manuscript No.
(will be inserted by the editor)

A Self-Adaptable Query Allocation Framework for Distributed Information Systems

Jorge-Arnulfo Quiané-Ruiz · Philippe Lamarre · Patrick Valduriez

Abstract In large-scale distributed information systems, where participants 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, preserving the participants' interests is also important. In this paper, we make the following contributions. First, we provide a model to define the participants' perception of the system regarding their interests and propose measures to evaluate the quality of query allocation methods. Then, we propose a framework for query allocation called *Satisfaction-based Query Load Balancing* (*SQLB*, for short), which dynamically trades consumers' interests for providers' interests based on their *satisfaction*. Finally, we compare *SQLB*, through experimentation, with two important baseline query allocation methods, namely *Capacity based* and *Mariposa-like*. The results demonstrate that *SQLB* yields high efficiency while satisfying the participants' interests and significantly outperforms the baseline methods.

Keywords distributed information systems, query allocation, query load balancing, satisfaction

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

J.-A. Quiané-Ruiz · P. Lamarre · P. Valduriez
Atlas group, INRIA and LINA – Université de Nantes
2 rue de la Houssinière
44322 Nantes, France

J.-A. Quiané-Ruiz
E-mail: Jorge.Quiane@univ-nantes.fr
P. Lamarre
E-mail: Philippe.Lamarre@univ-nantes.fr
P. Valduriez
E-mail: Patrick.Valduriez@inria.fr

1 Introduction

We consider distributed information systems with a mediator that allows consumers to access information providers through queries [23,36]. Consumers and providers (which we refer to participants) are autonomous in the sense that they are free to leave the mediator at any time and do not depend on anyone to do so. In the context 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.

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. Data heterogeneity means that providers provide different data and thus produce different results for the same query. Providers declare their *capabilities* for performing queries to the mediator. Then, the main function of the mediator is to allocate each incoming query to the providers that can satisfy it. Much work in this context has focused on distributing the query load among the providers in a way that maximizes overall performance (typically throughput and response time), i.e. *query load balancing* (*qlb*) [6,12,24, 35,40]. 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. Consumers' preferences may represent e.g. their interests towards providers (based on reputation for example), preferred providers, or quality of service. Providers' preferences may represent, for example, their topics of interests, relationships with other participants, or strategies.

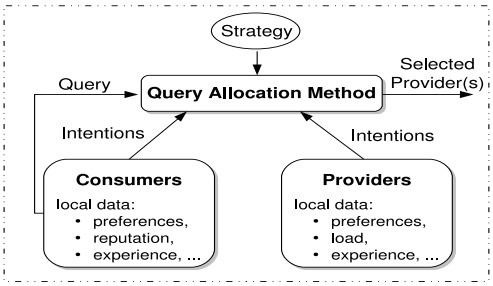


Fig. 1 Overview of Query Allocation.

Example 1 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, its 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* is a problem since it may lead participants to leave the mediator. Thus, it is 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, the participants' *preferences* must be taken into consideration when balancing 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 static data, i.e. long-run, while the desire of a participant to allocate and perform queries may depend on its context and thus is more dynamic, i.e. short-run. 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 1).

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' expectations in the long-run. Economic approaches consider *utility* and *individual rationality*, but utility is not normalized and is commonly related to economic notions (e.g. money), and individual rationality does not capture long-run aspects.
- 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.
- Unlike several economic models [9,10,42], 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.
- 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 our knowledge, this problem has not been addressed completely before. Thus, our first objective is to propose a model that provides a satisfaction notion to reflect how well the mediator meets the participants' expectations in the long-run. Then, our second objective is to propose a query allocation framework that considers both satisfaction and intentions of participants.

1.1 Motivating Example

Consider a public e-marketplace where thousands of companies can share information and do business (such as ebay-business [1] and freightquote [3]). 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, an e-marketplace should *(i)* protect, in the long-run, the participants' intentions for doing business, *(ii)* allow consumers to quickly obtain results, and *(iii)* allocate queries so that providers should have the same possibilities for doing business, i.e. to avoid *starvation* [11].

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 [3]. 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

Table 1 Providers for $eWine$'s query.

Providers	Prov.'s Int.	Cons.'s Int.	Avail. Cap.
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

in those negotiations that involve queries meeting their intentions. In this scenario, the mediator must perform several tasks.

First, 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. [17,20], so we do not focus on this problem in this paper.

Second, the mediator should obtain $eWine$'s intentions to deal with such providers and the providers' intention to deal with $eWine$'s query[1]. This can be done following the architecture proposed in [18]. Assume that the resulting list contains, for simplicity, only 5 providers: $p_1, \ldots, p_5$. Table 1 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 1 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.

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 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 it may considerably hurt response time, which dissatisfies $eWine$ with a poor response time and p_5 by overloading it. Again, whether this occurs several times 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? or the providers' available capacity?* In this paper, we address this question so that a query allocation method can decide on the fly what to do according to the participants' status.

[1] For simplicity, we assume in this example that the intentions values are binary.

1.2 Contributions and Organization

The rest of this paper is organized as follows. After defining the problem in Section 2, we present the main contributions of this paper:

- We propose a new model to characterize the participants' expectations in the long-run, which allows evaluating a system from a satisfaction point of view. This model facilitates the design and evaluation of *qlb* methods when confronted to autonomous participants (Section 3).
- We define the properties that allow evaluating the quality of *qlb* methods and propose measures to do so (Section 4).
- We propose *Satisfaction-based Query Load Balancing* ($SQLB$, in short), a flexible framework with *self-adapting* algorithms to allocate queries while considering both *qlb* and participants' intentions. $SQLB$ affords consumers the flexibility to trade their preferences for the providers' reputation and providers the flexibility to trade their preferences for their utilization. It also allows the mediator to trade consumers' intentions for providers' intentions. Furthermore, $SQLB$ affords the mediator the flexibility to adapt the query allocation process to the application by varying several parameters (Section 5).
- We demonstrate, through experimental validation, that $SQLB$ significantly outperforms baseline methods, the *Capacity based* and *Mariposa-like* methods, and yields significant performance benefits. We demonstrate the self-adaptability of $SQLB$ 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 6).

Then, we survey related work in Section 7 and conclude the paper in Section 8.

This paper is an extended version of [32] with the following added value. We present new global character-

istics that allow evaluating the query allocation method in a more objective way (Section 3.3) and discuss in Section 3.4 the two possible levels of satisfaction that a participant can have. We also define in Section 5.3.2 a strategy that allows the mediator to adapt the query allocation process to applications independently of the way in which participants compute their intentions. Furthermore, we analyze the *SQLB* communication cost in Section 5.3.4. Finally, we run new experimentations to demonstrate the adaptability of *SQLB* to the participants' expectations (Section 6.3.3) as well as to validate the proposed strategy (Section 6.3.4).

2 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. 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, and $q.n \in \mathbb{N}^*$ is the number of providers to which the consumer wishes to allocate its query. 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 earlier, such techniques are out of the scope of this paper and thus we assume there exists one in the system, e.g. [17,20], 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. 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 $\overrightarrow{Allocc}$ of length N, or $\overrightarrow{Allocc}_q$ and N_q if there is an ambiguity on q, such that,

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

As we assume that queries should be treated if possible, this leads to $\sum_{p \in P_q} \overrightarrow{Allocc}[p] = min(q.n,\ N)$. In the following, the set of providers such that $\overrightarrow{Allocc}[p] = 1$ is noted $\widehat{P_q}$. 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. Providers have a finite *capacity* to perform queries, denoted by function $cap > 0$. The capacity of a provider denotes the number of computational units that it can have. Thus, the *utilization* of a provider p at time t, denoted by function $\mathcal{U}_t(p)$, is defined as p's load with regards to its capacity.

A consumer $c \in C$ is free to express its intention $ci_c(q, p)$ 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 $pi_p(q)$ for performing a query q. Values of participants' intention are in 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[2]. A null value, i.e. a 0 value, denotes a participant's indifference. It is up to a participant to compute its own intentions by combining different local and external criteria (e.g. utilization, preferences, response time, reputation, past experience, etc.). The way in which a participant computes its intentions is considered as private information and is not revealed to other participants.

In these environments, where participants are autonomous, it is crucial that a query allocation method considers participants' intentions in order 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 query allocation problem as follows.

Problem Statement. Given a mediator dealing with autonomous participants, the problem we address is computing and using participants' intentions to perform query allocation at the mediator such that response time, system capacity, and participants' satisfaction are ensured.

3 Pariticpants' Characterization

We define, in this section, a model that allows comparing query allocation methods having different approaches to regulate the system, such as economic and *qlb* methods. 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:

- the system adequation to a participant, e.g. a system where a provider (respectively consumer) can-

[2] It is worth remembering that this does not means it can refuse to perform (resp. allocate) the query.

not find any query (resp. provider) it desires is considered inadequate to such a participant, and
– the participant's adequation to the system, 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.

Let us illustrate both adequation notions 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 many consumers do not want to deal with it.

Both adequation notions are needed to evaluate whether 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.

The second characteristic is *satisfaction*. As for adequation, two kinds of satisfaction could be considered:

– the satisfaction of a participant with what it gets from the system, e.g. a provider (respectively consumer) that receives queries (resp. results from the providers) it does not want is not satisfied, and
– the participant's satisfaction with the job that the query allocation method does, e.g. a provider (respectively consumer) that performs queries (resp. results from the 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 the only 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. In addition to the two kinds of adequation and satisfaction, we are interested in two other global characteristics: *Allocation Efficiency w.r.t. a Consumer* and *Allocation Efficiency w.r.t. a Provider*. In the following, we define all previous notions with regards to what a participant can observe in Sections 3.1 and 3.2. We then define the previous global notions in Section 3.3, which are only observable by the mediator.

It is worth noting that, because of autonomy, preserving the participants' intentions is quite important so that they have some interest in staying 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 some past 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.

Let us make two other 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 (dynamic data) or with respect to their preferences (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.

3.1 Local Consumer Characterization

Our characterization considers only the information that a consumer can obtain from the system. This characterization needs to use the memory of each consumer $c \in C$, which is denoted by set IQ_c^k. Intuitively, the characteristics we present in this section are useful to answer the following questions:

– "How well do the expectations of a consumer correspond to the providers that were able to deal with its last queries?" – *System-Consumer Adequation* – ,

– "How far do the providers that have dealt with the last queries of a consumer meet its expectations?" – *Consumer Satisfaction* – , and
– "Does the query allocation method do a good job for a consumer?" – *Consumer Allocation Satisfaction* –.

3.1.1 Adequation

The system adequation to a consumer characterizes the perception that the consumer has from the system. For example, in our motivating example of Section 1.1, $eWine$ 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, the system adequation w.r.t. a consumer $c \in C$ and concerning a query q, denoted by $\delta_{sca}(c, q)$, is defined as the average of c's intentions towards set P_q (Equation 1). Its values are in the interval [0..1].

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

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

Definition 1 System-Consumer Adequation

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

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

3.1.2 Satisfaction

This notion evaluates whether a mediator is allocating the queries of a consumer to the providers from which it wants to get results. To define the consumer's satisfaction 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. 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 $||\widehat{P_q}||$.

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

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

Definition 2 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..∞].

Definition 3 Consumer Allocation Satisfaction

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

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 possible characterization of a provider according to the information that it can obtain from the system. To this end, 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. Intuitively, this characterization is useful to answer the following questions:

– "How well do the expectations of a provider correspond to the last queries that the mediator has proposed to it?" – *System-Provider Adequation* – ,
– "How well do the last queries that a provider has treated meet its expectations?" – *Provider Satisfaction* – , and

– "Does the query allocation method do a good job for a provider?" – *Provider Allocation Satisfaction* –.

3.2.1 Adequation

The system adequation w.r.t. a provider evaluates if the system corresponds to the expectations of a provider. 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 $p \in P$, $\delta_a(p)$, as the average of p's shown intentions towards set PQ_p^k.

Definition 4 System-Provider Adequation

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

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.

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 have been proposed to it. To illustrate this notion, suppose that in our motivating 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. 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 formally define the satisfaction of a provider $p \in P$ in Definition 5. Set SQ_p^k ($SQ_p^k \subseteq PQ_p^k$) denotes the set of queries that provider p performed among the set of proposed queries (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

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

The satisfaction notion evaluates whether the system is giving queries to a provider according to its (those of the provider) expectations 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 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 (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.

3.3 Global Characterization

Conversely to Sections 3.1 and 3.2 that evaluate the query allocation method regarding what a participant perceives from the system, this section allows evaluating the query allocation method from a general point of view. For example, it is possible that a consumer (respectively a provider) is not satisfied with the job the query allocation is doing because providers (resp. consumers) generally do not want to deal with its queries (resp. do not want to get results from it). This global characterization considers this point. The goal of these characteristics is to answer the following questions:

– "How well do the last queries of a consumer correspond to the expectations of the providers that were able to deal with?" – *Consumer-System Adequation* – ,
– "How well does a provider correspond to the consumer's expectations?" – *Provider-System Adequation* – , and
– "How well does the query allocation method perform w.r.t. a consumer or a provider?" – *Allocation Efficiency w.r.t. a Consumer* – and – *Allocation Efficiency w.r.t. a Provider* – , respectively.

The consumer's adequation to the system evaluates how much providers are interested in the queries of a consumer. Going back to our motivating example, we can say that $eWine$ is adequate to the system 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 $\delta_{csa}(c, q)$, is defined as the average of the intentions shown by set P_q towards its query q (Equation 3). Its values are between 0 and 1. Vector $\overrightarrow{PI}_q$ denotes the P_q's intentions to perform q.

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

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

Definition 7 Consumer-System Adequation

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

Having formally defined the *consumer-system adequation* global notion, the *query allocation efficiency w.r.t. a consumer* $c \in C$, $\delta_{ae}(c)$, is then defined as in Definition 8. Its values are between 0 and ∞.

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

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

On the one hand, 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. This evaluation is objective since it considers the consumer's adequation to the system in addition to the system's adequation to the consumer. On the other hand, the *query allocation efficiency w.r.t. a provider* objectively evaluates (since it also considers the provider's adequation to the system) how much the query allocation method strives to give interesting queries to providers. To define this latter global notion, as for a consumer, we need to know how much a provider is adequate to the system.

The adequation of a provider to the system allows to evaluate if consumers are interested in interacting with it. To illustrate the *Provider-System 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 in Definition 9. Its values are in the interval [0..1]. The closer the value to 1, the greater the adequation of a provider to the system.

Definition 9 Provider-System Adequation

$$\delta_{psa}(p) = \begin{cases} \dfrac{1}{||PQ_p^k||} \cdot \displaystyle\sum_{q \in PQ_p^k} \Big(\big(\overrightarrow{CI}_q[p] + 1 \big) \Big/ 2 \Big) \\ 0 \qquad\qquad\qquad if\ PQ_p^k = \emptyset \end{cases}$$

We then define the efficiency of the query allocation w.r.t. a provider $p \in P$, denoted by the function $\delta_{ae}(p)$, 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 Allocation Efficiency w.r.t. a Provider

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

If the efficiency value of the query allocation with regards to a participant (consumer or provider) is greater than 1, the query allocation method does a good job for the participant (considering its adequation to the system). If the value is smaller than 1, the efficiency of the query allocation is not good. In the case the value is 1, the query allocation method is neutral to a participant.

3.4 Discussion

The proposed model can be applied with different purposes. First, to evaluate how well a query allocation method satisfies the participants' expectations. 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 inadequate to them. To do so, one has to apply measures, which reflect a global behavior, over all concepts of the model: adequation, satisfaction, and allocation efficiency (see Section 4). Third, to design new self-adaptable query allocation methods that meet the participants' expectations in the long-run (see Section 5).

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.

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 and explain all the possibilities to compute participants' satisfaction is well beyond the scope of this paper. In fact, this could be the subject of a full paper. We thus report this to future work.

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.

4 System Measures

The measures we use are the same for consumers and providers, and can be used to evaluate the δ_{sca}, δ_{csa}, 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 3) are additive values and may take zero values, we utilize the arithmetic mean to obtain this representative number (Equation 4).

$$\mu(g, S) = \frac{1}{||S||} \cdot \sum_{s \in S} g(s) \tag{4}$$

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 [14] (defined in Equation 5). Its values are between 0 and 1.

$$f(g, S) = \frac{\big(\sum_{s \in S} g(s)\big)^2}{||S||\big(\sum_{s \in S} g(s)^2\big)} \tag{5}$$

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.

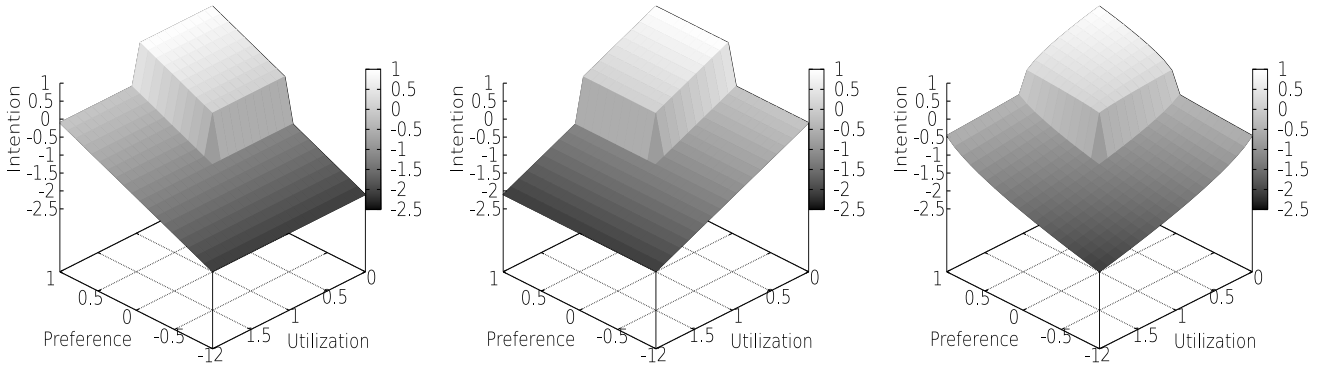
Finally, a traditional measure that reflects the ensured balance by a query allocation method is the *Min-Max* ratio. The Min-Max ratio σ is defined in Equation 6 (where $c_0 > 0$ is some fixed constant). Its values are between 0 and 1. The greater the balance value of a set S of g values, the better 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.

$$\sigma(g, S) = \frac{\min_{s \in S} g(s) + c_0}{\max_{s' \in S} g(s') + c_0} \tag{6}$$

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.

5 The SQLB Framework

We now present $SQLB$, a flexible framework for balancing queries while considering the participants' intentions. A salient feature of $SQLB$ is that it affords consumers the flexibility to trade their preferences for the providers' reputation (Section 5.1) and providers



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

Fig. 2 Tradeoff between *preference* and *utilization* for getting providers' *intention*.

the flexibility to trade their preferences for their utilization (Section 5.2). Then, a mediator allocates queries in accordance to the intentions and satisfaction of participants (Section 5.3). In this way, *SQLB* continuously adapts to changes in participants' expectations and workload. Without any loss of generality, participants may differently obtain their intentions.

5.1 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 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 paper.

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 11. 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 11 Consumer's Intention

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

Parameter $\epsilon > 0$, usually set to 1, prevents the consumer's intention from taking zero values when the consumer's preference or provider's reputation values are 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.

5.2 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 expectations. On the other hand, if a provider is dissatisfied, it does not pay so much attention to its utilization and focuses on its preferences so as to obtain queries that meet its expectations. 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 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 im-

plements 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 define the intention of a provider $p \in P_q$ to deal with a given query q as in Definition 12. Parameter $\epsilon > 0$, usually set to 1, prevents the intention of a provider from taking 0 values when its preference is equal to 1 whatever its utilization is.

Definition 12 Provider's Intention

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

Figure 2 illustrates the behavior that function pi takes for different provider's satisfaction values. We can observe in Figure 2(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(b)). In the case that a provider has a satisfaction of 0.5 (Figure 2(c)), we observe that its preferences and utilization have the same importance for it. Moreover, we can observe in Figure 2 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.

5.3 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, $SQLB$ allows the mediator to trade consumers' intentions for providers' intentions according to their satisfaction (Section 5.3.1). Furthermore, $SQLB$ 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 several parameters (Section 5.3.2). In Section 5.3.3, we describe the query allocation process and analyze, in Section 5.3.4, the number of messages that the mediator transfers over the network to perform q.

5.3.1 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.

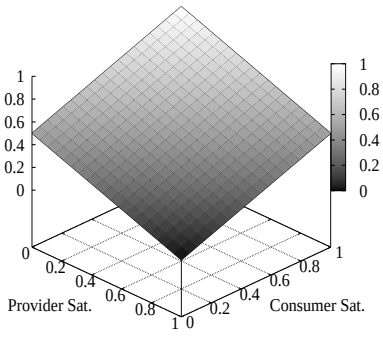


Fig. 3 The values that parameter ω can take.

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 13).

Definition 13 Provider's Score

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

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 7. 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.

$$\omega = \Big(\big(\delta_s(c) - \delta_s(p)\big) + 1\Big) \Big/ 2 \quad (7)$$

Figure 3 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.

5.3.2 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 short response times to consumers. To allow this, we assume that the mediator uses the K_nBest strategy that we proposed in [31]. K_nBest is inspired by the *two random choices (TRC)* paradigm [25,6]. 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[3]. 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 5.3.3. 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*).

Theorem 1 summarizes the K_nBest's properties that bound its behavior.

Theorem 1 *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 several 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 [25]) 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

[3] We can indifferently assume that k' and k_n values are predefined by the administrator or defined on the fly by the mediator.

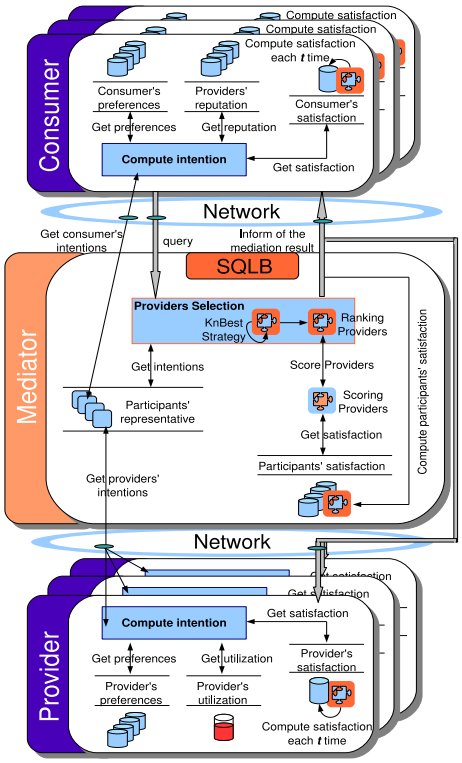


Fig. 4 SQLB system architecture.

$K_n Best$ 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 $K_n Best$ as in property *(iii)*.

In the rest of this paper, as we focus on heterogeneous distributed information systems, we assume, for the sake of simplicity, that k' is always equal to N (i.e. we discard the random selection phase).

5.3.3 Query allocation principle

We now describe how the mediator allocates queries. Figure 4 illustrates the general *SQLB* system architecture and Algorithm 1 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 1). 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

Algorithm 1: QueryAllocation

```
   Input  : q, k_n, P_q
   Output: All⃗c_q
1  begin
       // Consumer's intentions
2      fork ask for q.c's intentions;
       // Providers' intention
3      foreach p ∈ P_q do
4          fork ask for p's utilization and intention w.r.t. q ;
5      waituntil CI⃗_q, U⃗, and PI⃗_q be calculated or timeout;
       // qlb regulation
6      K_n ← select k_n less utilized providers from set P_q ;
       // Scoring and ranking providers
7      foreach p ∈ K_n do
8          compute p's score concerning CI⃗_q[p] & PI⃗_q[p] ;
9      rank set K_n of providers regarding scr_p(q), R⃗_q ;
       // Query Allocation
10     for i = 1 to min(n, k_n) do All⃗c[R⃗_q[i]] ← 1 ;
11     for j = min(n, k_n) + 1 to N do All⃗c[R⃗_q[j]] ← 0 ;
12 end
```

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 1 can be optimized, but our goal is to show the steps involved in the query allocation process.

5.3.4 Communication Cost

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

Theorem 2 *The total number of transferred messages by the mediator 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 $mssg_0 + mssg_1 + mssg_2 = 3(N + 1) + n$. □

We can further reduce the number of messages by using participants' *representatives* [18] or by introducing again the random selection phase (see Section 5.3.2). However, the problem of reducing communication cost is orthogonal to the problem we address in this paper.

5.4 Discussion

We pointed out in Sections 5.1 and 5.2 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 the 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, $SQLB$ is quite general, self-adaptable to the interests of participants, and adaptable to the application. This allows $SQLB$ 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.

We assumed a mono-mediator system, i.e. a system that contains only one mediator to allocate queries. Clearly, a mediator may become a single point of failure and a performance bottleneck and thus one may desire to have more than one mediator in the system to allocate queries. In this case, $SQLB$ 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. Hence, when allocating a query, a mediator should keep informed all other mediators of the mediation result to update participants' satisfaction. This tends to significantly increase the network traffic. A way to avoid such a traffic overhead between mediators is that providers express their interest for queries through "monetary" bids so that 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 context, we are currently doing some work to show how to adapt $SQLB$ to use virtual money and demonstrate that such an economic version can easily scale up to several mediators. However, a further discussion on this subject is not the focus of this paper (see [33,34]).

Finally, in large-scale distributed information systems participants may fail, usually because of network failures. Nevertheless, we do not deal with fault-tolerance issues in this paper since the main focus of this work is to evaluate $SQLB$ from a satisfaction point of view. To make $SQLB$ fault-tolerant is the focus of one of our forthcoming work. Generally speaking, the idea is to enable a mediator to set, in a predictive way, the number of replicas that should be created for an incoming query based on participants' satisfaction.

6 Experimental Validation

Our experimental validation has three main objectives: *(i)* to evaluate how well query allocation methods operate, *(ii)* to analyze if $SQLB$ 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 the initial criteria, and *(iii)* 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 as well as the computed measures. Second,

we evaluate the impact of participants' *autonomy* on performance. Third, we evaluate the self-adaptability of $SQLB$ to participants' expectations. Finally, we analyze the effects of varying the values of k_n parameter, i.e. we evaluate the $SQLB$'s adaptability to different kinds of applications.

6.1 Setup

We built a Java-based simulator and simulate a *mono-mediator* distributed information system, which follows the mediation system architecture presented in [18]. For all the query allocation methods we tested, the following configuration (Table 2) 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.* [29] 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 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 consumers and providers compute their intentions as defined in Sections 5.1 and 5.2, 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 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-provider adequation* notion we defined in Section 3.2.1. 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 [37]), but this is beyond the scope of this paper and orthogonal to the problem we address here. Without any loss of generality, the participants' expectations, in the long run, are static in our simulations. We assume this to evaluate the query allocation methods in a long-run trend, but our model allows expectations to be dynamic.

Table 2 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

We set the providers' capacity heterogeneity following the results presented in [39]. 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 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 consider in our experiments, without any loss of generality, that providers offer computational services to consumers. Thus, inspired from [12], we assume that providers compute their utilization as in Equation 8. Set Q_p denotes the set of queries that have been allocated to p but have not already been treated, i.e. the pending queries at p. Function $cost_p(q)$ represents the computational resources that a query $q \in Q_p$ consumes at provider p.

$$\mathcal{U}_t(p) = \frac{\sum_{q \in Q_p} cost_p(q)}{cap(p)} \tag{8}$$

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 [22]. Since our main focus is to study the way in which queries are allocated, we do not consider in this paper 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.

6.2 Baseline Methods

We briefly justify, in this section, the choice of the algorithms to which we compare $SQLB$.

6.2.1 Capacity based method

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* [12,6] methods since, unlike *Capacity based*, they inherently assume that providers and queries are homogeneous. In *Capacity based* [24,35,40] 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.

6.2.2 Economical method

Economical models have been shown to provide efficient query allocation in heterogeneous systems [9,10,42]. Mariposa [42] 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 $SQLB$. 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.

6.3 Results

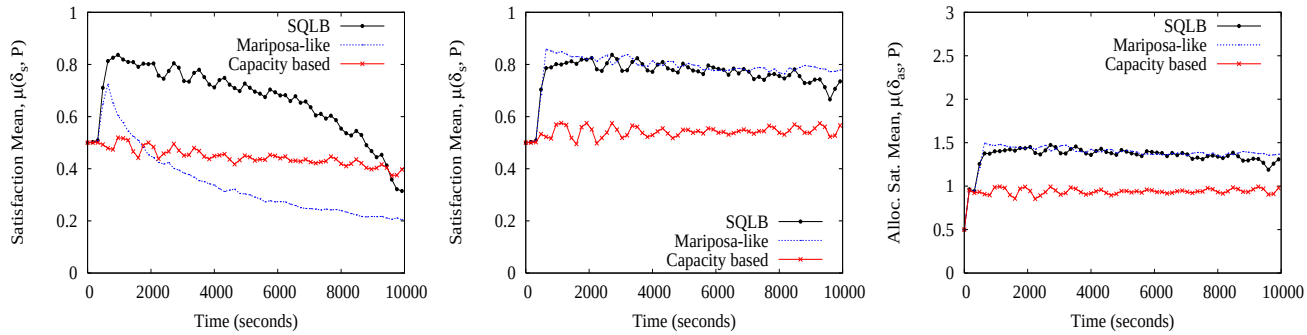
We start, in Section 6.3.1, by evaluating the quality of the three query allocation methods with regards to satisfaction and *qlb*. In Section 6.3.2, we evaluate how well these methods deal with the possible participants' departure by *dissatisfaction*, *starvation*, or *overutilization*. Then, in Section 6.3.3, we show the self-adaptability of $SQLB$ to participants' expectations. 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 6.3.4, we study the adaptability of $SQLB$ to the kind of application by varying parameter k_n.

6.3.1 Quality results without autonomy

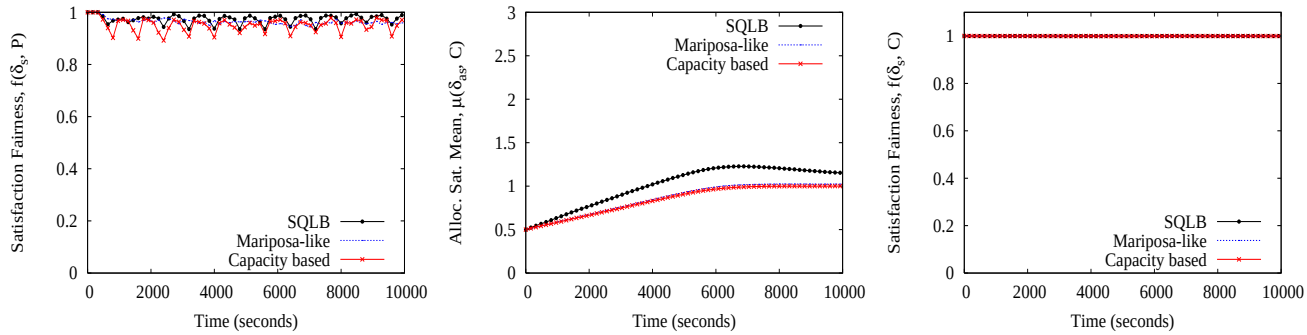
If participants are autonomous, they may leave the system by dissatisfaction, starvation, or overutilization. Nevertheless, the choice of such departure's thresholds is very subjective and may depend on several external factors. Thus, for these first experiments, we consider *captive* participants, i.e. they are not allowed to leave the system. To measure the quality of the three methods, we apply the measures defined in Section 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 5(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 $SQLB$ 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, $SQLB$ 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.

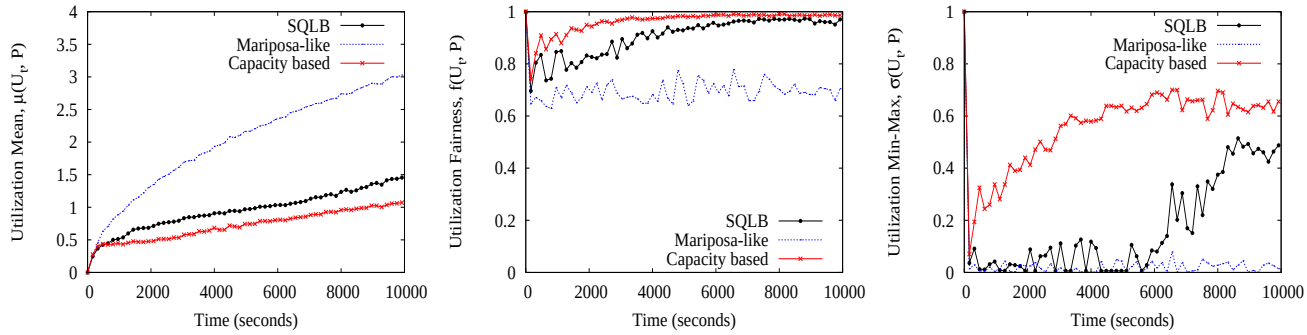
Nonetheless, this does not reflect what providers really feel with respect to their preferences. To show this, we need to measure the mean ensured by the three methods concerning the providers' satisfaction based on their preferences. Although we can measure



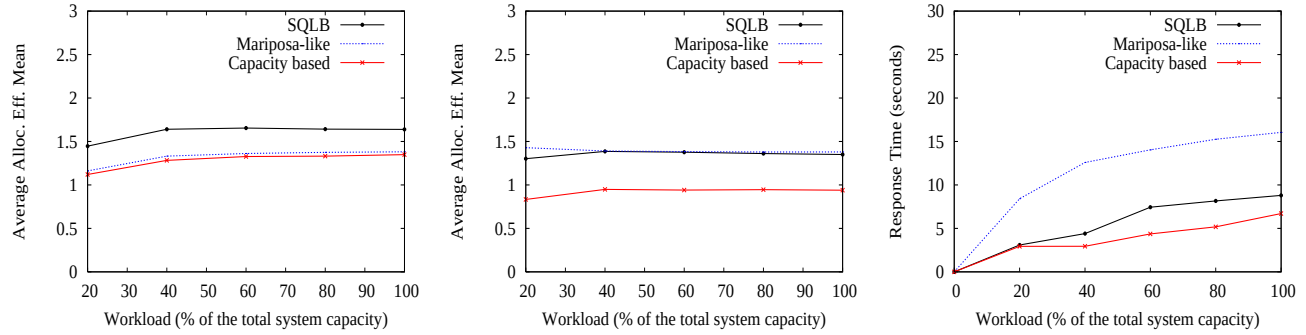
(a) Providers' satisfaction mean based on intentions. (b) Providers' satisfaction mean based on preferences. (c) Providers' allocation satisfaction.



(d) Provider satisfaction fairness. (e) Consumers' allocation satisfaction. (f) Consumer satisfaction fairness.



(g) Query load mean. (h) Query load fairness. (i) Query load min-max.



(j) Allocation efficiency w.r.t. Consumers. (k) Allocation efficiency w.r.t. Providers. (l) Response times.

Fig. 5 Results with *captive* participants. (a)-(i): quality results for a *workload* range from 30 to 100% of the total system capacity, (j)-(k): allocation efficiency results for different workloads, and (l): ensured response times for different workloads.

such a satisfaction in our simulations, this is not always possible since such preferences are usually considered as private. Figure 5(b) shows the results of these measurements. We observe that $SQLB$ 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 $SQLB$. 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 $SQLB$ 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 5(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 5(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 5(e) illustrates the allocation satisfaction mean concerning the consumers' intentions. We observe that while $SQLB$ 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 $SQLB$ does not. The $SQLB$'s mean decreases for high workloads because of providers. Remember that providers' satisfaction decrease because they take care of their utilization. So, $SQLB$ pays more attention to providers' satisfaction than to consumers' satisfaction. Nonetheless, consumers are never penalized! Conversely to providers, we can observe in Figure 5(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 $SQLB$ and *Mariposa-like* (see Figure 5(g)). We can observe that $SQLB$ performs well, while *Mariposa-like* has serious problems to balance queries. Thus, *Mariposa-like* may lose providers by starvation or overutilization reasons. Figure 5(h) shows that $SQLB$ has some difficulties to be fair (w.r.t. *qlb*) for workloads under 40%. In contrast, when the workload increases, $SQLB$ pays more attention to *qlb* and becomes fairer. This is clearly illustrated in Figure 5(i), which shows the results about the utilization Min-Max. The reason that $SQLB$ 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 $SQLB$ to the variations in the workloads.

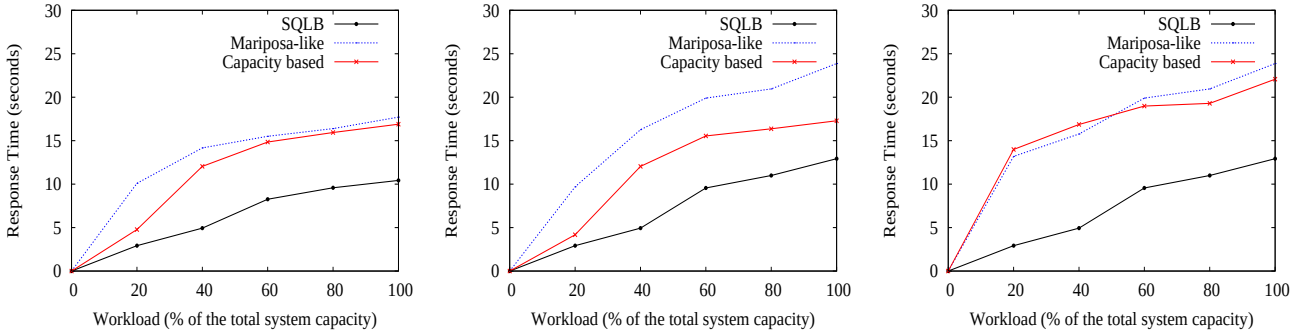
Figures 5(j) and 5(k) illustrate the allocation efficiency with respect to consumers and providers for different workloads. These results clearly illustrate the superiority of $SQLB$ over *Capacity based* and *Mariposa-like* since we can observe, *(i)* on the one hand, that $SQLB$ significantly outperforms *Capacity based* in both cases; and *(ii)* on the other hand, that $SQLB$ and *Mariposa-like* have the same allocation efficiency w.r.t. providers, but $SQLB$ significantly outperforms *Mariposa-like* in the consumers' case, which demonstrates the equity at both levels of $SQLB$.

Finally, Figure 5(l) 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 $SQLB$ 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 $SQLB$'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, $SQLB$ ensures quite good response times and pays attention to the quality of results and queries that consumers and providers get from the system, respectively.

6.3.2 Dealing with autonomy

To validate our measurements and intuitions of Section 6.3.1, 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. We evaluate the ensured response times by the three methods in au-



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

Fig. 6 Impact on performance of providers' departures.

tonomous environments and compare it with those of the captive environments (see Figure 5(l)).

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, we assume that participants support high degrees of dissatisfaction, 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 *(i)* by dissatisfaction, if its satisfaction value is 0.15 smaller than its adequation, *(ii)* by starvation, if its utilization is smaller than 20% of its optimal utilization, and *(iii)* by overutilization, if its utilization is greater than 220% of its optimal utilization. With a workload of 80% of the total system capacity, the optimal utilization of a provider is 0.8.

We ran a first series of experiments with different workloads where providers are allowed to leave the system by dissatisfaction only (see Figure 6(a)). We can see that our approach outperforms both *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 5(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 6(b) illustrates these results. We observe again that *SQLB* 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 *SQLB* 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.

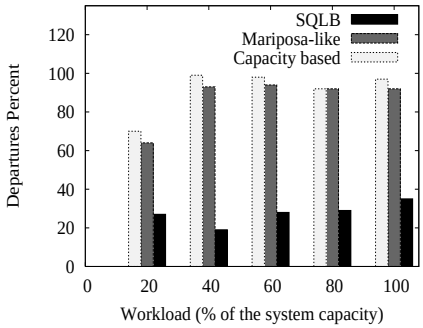


Fig. 7 Providers' departures.

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 6(c). We observe that while *SQLB* and *Mariposa-like* degrade their performance only by a factor of 1.4 in average (w.r.t. Figure 5(l)), *Capacity based* does it by a factor of 3.5! Figure 7 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 *SQLB* only loses 28% of providers in average! This demonstrates the high efficiency of *SQLB* in autonomous environments.

Table 3 Provider's departures reasons for a workload of 80% of the total system capacity.

		SQLB				*Capacity based*				*Mariposa-like*			
		low	med	high	total	low	med	high	total	low	med	high	total
	Cons. Interest to Prov.	1%	5%	13%		5%	16%	31%		1%	7%	11%	
Dissat.	Providers' Adequation	2%	9%	8%	19%	3%	34%	15%	52%	0%	15%	4%	19%
	Providers' Capacity	13%	6%	0%		13%	30%	9%		5%	12%	2%	
		low	med	high	total	low	med	high	total	low	med	high	total
	Cons. Interest to Prov.	0%	0%	4%		0%	0%	0%		0%	2%	6%	
Starv.	Providers' Adequation	4%	0%	0%	4%	0%	0%	0%	0%	3%	3%	2%	8%
	Providers' Capacity	2%	2%	0%		0%	0%	0%		3%	5%	0%	
		low	med	high	total	low	med	high	total	low	med	high	total
	Cons. Interest to Prov.	0%	0%	6%		0%	0%	38%		0%	0%	65%	
Overuti.	Providers' Adequation	0%	3%	3%	6%	3%	8%	27%	38%	1%	15%	49%	65%
	Providers' Capacity	1%	4%	1%		0%	18%	20%		0%	30%	35%	

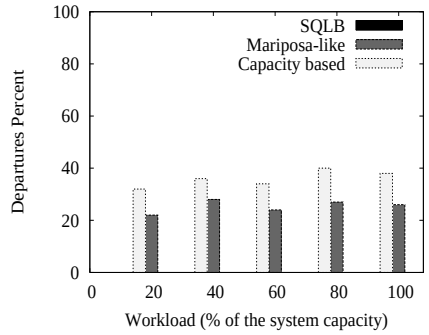


Fig. 8 Consumers' departures.

We show, in Table 3, an analysis of providers' reasons to leave the system when the workload is 80%. We observe that, as predicted in Section 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 *SQLB*, providers leave the system by dissatisfaction, but such providers are mainly those that are *low*-capacity. In fact, we can see that *SQLB* mainly maintains the *high*-interest, *high*-adaptation, and *high*-capacity providers in the system.

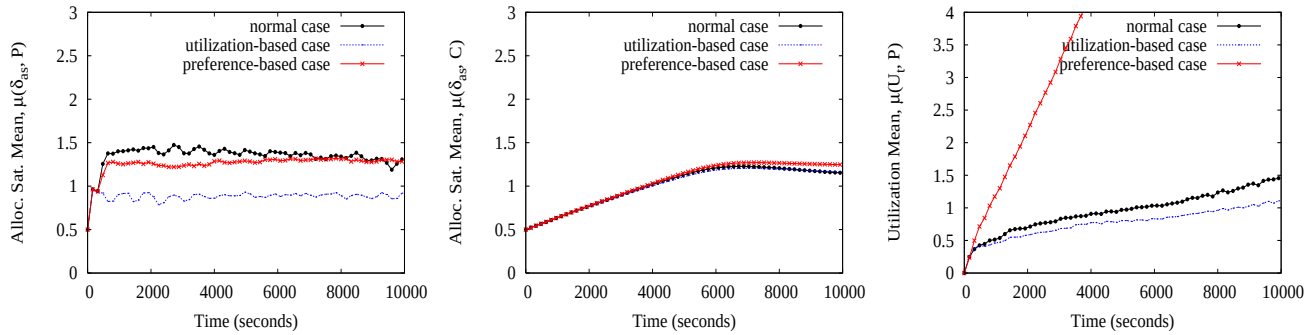
Finally, Figure 8 shows the consumers' departure by dissatisfaction with these three methods. Again, *SQLB* is a clear winner with no consumer's departures! Note that, the consumer's departures have also a direct impact on performance since the less the incoming queries, the less the chances for satisfying providers.

6.3.3 Adaptability to participants' interests

Our objective in this section is to study how well *SQLB* adapts to different participants' *intentions*. With this in mind, we consider again captive environments such as in Section 6.3.1. For simplicity, we evaluate in this paper 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 *SQLB* 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 6.3.1.

Figure 9 shows the results of these experiments with a workload range from 30 to 100% of the total system capacity. We can observe in Figures 9(a) and 9(b) that the results are strongly related to the participants' expectations. We can observe in Figure 9(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 *SQLB* 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 9(b), we observe that consumers have the same degree of satisfaction in the three cases, 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,



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

Fig. 9 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*).

for high workloads, the query allocation pays more attention to providers and thus the consumers' satisfaction decreases in these both cases.

Now, concerning *qlb*, *SQLB* performs well in the *utilization-based* and *normal cases* while, in the *preference-based case*, *SQLB* significantly degrades the providers' utilization because providers have no consideration for *qlb*. On the other side, observe that, in the *utilization-based case*, *SQLB* follows the behavior of the *Capacity based* approach (see Figures 9(a) and 9(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 *SQLB* 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 *SQLB* 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, *SQLB* allows them to reach their expectations.

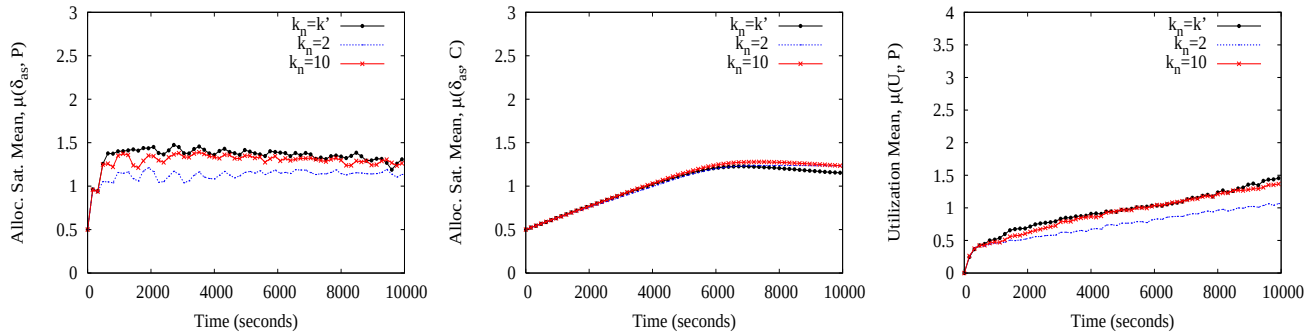
6.3.4 Query load balance control

We finally discuss how to adapt *SQLB* to different applications by varying parameter k_n. 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, for the first kind of application, the mediator sets parameter $k_n = 2$ and it sets $k_n = 10$ for the second one.

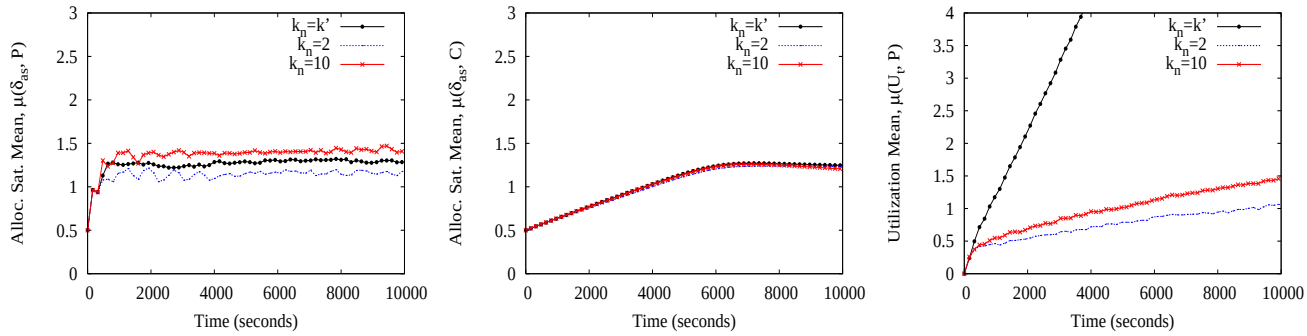
To clearly see the impact of parameter k_n, we compare both results (i.e. when $k_n = 2$ and $k_n = 10$) with the case where the mediator has no control to regulate the system (i.e. when $k_n = k'$). Notice that the previous sections assumed that $k_n = k'$, thus the results of *SQLB* in the *normal* and *preference-based cases* that we present in this section are the same as those we presented in Sections 6.3.1 and 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 10(a) and 11(a) that for all three different k_n values and both *normal* and *preference-based cases*, providers are generally satisfied with the job done by our approach, 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 10(a)), but this is not the case for providers in the *preference-based case* when $k_n = k'$ and $k_n = 10$ (see Figure 11(a)). 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



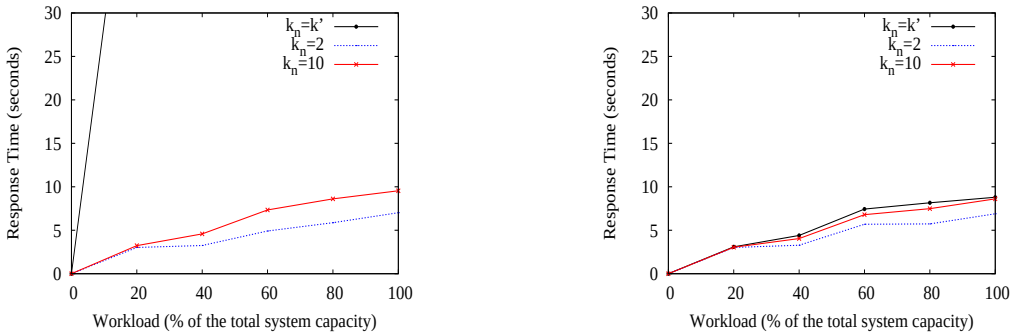
(a) Providers' allocation satisfaction mean. (b) Consumers' allocation satisfaction mean. (c) Query load mean.

Fig. 10 Quality results for a *workload* range from 30 to 100% of the total system capacity when participants are *captive* and providers compute their intentions based on their preferences and utilization (the *normal case*).



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

Fig. 11 Quality results for a *workload* range from 30 to 100% of the total system capacity when participants are *captive* and 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*).

Fig. 12 Performance results with captive participants.

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, *SQLB* performs much better than the *Capacity based* approach because it satisfies both consumers and providers (see Figures 5(c) and 5(e) for *Capacity based*). In fact, we can observe in Figures 10(b) and 11(b) 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 10(c) and 11(c) 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 $SQLB$ 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 10(c)). This is not the case for the *preference-based case*, when $k_n = k'$, even if providers' preferences are the same (see Figure 11(c)). But, by setting small k_n values, $SQLB$ can ensure short 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 12(a) and 12(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 $SQLB$ to applications where the mediator needs to regulate the system w.r.t. a given predefined function (*qlb* in this work) without mattering how participants compute their intentions. With high k_n values, one can adapt $SQLB$ to applications where the mediator has to meet the participants' expectations.

7 Related Work

The query allocation problem, which appears as a subproblem of query processing [16], 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 system 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 remainder of this section, we discuss five main approaches related to our query allocation framework: economics, data mediators, multi-agent systems, load balancing approaches, and web services. Notice that the scope of this paper goes well beyond related work by characterizing the participants' expectations in the long-run, proposing measures to analyze them and new algorithms to exploit them.

7.1 Economics

Economics is a social science concerned mainly with description and analysis of the production, distribution, and consumption of goods and services. It is subdivided into microeconomics, which studies how individuals make decisions to allocate limited resources, and macroeconomics, which studies aggregated indicators to understand how the whole economy functions. We are interested in the former in this work. In the following, we first discuss some microeconomic properties that are closely related to the satisfaction notion we proposed in this paper. Then, we present some microeconomics-based approaches to allocate queries.

7.1.1 Theory

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 [21]. 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. In other words, as the satisfaction notion, the total utility is an abstract concept that measures the happiness or gratification of participants by consuming or performing queries. Furthermore, the total 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, total utility is clearly related to the notion of satisfaction we presented in this paper, 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 total utility is neither bounded nor normalized. Therefore, one can easily compare the satisfaction of participants. Second, while total 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, total utility is generally reduced to monetary

concerns only, which is not the case for the satisfaction notion.

Moreover, most economic properties (e.g. pareto-optimality, nash equilibrium, and individual rationality) focus on only one interaction. As a result, most of the economic approaches [9], 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 happiness of participants in several interactions, i.e. in the long-run. Also, in the field of distributed rational decision making [38], 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.

7.1.2 Approaches

Economic approaches can claim to take into account the participants' intentions and have been shown to provide efficient query allocation in heterogeneous systems [10, 42]. A survey of economic models for various aspects of distributed system is presented in [9].

Mariposa [42] 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 [28], 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 [18], 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 $SQLB$, by computing bids with respect to intentions [33,34].

7.2 Data Mediators

Over the last years, data mediator systems [44] 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 [13] and Information Manifold [19]. In data mediator systems, the mediator allocates queries to providers and integrates results for consumers, much like distributed database systems [27]. Nevertheless, data mediators require some global information such as global schemas [43], which is difficult to maintain in dynamic systems because source schemas change frequently. $SQLB$ does not require any global knowledge, but it does not address the integration problem.

7.3 Multi-agent Systems

In multi-agent systems, the *Contract Net Protocol (CNP)* [41] is often mentioned as a way to allocate queries. However, *CNP* is a simple protocol and there is no control to regulate the system. Besides, it is generally assumed a rather small number of participants and a detailed description of the conditions of execution, which is not our case. Several approaches of middle-agents have been defined [8,17,26] and a survey can be found in [15]. A classical goal of middle-agents is to find the providers that are able to deal with a given query by matching providers' capabilities advertisements with

the given query. All these works are efficient but the number of selected providers may remain too large.

Thus, some works have investigated the possibility of reducing the list of selected providers. For example, *Z. Zhang and C. Zhang* [45] propose to perform classical matchmaking and then refine the result list of providers by considering the providers' quality. Nonetheless, the participants' intentions are not considered by the providers selection procedure, which does not allow a participant to have an active participation in the selection process.

D. Bernstein et al. [7] 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.

7.4 Load Balancing Approach

In the context of large-scale and heterogeneous distributed systems, most of the work on query allocation has mainly dealt with the problem of balancing queries among providers. There exist two well-known approaches to balance queries across providers: *Load Based* and *Capacity based* methods. *Load Based* methods [12,6] are not suitable for heterogeneous systems since they inherently assume that providers and queries are homogeneous. *Capacity based* methods [24, 35,40] allocate queries in accordance to the providers' utilization, i.e. they allocate each incoming query to providers with the most available capacity. Nevertheless, all these approaches have no consideration for the participants' intentions.

In [30] and [31], we propose a method and strategy, respectively, to balance queries among providers by considering providers' intentions and satisfaction, but no notion of intentions nor satisfaction of consumers is considered.

7.5 Web Services

To locate and select services, web services (the providers) have to describe properly all their proposed services [5]. Once services have been properly described, these descriptions are made available, via a service directory (the registry), to those interested in using them (the consumers). These directories can be hosted and managed by a trusted entity (centralized approach) or each provider can host and manage them (peer-to-peer approach). When a consumer has located the providers providing the service it desires, it then selects the provider that it wants. Then, the consumer selects the provider with respect to its interests, e.g. with the highest score among the providers in the registry. However, providers cannot express their intentions to perform queries and are considered as captive values.

8 Conclusion

We considered distributed information systems where participants are *autonomous* to leave the system at will. In this context, it is crucial to consider the participants' *intentions* to allocate and perform queries so that their expectations, response times, and system capacity are ensured. We presented, in this paper, a general and complete solution to allocate queries among providers by considering the participants' intentions and *query load balancing* (*qlb*). Our work carried out fourth main contributions.

First, we characterized the participants' expectations 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 design and evaluation of new query allocation methods for these environments. The proposed model is general, and thus, can be used for any distributed systems architecture.

Second, we proposed three different measures to evaluate the quality of *qlb* 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.

We proved that using these proposed measures together, one can predict possible consumers' and providers' departures from the system.

Third, we presented the $SQLB$ framework for balancing queries in these environments. The originality of $SQLB$ is to perform all query demand while satisfying participants' expectations. $SQLB$ strongly differs from the related work in several ways:

- It allows providers to trade their preferences for their utilization while keeping their strategic information private.

- It affords consumers the flexibility to trade their preferences for providers' reputation.
- It allows trading consumers' intentions for providers' intentions.
- It strives to balance queries at runtime via the participants' satisfaction, thus reducing *starvation*.
- It affords the mediator to regulate the system with respect to some predefined function and can adapt the query allocation process to the kind 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 expectations in the system whether they correctly work out their intentions.

Fourth, we evaluated and compared $SQLB$ 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, $SQLB$ significantly outperforms both baseline methods. We observed that participants are, in general, very satisfied with $SQLB$ 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, $SQLB$ 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), $SQLB$ has no consumer's departures! We showed the self-adaptability of $SQLB$ to the expectations and satisfaction of participants. We also discussed its adaptability to different kinds of applications. All these results demonstrate that $SQLB$ can scale up with autonomous participants, while *Capacity based* and *Mariposa-like* cannot.

As future work, we plan to address fault-tolerant issues so that $SQLB$ ensures good system performance even in the presence of participants' failures. We also plan to study the impact, on performance and participants' satisfaction, of introducing the random providers selection phase (discussed in Section 5.3.2). In our experimentations, we also noted that self-organizing phenomenons occur. When the system is composed of several mediators and the participants have divergent interests, the participants that share the same interests gather on the same mediator. We wish to explore this phenomenon that we not observe with the other approaches. Finally, the problem addressed in this paper as that addressed by economic approaches is to regulate a distributed information system while satisfying participants. We then plan to develop an economic version of $SQLB$, based on [18], to scale up to several mediators and to analyze in detail the contributions of the use of money for allocating queries.

References

1. The eBay System. http://business.ebay.com.
2. The Freenet Project. http://freenetproject.org.
3. The Freightquote System. http://www.Freightquote.com.
4. The Gnutella Project. http://www.gnutella.com.
5. Alonso G., Casati F., Kuno H., and Machiraju V.: Web Services: Concepts, Architecture, and Applications. Springer-Verlag, (2004).
6. Azar Y., Broder A. Z., Karlin A. R., and Upfal E.: Balanced Allocations. SIAM Journal on Computing. 29(1), 180–200 (1999).
7. Bernstein D. S., Feng Z., Levin B. N., and Zilberstein S.: Adaptive Peer Selection. In Proceedings of the International Workshop on Peer-to-Peer Systems, IPTPS, (2003).
8. Decker K., Sycara K., and Williamson M.: Middle-Agents for the Internet. In Proceedings of the International Joint Conferences on Artificial Intelligence, IJCAI, (1997).
9. Ferguson D., Nikolaou C., Sairamesh J., and Yemini Y.: Economic Models for Allocating Resources in Computer Systems. In: S. H. Clearwater (ed) Market-Based Control: A Paradigm for Distributed Resource Allocation, World Scientific, (1996).
10. Ferguson D., Yemini Y., and Nikolaou C.: Microeconomic Algorithms for Load Balancing in Distributed Computer Systems. In Proceedings of the International Conference on Distributed Computing Systems, ICDCS, (1988).
11. Fong T., Fowler D., and Swatman P.: Success and Failure Factors for Implementing Effective Electronic Markets. Journal of Electronic Commerce and Business Media. 8(1), 45–47 (1998).
12. Ganesan P., Bawa M., and Garcia-Molina H.: Online Balancing of Range-Partitioned Data with Applications to Peer-to-Peer Systems. In Proceedings of the Very Large Data Bases Conference, VLDB, (2004).
13. Garcia-Molina H., Papakonstantinou Y., Quass D., Rajaraman A., Sagiv Y., Ullman J. D., Vassalos V., and Widom J.: The TSIMMIS Approach to Mediation: Data Models and Languages. Journal of Intelligent Information Systems. 8(2), 117–132 (1997).
14. Jain R. K., Chiu D.-H., and Hawe W. R.: A Quantitive Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems. Digital Equipment Corporation, Technical Report. DEC-TR-301 (1984).
15. Klusch M. and Sycara K.: Brokering and Matchmaking for Coordination of Agent Societies: A Survey. In: Coordination of Internet Agents: Models, Technologies, and Aplications, Springer-Verlag, (2001).
16. Kossmann D.: The State of the Art in Distributed Query Processing. ACM Computing Surveys. 32(4), 422–46 (2000).
17. Kuokka D. and Harada L.: Matchmaking for Information Agents. In Proceedings of the International Joint Conferences on Artificial Intelligence, IJCAI, (1995).
18. Lamarre P., Cazalens S., Lemp S., and Valduriez P.: A Flexible Mediation Process for Large Distributed Information Systems. In Proceedings of the Cooperative Information Systems Conference, CoopIS, (2004).

19. Levy A. Y., Rajaraman A., and Ordille J.J.: Querying Heterogeneous Information sources Using Source Descriptions. In Proceedings of the Very Large Data Bases Conference, VLDB, (1996).
20. Li L. and Horrocks I.: A Software Framework for Matchmaking Based on Semantic Web Technology. In Proceedings of the World Wide Web Conference, WWW, (2003).
21. Mas-Colell A., Whinston M., and Green J.: Microeconomic Theory. Oxford University Press, (1995).
22. Markatos E. P.: 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 (2002).
23. Miller R.: Special Issue on Integration Management of the IEEE Data Engineering Bulletin. 25(3), (2002).
24. Mirchandaney R., Towsley D. F., and Stankovic J. A.: Adaptive Load Sharing in Heterogeneous Distributed Systems. Journal of Parallel and Distributed Computing, JPDC. 9(4), 331–346 (1990).
25. Mitzenmacher M.: The Power of Two Choices in Randomized Load Balancing. PhD. Thesis, UC Berkeley (1996).
26. Nodine M. H., Bohrer W., and Ngu A. H.: Semantic Brokering over Dynamic Heterogeneous Data Sources in InfoSleuth. In Proceedings of the International Conference on Data Engineering, ICDE, (1999).
27. Özsu T. and Valduriez P.: Principles of Distributed Database Systems, Second Edition. Prentice-Hall, (1999).
28. Pentaris F. and Ioannidis Y.: Query Optimization in Distributed Networks of Autonomous Database Systems. ACM Transactions on Database Systems, TODS. 31(2), 537–583 (2006).
29. Pieper S., Paul J., and Schulte M.: A New Era of Performance Evaluation. IEEE Computer. 40(9), 23–30 (2007).
30. Quiané-Ruiz J.-A., Lamarre P., and Valduriez P.: Satisfaction Based Query Load Balancing. In Proceedings of the Cooperative Information Systems Conference, CoopIS, (2006).
31. Quiané-Ruiz J.-A., Lamarre P., and Valduriez P.: K_nBest - A Balanced Request Allocation Method for Distributed Information Systems. In Proceedings of the Database Systems for Advanced Applications Conference, DASFAA, (2007).
32. Quiané-Ruiz J.-A., Lamarre P., and Valduriez P.: SQLB: A Query Allocation Framework for Autonomous Consumers and Providers. In Proceedings of the Very Large Data Bases Conference, VLDB, (2007).
33. Quiané-Ruiz J.-A., Lamarre P., Sylvie Cazalens, and Valduriez P.: Satisfaction Balanced Mediation. In Proceedings of the Conference on Information and Knowledge Management, CIKM, (2007).
34. Quiané-Ruiz J.-A., Lamarre P., Sylvie Cazalens, and Valduriez P.: Managing Virtual Money for Satisfaction and Scale Up in P2P Systems. In Proceedings of the EDBT Workshop on Data Management in Peer-to-Peer Systems, DAMAP, (2008).
35. Rahm E. and Marek R.: Dynamic Multi-Resource Load Balancing in Parallel Database Systems. In Proceedings of the Very Large Data Bases Conference, VLDB, (1995)
36. Roth M. and Schwarz P.: Don't Scrap It! Wrap It! A Wrapper Architecture for Legacy Data Sources. In Proceedings of the Very Large Data Bases Conference, VLDB, (1997).
37. Sah A., Blow J., and Dennis B.: An Introduction to the Rush Language. In Proceedings of the TCL Workshop (1994).
38. Sandholm T. W.: Distributed Rational Decision Making. In: G. Weiss (ed) Multiagent Systems, a Modern Approach to Distributed Artificial Intelligence, The MIT Press, (2001).
39. Saroiu S., Gummadi P. K., and Gribble S. D.: A Measurement Study of Peer-to-Peer File Sharing Systems. In Proceedings of the Multimedia Computing and Networking Conference, MMCN, (2002).
40. Shivaratri N. G., Krueger P., and Singhal M.: Load Distributing for Locally Distributed Systems. IEEE Computer. 25(12), 33–44 (1992).
41. Smith R. G.: The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. IEEE Transactions on Computers. C-29(12), 1104–1113 (1981).
42. Stonebraker M., Aoki P., Litwin W., Pfeffer A., Sah A., Sidell J., Staelin C., and Yu A.: Mariposa: A Wide-Area Distributed Database System. Journal on Very Large Data Bases, VLDBJ. 5(1), 48–63 (1996).
43. Tomasic A., Raschid L., and Valduriez P.: Scaling Access to Heterogeneous Data Sources with DISCO. IEEE Transanctions on Knowledge and Data Engineering, TKDE. 10(5), 808–823 (1998).
44. Wiederhold G.: Mediators in the Architecture of Future Information Systems. IEEE Computer. 25(3), 38–49 (1992).
45. Zhang Z. and Zhang C.: An Improvement to Matchmaking Algorithms for Middle Agents. In Proceedings of the Autonomous Agents and Multiagent Systems Conference, AAMAS, (2002).