Enlarge Research in cSON, a Maximal Covering Tree-Based Approach

Gilles Nachouki[1], Mohamed Quafafou[2]
[1]LINA-UMR CNRS 2729
Nantes University, France
[2]LSIS-UMR CNRS 6168
Marseille University, France
gilles.nachouki@univ-nantes.fr, mohamed.quafafou@univmed.fr

ABSTRACT: *We propose to enlarge research in* cSON *(community Semantic Overlay Network) a semantic overlay network that requires an organization of peers into communities.* cSON *is developed for an efficient research in unstructured Peer-to-Peer (P2P) system. In this paper, a community is composed of one super-peer (e.g. the administrator of the community) and of several peers: the super-peer describes a domain (e.g. ontology) and a peer (with its own data schema) joins one community which is in accordance with its own domain. One challenge in* cSON *is how to efficiently extend the scope of the research to all communities (and not only to neighboring communities). We propose an algorithm that builds, starting from the administrator of a community, a Maximal-affinity Covering Tree (MCT). The obtained MCTs are used later by queries in order to search in each community the pertinent peers. We give a performance evaluation concerning the creation of a MCT and we compare then our routing algorithm with the one (called baseline) which consists to send queries to only neighboring communities which are able to treat them.*

Keywords: cSON, P2P, Maximal-Affinity Covering Tree, Queries Routing, Simulation

Received: 1 February 201, Revised 21 Marc 2012, Accepted 3 April 2012

1. Introduction

1.1 Overlay network

A Peer is an autonomous entity with a capacity of storage and data processing. In a computer network, a peer may act as a client or as a server. Peer-to-peer (P2P) systems is characterized as decentralized self-organizing and resource usage. P2P systems have emerged as a popular way to share huge volumes of data. P2P systems is build on top of the physical one (typically the Internet) and thus referred to as overlay network. The degree of decentralization and the topology of the overlay network strongly impact the properties of the P2P systems, such as fault-tolerance, self-maintainability, performance, scalability and security. For simplicity, two main classes of P2P networks are considered: unstructured and structured. Unstructured P2P networks, the overlay network is created in no deterministic manner and data placement is completely unrelated to the overlay topology. Each peer knows its neighbors, but does not know the resources they have. Query routing is typically done by

flooding the query to the peers that are in limited hop distance from the query originator. Structured networks have emerged to solve the scalability problem of unstructured networks. They achieve this goal by tightly controlling the overlay topology and data placement. Data are placed at precisely specified locations and mappings between data and their locations (file identifier is mapped to peer address) are provided in the form of a distributed routing table. Unstructured and structured P2P networks are considered 'pure' because all their peers provide the same functionality. In contrast, super-peer networks are hybrid between client-server systems and pure P2P networks. Like client-server systems, some peers, the super-peers, act as dedicated servers for some other peers and can perform complex functions such as indexing, query processing, access control, and meta-data management. Using only one super-peer reduces to client-server with all the problems associated with a single server. Like pure P2P networks, super-peers can be organized in a P2P fashion. Super-peers can be dynamically elected (e.g. based on bandwidth and processing power) and replaced in the presence of failures. Requirements for widely distributed information systems supporting virtual organizations have given rise to a new category of P2P systems called schema-based or PDMS (Peer Data Management System) where each peer exposes its own schema. A PDMS aim at overcoming the scalability problems of data integration systems by combining P2P and distributed database techniques.

1.2 Semantic overlay network

In unstructured P2P systems, queries flooding tends to be very expensive. In [2], a Semantic Overlay Network (SON) consists on creation of a network that improve query routing based on the semantic links between peers. In SON peers are organized into groups according to the content they share. Groups may overlap, since some peers are belong to different groups and queries are sent to relevant neighboring groups increasing the chances that matching files will be found quickly. A disadvantage of SON, specially when each peer share data item that respect a data schema, is that it requires to build semantic links between peers. To overcome the problem of matching schemas of peers, we consider in our work a system composed of peers and super-peers. Each super-peer is responsible of a domain and peers are organized around super-peers according to their domains. In this organization, schema of a peer is matched only with the domain (i.e. ontology) of its super-peer. Another disadvantage in *cSON* is that queries routing is restricted to only peers which are in the same group (or in neighboring groups) so that it limits the number of pertinent peers reached in the network.

1.3 Contribution

cSON [7] is a community Semantic Overlay network designed around peers and super-peers. In this architecture, super-peers are responsible of communities and peers are members of these communities where each peer is assigned to only one community. A super-peer (administrator of a community) joins the network with its schema (i.e. ontology) that describes a specific domain (the domains held by communities are supposed not disjoint). Super-peers match their respective schemas to discover their acquaintances for effective data sharing and peers (having their data schemas) are affected to communities according to their semantic domains. A peer formulates its query (using its own query language) and submits it directly to its community and this last forward this query to only neighboring communities which are able to treat this query. The major problem is how to efficiently enlarge research to all communities in cSON. To address this problem, we propose an algorithm that returns a *Maximal-affinity* Covering Tree (for simplicity we call this tree: *Maximal Covering Tree* or MCT) requested by a community. The vertices of this tree are (super-)peers (i.e. administrators of communities) and values associated to edges measure affinities (i.e. semantic links) between them. *MCT* is constructed under the condition that the affinity between all communities is maximal. The maximal affinity of a MCT is obtained by the sum of values of all the edges which compose it.

The advantages of this approach are twofold: 1. a community can choose to use the *MCT* of one of its neighboring communities or it can choose to build its own tree. In fact, the *MCT* of a community may be different from the *MCT* of its neighbors but the value of the *maximal affinity* is still the same and the uniqueness of the *MCT* in *cSON* is granted only when the values of edges -which connect the communities between them- are all distinct; 2. when a community receives a query, it forwards this query to other communities through its *MCT*: the query traverses all the communities without forming a cycle and exploiting the maximum affinity existing between communities. This feature has the effect of privileging communities that are more likely to respond to queries. For example, suppose that A, B and C are three communities in cSON where A is linked semantically to B and C: the affinity between A and B is equal to 2 (i.e. 2 elements in A and B are found similar) and the affinity between A and C is equal to 20 (i.e. 20 elements are found similar). It is obvious that, for a query Q received by the community A, C is more likely to answer Q than B. Compared to the baseline approach[1] in which a query is forwarded to only neighboring communities, our approach (MCT) extends efficiently this research of all communities. The following section summarizes related works to queries routing in unstructured P2P systems.

[1]*Baseline is the first approach implemented in cSON*

2. Related Work

In this section, we review some methods and approaches developed in the literature for queries routing in unstructured P2P systems. Queries routing in this context follows, in general, one of the three approaches: blind, informed or using shortcuts. In *Blind approach*, queries routing consists to forward the query to their neighbors. This approach floods the network with a TTL (Time To Live) which makes that the choice of routing is arbitrary. BFS (Breadth-first search) [20] follows blind approach. This approach is used in Gnutella [23] for data discovery, floods the query to all accessible peers within a TTL (Time To Live). By continuing this procedure, all accessible peers whose hop distance from the query originator is less than or equal to TTL receive the query. Each peer that receives the query executes it locally and returns the answers directly to the query originator.

Others methods follows the blind approach, we can cite: Modified BFS (MBFS) [11], Iterative Deeping (ID) [24], Random Walks (RW) [15] etc. In *Informed approach*, routing queries is ameliorate using diverse information i.e. information concerning neighbors (indices, tables etc.), information about the historic of queries, the probability of selection of a peer or semantic links. Intelligent BFS (IBFS) [11] follows the informed approach. Others methods are developed in this context, we can quote: APS (Adaptive Probabilistic Search) [21], LI (Local Indices) [3], DRLP (Distributed Resource Location Protocol) [17], BFBI (Bloom Filter based Indices) [1], RI (Routing indices) [3], HSON (Hierarchical Semantic Overlay Network) [12] etc. *Shortcuts* are created between peers in order to forward queries directly to peers that are able to process them. Shortcuts presents a considerable gain in efficiency of queries routing. REMINIDIN [22] follows an informed approach that uses shortcuts. Other approaches use shortcuts such as RPBS (Ranking Peers Based Shortcuts) [14] and IBS (Interest Based Shortcuts) [18].

In the following section we introduce briefly cSON. In section 4 we present the background of our approach and we give an algorithm that implies all the communities in order to build a *MCT*. Section 5 compares the results of our evaluations with the baseline approach and section 6 concludes and gives some future works.

3. Overview of cSON

3.1 Model

Formally, we define cSON as follows:

- $\mathcal{G} = \{(SP, L_{sp})\} \cup \{SP \cup P, L_p\}$ where
  - $SP = \{SP_1, \ldots, SP_n\}$ is a set of communities where each one describes its own schema,
  - $Lsp = \{(SP_i, SP_j) \mid SP_i, SP_j \mid \in SP^2\}$ is a set of semantic links (between communities),
  - $P = \{P_1, \ldots, P_m\}$ is a set of peers, each one describes its own schema and
  - $Lp = \{(SP_i, P_j) \mid SP_i \in SP, P_j \in P\}$ is a set of semantic links between peers and communities;
- $\mathcal{M} = \{\mathcal{M}^{ij}_{SP/SP}\} \cup \{\mathcal{M}^{kl}_{SP/P}\}$ is a set of correspondence matrices where each one is associated to $sl(i, j)$ or $sl(k, l) \in \{L_{sp}\} \cup \{L_p\}$ that saves semantic links between communities.
- $\varepsilon = \{E_{SP/SP}\} \cup \{E_{SP/P}\}$ is a set of expertise tables between communities ($E_{SP/SP}$) or between a community and its members ($E_{SP/p}$). An expertise describes a part of the schema of a community (or a peer) to share with other communities (or peers).

3.2 Semantic links

The purpose of cSON is the sharing of resources distributed on peers. Each peer is supposed to hold a database (or an XML document, etc..) with a data schema. A community provides a schema describing a specific domain to a group of peers where each peer is supposed belong to one community. We adopt a community network topology that combines the efficiency of centralized search with the autonomy, load balancing and robustness of distributed search. The domains of communities are not necessarily separated. In general, the search for correspondence between two schemas S_1 and S_2 consists to find for each concept in S_1 (or S_2) a correspondent in S_2 (or S_1) which is nearest semantically. We can define the concept of mapping (Map) between schemas as follows:

$$Map : S_1 \Rightarrow S_2, Map(c_{s1}) = c_{s2} \; if \; Sim(c_{s1}, c_{s2}) > \in_{acc} \quad (1)$$

where c_{s1}: element of schema S_1; c_{s2}: element of schema S_2; $\in_{acc}$ is the acceptable threshold; Sim(c_{s1}, c_{s2}) is a function, that measure the similarity between two concepts c_{s1} and c_{s2}, given as follows:

$$Sim : S_1 \; X \; S_2 \Rightarrow [0, 1] \quad (2)$$

We distinguish two particular cases: *Sim* (c_{s1}, c_{s2}) = 1 describes two similar elements; *Sim* (c_{s1}, c_{s2}) = 0 describes two distinct elements.

In this work, we use the similarity function proposed in [5]. The similarity function *Sim* (n_p, n_k) is a weighted function that sums linguistic similarity $S_l(n_p, n_k)$ and neighborhood similarity $S_v(n_p, n_k)$ of two nodes n_p and n_k. The coefficients λ_l and λ_v are respectively the weights associated to linguistic and neighborhood similarities of these elements.

$$Sim(n_p, n_k) = \lambda_l . S_l(n_p, n_k) + \lambda_v . S_v(n_p, n_k) \quad \text{where } \lambda_l, \lambda_v \geq 0 \quad \text{and } \lambda_l + \lambda_v = 0 \tag{3}$$

The linguistic similarity $S_l(n_p, n_k)$ of two elements n_p and n_k permits to evaluate the linguistic affinity between their names. We suppose that, the linguistic similarity $S_l(n_p, n_k)$ is the similarity $S_s(n_p, n_k)$ of the two sets of synonymous corresponding to n_p, n_k for which we add the similarity of types between them:

$$S_l(n_p, n_k) = \omega_s \times S_s(n_p, n_k) + \omega_t \times S_t(n_p, n_k) \tag{4}$$

The similarity $S_s(n_p, n_k)$ between the synonyms sets combines the measure proposed by Tversky [19] and the distance (ed) of Levenstein [16]. This is a convenient measure, but it takes into account only exact matching between terms. For example the terms *Numero_assurance* and *NumeroAssurance* are considered different with this formula, but in our case, we consider them equivalent to a nearly transformation. In [5] we proposed to extend it by making an approximation matching between terms present in sets of synonyms, rather than an exact match of those terms. So, given *A* and *B* two sets of synonyms, we define the intersection $A \cap_f B$ and the difference $A -_f B$ as follows:

$$A \cap_f B = \left\{ a \in A / \max_{b \in B} SM(a,b) > \in_{acc} \right\} \tag{5}$$

Algorithm 1 Generation of the correspondence matrix Super-Peer/Super-Peer

Require: Input : The schemas *S*1 and *S*2 of *SP*1 and *SP*2.
Output : Mat^{12}_{SPSP} : Correspondence matrix between
for all $n^1_g \in$ *NoeudInterne*(*S*1) **do**
  for all $n^2_g \in$ *NoeudInterne*(*S*2) **do**
    $x \leftarrow \omega_s \ sim_s(n^1_g, n^2_g)$
    $y \leftarrow sim_v(n^1_g, n^2_g)$
    $sim(n^1_g, n^2_g) \leftarrow \alpha \times x + (1 - \alpha) \times y$
    if $sim(n^1_g, n^2_g) > \in_{acc}$ **then**
      add (Mat^{12}_{SPSP},n^1_g, n^2_g,*Sim-Value*)
    else
      add (Mat^{12}_{SPSP}, n^1_g, n^2_g, *null*)
    end if
  end for
end for

$$A -_f B = \left\{ a \in A / \max_{b \in B} SM(a,b) \leq \in_{acc} \right\} \tag{6}$$

$$SM(a,b) = \max\left(0, \frac{\min(|a|,|b|) - \text{ed}(a,b)}{\min(|a|,|b|)}\right) \in [0,1] \tag{7}$$

The function *SM* computes the similarity of the strings, taking into account the number of atomic actions (add, deletion of character changes) needed to transform one of string in another string. The number of actions is calculated according to distance *ed* of Levenshtein. $\in_{acc}$ is the threshold above which the calculated similarity is considered acceptable. The similarity $S_s(n_p, n_k)$ between two sets of synonyms *syn* (n_p) and *syn* (n_k) of two nodes n_p et n_k is expressed as follows :

$$S_s\left(n_p, n_k\right) = \frac{X}{X + \alpha Y + (1-\alpha)Z} \; where X = |\, syn(n_p) \cap_f syn(n_k)\,|\; Y = |\, syn(n_p) -_f syn(n_k)\,|\; Z = |\, syn(n_k) -_f syn(n_p)\,| \quad (8)$$

The global similarity of two nodes depends on the similarity of their contexts. It is based on the following intuition: similar nodes are linked to similar nodes in two schemas. More specifically, the nodes n_1 and n_2 are similar if they are linked to respectively nodes m_1 and m_2 which are also similar. Let *S* be a *Schema* and $n_p \in S$, the neighborhood $V(n_p)$ of n_p in S is a node having a direct semantic link with n_p, let : $V(n_p) = \{n_i \in InternalNode(S) / \exists r \in S \wedge r(n_i, n_p)\}$. *InternalNode*(*S*) consists of the nodes set of document S and $r(n_i, n_p)$ means that there exists a direct link from n_i to n_p. The neighborhood similarity compares nodes based on their linguistic similarity (i.e. S_l function). By approximate matching of neighborhoods, we get the neighborhood similarity as follows:

$$S_v\left(n_p, n_k\right) = \frac{\left|\{x \in V(n_p) / \exists y \in V(n_k) \wedge S_l(x, y) > \in_{acc}\}\right|}{\left|V(n_p) \cup V(n_k)\right|} \quad (9)$$

Algorithme 1 describes an algorithm that generate the correspondence matrix between two super-peers given their respective ontologies.

Example* (*Semantic links*):* Consider the two following schemas S_1 and S_2 of two communities given in figure 1. A vertex in S_1 (or S_2) represents an element (i.e. concept) and a link between two elements represents a semantic link between them. In the first schema (S_1), the concept *Journals* concerns the *articles* published in journals. In the second schema S_2, *Proceedings* concerns the articles published in conferences.

In this example, we apply the similarity function given above in order to generate the corresponding matrix SP/SP. For example, the semantic links between the two elements *article* and *paper* is obtained as follows (for $\lambda_l = \lambda_v = \omega_s = \omega_t = 0.5$):

$$Sim\,(paper, article) = 0.5 \times S_l\,(paper, article) + 0.5 \times S_v\,(paper, article).$$

We consider $\alpha = 0.5$ in equation (8), the similarity of synonymous is expressed as follows:

$$Sim\,(paper, article) = \frac{|\{subject\}|}{|\{subject\}| + 0.5\,|\{paper\}| + 0.5\,|\{article\}|} = \frac{1}{2}.$$

Consequently $S_l\,(paper, article) = 0.5 \times \frac{1}{2} + 0.5 \times 1 = 0.75$ (the similarity of types is supposed equal to 1). The neighbors similarity $S_v\,(paper, article)$ can be expressed as follows:

$$S_v\,(paper, article) = \frac{|\{x \in V\,(paper) / \exists y \in V\,(article) \wedge S_s\,(x,y) > \in_{acc}\}|}{|V\{paper\} \cup V\{article\}|}$$

where $V\,(article) = \{\, journals\}$ and $V\,(paper) = \{proceedings\}$.

We find that $S_v(\, paper, article) = \frac{1}{2}$ and hence $Sim(paper, article) = 0.5 \times 0.75 + 0.5 \times \frac{1}{2} = 0.62 > \in_{acc} = 0.5$.

3.3 Semantic overlay network formation

In the context of SON, each peer should be able to advertise its base to other peers [13]. In our context, a new peer P_i advertises its content by sending to the super-peer backbone through its access point a membership advertisement $MA_i = (PID, E_i, T)$ that

contains the peer's ID *PID*, its topic of interest *T* and its expertise E_i. When the *godfather* (i.e. the super-peer) of P_i has not joined the system yet, the advertisement is cached by the access point so that it will be able to advertise the *orphan* peer when its *godfather* joins the network. While the *godfather* of its semantic domain has been found, the matching process is done by the *godfather* to find similarity links between the peer's *expertise* and its ontology/schema using equation 4. When the super-peer accepts the membership request, it stores the peer's expertise in its *ESP/P* expertise table and sends a membership approval to the peer P_i. A new super-peer SP_j advises its domain by sending to its neighbors a domain advertisement SDA_j = (*SID*, E_j , T_j, $\in_{acc}$,*TTL*) containing the Super-peer ID *SID*, the corresponding expertise E_j , the topic area of interest T_j , the minimum semantic similarity value required to establish semantic links between its expertise and those of other (super-) peers. When receiving a domain advertisement, a super-peer SP_r invokes the semantic matching process to find semantic links between its suggested expertise E_r and the received suggested expertise E_j. When the super-peer SP_r accepts the membership request it stores the super-peer's expertise E_j in its $E_{SP/SP}$ expertise table. The semantic links found are stored in SP_r's correspondence matrix $M_{SP/SP}$ and sent also to SP_j which can approve or reject them. If the collaboration has been accepted, each one stores the other's expertise in its expertise table $E_{SP/SP}$. SP_r forwards also the advertisement SDA_j to the peers it has early cached their advertisement to be interest by topic T_j.

Example (Network configuration): Figure 2 shows an example of the semantic network topology. In this figure the super-peer SP_A describes bibliographical references of a specific topic *Database*. This topic is part of the general topic *Information Systems*. The domain of the super-peer SP_A involves the ACM computing classification system (http://www.acm.org/about/class /ccs98-html). In the same way, SP_B describes the same domain but it is interested on another topic *Data mining* etc. In this example, a super-peer can support one or several topics (e.g. Logic and database). The interest's domain of peer P_1 is *Database*. P_1 sends a membership advertisement to its access point SP_B. This last forwards the messageMA_1 to SP_A which has joined yet the network. SP_A contacts P_1 in order to study the membership of P_1.

3.4 Affinity measures

A semantic domain appears when a new community SP_j joins the network with a suggested schema. First, the community defines its expertise (i.e. the part of the Schema to be share with the other communities) and publishes a domain advertisement. Semantic links between communities are established by using the *Similarity* function given in equation 4. The affinity function *Aff* is proposed in order to measure the semantic affinity between two communities SP_j and SP_r. This function is defined as follows:

$$Aff\left(M_{SP/SP}^{j,r}\right) = \sum_{M_{SP/SP}^{j,r}(n^{S_j}, m^{S_r})} Sim\left(n^{s_j}, m^{s_r}\right) \tag{10}$$

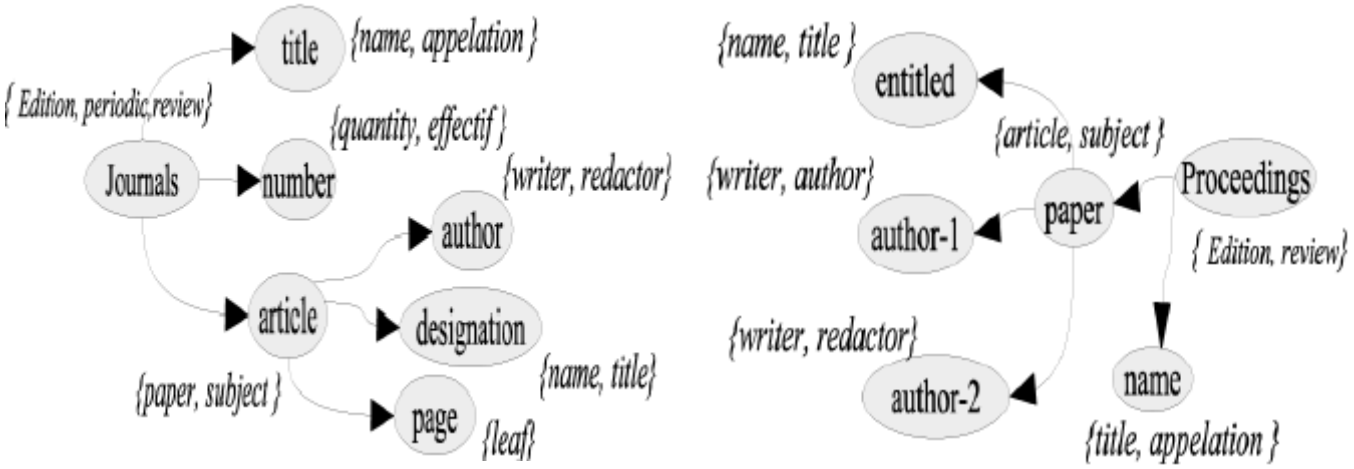
where n^{S_j} and m^{S_r}designate respectively two elements in schemas S_j and S_r and $M_{SP/SP}^{j,r}$ is the corresponding matrix concerning the two communities SP_j and SP_r. This matrix contains the results of matching of elements belonging respectively to S_j and S_r. *Aff* function returns the sum of values in $M_{SP/SP}^{j,r}$ that represents the similarities between elements.

3.5 Semantic queries routing (baseline)

Queries routing exploits the semantic overlay and the expertise's tables of the community for both intra-community and intercommunity query propagation. The set of expertise of communities are stored respectively in their expertise tables $E_{SP/SP}$ and $E_{SP/P}$. Assume that peer P_i of community SP_j , issues a query *Q* on its schema with its query language, we give briefly the principle of the main steps:

***Step 1*. Query subject extraction**. The query is first expressed in common format denoted Q_s (e.g. a graph model) [5]. Then, the resulting query Q_s is routed to the community which extracts the query subject. The subject of a query Q_s is an abstraction of the query in terms of nodes of the query.

***Step 2*. Peer selection**. A community selects each neighbor community (or each member of its community) which is able to treat



	Journal	title	number	article	author	Page	destination
Proceedings	0.70						
name		0.60					
Paper				0.62			
entitled							0.90
.author-1					0.70		

Figure 1. Semantic reconciliation between schemas/ontologies

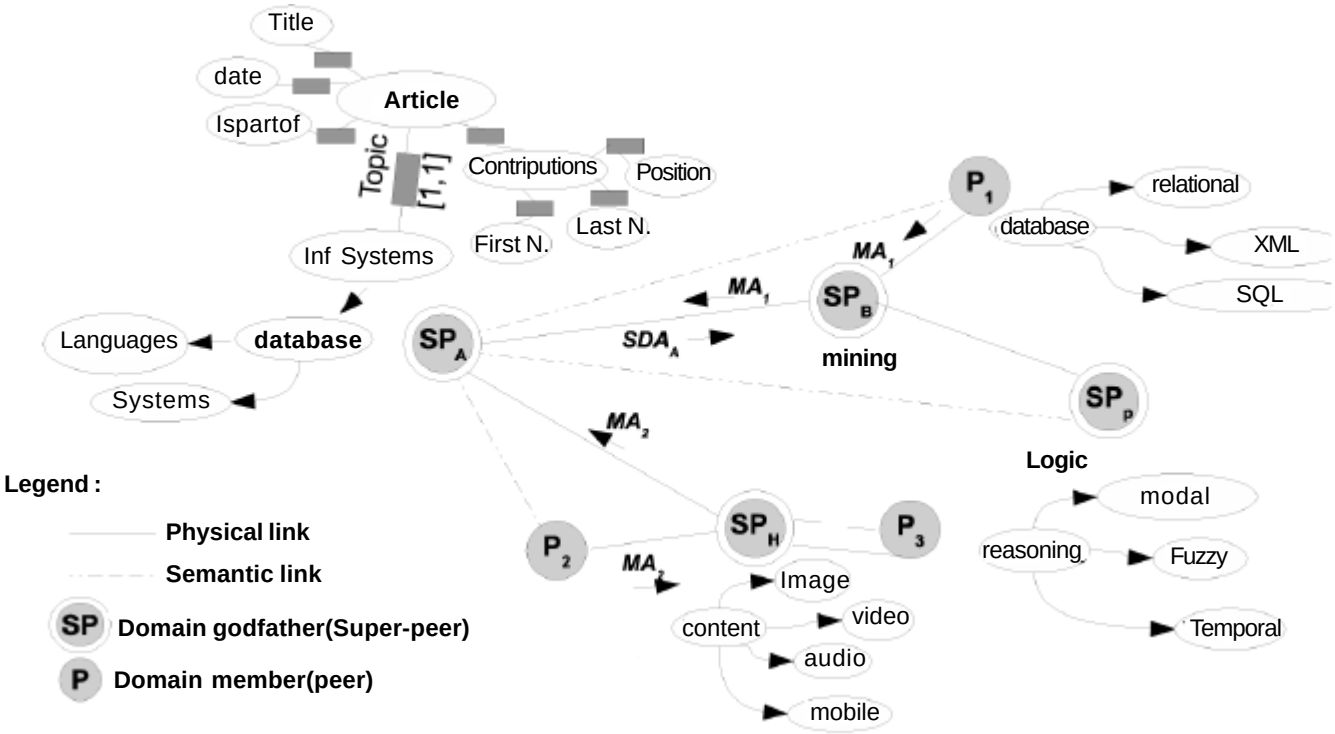


Figure 2. Semantic overlay network formation

a query by matching the query subject to the expertise of this community (or to the expertise of a member of its community). The selection is based on a function *Cap* that measures the capacity of a community (or a peer) of expertise *Exp*(*P*) on answering a given query of subject *Sub*(*Q*).

$$Cap(P,Q) = \frac{1}{|Sub(Q)|}\left(\sum_{s \in Sub(Q)} \max_{e \in Exp(P)} Sim(s,e)\right) \quad (11)$$

The function *Cap* is based on the similarity function *Sim* associated to the elements of *Sub*(*Q*) and *Exp*(*P*). Each selected community applies *Step 2* (peer selection) for finding promising peers for the incoming query *Q*. After peer selection, the original query with its structure is rewritten towards the selected peers.

***Step 3*. Query routing.** Once the set of relevant communities has been identified, the community SP_j of P_i sends the query to those promising peers or communities closed to them by using their ID, IP addresses and the underlying physical network.

4. Enlarge research in cSON

4.1 Background

In cSON each community is linked semantically to at least one other community and in addition we suppose that, for any two communities there exist a path that connect them. In this context, we modeled cSON as an edge-weighted connected and undirected graph G(*N*, *E*) where *N* is the set of vertices in G and *E* is the set of edges. The weight of each edge measures the affinity between two communities. The objective, in this paper, is to build a maximal covering tree associated to G such that the sum of affinities between communities is maximal. The aim of this tree, is to enlarge the research in cSON with a minimum exchange possible of messages between communities and a large number of relevant responses returned by the peers. Indeed, a query that follows this tree discovers other responses which can not be found by the baseline algorithm. Several algorithms are proposed in centralized settings in order to build a such tree. We can quote: Prim or Kruskal algorithms [8] [4]. The principle of these algorithms consist to find a covering tree with a minimum (or maximum) cost. The most important properties of a such tree are the following: *unique* and *acyclic*. The uniqueness of the tree is granted when the values associated to edges in G are all distinct. The uniqueness means that if we can find another covering tree *T'* then the cost of *T'* is not minimal (or maximal). When some values associated to edges in G are identical then different trees can be obtained with the same minimum (or maximum) cost; Acyclic ensures that each query is received by each community only one time. The complexity of Prim is given about O(| E | log(| N |)) where E is the number of edges and N is the number of vertices. The results of comparison, between Prim and others methods like Kruskal, shows that when the number of vertices is less than 100, judging from the space complexity, Kruskal algorithm is relatively superior to Prim algorithm [9]; however, when this number is greater than 100, Prim algorithm is more superior. In the rest of this paper we propose an implementation of the Prim algorithm in a context such as cSON. The proposed algorithm build a maximal-affinity covering tree that can be used later in order to efficiency enlarge the search to all communities. When a community receives a query, it forwards this query to other communities via the *MCT* without cycle (when browsing) and by exploiting the maximum links existing between communities in order to privilege communities that are more likely to respond to queries.

4.2 Construction of the MCT

In this section, we present briefly a basic version of the distributed algorithm then we present optimized version.

4.2.1 Basic algorithm

The principle of the proposed algorithm consists to ask each existing community in MCT to return its best neighbors (we suppose, at beginning, MCT contains the community SP_0, this step returns the community which have a maximal affinity with SP_0). The best returned neighbor (among existing communities in MCT) is added into MCT and this step is repeated until the end (i.e. all the communities in *G* are included in MCT). This version of the algorithm is called naive, since a community, with all its neighbors are already in the tree, is always asked to return its best neighbor. Therefore, the number of messages exchanged between communities increases dramatically with the augmentation of numbers of communities.

4.2.2 Optimized algorithm

This algorithm differs from the first version (naive version) by the fact that when a community *SP* returns its best neighbor (e.g. SP_b) and that this neighbor (SP_b) is the last one for *SP* which is selected to join the tree, then *SP* shouldn't receive more

messages later. Algorithm 2 shows this optimized version: *R* represents the maximal covering tree (MCT) where *S* is the set of vertices and *A* is the set of edges. We start from SP_0 (*i.e.R.S* contains SP_0 and R.A is ϕ). Responses contains the best neighbors returned by communities in R. *V* contains the set of communities in R which have at least one neighbor in *G* which is not yet in R. Initially, *V* is supposed contains SP_0 (i.e. SP_0 is not the only community in the graph) and *Responses* is initialized to empty. The algorithm asks each community in V to return its better neighbor (e.g. SP_0 is asked to return its best neighbor). The better returned neighbors are added to *Responses* (lines 6 to 13). Then, Line 14 orders the obtained communities in *Responses* in ascending with respect to their respective weights (i.e. their measures of affinities) and the first one (the community having the maximal affinity) is selected (line 15). This selected community (the_better) is added into *R* and therefore this community (the_better) is added in V only if it has at least one neighbor (lines 18 to 20) again in *G* which is not in R. This part of the algorithm is repeated while still in V one element. A Community SP_i receives a request *Ask-Better-Neighbor* from another community (e.g. SP_0) considers only its neighbors which are not in yet R.*S* (Line 25). This function checks if *Res* is empty. In this case it returns an empty message to the community (SP_0). Otherwise, it checks if the size of *Res* is equal to one. If that the case, SP_i has one neighbor and the boolean variable *Last* is positioned to *true*; else (i.e. the case where there are several neighbors) the neighbors are ordered in ascending with respect to their respective weights and finally, the better neighbor which has the maximum weight is returned to the claimant community (SP_0). The application of this algorithm in the graph given in figure 3 return the MCT (reported in bold in this figure).

4.3 Semantic queries routing (MCT)

The proposed queries routing algorithm takes in input: the query *Q* submitted by a peer to his community SP_k and the obtained tree MCT. The principle of this algorithm is given as follows: 1. SP_k researches locally the other peers that are able to process this query: local research consists to measure the capacity (formulas given in 11) of each peer to process this query; 2. SP_k sends the query through MCT to other communities. Each community in its turn runs these two steps and the results are returned to SP_k.

Example (Queries routing): In Figure 3, SP_2 receives a query from one of its peers. SP_2 checks if at least one of its peers is able to process this query and sends the query to SP_4, SP_6 and SP_7. SP_4 in its turn processes the query as follows: it checks if at least one of its peers is able to process the query and sends the query to *SP*9 and SP_5. SP_9 returns the results of local research to (SP_2). SP_5 continues the propagation of this query thought MCT.

5. Validation

In this section, we evaluate our algorithms with SimJava-based simulator. Firstly, for the creation of the tree, we compare the performance between the two versions of the algorithm: naive and optimized versions. In our evaluations, we consider a constant number of peers and we varying the number of communities from 10 to 1000. We measure the time of the creation of MCT and the total number of messages exchanged between communities. Secondly, for evaluations concerning the routing algorithms, the number of communities is supposed a constant and we varying the number of peers from 100 to 4000. For each one of these algorithms, we measure the time to get the responses of queries submitted by peers, the total number of messages exchanged between communities and the average number of responses returned by each query. All our experiments are realized under Microsoft Windows XP, x86 family of roughly 2000Mhz and a total physical memory of 2048Mo.

Figure 4(a) shows the number of messages exchanged between peers in order to create the MCT and Figure 4(b) measures the time to create this tree. We varying the number of communities from 10 to 1000 and the number of peers still constant because peers are not solicited in the creation of this tree. We constat that, the number of messages exchanged using the basic algorithm is more important than the number of message exchanged using the optimized version. The time to create the MCT increases with the augmentation of communities and this time in the second version (optimized) is better. We observe that, the time of creation of the MCT, returned by the two algorithms is low when the number of communities is less than 200 and this time increases when the number of communities exceeds 200.

Figure 5 evaluates the performance of routing algorithms: baseline (denoted cSON) and MCT. Figure 5a shows the total number of message exchanged between communities using MCT. This number is more important than the number of messages exchanged between peers in baseline approach. Figure 5b shows the time of queries routing to relevant peers. Figure 5c shows the average number of responses returned by each query submitted by a peer. This number is negligible when the number of peers is low and

Algorithm 2 Construction of the MCT: optimized version

Require: *cSON - Weighted connected and undirected graph*

Output: MCT: R(S,A) where S is the set of communities and A is the set of semantic links between communities. We denote by W, the set of weights assigned to semantic links.

{Start with SP_0 }

1: ***Module*** (SP_0)*:*
{***Initialization***}
2: R.S={} ; R.A = {};
3: Add(SP_0, R.S) ;
4: **repeat**
5: Responses = ϕ
6: **for all** SP_i in V **do**
7: m_v = ask-Better-Neighbor (SP_i, R)
8: **if** m_v == ϕ **then**
9: remove(SP_i,V)
10: **else**
11: Response = Responses $\cup$ m_v
12: **end if**
13: **end for**
14: SORT(Responses, Weight)
15: The_better = first (Responses)
16: R = R $\cup$ The_better . neighbor
17: V = V $\cup$ The_better . neighbor . S
18: **if** The_better.last **then**
19: remove(The_better . SP, V)
20: **end if**
21: **until** V ! = ϕ
22: ***Module*** (SP_i) *:*
23: *Ask-Better-Neighbor* (*R*)
24: last = false
25: Res = Neighbors - R.S {Res contains the set of Neighbors of SP_i in *G* minus the set of vertices in R}
26: if Empty(Res) then
27: Return Message (Empty)
28: **else**
29: **if** size (Res) = 1 **then**
30: last = true
31: **else**
32: sort (Res,weight)
33: **end if**
34: Return Message (SP_i , First (Res), last)
35: **end if**

this number increases with the augmentation of the number of peers. For example, for 1000 peers the difference of number of responses returned by a query using each algorithm is an average roughly 10, and for 4000 peers this difference is roughly 50. In the first case (1000 peers) the total number of responses increases roughly of 10000 and in the second case (4000 peers) this number increases roughly of 200000.

6. Conclusion

In this paper we considered a semantic P2P system such as cSON. Communities in cSON advertise the content they want to

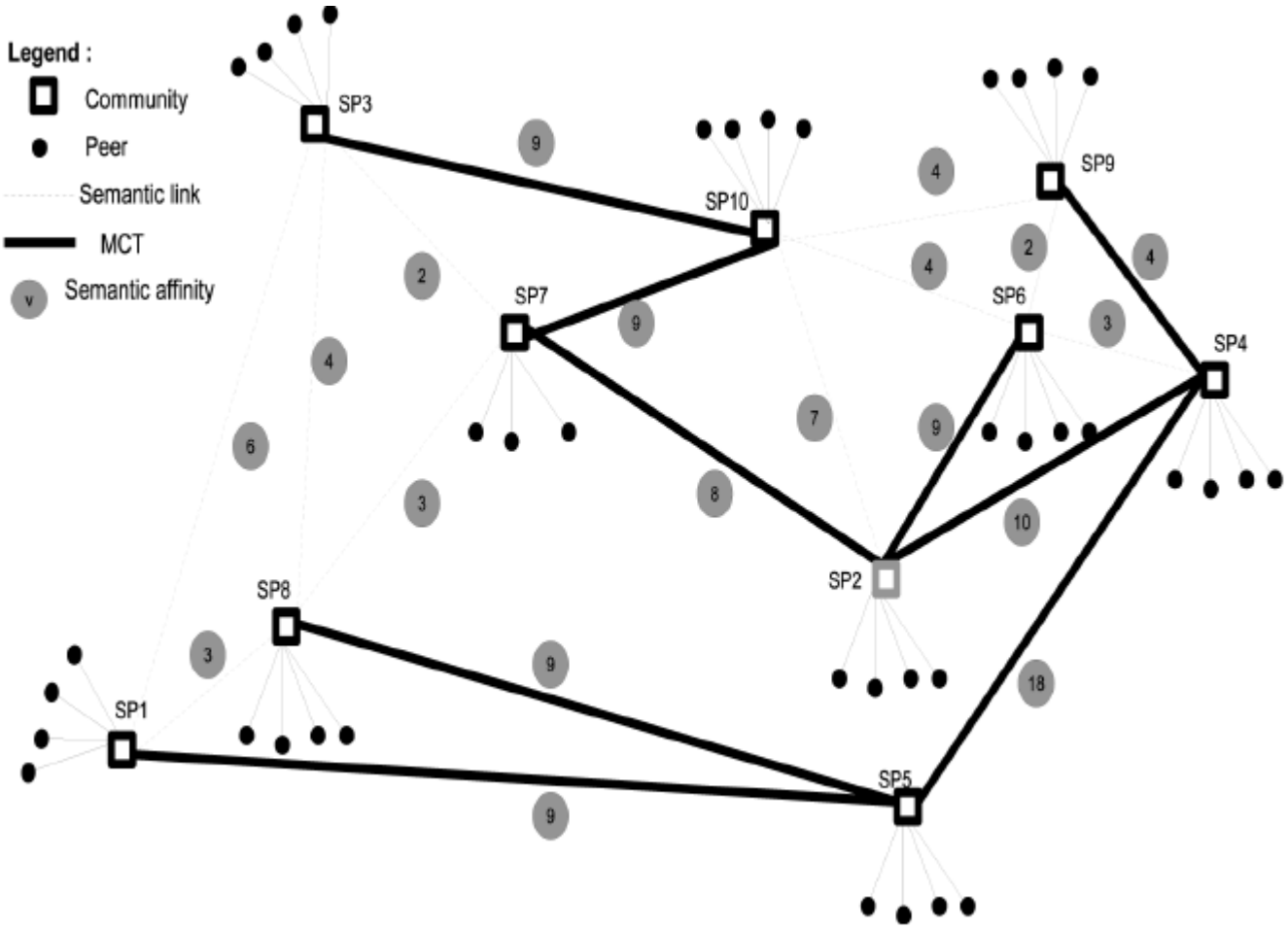
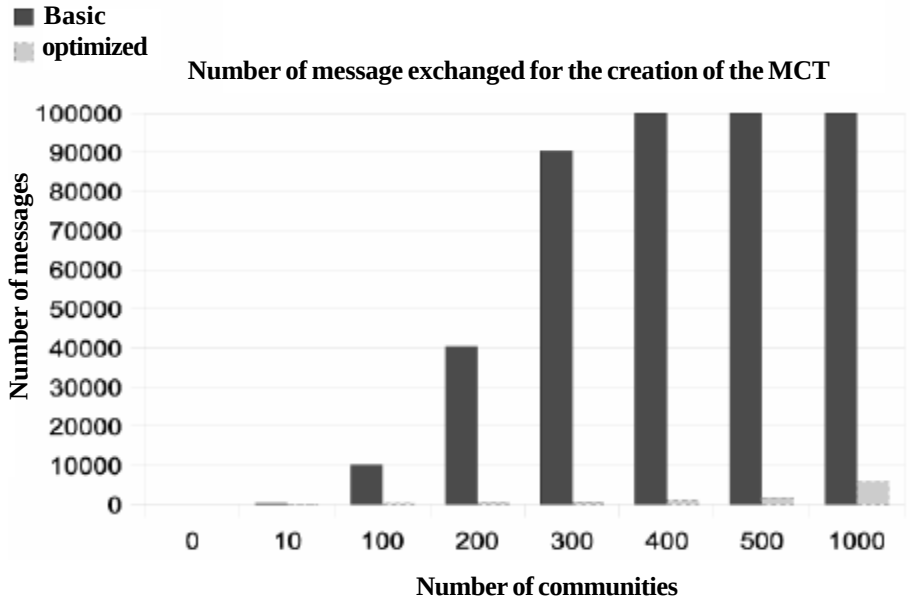
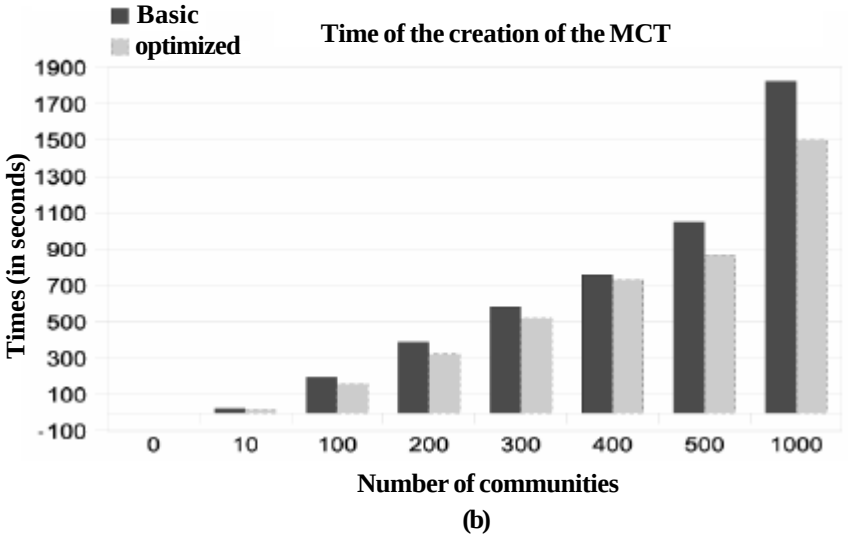


Figure 3. cSON and the MCT corresponding to the community SP2

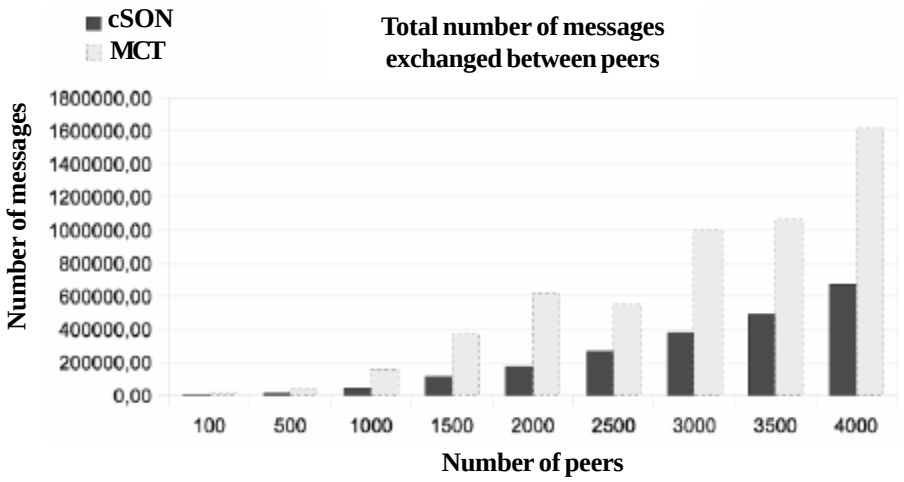


(a)

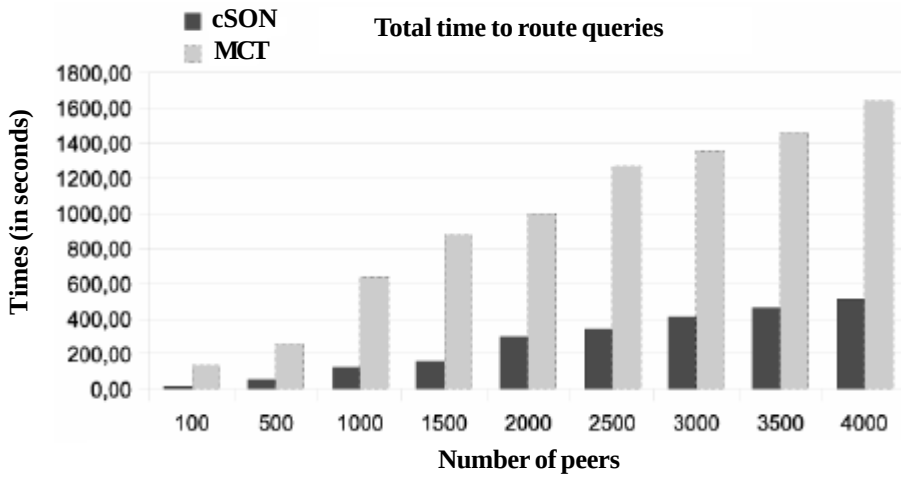


(b)

Figure 4. (a) Number of messages exchanged (b)Times to create the MCT



(a)



(b)

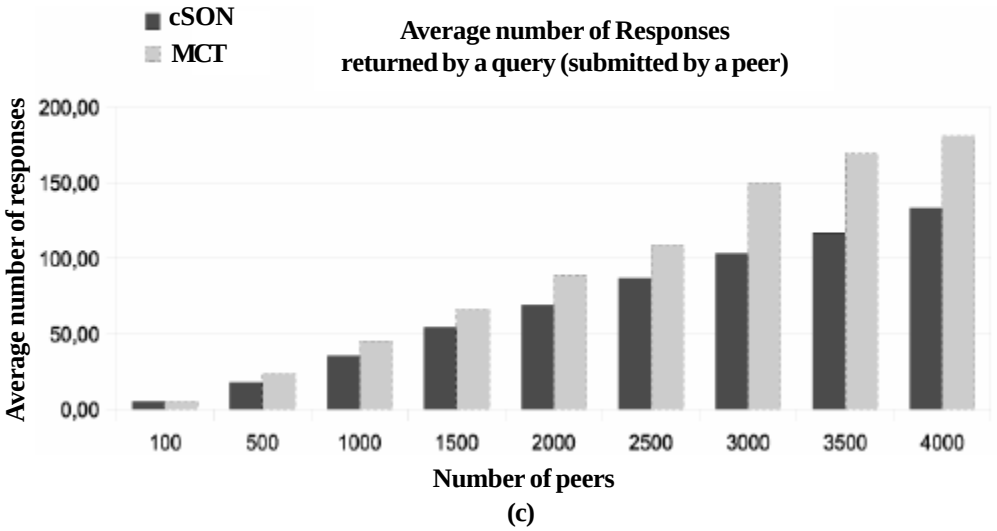


Figure 5. (a) Number of messages exchanged (b)Times to route queries (c) average number of response returned by a query

share by their expertise and discover communities by schema matching techniques. In cSON, within its basic queries routing algorithm, a query submitted by a peer is sent only to neighboring communities (linked together semantically). In this paper, we proposed an algorithm (based on PRIM'algorithm) over cSON that returns a maximal-affinity covering tree denoted MCT. We proposed a queries routing algorithm based on MCT in order to enlarge research to all communities in *cSON*. Our experiments show that the MCT-based routing algorithm combined with the semantic links established between communities outperforms significantly the baseline queries routing algorithm. Future works consists to compare our routing algorithm with others approaches (e.g. flooding algorithm), to generalize the notion of communities to several super-peers and to study the case where new communities joins or leaves the network dynamically.

References

[1] Bloom, B. (1970). Space/time tradeoffs in hash coding with allowable errors, *Communications of the ACM*, 13 (7) 422–426.
[2] Crespo, A., Garcia-Molina, H. (2004). Semantic Overlay Networks for P2P Systems. *In:* Agents and Peer-to-Peer Computing (AP2PC).
[3] Crespo, A., Garcia-Molina, H. (2002). Routing indices for peer-to-peer systems. *In:* IEEE Int. Conf. on Distributed Computing Systems, p. 23–33.
[4] Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. (2001). Introduction to Algorithms, Second Edition. *In:* MIT Press and McGraw Hill, p.561–579.
[5] Faye, D. C., Nachouki, G., Valduriez, P. (2006). Un système pair-à-pair de médiation de données. *In:* journal ARIMA, p. 24–48
[6] Faye, D. C., Nachouki, G., Valduriez, P. (2007). Semantic Query Routing in SenPeer, a P2P Data Management System. *In:* NBIS Conference, p. 365–374.
[7] Nachouki, G., Quafafou, M. (2012). Efficient Research in cSON a Semantic P2P Network. *In:* NDT conference, proceedings published in the Communications in Computer and Information Science (CCIS 7899) Series of Springer LNCS.
[8] Eppstein, D. (2000). Spanning Trees and Spanners. *In:* Handbook of Computational Geometry (Ed. J.-R. Sack and J. Urrutia), p. 425–461.
[9] Huang, F., Gao, P., Wang, Y. (2009). Comparison of Prim and Kruskal on Shanghai and Shenzhen 300 Index Hierarchical Structure Tree. *In:* Web Information Systems and Mining, p. 237–241.
[10] Ismaïl, A. (2010). communities in Semantic P2P Networks (in French), PHD Thesis, Marseille, France.
[11] Kalogeraki, V., Gunopoulos, D., Zeinalipour-Yazti, D. (2002). A local search mechanism for peer-to-peer networks. *In:* ACM Int. Conf. on Information and Knowledge Management (CIKM), p. 300–307.
[12] Kang, C. (2010). Survey of search and optimization of P2P networks. *In:* Peer-to-Peer Net. Appl.
[13] Kokkinidis, G., Christophides, V. (2004). Semantic query routing and processing in p2p database systems: The icsforth sqpeer middleware. *In:* EDBT Workshops, p. 486–495.

[14] Löser, A., Tempich, C. (2005). On Ranking Peers in Semantic Overlay Networks. *In:* 3rd Conference on Professional Knowledge Management.
[15] Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S. (2002). Search and replication in unstructured peer-to-peer networks. *In:* ACM Int. Conf. on Supercomputing, p. 84–95.
[16] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *In:* Soviet Physics Doklady, 8 (10).
[17] Menascé, D., Kanchanapalli, L. (2002). Probabilistic scalable P2P resource location services. SIGMETRICS Performance Evaluation Review, 30 (2) 48–58.
[18] Sripanidkulchai, K., Maggs, B., Zhang, H. (2002). Efficient Content Location Using Interest-Based Locality in Peerto- Peer Systems. *In:* Carnegie Mellon University, Pittsburgh.
[19] Tversky, Amos, (1977). Features of Similarity, *Psychological Review*, 2 (84).
[20] Tsoumakos, D., Roussopoulos, N. (2003). A Comparison of Peer-to-Peer Search Methods, InternationalWorkshop on the Web and Databases, p. 61–66.
[21] Tsoumakos, D., Roussopoulos, N. (2003). Adaptive probabilistic search (APS) for peer-to-peer networks. *In:* Technical Report, University of Maryland.
[22] Tempich, C. (2004). REMINIDIN': Semantic Query Routing in P2P Networks based on Social Metaphors. *In:* World Wide Web Conference.
[23] Taylor, I. (2001). Extract of Peer-To-Peer: Harnessing the Power of Disruptive Technologies. *In:* Edited by Andy Oram, published by O'Reilly.
[24] Yang, B., Garcia-Molina, H. (2002). Improving search in peer-to-peer networks. *In:* Proc. of the IEEE Int. Conf. on Distributed Computing Systems, p. 5–14.