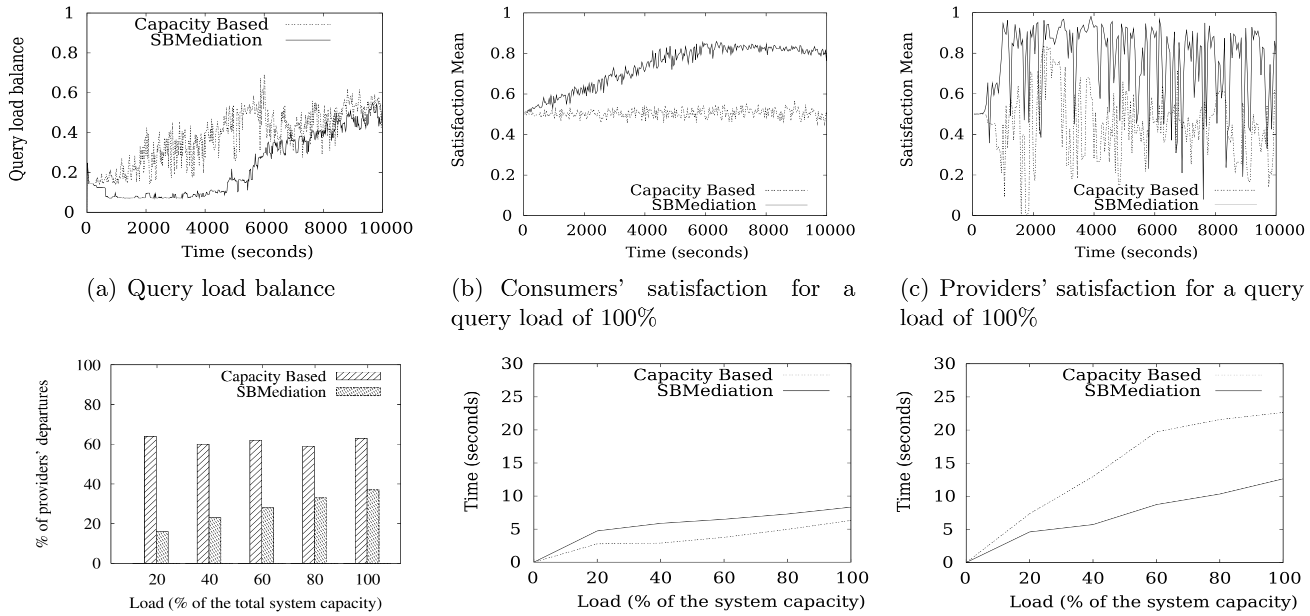
Table 5: Simulation parameters.

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

5.2. Results

We analyze $SBMediation$ from three points of view: *QLB*, *satisfaction balance*, and *performance*. The results, for *QLB*, presented here are for a workload of 10% (at the beginning of the simulation) increasing to 100%. Concerning *QLB*, we observed that, for *workloads* from 10% to 60% of the total system capacity (i.e. the aggregate capacity of all providers) $SBMediation$ approach is under, and so worse than $Capacity$ *based* one (see Figure 2(a)). Nonetheless, we observed that, for *workloads* from 60% to 100% of the total system capacity, $SBMediation$ almost ensures the same query load balance than $Capacity$ *based*. This is because, for high *workloads*, providers start to pay more attention to their *utilization* than to their *preferences*. Moreover, we observed that, for query arrival rates from 10% to 40% of the total system capacity, providers suffers from query *starvation* with $Capacity$ *based*. This is because

[2] Which denotes the k last issued queries by consumers and the $||SQ_p^k||$ last treated queries by providers, respectively.



(a) Query load balance

(b) Consumers' satisfaction for a query load of 100%

(c) Providers' satisfaction for a query load of 100%

(d) Providers' departures

(e) Response time with captive participants

(f) Response time with autonomous participants

Figure 2: Satisfaction and performance results.

the most capable providers monopolize queries. Since *SBMediation* considers their *satisfaction*, no provider monopolizes queries. The next results show that things are still much better for *SBMediation*.

Concerning *satisfaction*, we observed through our experiments that participants' *satisfaction* is almost the same for different query arrival rates (from 10% to 100% of the total system capacity). By this fact, we only present the *satisfaction* results for a query arrival rate of 100% of the total system capacity. Consumers can observe that *Capacity based* is neutral (because their *satisfaction* is equal to their *adequation*), but they benefit from paid attention to their *intentions* by *SBMediation* (see Figure 2(b)). In fact, *SBMediation* gives always better *satisfaction* to consumers than *Capacity based* because *Capacity based* proceeds in a blind way as far as this point is concerned. On the other side, while *Capacity based* satisfies providers under their *adequation*, *SBMediation* satisfies them almost all the time over their *adequation* (see Figure 2(c)). This means that *SBMediation* gives, in average, interesting queries to providers and that *Capacity based* punishes them with uninteresting ones. Notice that, the values of the providers' *satisfaction* suffer from greater oscillations than those of consumers, because of natural competition of providers for performing queries.

Now, we proceed to study the impact on performance of the providers' autonomy. We can observe in Figure 2(d) that while *Capacity based* looses in average 60% of providers for all query arrival rates, *SBMediation* looses only a 27% of providers! Indeed, such provider's departures are reflected on the ensured response time[3]. To better illustrate the departures impact, we show in Figure 2(e) the response times ensured by both *SBMediation* and *Capacity based* when participants are captive, i.e. they do not leave the system by dissatisfaction. As expected, *Capacity based* is better than *SBMediation* in captive environments, but things are not the same when confronted to autonomous participants (see Figure 2(f)). We observed that while *SBMediation*

[3] As is conventional, it is defined as the elapsed time from the moment that a query q is issued to the moment that the $q.c$ site receives the response of q.

degrades its performance only by a factor of 1.3, *Capacity based* does it by a factor of 4! Therefore, *SBMediation* can scale up in such environments while *Capacity based* cannot.

All above results demonstrate the great superiority, in all the cases, of *SBMediation* against the *Capacity based* when confronted to autonomous participants.

6. Related Work

In the context of large distributed information systems, the problem of balancing queries while respecting the participants' *expectations* has not received much attention and is still an open problem. Most of the work on query load balancing [8, 15] has only dealt with the problem of minimizing providers' *utilization*.

Several solutions [2, 5, 17] strive to deal with such *intentions* in query allocations by means of economical models. Mariposa [17] pioneered the use of a market approach for dealing with the query allocation problem. Nevertheless, its query allocation procedure is simple and limited. Consumers cannot freely express their *intentions* since it is inherently assumed that they are just interested in response times and low prices for acquiring services. Furthermore, some queries may not get processed although relevant providers exist and it is unclear how this technique really ensures the *QLB* in the system. A survey of economic models for various aspects of distributed system is presented in [4]. The notion of *utility* is clearly linked to satisfaction. However, it is generally reduced to monetary concerns only. Most of the processes that are proposed in the field of distributed rational decision making [13] are *individually rational*: the utility of any participant in the process is no less than the utility it would have by not participating. This property is not relevant in cooperative contexts where some participants may be imposed, which implies having a lower utility in participating. Thus, *satisfaction* is still relevant in such contexts because it is a long-run notion.

Auctions are widely recognized as a way to manage negotiation among participants. Several kinds of auction mechanisms exist [13, 20]. In the purely competitive case our work looks like this generalized Vickrey auction, but it pushes generalization further because it takes into account the consumer's *intention* factor via ranking and theoretical bid. Multi-attribute auctions [3, 19] are another kind of generalization, which help finding goods providers, without considering requisition.

Imposition occurs any time a participant is required to perform a query that it does not want to. The basic idea of fair imposition [16] is that all the participants must support the imposed one. The problem is tackled from a purely economical point of view, each participant sending its cost to perform the task. Fairness is obtained because the invoicing asks all participants to pay the same amount and gives a compensation to the imposed one. In *SBMediation*, the requisition case generalizes the fair imposition mechanism, with the notion of consumer's *intention* and to n selected providers.

In our proposal presented in [11], the providers' *intention* is already considered, but the consumers' one does not. In [12], we proposed a set of strategies for balancing queries considering such *intentions*. The work [12] is complementary to the contributions of this paper. One can apply, for example, such strategies before performing the query allocation process proposed in this work.

7. Conclusion

In this paper, we addressed the query allocation problem in distributed information systems from a new point of view, by considering not only query load but also participants' *satisfaction*. To our knowledge, this is the first work that studies this problem in its whole generality. Our work brings several contributions.

First, we proposed a complete model that characterizes the participants' interests for allocating and performing queries. These definitions are original, considering long-run notions: *adequation* and *satisfaction*. They are independent of how the participants' *intentions* are computed and how the mediation process considers them. The proposed model was designed to be general, and thus, can be used for any distributed system architecture.

Second, we propose a mediation process, called *SBMediation*, that considers consumers' *intentions* and providers' bids to allocate queries. We discussed how query allocation and invoicing steps leads to participants' *satisfaction*. The originality of *SBMediation* is to satisfy both participants' expectations and query demand.

Third, we evaluated and compared, through experimentation, the behavior of *SBMediation* against *Capacity based*. We demonstrated that *SBMediation* significantly outperforms *Capacity based*. We discussed that *Capacity based* suffers from *query starvation* problems for low query arrival rates while *SBMediation* does not. We showed that participants are, in general, very satisfied with *SBMediation*. This is not the case for *Capacity based* which suffers from several providers' departures due to dissatisfaction. Furthermore, the results demonstrate that *SBMediation* can scale up in these systems while *Capacity based* cannot. Finally, since *SBMediation* considers the consumers' *intentions* and providers' bid without any consideration about how they are computed, it is self-adaptable to the changes in their expectations.

References

[1] Grid4All project. http://www.grid4all.eu.

[2] R. Buyya, H. Stockinger, J. Giddy, and D. Abramson. Economic Models for Management of Resources in Grid Computing. *The Computing Research Repository (CoRR)*, cs.DC/0106020(1), 2001.

[3] E. David, R. Azoulay-Schwartz, and S. Kraus. Protocols and Strategies for Automated Multi-Attribute Auctions. In *Procs. of AAMAS*, 2002.

[4] D. F. Ferguson, C. Nikolaou, J. Sairamesh, and Y. Yemini. Economic Models for Allocating Resources in Computer Systems. In S. Clearwater, editor, *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, 1996.

[5] D. F. Ferguson, Y. Yemini, and C. Nikolaou. Microeconomic Algorithms for Load Balancing in Distributed Computer Systems. In *ICDCS*, 1988.

[6] P. Lamarre, S. Cazalens, S. Lemp, and P. Valduriez. A Flexible Mediation Process for Large Distributed Information Systems. In *Procs. of CoopIS*, 2004.

[7] L. Li and I. Horrocks. A Software Framework for Matchmaking Based on Semantic Web Technology. In *Procs. of the WWW Conf.*, 2003.

[8] R. Mirchandaney, D. Towsley, and J. Stankovic. Adaptive Load Sharing in Heterogeneous Distributed Systems. *Parallel and Distributed Computing*, 9(4), 1990.

[9] M. H. Nodine, W. Bohrer, and A. H. Ngu. Semantic Brokering over Dynamic Heterogeneous Data Sources in InfoSleuth. In *Procs. of the ICDE Conf.*, 1999.

[10] T. Özsu and P. Valduriez. *Principles of Distributed Database Systems (2nd ed.)*. Prentice-Hall, 1999.

[11] J.-A. Quiané-Ruiz, P. Lamarre, and P. Valduriez. Satisfaction Based Query Load Balancing. In *Procs. of CoopIS Conf.*, 2006.

[12] J.-A. Quiané-Ruiz, P. Lamarre, and P. Valduriez. K$_n$Best - A Balanced Request Allocation Method for Distributed Information Systems. In *Procs. of the DASFAA Conf.*, 2007.

[13] T. W. Sandholm. *Multiagent Systems, a modern approach to Distributed Artificial Intelligence*, chapter Distributed Rational Decision Making. The MIT Press, 2001.

[14] S. Saroiu, P. Gummadi, and S. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Procs. of Multimedia Computing and Networking*, 2002.

[15] N. G. Shivaratri, P. Krueger, and M. Singhal. Load Distributing for Locally Distributed Systems. *IEEE Computer*, 1992.

[16] Y. Shoham and M. Tennenholtz. Fair Imposition. In *Procs. of IJCAI*, 2001.

[17] M. Stonebraker, P. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu. Mariposa: A Wide-Area Distributed Database System. *VLDB*, 5(1), 1996.

[18] W. Vickrey. Counterspeculation, Auctions, and Competitive Sealed Tenders. *Finance*, 16(1), 1961.

[19] N. Vulkan and N. R. Jennings. Efficient Mechanisms for the Supply of Services in Multi-Agent Environments. *Decision Support Systems*, 28, 2000.

[20] E. Wolfstetter. Auctions: and introduction. *Economic Surveys*, 10(4), 1996.