Managing Virtual Money for Satisfaction and Scale Up in P2P Systems

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

To cite this version:

HAL Id: hal-00374997
https://hal.archives-ouvertes.fr/hal-00374997
Submitted on 10 Apr 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers. L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Managing Virtual Money for Satisfaction and Scale Up in P2P Systems

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

ABSTRACT

In peer-to-peer data management systems query allocation is a critical issue for the good operation of the system. This task is challenging because participants may prefer to perform some queries than others. Microeconomic mechanisms aim at dealing with this, but, to the best of our knowledge, none of them has ever proposed experimental validations that, beyond query load or response time, use measures that are outside the microeconomic scope. The contribution of this paper is twofold. We present a virtual money-based query allocation process that is suitable for large-scale super peer systems. We compare a non microeconomic mediation with micro-economic ones from a satisfaction point of view. The experimental results show that the providers’ invoice phase is as much important as the providers’ selection phase for a virtual money-based mediation.

1. INTRODUCTION

We consider a peer-to-peer (P2P) data management system with mediators that enable consumers to access distributed information providers through queries [6], such as super peer systems. Consumers and providers (for clarity, we refer to both together as participants) are autonomous in the sense that they are free to enter and leave the system at will and do not depend on anyone to do so. The main function of the mediator is to allocate each incoming query to the providers that can answer it. In this context, it is crucial to ensure good system performance, e.g. high throughput or low response times. Nevertheless, participants may get satisfied by the mediator because of certain expectations that are not only performance-related. These participants’ expectations are clearly illustrated by Google AdWords [1], which proposes relevant commercial providers to consumers according to some keywords of their interest.

To capture this intuition, we have proposed a definition of satisfaction [9] of the participants. The satisfaction of a provider after several query allocations is evaluated considering its intentions w.r.t. the queries that it has been allocated. Satisfaction is defined as a long run notion, to capture the fact that the provider is “globally” satisfied even if it is sometimes allocated queries it does not intend to treat. We have proposed a same kind of definition to characterize the consumers’ satisfaction. We have also proposed a mediation process, called SQLB mediation, which allocates a query considering the participants’ intentions with respect to the query and their current satisfaction. We implemented the SQLB mediation within a mediator and run experiments with a single mediator and many participants. The results show that SQLB has very good system performance in maintaining a good satisfaction for participants (as far as they are adequate to the system) and low response times.

However, directly considering the participants’ intentions and current satisfaction in the mediation process may have some drawbacks. First, the process suits only in cases where the participants agree to show their intentions with respect to the queries. Second, scaling up to several mediators requires a high message traffic. Indeed, the mediator can no longer calculate the providers’ satisfactions itself, because, in case of a provider using several mediators, its satisfaction results from the queries obtained with the different mediators. Thus either each provider has to inform each mediator of its current satisfaction or the mediators have to exchange information about the providers’ satisfaction. In both cases, network traffic increases significantly.

A way to regulate query allocation with less information exchange is to use virtual money. This latter expression means that this money is totally disconnected from the real money we use in current life. When using virtual money, the providers no longer show their intentions directly, but rather bid on the incoming queries. Thus, the mediation process considers the providers’ bids to allocate the queries. In case of several mediators, the same virtual money can be used by the different mediators. It can be considered as the common denominator of the different mediation processes, which use the bids, and possibly other elements to decide the allocation. On the other side, a participant’s satisfaction still depends on the work of the different mediators and thus it is still important to satisfy participants.

In this context, several works use microeconomic mechanisms to allocate resources in distributed systems. However, to the best of our knowledge, none of them has ever proposed experimental validations, which, beyond query load
or response time, use measures that are outside the microeconomic scope. Thus, the main contribution of this paper is twofold. First, we present a virtual money-based query allocation process that (i) is participant-centric, (ii) behaves as good as SQLB, and (iii) scales up to several mediators. Second, from a methodological point of view, it is interesting to compare a non microeconomic mediation (namely SQLB) with microeconomic ones (namely FPSB, VMbQA, and VMbQA\*), using satisfaction as a "money independent" evaluation measure in both cases.

This paper is organized as follows. In Section 2, we give some preliminary concepts and definitions. In Section 3, we discuss the use of virtual money when designing a mediation process. We present in Section 4 a virtual money-based query allocation process and improve this mediation process in Section 5. In Section 6, we discuss how the mediation process we propose can normally perform in super peer systems. Finally, we present related work and conclude this paper in Sections 7 and 8, respectively.
that can deal with an incoming query \( q \). We only consider the arrival of those queries for which \( \mathcal{P}_q \neq \emptyset \). Thus, the allocation of query \( q \) is denoted by a vector \( \text{All}oc_q \) of length \( N \) (with \( N = |\mathcal{P}_q| \)) such that,

\[
\forall p \in \mathcal{P}_q, \text{All}oc[p] = \begin{cases} 1 & \text{if } p \text{ gets the request} \\ 0 & \text{otherwise} \end{cases}
\]

Set \( \mathcal{P}_q \) denotes the set of providers such that \( \text{All}oc_q = 1 \), whose length is \( \min(q,n,N) \). In other words, if the number of providers that are able to deal with a query \( q \) is smaller or equal than the number of desired providers by a consumer, all the providers in \( \mathcal{P}_q \) must perform \( q \). Given all the above, we can then state the query allocation problem in mediator systems with autonomous participants as follows.

**Definition 1. Query Allocation Problem** Given a mediator \( m \) confronted to \( |\mathcal{Z}| \) autonomous participants, \( m \) should allocate each incoming query \( q \) to a set \( \mathcal{P}_q \) such that \( |\mathcal{P}_q| = \min(q,n,N) \) and good system performance as well as participants’ intentions are satisfied in the long-run.

### 2.4 SQLB Mediation

SQLB allocates queries by balancing consumers’ and providers’ intentions in accordance to their satisfaction. In that work, we experimentally proved that SQLB ensures good system performances and satisfies, in the long-run, the participants’ intentions, which allows it to scale up in environments with autonomous participants. This is why we consider it as a baseline method in the remainder of this paper. The general system architecture is shown by Figure 1(a) and the principle of SQLB is the following. A consumer (the user) sends a query \( q \) to the mediator, which must find set \( \mathcal{P}_q \) and ask for \( q.c \)'s intentions regarding each provider in \( \mathcal{P}_q \). Also, it must ask for \( \mathcal{P}_q \)'s intention for performing \( q \). Once the mediator obtains such information, it computes the score of each provider \( p \in \mathcal{P}_q \) by making a balance between \( q.c \)'s and \( p \)'s intentions and computes the ranking of providers in \( \mathcal{P}_q \) based on their score. Finally, it allocates \( q \) to the \( q.n \) best scored providers in \( \mathcal{P}_q \) and informs all participants of the mediation result.

### 3. USE OF VIRTUAL MONEY

We now consider the introduction of virtual money within the system so as to allow a system to scale up more easily. Notice that this money is purely virtual. This point has to be stressed upon for two main reasons. First, we do not focus on any particular business model: we only use the virtual money as a means to regulate the query allocation in the system. Indeed, after a consumer has decided which providers it chooses, it might give real money to them because it uses their services. This point is far beyond the focus of this paper. Second, when using real money, one can assume that consumers and providers get money from elsewhere. For example, when designing an auction mechanism for e-commerce one can assume that people spend the money they have earned by working (in real life). When dealing with virtual money, one can no longer make such assumptions. Therefore, in our case, to correctly regulate the system, we must precise the way in which the virtual money circulates within it. This is a macro-economic concern for which we adopt a simple solution for the mechanism we propose. First, only the providers and the mediator deal with virtual money, i.e. the consumers still show their intentions. The policy used to regulate the money flow depends on the mechanism. We decide that, in case the mediator earns money while the providers get poorer and poorer, the mediator distributes the money to the providers; That is, to avoid that the lack of virtual money influences the regulation effect of our mechanism, when the virtual money earned by the mediator is above some threshold, it is distributed again among all providers in an equitable way.

Now, designing the allocation mechanism used by the mediator is a micro-economic concern. Providers no longer express their intentions directly to the mediator. Instead, they have to express their intentions through bids, which also considers their current virtual money balance as well as their strategy to bid. Notice that the general architecture of the virtual money-based system is almost the same that when one does not use virtual money (see Figure 1(b)). The only thing that changes is that the mediator now asks providers for their bids instead of their intentions and has to invoice providers at the end of each query allocation. Therefore, the design of a specific allocation mechanism first requires to define the way in which the mediator merges the consumers’ intentions with the providers’ bids so as to select providers. This is the selection phase. Second, we consider in this work systems where providers have to “pay” for performing or receiving queries. Thus, in such models, one has to define the way in which the mediator invoices providers after each query allocation. This is the invoicing phase. Finally, one has to define how providers compute their bids.

### 4. VM\(_b\)QA MEDIATION

We present in this section Virtual Money-based Query Allocation (VM\(_b\)QA), a particular query allocation method that takes into account both consumers’ intentions and providers’ bids. VM\(_b\)QA assumes that consumers show their intentions (denoted by vector \( \bar{I} \)) to the mediator, while providers bid on queries instead (which is a means to reflect their intentions while keeping them private). In Section 4.1, we define the process itself and then define how a provider computes its bids in Section 4.2. Finally, we evaluate in Section 4.3 how well VM\(_b\)QA allocates queries among providers.

#### 4.1 Definition of the Process

Let us consider the allocation of some query \( q \) initiated

\footnote{This is in some way similar to an invoicing such as the one of Google AdWords \cite{AdWords}, except we consider virtual money.}
by some consumer \( c \in C \). The providers in \( P_q \) bid on \( q \) and such bids are not public to others. The bids are represented by a vector \( \overrightarrow{B} \), with \( B_p \in \mathbb{R} \) for all \( p \in P_q \). If a bid is positive, the higher it is, the more \( p \) wants to be allocated \( q \). If it is negative, the lower it is the less \( p \) wants to treat \( q \). Intuitively, the bid of \( p \) reflects its intention to perform \( q \). This should lead to the providers’ satisfaction. We detail the way in which \( VM_qQA \) allocates queries among providers in Section 4.1.1 and define the way in which it invoices providers in Section 4.1.2.

### 4.1.1 Selection phase

Let us remember that a query \( q \) is allocated to the \( \min(n, N) \) providers, which are given by vector of ranking \( \overrightarrow{R} \). Intuitively, \( \overrightarrow{R}[1] = p \) if and only if \( p \) is the best ranked, \( \overrightarrow{R}[2] \) stands for the second best ranked and so on. Hence, \( \forall i, \overrightarrow{R}[i] = p \) and \( i \leq \min(n, N) \). Traditionally, vector \( \overrightarrow{R} \) is computed with respect to the providers’ bids only. Then, a simple solution to select providers may be the first-price sealed-bid approach, which selects the provider having made the highest bid. However, this is a provider-centric approach that penalizes consumers’ intentions. Thus, to satisfy consumers, \( VM_qQA \) allocates queries based on the providers’ level and computes vector \( \overrightarrow{R} \) with regards to these levels (denoted by vector \( \overrightarrow{L} \)) such that, given an incoming query \( q \), the provider with the highest level is allocated \( q \). We formally define providers’ level, denoted by vector \( \overrightarrow{L} \), concerning a query \( q \) as Definition 2. Given this definition, a query might be allocated to a provider that did not want it. We call this an imposition case, otherwise, we have a competition case.

**Definition 2. Provider’s Level**

Given an incoming query \( q \), the level of each \( p \in P_q \) is defined as follows,

\[
\overrightarrow{L}[p] = \begin{cases} 
( \overrightarrow{B}[p] + 1)^{-\omega} \times (CI_{qa}[p] + 1)^{1-\omega} \quad & \text{if } \overrightarrow{B}[p] \geq 0 \\
-(-\overrightarrow{B}[p] + 1)^{-\omega} \times (CI_{qa}[p] + 1)^{1-\omega} \quad & \text{otherwise}
\end{cases}
\]

Parameter \( \omega \), whose values are in the interval \([0.1\ldots1]\), ensures the balance between consumers’ intentions and providers’ bids. In SQLB a mediator set this parameter according to participant’s satisfaction to regulate the system. When using virtual money, this is no more necessary because the virtual money itself is a natural means of regulation. Thus, in \( VM_qQA \), the mediator sets \( \omega \) according to the importance it wants to give to consumers’ intentions and providers’ bids. If \( \omega = 0 \), only the consumer’s intentions are considered by the mediator, thus leading to providers dissatisfaction. Conversely, if \( \omega = 1 \), the mediator only considers bids, leading to consumers dissatisfaction. This is why the mediator should set parameter \( \omega \) according to the balance between both consumers’ and providers’ satisfaction that it wants to reach.

Table 1 shows the case of a competition. The consumer asks for two providers, and more than two of them bid positively. Providers \( p_2 \) and \( p_3 \) are allocated the query because they get the two highest levels, respectively 2.28 and 2.24. Notice that the consumer’s intention with respect to \( p_3 \) is lower than its intention with respect to \( p_2 \). Thus, \( p_3 \) only got the query because of its bid (2.25) which is higher than \( p_2 \)’s bid (1.79), meaning that it wanted the query more than \( p_2 \). Table 2 shows an imposition case where no provider but \( p_4 \) wants to treat the query, whereas the consumer asks for two providers. Provider \( p_3 \) is imposed the query because of both its bid (which is the highest negative bid) and the consumer’s intention with respect to it, which leads to the value 1.47 of its level.

### 4.1.2 Invoicing phase

As start point, a natural strategy to invoice a provider \( p \) that has been allocated a query \( q \) is that \( p \) pays what it has bid for \( q \), which is denoted by \( \overrightarrow{B}[p] \). Thus, given a query \( q \), the mediator computes the providers’ invoice as follows.

**Definition 3. Provider’s Invoice**

If a provider \( p \in P_q \) is allocated a given query \( q \), then it must pay the amount of virtual money it has offered to get \( q \). Then, after the payment for \( q \), the new virtual money balance of \( p \), \( bal_p \), is set as follows, \( bal_p = bal_p - \overrightarrow{B}[p] \)

### 4.2 Bid Computation

A simple way to obtain providers’ bid, is that each provider maintains a local bulletin board, which contains a billing rate for its resources based on its preferences to perform queries (denoted by function \( prfF \)). Then, a provider’s bid for getting a query may be the product of its current utilization by the billing rate, i.e. its preferences for the query (such as in [13]). In our case, the context of a provider is more complex: we have to consider its preferences, load, current satisfaction, and current virtual money balance. Thus, given a query \( q \), a provider first works out its intention to perform \( q \) by considering its preferences (denoted by function \( prfF_p(q) \)), whose values are in the interval \([-1..1]\), its current utilization, and its current satisfaction. Intuitively, on the one hand, a provider may not pay so much attention to its preferences and accepts sometimes queries it does not desire if it is satisfied. On the other hand, if a provider is not satisfied, it may focus on its preferences to obtain desired queries. Then, a provider computes its intention as follows.

**Definition 4. Provider’s Intention**

Given an incoming query \( q \), a provider \( p \in P_q \) computes its intention towards

![Image](image-url)
\( p_i(q) \), as follows,

\[
p_i(q) = \begin{cases} 
(\text{prf}_q(1 - \delta_s(p))) \times (1 - \text{U}_p(t))^{\delta_s(p)}, & \text{if } \text{prf}_q(q) > 0 \land \text{U}_p(t) < 1 \\
-((1 - \text{prf}_q(q) + 1)^{1 - \delta_s(p)} \times (\text{U}_p(t) + 1)^{\delta_s(p)}), & \text{else} 
\end{cases}
\]

Once a provider obtains its intention with respect to a query, it then proceeds to work out its bid to perform such a query. Intuitively, the bid of a provider \( p \) is the product of its intention by its current virtual money balance (\( \text{bal}_p \)). Nonetheless, such a simple procedure may lead a provider to spend all, or almost all, its money on only one query. To avoid such a behavior, a provider offers at most a defined percent of its current virtual money balance, denoted by constant \( c_0 \), \( 0 < c_0 \leq 1 \). Then, we define a provider's bid as in Equation 3, where constant \( c_1 \) is set to the initial virtual money balance of a provider.

**Definition 5.** Provider's Bid Given an incoming query \( q \), a provider \( p \in P_q \) computes its bid to perform \( q \), \( b_p(q) \), as follows,

\[
b_p(q) = \begin{cases} 
\text{pi}_p(q) \cdot \text{bal}_p \cdot c_0, & \text{if } \text{pi}_p(q) > 0 \\
\text{pi}_p(q) \cdot c_1, & \text{otherwise} 
\end{cases} \quad (3)
\]

The idea behind the above definition is that a provider always sets a positive bid when it desires to perform queries and it is not overutilized, otherwise it sets a negative bid. Notice that in traditional economic-based methods providers just do not bid (or give a null bid) when they do not desire to perform a query. However, this does not allow them to express how much unpleasant it is for them to perform a query and how much overloaded they are. Thus, Equation 3 allows a provider to preserve its preferences while good response times are also ensured to consumers. At first glance, there is no difference between providers' bids and intentions, but by showing bids providers can keep private its real intentions.

### 4.3 Evaluation of VM\(_b\)QA

Our aim in this evaluation is to analyze the impact of using virtual money to address the query allocation problem defined in Section 2.3. To clearly see such an impact, we compare VM\(_b\)QA mediation with SQLB mediation, which has been proved to perform well in autonomous environments. Also, to analyze the possible loss that providers may have with respect to providers due to the attention they pay to consumers' intentions, we compare VM\(_b\)QA with a provider-centric mediation: the first-price sealed-bid mediation (FPSB for short). Unlike VM\(_b\)QA that allocates queries by balancing consumers' intentions with providers' bid, FPSB allocates each incoming query to those providers that have made the highest bid. In contrast, as in VM\(_b\)QA as in FPSB, a provider pays for a query what it has bid.

In Section 4.3.1, we describe the experimental setup and, in Section 4.3.2, present the experimental results.

#### 4.3.1 Evaluation Setup

The system consists of 200 consumers, 400 providers, and only one mediator allocating all the incoming queries. We assigned sufficient resources to the mediator so that it does not cause bottlenecks in the system. We assume that consumers compute their intentions by considering only their preferences. Concerning a provider in SQLB mediation, on the one hand, we assume that it computes its intentions as defined in [9]. On the other hand, a provider in both FPSB and VM\(_b\)QA mediations computes its bid as defined in the Section 4.2. To study all three mediations, we implemented an observer that computes participants' satisfaction as presented in Section 2.2. It initializes participants' satisfaction with a value of 0.5, which evolves with their last 200 issued queries and 500 queries that have passed through providers (i.e., \( k = 200 \) for a consumer and \( k = 500 \) for a provider).

We set the heterogeneity of the providers' capacity in accordance to the results presented in [11]. 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 high-capacity providers perform in 1.3 and 1.5 seconds, respectively. We consider that queries arrive to the system in a Poisson distribution, as found in dynamic autonomous environments [5]. 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 a provider's degree of interest to perform incoming queries. Providers randomly obtain their preferences between −.2 and 1 (high-adaptation), between −.6 and .6 (medium-adaptation) or between −1 and 2 (low-adaptation).

Concerning participants' departures, we assume on the one hand that a consumer leaves the system, by dissatisfaction if its satisfaction is smaller than .4. On the other hand, we assume a provider may leave the system by three reasons: (i) dissatisfaction if its satisfaction is smaller than 0.3, (ii) by query starvation if, in an interval of 1 minute, it does not perform a set of queries towards which it has a preference of at least 0.2 in average, and (iii) by overutilization if its utilization is greater than 1.5 if it is a high-capacity provider, 2.5 if it is a medium-capacity provider, or 3 if it is a low-capacity provider. Without any loss of generality, the participants' expectations, in the long run, are static in our simulations. We assume this to evaluate all three mediations in a long-term trend.

Finally, let us make the following assumptions. First, as our main focus is to study if VM\(_b\)QA performs equal or better than SQLB, we do not consider the bandwidth problem and assume that all participants have the same network capacities. Second, for the sake of simplicity, we assume that consumers only ask for one informational answer (i.e., \( n = 1 \)) and that all the providers in the system are able to perform all the incoming queries. Above assumptions do not harm with the generality of our proposal.

#### 4.3.2 Evaluation Results

Before going on to discuss the results, let us say that, for space reasons, we do not present all results we obtained from
this experimentation but the most relevant. That is, we only discuss the results concerning the participants’ satisfaction and response times ensured by all three mediations.

Figure 2(a) illustrates the consumers’ satisfaction results. We can observe that VM₆QA satisfies consumers much better than FPSB and is not so far from SQLB. Obviously, VM₆QA outperforms FPSB because FPSB has no consideration of the consumers’ intentions. However, when regarding the providers’ satisfaction in Figure 2(b), we can observe that VM₆QA once again outperforms FPSB. This proves that VM₆QA has no loss in performance w.r.t. providers’ satisfaction while it has benefits from the consumers’ point of view. Nonetheless, during our experimentation we observed that FPSB better balances queries among providers than VM₆QA. This is because VM₆QA also considers the consumers’ intentions and thus, given the experimental setup, VM₆QA tends to allocate queries to the most reputed providers (which are those providers that consumers prefer in general) causing a query load imbalance in the system. This is reflected in the response time ensured by both two mediations, which is illustrated by Figure 2(c). We can observe in those results that FPSB ensures lower response times than VM₆QA. It is however so far from SQLB, which outperforms it by a factor of 2.

5. IMPROVEMENT OF VM₆QA

Although VM₆QA has no loss in performance concerning providers’ satisfaction and only a small loss concerning response times w.r.t. FPSB, it is still far from SQLB in all cases: participants’ satisfaction and ensured response times. An intuition to improve VM₆QA is to change the way in which it invoices providers. In this section, we present another way to invoice providers [4] in Section 5.1 and evaluate it in Section 5.2.

5.1 Invoicing Providers

We now discuss another manner to compare bids to rank providers regarding their bids. Notice that bids cannot be directly compared because of consumers’ intentions. Thus, to overcome this difficulty, the theoretical bid is introduced ($B^T(p, l)$), which corresponds to the amount that provider $p$ should bid for reaching level $l$. With $\omega \neq 0$ and $\alpha = 1$ if $l \geq 0$, and $\alpha = -1$ otherwise, $B^T(p, l)$ is given by the following formula,

$$B^T(p, l) = \alpha \max((\alpha \times l)^{\frac{1}{\omega}}(\min\{l, 1\}^\omega + 1)\frac{\alpha^{\omega - 1}}{\omega} - 1), 0)$$

For example, in Table 1, we have already noticed that provider $p_3$ gets a level slightly higher than $p_1$’s, because of its higher bid and despite the lower consumer’s intention.

In fact, to come exactly to $p_3$’s level, $p_5$ should bid 2.136 (theoretical bid). Now, given a provider $p' \in P_q$, $PT[p, p']$ denotes what $p'$ owes due to the allocation of $q$ to $p \in P_q$ ($PT[p, p'] = 0$ if $p$ is not allocated $q$). Then, the total amount paid by $p'$ is defined by a sum (Equation 5).

$$\forall p' \in P_q, \quad Trans[p'] = \sum_{p \in P_q} PT[p, p']$$

There is a competition among providers when there are enough providers that want to be allocated the query. In that case, each of them pays the amount of its theoretical bid to reach the level of the best provider which has not been selected (in the spirit of a generalized Vickrey auction [14] except that the consumer’s intention is considered). In Table 1, only providers $p_5$ and $p_3$ pay (respectively 2.09 and 1.75) to the mediator, thus decreasing their own money balance (bal). A requisition case occurs when at least one provider is imposed the query. Obviously, being imposed does not meet at all the expectations of the imposed providers. Hence, to keep them satisfied in the long run, the idea is then to distribute the cost of the imposition on all the providers in $P_q$ (in the spirit of [12], but also considering the consumer’s intentions). Then, having obtained a reward, the imposed providers are more likely, in the future, to obtain the queries they expect (because they have more money) so leading to their satisfaction. The formal definitions of the transfers in the imposition case are as in definition below.

**Definition 6.** Partial Transfers in a Requisition

Case If provider $p \in P_q$ is allocated $q$ and $\bar{B}[p] < 0$, then for all $p' \in P_q$,

$$\overline{PT}[p, p'] = \begin{cases} 
-\frac{B^T(p, \bar{L}[\min\{n + 1, N\}])}{N} & \text{if } p \neq p' \\
\frac{B^T(p, \bar{L}[\min\{n + 1, N\}])}{N} & \text{else} 
\end{cases}$$

In the example of Table 2, $p_5$ is imposed and thus gets a 4.89 reward. All the providers contribute to this reward. Notice also in both Table 1 and Table 2 that the mediator gets some money left, which is of no use for it. We discussed this point in Section 3.

5.2 Evaluation of VM₆QA⁺

We implemented VM₆QA⁺ mediation, which selects providers as VM₆QA does but invoices them as defined in
previous section. We compare $VM_{QA}^+$ with SQLB to evaluate whether it can ensure the same system’s performance as SQLB. Particularly, we evaluate whether the way in which $VM_{QA}^+$ invoices providers improves the performance of $VM_{QA}$. To be comparable with evaluated mediations in Section 4.3, we consider the same experimental setup we used in Section 4.3.1 to run these experimentations.

5.2.1 Results

We can observe in Figures 3(a) and 3(b) that $VM_{QA}^+$ ensures the same participants’ satisfaction, with only a quite small difference for low workloads, as SQLB. This is because the way in which $VM_{QA}^+$ invoices providers avoids that some providers monopolize queries, thereby allowing to better balance queries among providers. This allows $VM_{QA}^+$ better satisfying providers without penalizing consumers’ intentions. Figure 3(c) illustrates the response times ensured by $VM_{QA}^+$. We observe that $VM_{QA}^+$ has the same performance as SQLB, however we can observe that for a workload of 20% of the total system capacity SQLB slightly outperforms $VM_{QA}^+$. Even this small difference we can see that $VM_{QA}^+$ generally performs as well as SQLB. Thus, we can conclude that we can introduce virtual money, without any loss of system’s performance, to regulate a system as long as we care about the way in which providers are selected and invoiced.

6. DEALING WITH SEVERAL MEDIATORS: SUPER-PEER NETWORKS

We experimentally proved in previous section that $VM_{QA}^+$ performs as well as SQLB in mono-mediator systems. In this section, we demonstrate that $VM_{QA}^+$ performs well in multi-mediator systems, such as in super-peer systems. A super-peer network is a pure peer-to-peer (P2P) network, i.e. an unstructured P2P network, but each peer in the network is actually a super-peer. A super-peer operates as a server to a cluster of peers (participants) and as equal participant with respect to other super-peers. We call a set of participants with at least one of them playing the role of super-peer a virtual organization (VO). Figure 4 illustrates a super-peer network with 6 super-peers, which are represented by the SP peers.

Figure 5: A virtual organization (VO) with 3-redundancy, where $p$ peers represent those participants that do not play the super-peer role.

Figure 3: (a-b) participants’ satisfaction results and (c) ensured response times, for different workloads.
participants’ satisfaction is not the same at all mediators. Moreover, these update messages considerably increase the network traffic and hence hurt response times, which does not allow the system to scale up. In contrast, \( VM_{QA}^{+} \) has no message to exchange among mediators because the system regulation is done by the flow of virtual money instead of by satisfaction. In other words, \( VM_{QA}^{+} \) has no additional cost when performing in a multi-mediator system. Therefore, \( VM_{QA}^{+} \) allows a VO, and hence a super-peer system, to scale up with no loss in system performance. We state this in the following Theorem.

**Theorem 1.** \( VM_{QA}^{+} \) always satisfies (i) consumers and (ii) providers in a \( x - \) redundant VO as well as in a mono-mediator system.

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

(i) Assume to the contrary that, for some query \( q \), consumer \( q_{c} \in T \) is not equally satisfied by \( S_{vo} \) and \( S_{m} \). If this is the case, we can know, by Equation 1, that \( S_{vo} \) allocated \( q \) to a set \( \widetilde{P}_{q} \) such that \( \exists p_{\in} \widetilde{P}_{q}, q \notin \widetilde{P}_{q} \), where \( \widetilde{P}_{q} \) is the set of providers selected by \( S_{m} \). Hence, we can know that the set of relevant providers found by \( S_{vo} \) is different to the set found by \( S_{m} \). This implies that provider \( p \) is not connected to the super-peer that allocated \( q \) in \( S_{vo} \), which contradicts the definition of a \( x - \) redundant VO.

(ii) Assume to the contrary that a provider \( p \in T \) is not equally satisfied by \( S_{vo} \) and \( S_{m} \). Then, by Equation 2, we can know that \( p \) did not perform the same set of queries in \( S_{vo} \) as in \( S_{m} \). This means that \( p \) is not connected to all super-peers in \( S_{vo} \) so as to receive all queries it can perform, which contradicts the definition of a \( x - \) redundant VO. □

To discuss how queries may be forwarded to other VOs is well beyond the scope of this paper.

### 7. RELATED WORK

Several approaches based on microeconomics have been proposed [2, 3, 8, 13]. A survey of economic models for various aspects of distributed system is presented in [3]. In economic models the notion of *utility* is clearly linked to satisfaction. Nonetheless, it is generally reduced to the allocation of only one query and to monetary concerns. In the field of distributed rational decision making [10], most of the processes are *individually rational*: the utility of any participant in the process is no less than the utility it would have by not participating. This property is not relevant in contexts where some participants may be imposed, which implies having a lower utility in participating. Furthermore, participants may have the interest that the system be effective and, in certain query allocations, some participants may be interested in participating, even if this means to have a low utility. Thus, to consider participants’ satisfaction is still relevant in these contexts because it is a long-run notion. In [7], the authors focus on the optimization algorithms for *buying* and *selling* query answers, and the negotiation strategy. However, this way of dealing with subqueries optimization is orthogonal to our proposal and one may combine them to improve performance. In [4], we proposed an economic flexible mediation that allocates queries by taking into account the *quality* and *bids* of providers, but it is inherently assumed that consumers are only interested in the quality of providers. In a recent work [9], we provide a query allocation framework for distributed information systems with autonomous participants, but we considered VOs with only one super-peer, i.e., mono-mediator systems.

### 8. CONCLUSION

We addressed the query allocation problem in super-peer data management systems. We presented a virtual money-based query allocation mediation that behaves as good as SQLB. We compared a non micro-economic mediation (SQLB) with micro-economic ones (FPSB, \( VM_{QA} \) and \( VM_{QA}^{+} \)), by evaluating the participants’ satisfaction and response times. From the results point of view, we showed that, with both an adequate selection and invoice, a virtual money based mediation shows results that are as good as those of SQLB. This enables scaling up to a super peer network where all the peers use the same virtual money, without any additional cost for the allocation of a query.

### 9. REFERENCES


