Summary Management in Unstructured P2P Systems

Rabab Hayek, Guillaume Raschia, Patrick Valduriez, Noureddine Mouaddib

Summary Management in Unstructured P2P Systems

Rabab Hayek* — **Guillaume Raschia*** — **Patrick Valduriez**** — **Noureddine Mouaddib***

Atlas Team (INRIA - LINA), University of Nantes
2 rue de la Houssiniere – B.P. 92208, 44300 Nantes, France
* *surname.name@univ-nantes.fr*
** *Patrick.Valduriez@inria.fr*

ABSTRACT. *In this paper, we propose managing data summaries in unstructured P2P systems. Our summaries are intelligible views with two main virtues. First, they can be directly queried and used to approximately answer a query. Second, as semantic indexes, they support locating relevant nodes based on data content. The performance evaluation of our proposal shows that the cost of query routing is minimized, while incurring a low cost of summary maintenance.*

RÉSUMÉ. *Dans ce travail, nous proposons de maintenir des résumés de données dans les systèmes P2P non structurés. Nos résumés sont des vues intelligibles ayant un double avantage en traitement de requête. Ils peuvent soit répondre d'une manière approximative à une requête, soit guider sa propagation vers les pairs pertinents en se basant sur le contenu des données. L'évaluation de performance de notre proposition a montré que le coût de requêtes est largement réduit, sans induire des côuts élevés de maintenance de résumés.*

KEYWORDS: *P2P Systems, DB Summarization*

MOTS-CLÉS : *Systèmes pair-à-pair, Résumés des bases de données*

RSTI - ISI – 13/2008, pages 83 à 107

1. Introduction

Today's information systems are facing two main problems. First, they host a large number of data sources that are highly distributed, autonomous, and dynamic. Second, modern applications generate huge amount of information stored into the connected data sources, which become more and more voluminous. These applications require efficient and flexible querying facilities to access semantically rich data. Therefore, information systems need to scale up in terms of both number of participants and amount of shared data, without losing the ability to handle complex queries.

While distributed systems such as database, integration and parallel systems have reached their maturity and only supported a limited number of users, P2P systems allow data sharing on a world wide scale with many advantages like decentralization, self-organization, autonomy, etc. Initially developed for file-sharing applications, P2P technology is now evolving to support more advanced applications. Such applications must deal with high-level semantics of shared data by the use of advanced query languages and access methods. As a potential example of applications, consider the cooperation of scientists who are willing to share their private data for the duration of a given experiment. However, a major problem in the operation of P2P systems as distributed systems is object locating. Unstructured P2P search systems rely on flooding mechanism and its variations. Though simple and robust, this approach suffers from high query execution cost and poor query recall. Initial works have led to structured P2P systems (e.g. (Stoica *et al.*, 2001)), which are based on Distributed-Hash-Table functionalities. These systems achieve the goal of improving search efficiency, but they compromise peer autonomy and may restrict query expressiveness.

So far, data localization has been the main issue addressed in P2P data sharing systems, whose scalability is constrained by the employment of efficient search techniques. But, nowadays we are asking the following question: with the ever increasing amount of information stored each day into data sources, are these techniques still sufficient to support advanced P2P applications? To illustrate, in a scientific collaborative application, a doctor may require information about patients diagnosed with some disease, without being interested in individual patient records. A user in such collaborative or decision-support applications may prefer an approximate but fast answer, instead of waiting a long time for an exact one. Therefore, reasoning on compact data descriptions that can return approximate answers like "dead Malaria patients are typically children and old" to queries like "age of dead Malaria patients", is much more efficient than retrieving raw records, which may be very costly to access in highly distributed, massive databases.

This work aims at managing summaries over shared data in P2P systems. Data summaries are synthetic, multidimensional views with two main virtues. First, they provide an intelligible representation of the underlying data such that an approximate query can be processed entirely in their domain; that is, inputs and outputs are summaries. Second, as indexing structures, they support locating relevant nodes based on their data descriptions. This paper makes the following contributions. First, we

define an efficient algorithm for partitioning an unstructured P2P network into domains, in order to optimally distribute summaries in the network. Then, we propose distributed algorithms for managing data summaries in a given domain. We validated our proposal through simulation, using the BRITE topology generator and SimJava. The performance results show that the cost of query routing is minimized, while incurring a low cost of summary maintenance. The rest of this paper is organized as follows. Section 2 describes our summary model for P2P systems. Section 3 presents the algorithm for network organization, while section 4 presents the algorithm for summary management. Section 5 discusses query processing in the context of summaries. Section 6 gives a performance evaluation with a cost model and a simulation model. Section 7 compares our solution with related work. Section 8 concludes.

2. Summary model for P2P systems

In this section, we first present our summary model architecture. Second, we describe the summarization process that allows generating summaries of a relational database. Then, we formally define the notion of data summary in a P2P network.

2.1. *Model architecture*

Data indexes are maintained in P2P systems using one of the following approaches. A *centralized* approach maintains a global index over all the data shared in the network, and thus provides a centralized-search facility. A *hybrid decentralized* approach distributes indexes among some specialized nodes (e.g. supernodes), while a *pure decentralized* approach distributes indexes among all the participants in the network (e.g. structured DHTs, Routing Indices). Each of these approaches provides a different trade-off between the cost of maintaining the indexes and the benefits obtained for queries. In our work, we have adopted the second approach since it exploits peer heterogeneity, which is a central key to allow P2P systems scaling up without compromising their decentralized nature (Saroiu *et al.*, 2002). Let us examine the architecture of our summary model which is presented in Figure 1. The network is organized into *domains*, where a domain is defined as being the set of a supernode and its associated leaf nodes. In a given domain, peers cooperate to maintain a global summary over their shared data. The set of global materialized summaries and links between the corresponding domains, provide a virtual *complete* summary, which ideally describes all the data shared in the network. Obviously, an important issue here is how the network is organized into domains, *i.e.* how the superpeers are selected and other peers are grouped around them in a fully decentralized manner. This issue will be discussed in Section 3. However, we first give a brief description of the summarization process that generates summaries of relational databases.

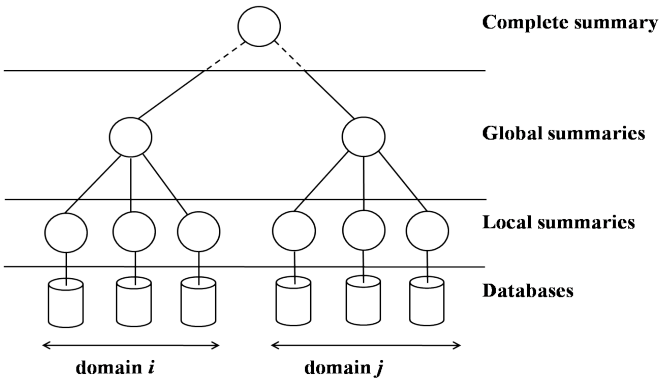


Figure 1. *Model architecture for hierarchical P2P networks*

2.2. *Summarization process*

A summarization process is integrated to each peer's DataBase Management System (DBMS) to allow constructing the local summary level of Figure 1. Our approach is based on SAINTETIQ (Raschia *et al.*, 2002), an online linguistic approach for summarizing databases. The SAINTETIQ system takes tabular data as input and produces multi-resolution summaries of records through a two-step process: online mapping and summarization. For illustration, consider a relational database which is reduced to a single *Patient* relation (Table 1).

Table 1. *Raw data*

Id	Age	BMI	Disease
t_1	15	17	*Anorexia*
t_2	20	20	*Malaria*
t_3	18	16.5	*Anorexia*

Table 2. *Grid-cells mapping*

Id	Age	BMI	tuple count
c_1	*young*	*underweight*	2
c_2	0.7/*young*	*normal*	0.7
c_3	0.3/*adult*	*normal*	0.3

2.2.1. *Mapping service*

The SAINTETIQ system relies on Zadeh's fuzzy set theory (Zadeh, 1965) and, more specifically on linguistic variables (Zadeh, 1975) and fuzzy partitions (Zadeh, 1999) to represent data in a concise form. The fuzzy set theory is used to translate records according to a *Background Knowledge (BK)* provided by the user. The Background Knowledge BK is a priori built over the attributes that are considered relevant to the summarization process. In the above relation, the selected attributes are AGE and BMI [1]. Basically, the mapping operation replaces the original values of every record in the table by a set of linguistic descriptors defined in the *BK*. For instance,

1. Body Mass Index (BMI): patient's body weight divided by the square of the height.

with a linguistic variable on the attribute AGE (Figure 2), a value $t.AGE = 20$ years is mapped to *{0.3/adult, 0.7/young}* where 0.3 is a membership grade that tells how well the label *adult* describes the value 20. Extending this mapping to all the attributes of a relation could be seen as locating the overlapping cells in a grid-based multidimensional space that map records of the original table. The fuzzy grid is provided by BK and corresponds to the user's perception of the domain.

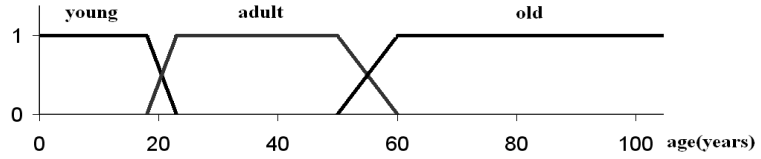


Figure 2. *Fuzzy linguistic partition on* `age`

Thus, tuples of Table 1 are mapped into three distinct grid-cells denoted by c_1, c_2 and c_3 in Table 2. *A priori*, the fuzzy label *underweight* provided by the BK on attribute BMI, perfectly matches (with degree 1) range $[15, 17.5]$, while the fuzzy label *normal* perfectly matches range $[19.5, 24]$ of raw values. Besides, tuple count column gives the proportion of records that belongs to the cell and 0.3/*adult* says that *adult* fits the data only with a small degree (0.3). It is computed as the maximum of membership grades of tuple values to *adult* in c_3. The fuzziness in the vocabulary definition of *BK* permits to express any single value with more than one fuzzy descriptor and thus avoid threshold effect thanks to the smooth transition between different categories. Besides, BK leads to the point where tuples become indistinguishable and then are grouped into grid-cells such that there are finally many more records than cells. Every new (coarser) tuple stores a record count and attribute-dependent measures (min, max, mean, standard deviation, etc.). It is then called a *summary*.

2.2.2. *Summarization service*

The summarization service is the last and the most sophisticated step of the SAINTETIQ system. It takes *grid-cells* as input and outputs a collection of summaries hierarchically arranged from the most generalized one (the root) to the most specialized ones (the leaves) (Raschia *et al.*, 2002). Summaries are clusters of grid-cells, defining hyperrectangles in the multidimensional space. In the basic process, leaves are grid-cells themselves and the clustering task is performed on K cells rather than N tuples ($K << N$).

From the mapping step, cells are introduced continuously in the hierarchy with a top-down approach inspired of D.H. Fisher's Cobweb (Thompson *et al.*, 1991), a conceptual clustering algorithm. Then, they are incorporated into best fitting nodes descending the tree. Three more operators could be apply, depending on partition's score, that are *create*, *merge* and *split* nodes. They allow developing the tree and updating its current state. Figure 3 represents the summary hierarchy built from the cells c_1, c_2 and c_3.

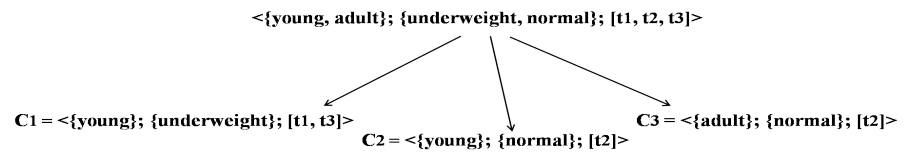


Figure 3. *Example of SaintEtiQ hierarchy*

2.2.3. *Scalability issues*

Memory consumption and time complexity are the two main factors that need to be taken care of in order to guaranty the capacity of the summary system to handle massive datasets. First, the time complexity of the SAINTETIQ process is in $O(K)$, where *K* is the number of cells to incorporate into a hierarchy of summaries. Here we note that, the number of cells that are produced by the mapping service depends only on the granularity and the fuzziness of the BK definition. A fine-grained and overlapping BK will produce much more cells than a coarse and crisp one. Besides, an important feature is that in the summary algorithm, raw data have to be parsed only once, and this are processed with a low time cost. Second, the system requires low memory consumption for performing the summary construction algorithm as well as for storing the produced summaries. On the other hand, the parallelization of the summary system is a key feature to ensure smooth scalability. The implementation of the summarization system is based on the Message-Oriented Programming paradigm. Each sub-system is autonomous and collaborates with the others through disconnected asynchronous method invocations. It is among the least demanding approaches in terms of availability and centralization. Thus, the autonomy of summary components allows for a distributed computing of the summary process.

2.3. *Distributed summary representation*

In this section, we introduce basic definitions related to the summarization process.

Definition 1 ***Summary*** *Let $E = \langle A_1, \ldots, A_n \rangle$ be a n-dimensional space equipped with a grid that defines basic n-dimensional areas called cells in E. Let R be a relation defined on the cartesian product of domains D_{A_i} of dimensions A_i in E. Summary z of relation R is the bounding box of the cluster of cells populated by records of R.*

The above definition is constructive since it proposes to build generalized summaries (hyper-rectangles) from cells that are specialized ones. In fact, it is equivalent to performing an *addition* on cells: $z = c_1 + c_2 + \ldots + c_p$, where $c_i \in L_z$, the set of

p cells (summaries) covered by z. A summary z is then an *intentional description* associated with a set of tuples R_z as its *extent* and a set of cells L_z that are populated by records of R_z. Thus, summaries are areas of E with hyper-rectangle shapes provided by BK. They are nodes of the summary tree built by the SAINTETIQ system.

Definition 2 ***Summary Tree*** *A summary tree is a collection S of summaries connected by $\preccurlyeq$, the following partial order: $\forall z, z' \in \mathcal{Z}, \quad z \preccurlyeq z' \iff R_z \subseteq R_{z'}$.*

The above link between two summaries provides a generalization/specialization relationship. And assuming that summaries are hyper-rectangles in a multidimensional space, the partial ordering defines *nested summaries* from the larger one to the single cells. General trends in the data could be identified in the very first levels of the tree whereas precise information has to be looked at near the leaves. For our purpose, we also consider a summary tree as an indexing structure over distributed data in a P2P system. Thus, we add a new dimension to the definition of a summary z, which provides the set of peers having data described by z.

Definition 3 ***Peer-extent*** *Let z be a summary in a given hierarchy of summaries S, and P the set of all peers who participated to the construction of S. The* peer-extent *P_z of the summary z is the subset of peers owning, at least, one record of its extent R_z: $P_z = \{p \in P \mid R_z \cap R_p \neq \emptyset\}$, where R_p is the view over the database of node p, used to build summaries.*

Due to the above definition, we extend the notion of *data-oriented* summary in a given database, to a *source-oriented* summary in a given P2P network. In other words, our summary can be used as a database index (e.g. referring to relevant tuples), as well as a semantic index in a distributed system (e.g. referring to relevant nodes). The summary hierarchy S will be characterized by its *Coverage* in the P2P system; that is, the number of data sources described by S. Relative to the hierarchy S, we call *Partner Peer* a peer whose data is described by at least a summary node of S.

Definition 4 ***Partner peers*** *The set of* Partner peers *P_S of a summary hierarchy S is the union of peer-extents of all summaries in S: $P_S = \{\cup_{z \in S} P_z\}$.*

For simplicity, in the following we designate by "summary" a hierarchy of summaries maintained in a P2P system, unless otherwise specified.

3. Network self-organization

Intuitively, the term "self-organization" describes the ability of a P2P network to organize its participants into a cooperative framework, without the need of external intervention or control. For our purposes, we understand *self-organization* as the capability of partitioning the network into domains, to optimally distribute data summaries, without using global information or restricting peer autonomy.

3.1. *Rationale*

(Ganesan *et al.*, 2005) have addressed the problem of maintaining distributed indexes in a P2P network. Nodes are partitioned into independent domains, and nodes within a domain build a global index over their shared data. However, the authors have assumed that there exists a fixed number of domains k, and a node is assigned to one of these domains at random. As a first issue, they have studied the optimal number of domains in order to minimize the total cost of queries and index maintenance. It has been shown that this number is a function of the the total number of nodes in the network.

In our work, we do not suppose that nodes are assigned to domains at random. Instead, we are interested in how domains are created and nodes are assigned to them. Here, we present a primitive function that enables to partition the network around high-connectivity nodes. A number of recent studies (Saroiu *et al.*, 2002), (Ripeanu *et al.*, 2002) have shown that the existing complex networks have common characteristics, including power law degree distributions, small diameter, etc. In these networks, called *power-law networks*, most nodes have few links and a tiny number of hubs have a large number of links. More specifically, the fraction of nodes with k links is proportional to $k^{-\beta}$, where β is called the exponent of the distribution.

Our solution for network organization is completely decentralized, and mainly exploits the power-law distribution of node degrees. It does not rely on any global information, however, it uses local information by considering that a node has only to know about the entities and the connectedness of its neighbors. The key idea is that random walks in power-law networks naturally gravitate toward the high-degree nodes. A *random walk* is a technique proposed by (Lv *et al.*, 2002) to replace flooding. At each step, a query message is forwarded to a randomly chosen neighbor until sufficient responses to the query are found. Although it makes better utilization of the P2P network than flooding, a random walk is essentially a blind search in that it does not take into account any indication of how likely it is the chosen node will have responses for the query. (Adamic *et al.*, 2001) addressed this problem and showed that a better scaling is achieved by intentionally choosing high degree nodes. We will refer to this routing technique as "*selective walk*".

In our work, we aim to identify the superpeers, which will be referred later as *summary peers*, in a power law network with an exponent β and maximum degree k_{max}. Summary peers are defined as being high degree peers, which will serve as *centers of summary-attraction*. Using a selective walk which naturally and rapidly gravitates toward high degree nodes, a peer p finds the nearest summary peer SP to which it sends a duplicate of its local summary LS. The set of peers that discover the same summary peer SP are grouped around it and form a domain. These peers become *partners* relative to a global summary GS obtained by merging their local summaries. In (Sarshar *et al.*, 2004), any node with degree k is considered as a high-degree node if $k \geq k_{max}/2$. However, k_{max} scales like $O(N^{1/\beta})$ (Aiello *et al.*, 2000) and is a global information. In the next section, we propose an IS_SUMPEER function

that is executed locally at each peer to decide whether it is a summary peer or not, using minimum local information.

3.2. *Algorithm*

For our purpose, we have extracted a general model of a high-degree node in a power law network. Thus, the IS_SUMPEER function consists in matching this model with each node of the network. Algorithm 1 shows the steps involved in making this matching. First, we check if the current peer p is among the highest-degree peers in its neighborhood. In other words, we check if the degree k of peer p is greater than the median value of the set of its neighbor's degrees (*i.e.* $|subset\_inf| \succ |subset\_sup|$). Then, we verify if the local maximum degree k_{max} in p's neighborhood does not exceed $2 \cdot k$. Finally, we examine if k is larger than the mean value of neighbor's degrees by a constant ct. This condition makes a difference in the matching result when the neighbor's degrees follow an asymmetric distribution with a positive skew, *i.e.* there are a small number of very large degrees. In that case, the mean value is greater than the median. The constant ct permits to tune the selectivity of the matching function. Larger is ct, less is the total number of summary peers. Indeed, peer p is considered as a summary peer if the above three conditions are satisfied simultaneously.

Algorithm 1 $Is\_SumPeer$

function $Is\_SumPeer(k, NL)$
    k is the degree of the current peer p, and NL is the Neighboring List that contains the identifiers of p's neighbors and their degrees.
    $subset\_inf$:= $p_i \; \forall \; 1 \leq i \leq |NL|$ such that $k_i < k$
    $subset\_sup$:= $p_i \; \forall \; 1 \leq i \leq |NL|$ such that $k_i > k$
    k_{max} := **max** $(k_i), \forall \; 1 \leq i \leq |NL|$
    k_{mean} := **mean** $(k_i), \forall \; 1 \leq i \leq |NL|$
    if $(|subset\_inf| > |subset\_sup|)$ **and** $(k > k_{max}/2)$ **and** $(k > ct \cdot k_{mean})$ **then**
        $Is\_SumPeer$:= true
    else $Is\_SumPeer$:= false
    end if
end function

Using the Brite topology generator[2], we simulate N-node P2P networks whose node degrees follow a power law distribution with a mean value of 4. Figure 4 shows the number of summary peers in function of the total number of peers N. We see that this number is proportional to $\sqrt{n}$ for network sizes smaller than 1024, and is proportional to n for larger networks. Since our domains are formed around the summary peers, thus figure 4 gives directly the number d of domains obtained in the network.

The shown results are similar to those found in (Ganesan *et al.*, 2005). It has been proved, theoretically and by simulation, that the optimal number of domains required

2. http://www.cs.bu.edu/brite/

to distribute a global index, is in $O(\sqrt{n})$ for total-lookup queries and in $O(n)$ for partial-lookup queries. A *total-lookup* query requires all results that are available in the system, whereas a *partial-lookup* query requires any m results, for some constant m. We believe that a total-lookup query is very difficult and costly in large P2P networks, and all queries are processed as being partial-lookup queries. Therefore, we conclude that using a local degree-based function, we can organize the network into an optimal number of domains: for total-lookup queries in small-sized networks, and for partial-lookup queries in larger-sized networks. Certainly, this work is not complete and requires further examination and discussion. But, it may be considered as the initial phase of other works like (Ganesan *et al.*, 2005). In other terms, it constitutes the beforehand organization of a P2P network, which is then associated with efficient mechanisms for maintaining such organization against dynamicity.

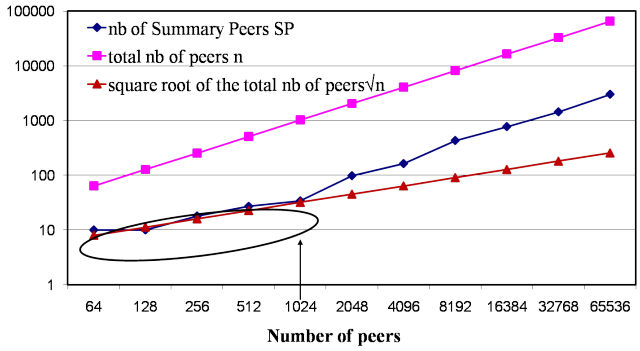


Figure 4. *Number of summary peers vs. number of peers*

4. Summary management

In this section, we present our algorithms for summary construction and maintenance in a given domain. First, we work in a static context. Then we address the volatility of peers and propose appropriate solutions.

4.1. *Summary construction*

We assume that each global summary is associated with a *Cooperation List* (CL) that provides information about its partner peers. An element of CL is composed of a partner identifier *PeerID*, and a *2*-bit value v that provides information about the description freshness as well as the availability of the corresponding database:

– value 0 (initial value): the descriptions are fresh relative to the original data,

– value 1: the descriptions need to be refreshed,

– value 2: the original data are not available.

Algorithm 2 shows the messages exchanged between peers in order to build a global summary GS. A summary peer SP broadcasts a SUMPEER message that contains its identifier, to indicate its ability to host summaries. Since SP is supposed to have high connectivity, a small value of TTL (Time-To-Live) is sufficient to cover a large number of peers (e.g. $TTL = 2$). The message contains also a hop value h, initialized to 0, which is used to compute the distances between SP and the visited peers. A peer p who received a first SUMPEER message, maintains information about the corresponding summary peer SP (*i.e.* Line 13). Then, p sends to SP a LOCALSUM message that contains its local summary LS, and thus becomes a partner peer in the SP's domain. Upon receiving this last message, SP merges LS to its current global summary GS, and adds a new element in the cooperation list.

However, a peer p who is already a partner may receive a new SUMPEER message. In that case, only if the new summary peer is nearer than the old one (based on latency), it chooses to drop its old partnership through a DROP message (*i.e.* Line 11), and p proceeds to participate to a new domain. We now suppose that a peer p does not belong to any domain, and wants to participate to a global summary construction. Using a selective walk, it can rapidly find a summary peer SP (*i.e.* FIND message). The information about SP, which is maintained at each of its partners, makes the selective walk even shorter. Once a partner or a summary peer is reached, the FIND message is stopped (*i.e.* Line 27).

Note that peers exchange summaries that are produced using local Background Knowledges (BKs). In our work, we consider the two following assumptions. First, each peer p owns some tuples in a global, horizontally partitioned relation R. Second, users that are willing to cooperate agree on a Background Knowledge BK, which represents their common perception of the domain. Thus, here we do not address the problem of semantic heterogeneity among peers, since it is a separate P2P issue on its own. Besides, our work mainly targets collaborative database applications where the participants are supposed to work on "related" data. In such a context, the number of participants is also supposed to be limited, and thus the assumption of a common BK seems not to be a strength constraint. An example of such BK in a medical collaboration is the Systematized Nomenclature of Medicine Clinical Terms (SNOMED CT) [3], which provides a common language that enables a consistent way of capturing, sharing and aggregating health data across specialties and sites of care. On the other hand, our summaries are data structures that respect the original data schemas (Saint-Paul *et al.*, 2005). Hence, we can assume that the techniques that have been proposed to deal with information integration in P2P systems (e.g. (Tartinov *et al.*, 2003), (Akbarinia *et al.*, 2006)) can be used here to overcome the heterogeneity of both data and summary representations.

3. http://www.snomed.org/snomedctn

Algorithm 2 *Global Summary Construction*

```
1: // Definition of different types of messages
2: SUMPEER=⟨sender⟩ ⟨id, h, TTL⟩; FIND=⟨sender⟩ ⟨h, TTL⟩
3: LOCALSUM=⟨sender⟩ ⟨LS⟩; DROP=⟨sender⟩ ⟨⟩
4: // Treatment of messages
5: Switch msg.type
6: // Receiving information about a summary peer
7: Case (SumPeer):
8:     msg.h++; msg.TTL−−
9:     if (this.SumPeer=null) or (this.SumPeer.h>msg.h) then
10:        if (this.IsPartner) then
11:            Send DROP message to this.SumPeer.id
12:        end if
13:        this.SumPeer:= ⟨msg.id, msg.h⟩
14:        LOCALSUM:= new msg (this.LS); Send LOCALSUM to msg.sender
15:        IsPartner:= True
16:    end if
17:    if msg.TTL > 0 then
18:        Send msg to all neighbors
19:    end if
20: end Case
21: // Searching for a summary peer
22: Case (Find):
23:    msg.TTL−−; msg.h++
24:    if (this.Is_SumPeer) then
25:        PEERSUM:= new msg (this.id, msg.h, 1); Send PEERSUM to msg.sender
26:    else
27:        if (this.SumPeer ≠ null) then
28:            PEERSUM:= new msg (this.SumPeer.id, (msg.h + this.SumPeer.h), 1)
29:            Send PEERSUM to msg.sender
30:        else
31:            if (msg.TTL > 0) then
32:                p′:= highest degree peer in N(p); Send msg to p′
33:            end if
34:        end if
35:    end if
36: end Case
37: // arrival of a new partner
38: Case (LocalSum):
39:    CoopList.add (msg.sender, 0); GlobalSum:= merge (GlobalSum, msg.LS)
40: end Case
41: // departure of a partner
42: Case (Drop):
43:    CoopList.remove (msg.sender)
44: end Case
```

4.2. *Summary maintenance*

A critical issue for any indexing structure is to maintain the index, relative to the current data instances, without incurring high costs. For a local summary, it has been demonstrated that the summarization process guarantees an incremental maintenance, using a *push* mode for exchanging data with the DBMS, while performing with a low complexity (Saint-Paul *et al.*, 2005). In this section, we propose a strategy for maintaining a global summary in a given domain, based on both *push* and *pull* techniques, in order to minimize the number of messages exchanged in the system. The algorithm is divided into two phases: data modification and summary reconciliation.

4.2.1. *Push: data modification*

Let GS be a global summary and P_{GS} the set of its partner peers. Each peer in P_{GS} is responsible for refreshing its own element in the GS's cooperation list. A partner peer p observes the modification rate issued on its local summary LS. When LS is considered as enough modified, the peer p sets its freshness value v to 1, through a *push message* to the corresponding summary peer SP. The value 1 indicates that the local summary version being merged while constructing GS does not correspond any more to the current instance of the database. An important feature is that the frequency of push messages depends on modifications issued on local summaries, rather than on the underlying databases. It has been demonstrated in (Saint-Paul *et al.*, 2005) that, after a given process time, a summary hierarchy becomes very stable. As more tuples are processed, the need to adapt the hierarchy decreases and hopefully, once all existing attribute combinations have been processed, incorporating new tuple consists only in sorting it in a tree. A summary modification can be detected by observing the appearance/disappearance of descriptors in summary intentions.

4.2.2. *Pull: summary reconciliation*

The summary peer SP, in its turn, observes the fraction of old descriptions (*i.e.* number of ones) in the cooperation list. Whenever this fraction exceeds a threshold value (*i.e.* our system parameter), the global summary GS must be refreshed. In that case, SP pulls all the partner peers to merge their current local summaries into the new version of GS, which will be then under reconstruction. The algorithm is described as follows. SP initiates a reconciliation message that contains a new summary *NewGS* (initially empty). The message is propagated from a partner to another (started at SP). When a partner *p* receives this message, it first merges NewGS with its local summary. Then, it sends the message to another partner (chosen from the cooperation list CL). If *p* is the last visited peer, it sends the message to SP who will store the new version of the gobal summary. All the freshness values in CL are reset to zero. This strategy distributes the charge of summary merging on all partners, instead of imposing on SP to receive all local summaries and to make the merging calculations alone. Furthermore, this strategy guarantees a high availability of the global summary, since only one update operation is performed at the end by SP.

4.3. *Peer dynamicity*

In large P2P systems, a peer connects mainly to download some data and may leave the system without any constraint. Therefore, the shared data can be submitted with a low modification rate, while the rate of node arrival/departure is very important. We now study the effect of this peer dynamicity on our summary management algorithms.

4.3.1. *Partner peer arrival/departure*

In unstructured P2P systems, when a new peer *p* joins the system, it contacts some existing peers to determine the set of its neighbors. If one of these neighbors is a partner peer, *p* sends its local summary LS to the corresponding summary peer SP, and thus becomes a new partner in the SP's domain. SP adds a new element to the cooperation list with a freshness value v equal to one. Recall that the value 1 indicates the need of pulling peer p to get new data descriptions. When a partner peer *p* decides to leave the system, it first sets its freshness value v to two in the cooperation list, through a push message. This value reminds the participation of the disconnected peer *p* to the corresponding global summary, but also indicates the unavailability of the original data. There are two alternatives to deal with such a freshness value. First, we can keep the data descriptions and use it, when a query is approximately answered using the global summary. A second alternative consists in considering the data descriptions as expired, since the original data are not accessible. Thus, a partner departure will accelerate the summary reconciliation. In the rest of this work, we adopt the second alternative and consider only a *1*-bit freshness value v: value 0 to indicate the freshness of data descriptions, and value 1 to indicate either their expiration or their unavailability. However, if peer *p* failed, it could not notify its summary peer by its departure. In that case, its data descriptions will remain in the global summary until a new summary reconciliation is executed. The reconciliation algorithm does not require the participation of a disconnected peer. The global summary GS is reconstructed, and descriptions of unavailable data will be then omitted.

4.3.2. *Summary peer arrival/departure*

In Section 3, we have presented our IS_SUMPEER function that is executed at each peer to decide whether it is a summary peer or not. This function is based on node connectivity, and thus variations of node degrees may incur modifications in the function results. Therefore, we suppose that a peer executes periodically the IS_SUMPEER function. However, we believe that the results of the IS_SUMPEER function do not change frequently. In fact, connections in P2P systems tend to be formed preferentially because peers tend to discover high-degree nodes in the network overlay (Saroiu *et al.*, 2002). Besides, although the nodes join and leave the network with a high rate, the total number of nodes does not significantly change and almost remains the same. Thus, the cases in which a summary peer becomes an ordinary peer, or a node is submitted to a significant degree variation rarely occur. However, when a new highly-available peer attracts many peer connections and becomes a summary peer, it simply diffuses this information as described in section 4.1, and a new domain starts to appear

around it. When a summary peer SP decides to leave the system, it sends a release message to all its partners. Upon receiving such a message, a partner p makes a selective walk to find a new summary peer. However, if SP failed, it could not notify its partners. A partner p who has tried to send push or query messages to SP will detect its departure and thus search for a new one.

5. Query processing

In this section, we describe how a query Q, posed at a peer p, is processed. Peer p first sends Q to the summary peer SP of its domain. SP proceeds then to query the available global summary GS. Summary querying allows to achieve two distinct tasks depending on the user/application requirements: *peer localization* to return the original results, and *approximate answering* to return approximate answers. Summary querying is divided into two phases: 1) query reformulation and 2) query evaluation.

5.1. *Query reformulation*

First, a selection query Q must be rewritten into a flexible query Q^* in order to be handled by the summary querying process. For instance, consider the following query Q on the Patient relation in Table 1: SELECT AGE FROM PATIENT WHERE BMI < 19 AND DISEASE = "ANOREXIA". This phase replaces the original value of each selection predicate by the corresponding descriptors defined in the Background Knowledge (*BK*). Therefore, the above query is transformed to Q^*: SELECT AGE FROM PATIENT WHERE BMI IN UNDERWEIGHT,NORMAL AND DISEASE = "ANOREXIA". Let QS (resp.QS^*) be the *Query Scope* of query Q (resp.Q^*) in the domain, that is, the set of peers that should be visited to answer the query. Obviously, the query extension phase may induce false positives in query results. To illustrate, a patient having a BMI value of 20 could be returned as an answer to the query Q^*, while the selection predicate on the attribute `BMI` of the original query Q is not satisfied. However, false negatives cannot occur, which is expressed by the following inclusion: $QS \subseteq QS^*$. In the rest of this paper, we suppose that a user query is directly formulated using descriptors defined in the BK (*i.e.* $Q = Q^*$). Thus, we eliminate potential false positives that may result from query extension.

5.2. *Query evaluation*

This phase deals with matching a set of summaries organized in a hierarchy S, against the query Q. The query is transformed into a logical proposition P used to qualify the link between each summary and the query. Proposition P is under a conjunctive form in which all descriptors appears as literals. In consequence, each set of descriptors yields on corresponding clause. For instance, the above query Q is transformed to $P =$ *(underweight OR normal) AND (anorexia)*. A valuation function

has been defined to valuate the proposition P in the context of a summary z. Then, a selection algorithm performs a fast exploration of the hierarchy and returns the set Z_Q of most abstract summaries that satisfy the query. For more details see (Voglozin *et al.*, 2004). Once Z_Q determined, the evaluation process can achieve two distinct tasks: 1) Peer localization, and 2) Approximate answering.

5.2.1. *Peer localization*

Since the extended definition of a summary node z provides a peer-extent, *i.e.* the set of peers P_z having data described by its intent (see Definition 3), we can define the set of relevant peers P_Q as follows: $P_Q = \{\cup_{z \in Z_Q} P_z\}$. The query Q is directly propagated to these relevant peers. However, the efficiency of this query routing depends on the completeness and the freshness of summaries, since stale answers may occur in query results. We define a *False Positive* as the case in which a peer p belongs to P_Q and there is actually no data in the p source that satisfies Q (*i.e.* $p \notin QS$). A *False Negative* is the reverse case in which p does not belong to P_Q, whereas there exists at least one tuple in its data source that satisfies Q (*i.e.* $p \in QS$).

5.2.2. *Approximate answering*

A distinctive feature of our approach is that a query can be processed entirely in the summary domain. An approximate answer can be provided from summary descriptions, without having to access original, distributed database records. The selected summaries Z_Q are aggregated according to their interpretation of proposition P: summaries that have the same required characteristics on all predicates (*i.e. BMI* and *disease*) form a class. The aggregation in a given class is a union of descriptors: for each attribute of the selection list (*i.e. age*), the querying process supplies a set of descriptors which characterize summaries that respond to the query through the same interpretation (Voglozin *et al.*, 2004). For example, according to Table 2, the output set obtained for the two classes $\{$*underweight*, *anorexia*$\}$, and $\{$*normal*, *anorexia*$\}$ is *age* = $\{$*young*$\}$. In other words, *all* patients diagnosed with *anorexia* and having an *underweight* or *normal* BMI are *young* patients. In the case where exact answers are required, suppose now that processing a query Q in a given domain d_i returns C_i results, while the user requires C_t results. Obviously, when C_i is less than C_t, the query should be propagated to other domains. To this end, we adopt the following variation of the flooding mechanism.

Let P_i the subset of peers that have answered the query Q in the domain d_i: $|P_i| = (1 - FP) \cdot |P_Q|$, where FP is the fraction of false positives in query results. The query hit in the domain is given by: $(|P_i| \,/\, |d_i|)$. As shown by many studies, the existing P2P networks have small-world features (Iamnitchi *et al.*, 2002). In such a context, users tend to work in groups. A group of users, although not always located in geographical proximity, tends to use the same set of resources (*i.e. group locality* property). Thus, we assume that the probability of finding answers to query Q in the neighborhood of a relevant peer in P_i, is very high since results are supposed to be *nearby*. This probability is also high in the neighborhood of the originator peer p

since some of its neighbors may be interested in the same data, and thus have cached answers to similar queries. Such assumptions are even more relevant in the context of interest-based clustered networks. Therefore, the summary peer SP_i of domain d_i sends a flooding request to each peer in P_i as well as to peer p. Upon receiving this request, each of those peers sends the query to its neighbors that do not belong to its domain, with a limited value of TTL. Once a new domain is reached or TTL becomes zero, the query is stopped. Besides, the summary peer SP sends the request to the set of summary peers it knows in the system. This will accelerate covering a large number of domains. In each visited domain, the query is processed as described above. When the number of query results becomes sufficient, or the network is entirely covered, the query routing is terminated.

6. Performance evaluation

In this section, we devise a simple model of the summary management cost. Then, we evaluate and analyze our model through simulation.

6.1. *Cost model*

A critical issue in summary management is to trade off the summary updating cost against the benefits obtained for queries.

6.1.1. *Summary update cost*

Here, our first undertaking is to optimize the update cost while taking into account *query accuracy*. In the next section, we discuss query accuracy which is measured in terms of the percentage of false positives and false negatives in query results. The cost of updating summaries is divided into: usage of peer resources, *i.e.* time cost and storage cost, and the traffic overhead generated in the network.

6.1.1.1. Time cost

A unique feature of SAINTETIQ is that the changes in the database are reflected through an incremental maintenance of the summary hierarchy. The time complexity of the summarization process is in $O(n)$ where n is the number of tuples to be incorporated in that hierarchy (Saint-Paul *et al.*, 2005). For a global summary update, we are concerned with the complexity of merging summaries. The MERGING method that has been proposed is based on the SAINTETIQ engine. This method consists in incorporating the leaves of a given summary hierarchy S_1 into an another S_2, using the same algorithm described by the SAINTETIQ summarization service (referenced in Section 2.2.3). It has been proved that the complexity C_{M12} of the MERGING(S_1, S_2) process is constant w.r.t the number of tuples. More precisely, C_{M12} depends on the maximum number of leaves of S_1 to incorporate into S_2. However, the number of leaves in a summary hierarchy is not an issue because it can be adjusted by the user

according to the desired precision. A detailed Background Knowledge (BK) will lead to a greater precision in summary description, with the natural consequence of a larger summary. Moreover, the hierarchy is constructed in a top-down approach and it is possible to set the summarization process so that the leaves have any desired precision.

6.1.1.2. Storage cost

We denote by k the average size of a summary z. In the average-case assumption, there are $\sum_{i=0}^{d} B^i = (B^{d+1} - 1)/(B - 1)$ nodes in a B-arity tree with d, the average depth of the hierarchy. Thus the average space requirement is given by: $C_m = k.(B^{d+1} - 1)/(B - 1)$. Based on real tests, $k = 512$ bytes gives a rough estimation of the space required for each summary. An important issue is that the size of the hierarchy is quite related to its stabilization (*i.e.* B and d). As more cells are processed, the need to adapt the hierarchy decreases and incorporating a new cell may consist only in sorting a tree. Hence, the structure of the hierarchy remains stable and no additional space is required. On the other hand, when we merge two hierarchies S_1 and S_2 having sizes of C_{m1} and C_{m2} respectively, the size of the resultant hierarchy is always in the order of the $max\,(C_{m1}, C_{m2})$. However, the size of a summary hierarchy is limited to a maximum value which corresponds to a maximum number of leaves that cover all the possible combinations of the BK descriptors. Thus, storing the global summary at the summary peer is not a strength constraint. According to the above discussion, the usage of peer resources is optimized by the summarization process itself, and the distribution of summary merging operation. Thus, we restrict our focus to the traffic overhead generated in the network.

6.1.1.3. Network traffic

Recall that there are two types of exchanged messages: *push* and *reconciliation*. Let local summaries have an average lifetime of L seconds in a given global summary. Once L expired, the node sends a (push) message to update its freshness value v in the cooperation list CL. The reconciliation algorithm is then initiated whenever the following condition is satisfied: $\sum_{v \in CL} v/|CL| \geq \alpha$, where α is a threshold that represents the ratio of old descriptions tolerated in the global summary. During reconciliation, only one message is propagated among all partner peers until the new global summary version is stored at the summary peer SP. Let F_{rec} be the reconciliation frequency. The update cost is: $C_{up} = 1/L + F_{rec}$ messages per node per second. In this expression, $1/L$ represents the number of push messages which depends either on the modification rate issued on local summaries or the connection/disconnection rate of peers in the system. Higher is the rate, lower is the lifetime L, and thus a large number of push messages are entailed in the system. F_{rec} represents the number of reconciliation messages which depends on the value of α. This threshold is our system parameter that provides a trade-off between the cost of summary updating and query accuracy. If α is large, the update cost is low since a low frequency of reconciliation is required, but query results may be less accurate due both to false positives stemming from the descriptions of non existent data, and to false negatives due to the loss of relevant data descriptions whereas they are available in the system. If α is small, the

update cost is high but there are few query results that refer to data no longer in the system, and nearly all available results are returned by the query.

6.1.2. *Query cost*

When a query Q is posed at a peer p, it is first matched against the global summary available at the summary peer SP of its domain, to determine the set of relevant peers P_Q. Then, Q is directly propagated to those peers. The query cost in a domain d is given by: $C_d = (1+|P_Q|+(1-FP)\cdot|P_Q|)$ messages, where $(1-FP)\cdot|P_Q|$ represents the query responses messages (*i.e.* query hit in the domain). Here we note that, the cooperation list CL associated with a global summary provides information about the relevance of each database description. Thus, it gives more flexibility in tuning the *recall/precision* trade-off of the query answers in domain d. The set of all partner peers P_H in CL can be divided into two subsets: $P_{old} = \{p \in P_H \mid p.v = 1\}$, the set of peers whose descriptions are considered old, and $P_{fresh} = \{p \in P_H \mid p.v = 0\}$ the set of peers whose descriptions are considered fresh according to their current data instances. Thus, if a query Q is propagated only to the set $V = P_Q \cap P_{fresh}$, then precision is maximum since all visited peers are certainly matching peers (no false positives), but recall depends on the fraction of false negatives in query results that could be returned by the set of excluded peers $P_Q \backslash P_{fresh}$. On the contrary, if the query Q is propagated to the extended set $V = P_Q \cup P_{old}$, the recall value is maximum since all matching peers are visited (no false negatives), but precision depends on the fraction of false positives in query results that are returned by the set of peers P_{old}. Now we consider that the selectivity of query Q is very high, such that each relevant peer has only one result tuple. Thus, when a user requires C_t tuples, we have to visit C_t relevant peers. The cost of inter-domain query flooding is given by: $C_f = ((1-FP)\cdot|P_Q|+2)\cdot\sum_{i=1}^{TTL} k^i$ messages, where k is the average degree value in an unstructured P2P system (e.g. average degree of 3.5, similar to Gnutella-type graphs). Remember that, the set of relevant peers who have answered the query (*i.e.* $(1-FP)\cdot|P_Q|$), the originator and the summary peers participate to query flooding. In this expression, we consider that a summary peer has on average k long-range links. As a consequence, the total cost of a query is:

$$C_Q = C_d \cdot \frac{C_t}{(1-FP)\cdot|P_Q|} + C_f \cdot (1 - \frac{C_t}{(1-FP)\cdot|P_Q|}) \quad [1]$$

In this expression, the term $C_t/((1-FP)\cdot|P_Q|)$ represents the number of domains that should be visited. For example, when $C_t = ((1-FP)\cdot|P_Q|)$, one domain is sufficient and no query flooding is required.

6.2. *Simulation*

We evaluated the performance of our solutions through simulation, based on the above cost model. First, we describe the simulation setup. Then we present simulation results to evaluate various performance dimensions and parameters: scale up, query accuracy, effect of the freshness threshold α.

Table 3. *Simulation parameters*

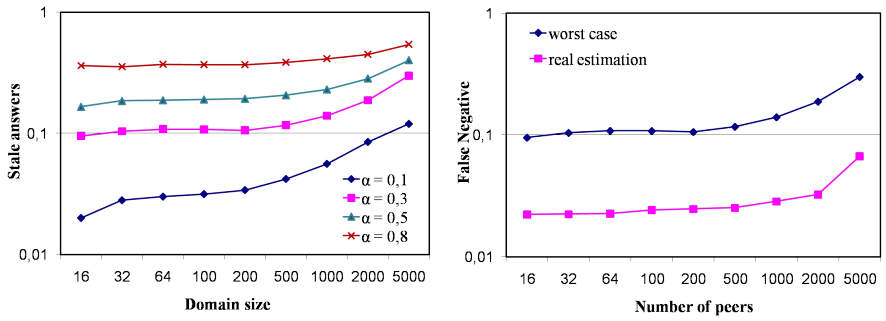
Parameter	value
local summary lifetime L	skewed distribution, Mean=3h, Median=1h
number of peers n	16–5000
number of queries q	200
matching nodes/query *hits*	10%
freshness threshold α	0.1–0.8

6.2.1. *Simulation setup*

We used the SimJava package (Howell *et al.*, 1998) and the BRITE universal topology generator to simulate a power law P2P network, with an average degree of 4. The simulation parameters are shown in Table 6.2.1. In our tests, we consider that local summary lifetimes are quite related to the node lifetimes, since the rate of node connection/disconnection is supposed to be greater than the modification rate issued on local summaries, and this for two reasons. First, in large P2P systems, we mainly deal with selection queries to locate and download required data. Thus, the original data are submitted to a low modification rate. Second, our summaries are even more stable than the original data (as we discussed before). Thus, the volatility of peers is, in reality, the main reason for a global summary reconciliation. Under this assumption, we consider that local summary lifetimes, like node lifetimes, follow a skewed distribution with a mean lifetime of 3 hours, and a median lifetime of 60 minutes. Our workload has 200 queries. The query rate is 0.00083 queries per node per second (one query per node per 20 minutes) as suggested in (Yang *et al.*, 2001). Each query is matched by 10% of the total number of peers. Finally, Our system parameter α that decides of the reconciliation frequency varies between 0.1 and 0.8.

6.2.2. *Update cost*

In this set of experiments, we quantify the trade-off between query accuracy and the cost of updating a global summary in a given domain. Figure 5(a) depicts the fraction of stale answers in query results for different values of the threshold α. Here, we illustrate the worst case. For each partner peer *p* having a freshness value equal to 1, if it is selected in the set P_Q then it is considered as false positive. Otherwise, it is considered as false negative. However, this is not the real case. Though it has a freshness value equal to 1, the peer *p* does not incur stale answers unless its database is changed relative to the posed query *Q*. Thus, Figure 5(a) shows the worst, but very reasonable values. For instance, the fraction of stale answers is limited to 11% for a network of 500 peers when the threshold α is set to 0.3 (30% of the peers are tolerated to have old/non existent descriptions). Moreover, simulation results (not presented here due to space limitations) have shown that the maximum domain sizes obtained in our self-organized network are approximatively less than 25% of the total number of peers. Thus, in Figure 5(a), the fraction of stale answers measured for a domain size of 256 peers, corresponds to a network of size 1024.



(a) Stale answers vs. domain size

(b) False negative vs. domain size

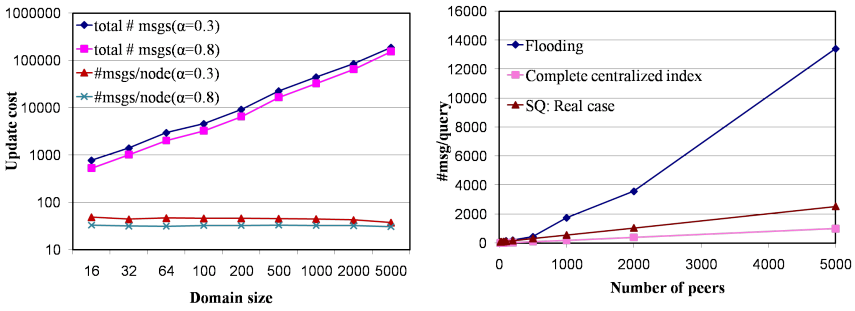
Figure 5. *Stale Answers*

As mentioned in Section 6.1.2, if we choose to propagate the query only to the set $V = P_Q \cap P_{fresh}$ we eliminate the possible false positives in query results. However, this may lead to additional false negatives. Figure 5(b) shows the fraction of false negatives in function of the domain size. Here we take into account the probability of the database modification relative to the query, for a peer having a freshness value equal to 1. We see that the fraction of false negatives is limited to 3% for a domain size less than 2000 (*i.e.* network size less than 8000). The real estimation of stale answers shows a reduction by a factor of 4.5 with respect to the preceded values. Figure 6(a) depicts the update cost in function of the domain size, and this for two threshold values. The total number of messages increases with the domain size, but not surprisingly, the number of messages per node remains almost the same. In the expression of C_{up}, the number of push messages for a given peer is independent of domain size. More interestingly, when the threshold value decreases (from 0.8 to 0.3) we notice a little cost increasing of 1.2 on average. For a domain of 1000 peers, the update cost increases from 0.01056 to 0.01296 messages per node per minute (not shown in figure). However, a small value of the threshold α allows to reduce significantly the fraction of stale answers in query results, as seen in Figure 5(a). We conclude therefore that tuning our system parameter, *i.e.* the threshold α, do not incur additional traffic overhead, while improving query accuracy.

6.2.3. *Query cost*

In this set of experiments, we compare our algorithm for query processing against centralized-index and pure non-index/flooding algorithms. A centralized-index approach is very efficient since a single message allows locating relevant data. However, a central index is vulnerable to attack and it is difficult to keep it up-to-date. Flooding algorithms are very used in real life, due to their simplicity and the lack of complex state information at each peer. A pure flooding algorithm consists in broadcasting the

query in the network till a stop condition is satisfied, which may lead to a very high query execution cost. Here, we limit the flooding by a value 3 of TTL. According to Table 6.2.1, the query hit is 10% of the total number of peers. For our query processing approach, which is mainly based on summary querying (SQ), we consider that each visited domain provides 10% of the number of relevant peers (*i.e.* 1% of the network size). In other words, we should visit 10 domains for each query Q. From equation 6.1.2, we obtain: $C_Q = (10 \cdot C_d + 9 \cdot C_f)$ messages. Figure 6(b) depicts the number of exchanged messages to process a query Q, in function of the total number of peers. The centralized-index algorithm shows the best results that can be expected from any query processing algorithm, when the index is complete and consistent, *i.e.* the index covers the totality of data available in the system, and there are no stale answers in query results. In that case, the query cost is: $C_Q = 1+2\cdot((0.1)\cdot n)$ messages, which includes the query message sent to the index, the query messages sent to the relevant peers and the query response messages returned to the originator peer p. In Figure 6(b), we observe that our algorithm SQ shows good results by significantly reducing the number of exchanged messages, in comparison with a pure query flooding algorithm. For instance, the query cost is reduced by a factor of 3.5 for a network of 2000 peers, and this reduction becomes more important with a larger-sized network. We note that in our tests, we have considered the worst case of our algorithm, in which the fraction of stale answers of Figure 5(a) occurs in query results (for $\alpha = 0.3$).



(a) Update cost vs. domain size

(b) Query cost vs. number of peers

Figure 6. *Update and Query Cost*

7. Related work

Current works on P2P systems aim to employ *content-based* routing strategies, since the content of data can be exploited to more precisely guide query propagation. These strategies require gathering information about the content of peer's data. However, the limits on network bandwidth and peer storage, as well as the increasing amount of shared data, call for effective summarization techniques. These techniques

allow exchanging compact information on peer's content, rather than exchanging original data in the network. Existing P2P systems have used keyword-based approaches to summarize text documents. For instance, in (Crespo *et al.*, 2002) documents are summarized by keyword vectors, and each node knows an approximate number of documents matching a given keyword that can be retrieved through each outgoing link (*i.e.* Routing Indices RIs). Although the search is very bandwidth-efficient, RIs require flooding in order to be created and updated, so the method is not suitable for highly dynamic networks. Other works (e.g. (Acuna *et al.*, 2003)) investigate Vector Space Model (VSM) and build *Inverted Indexes* for every keyword to cluster content. In this model, documents and queries are both represented by a vector space corresponding to a list of orthogonal term vectors called *Term Vector*. The drawback of VSM is its high cost of vector representations in case of P2P churns. In (Shen *et al.*, 2004), a *semantic-based* content search consists in combining VSM to Latent Semantic Index (LSI) model to find semantically relevant documents in a P2P network. This work is based on hierarchical summary structure over hybrid P2P architecture, which is closely related to what we are presenting in this paper. However, instead of representing documents by vector models, we describe structured data (*i.e.* relational database) by synthetic summaries that respect the original data schema. To the best of our knowledge, none of the P2P summarization techniques allows for an *approximate query answering*. All works have focused on facilitating content-based query routing, in order to improve search efficiency. We believe that the novelty of our approach relies on the fact that our data summaries allow for a semantic-based query routing, but also for approximately answering the query using their intentional descriptions.

8. Conclusion

We proposed a model for summary management in unstructured P2P systems. The innovation of this proposal consists in combining the P2P and database summarization paradigms, in order to support data sharing on a world wide scale. The database summarization approach that we proposed allows for data localization as well as for data description. We made two main contributions. First, we defined a new function for organizing the network into domains, in order to distribute summaries built over the shared data. Second, we proposed efficient algorithms for summary management in a given domain. Our performance evaluation showed that the cost of query routing in the context of summaries is significantly reduced in comparison with flooding algorithms, without incurring high costs of summary maintenance.

9. References

Acuna F., Peery C., Martin R., Nguyen T., " PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities", *HPDC-12*, 2003.

Adamic L., *et al*, " Search in power law networks", *Physical Review E*, 2001.

Aiello W., Chung F., Lu L., " A random graph model for massive graphs", *Proc of the thirty-second annual ACM symposium on Theory of computing (STOC)*, 2000.

Akbarinia R., Martins V., Pacitti E., Valduriez P., " Design and implementation of APPA", *Global Data Management (Eds. R. Baldoni, G. Cortese and F. Davide)*, IOS press, 2006.

Crespo A., Molina H., " Routing indices for peer-to-peer systems", *Proc. of the 28 tn Conference on Distributed Computing Systems*, July, 2002.

Ganesan P., Sun Q., Molina H., " Adlib: a self-tuning index for dynamic peer-to-peer systems", *Int. Conference on Data Engineering (*ICDE*)*, 2005.

Howell F., McNab R., " SimJava: a discrete event simulation package for java with the applications in computer systems modeling", *Int. Conf on Web-based Modelling and Simulation, San Diego CA, Society for Computer Simulation*, 1998.

Iamnitchi A., Ripeanu M., Foster I., " Locating Data in (Small-World?) Peer-to-Peer Scientific Collaborations", *IPTPS*, p. 232-241, 2002.

Lv Q., Cao P., Cohen E., Li K., Shenker S., " Search and replication in unstructured peer-to-peer networks", *ICS: international conference on Supercomputing*, 2002.

Raschia G., Mouaddib N., " A fuzzy set-based approach to database summarization", *Fuzzy sets and systems*, vol. 129, n° 2, p. 137-162, 2002.

Ripeanu M., Foster I., Iamnitchi A., " Mapping the gnutella network", *IEEE Internet Computing Journal*, 2002.

Saint-Paul R., Raschia G., Mouaddib N., " General purpose database summarization", *VLDB*, 2005.

Saroiu S., Gummadi P., Gribble S., " A Measurement Study of Peer-to-Peer File Sharing Systems", *Proc of Multimedia Computing and Networking (MMCN)*, 2002.

Sarshar N., Boykin P., Roychowdhury V., " Percolation Search in Power Law Networks: Making Unstructured Peer-to-Peer Networks Scalable", *P2P: International Conference on Peer-to-Peer Computing*, 2004.

Shen H., Shu Y., Yu B., " Efficient Semantic-Based Content Search in P2P Network", *IEEE Transactions on Knowledge and Data Engineering*, 2004.

Stoica I., Morris R., Karger D., Kaashoek M., Balakrishnan H., " Chord: A scalabale peer-to-peer lookup service for internet applications", *Proc ACM SIGCOMM*, 2001.

Tartinov I., *et al*, " The Piazza peer data management project", *SIGMOD*, 2003.

Thompson K., Langley P., " Concept Formation in Structured Domains", *Concept formation: Knowledge and experience in unsupervised learning*, Morgan Kaufmann, p. 127-161, 1991.

Voglozin W., Raschia G., Ughetto L., Mouaddib N., " Querying the SAINTETIQ Summaries–A First Attempt", *Int.Conf.On Flexible Query Answering Systems (FQAS)*, 2004.

Yang B., Molina H., " Comparing hybrid peer-to-peer systems", *Proc VLDB*, 2001.

Zadeh L., " Fuzzy Sets", *Information and Control*, vol. 8, p. 338-353, 1965.

Zadeh L., " Concept of a linguistic variable and its application to approximate reasoning-I", *Information Systems*, vol. 8, p. 199-249, 1975.

Zadeh L., " Fuzzy sets as a basis for a theory of possibility", *Fuzzy Sets and Systems*, vol. 100, p. 9-34, 1999.

ANNEXE POUR LE SERVICE FABRICATION
A FOURNIR PAR LES AUTEURS AVEC UN EXEMPLAIRE PAPIER
DE LEUR ARTICLE ET LE COPYRIGHT SIGNE PAR COURRIER
LE FICHIER PDF CORRESPONDANT SERA ENVOYE PAR E-MAIL

1. ARTICLE POUR LA REVUE :

RSTI - ISI – 13/2008

2. AUTEURS :

Rabab Hayek[*] *— Guillaume Raschia*[*] *— Patrick Valduriez*[**] *— Noureddine Mouaddib*[*]

3. TITRE DE L'ARTICLE :

Summary Management in Unstructured P2P Systems

4. TITRE ABRÉGÉ POUR LE HAUT DE PAGE MOINS DE 40 SIGNES :

Summary management in P2P systems

5. DATE DE CETTE VERSION :

November 5, 2008

6. COORDONNÉES DES AUTEURS :

– adresse postale :
Atlas Team (INRIA - LINA), University of Nantes
2 rue de la Houssiniere – B.P. 92208, 44300 Nantes, France
[*] surname.name@univ-nantes.fr
[**] Patrick.Valduriez@inria.fr
– téléphone : 00 00 00 00 00
– télécopie : 00 00 00 00 00
– e-mail : The Publisher

7. LOGICIEL UTILISÉ POUR LA PRÉPARATION DE CET ARTICLE :

LaTeX, avec le fichier de style `article-hermes.cls`,
version 1.23 du 17/11/2005.

8. FORMULAIRE DE COPYRIGHT :

Retourner le formulaire de copyright signé par les auteurs, téléchargé sur :
`http://www.revuesonline.com`

SERVICE ÉDITORIAL – HERMES-LAVOISIER
14 rue de Provigny, F-94236 Cachan cedex
Tél. : 01-47-40-67-67
E-mail : revues@lavoisier.fr
Serveur web : http://www.revuesonline.com