

Cluster-based Search Technique for P2P Systems

Rabab Hayek — Guillaume Raschia — Patrick Valduriez

N° 6782

December 2008

Thème COM

 *Rapport
de recherche*

Cluster-based Search Technique for P2P Systems

Rabab Hayek* , Guillaume Raschia* , Patrick Valduriez*

Thème COM — Systèmes communicants
Équipe-Projet Atlas

Rapport de recherche n° 6782 — December 2008 — 21 pages

Abstract: We consider network clustering as the way to improve the performance of locating data in unstructured P2P systems. Connectivity-based Distributed node Clustering (CDC), and SCM-based Distributed Clustering (SDC) are two major protocols that allow partitioning a network topology into clusters, based on node connectivity. These protocols focus on the accuracy of the clustering scheme, i.e. using the Scale Coverage Measure (*SCM*), and its maintenance against node dynamicity. However, they do not propose search techniques that may take advantage of their clustering information. Thus, their proposals have not been evaluated according to the motivation behind.

In this work, we propose a new, efficient *Cluster-based Search Technique* (CBST) for unstructured P2P systems. We use it to validate connectivity-based clustering schemes, according to the trade-off between cost of maintaining clusters, and benefit for query processing. Our experimental results show the efficiency of CBST implemented over the SDC protocol. By simply exploiting clustering features of the underlying network, a query can travel across a large number of nodes with a minimum number of messages. CBST eliminates a large portion of redundant messages, thus avoiding to overload the P2P network.

Key-words: P2P Systems, Network Clustering

* Atlas, INRIA/LINA-Université de Nantes

Une technique de recherche dans les systèmes pair-à-pair fondée sur le partitionnement du réseau

Résumé : Nous proposons une technique de recherche qui exploite l'organisation inhérente d'un réseau P2P afin de réduire le nombre de messages échangés. Dans la littérature, deux travaux majeurs ont proposés des protocoles d'organisation de réseaux P2P, basés sur la connectivité des noeuds (i.e. Connectivity-based Distributed node Clustering (CDC), et SCM-based Distributed Clustering (SDC)). Ces protocoles ont focalisé sur la précision du schéma d'organisation, et sa maintenance contre la volatilité des noeuds. Cependant, ils ne proposent pas des techniques de recherche qui peuvent exploiter des informations sur leur organisation du réseau. Leurs propositions ne sont pas donc évaluées selon la motivation sous-jacente.

Dans notre travail, nous proposons *CBST* (*Cluster-based Search Technique*), une technique de recherche pour des systèmes P2P non structurés. Cette technique est utilisée pour valider les protocoles d'organisation de réseaux P2P, selon le compromis réalisé entre le coût de maintenir les groupes de noeuds (i.e. *clusters*) et les gains en traitement de requêtes. L'évaluation de performance montre l'efficacité de notre technique de recherche. Une requête est propagée vers un grand nombre de noeuds, avec un nombre optimal de messages. *CBST* élimine une grande portion de messages redondants, évitant donc de surcharger le système P2P.

Mots-clés : Systèmes pair-à-pair, partitionnement de graphe

1 Introduction

P2P networks allow to share data on a world-wide scale with many advantages such as decentralization, self-organization, etc. However, a key challenge is to implement efficient search techniques. Initially, P2P search systems relied on flooding mechanism. Despite of its simplicity and high network coverage (i.e. a large number of nodes could be visited within small values of Time-To-Live (*TTL*)), the flooding mechanism suffers from high bandwidth consumption. There are two major concerns with flooding.

- **Search Blindness:** a peer forwards a query message to its neighbors without any information on how these neighbors may contribute to query answers.
- **Message Redundancy:** a peer may receive the same query message multiple times. This is due to the ad hoc nature of P2P connections, i.e. the neighbor selection process is random and non-discriminant.

Many techniques have been proposed in order to improve the performance of P2P systems. Initial works have led to structured systems, which mainly act as global distributed indexes (e.g. [7], [14]). These systems address the problem of topology randomness by imposing a specific network structure, and remedy to the blindness problem by making a tight control on data (or data pointers) placement. Hence, these systems provide an efficient, deterministic search. However, they compromise node autonomy and may restrict search expressiveness. Other data indexing techniques have been proposed in fully unstructured systems such as [1], [3]. Each of these techniques achieves a different trade-off between the cost of maintaining indexes and the benefits obtained in query processing.

Another research axis has focused on network clustering which aims to introduce some structure to, or extract inherent structural patterns from fully unstructured P2P networks. A network clustering scheme consists in organizing the nodes into clusters, based on a given criterion. A clustering criterion could be a physical network metric (e.g. bandwidth, latency), some peer property/behavior (e.g. node connectivity/stability), or even defined at the application layer (e.g. similar interests).

Our contribution consists in exploiting the characteristics of the overlay topology, namely its clustering features, in order to reduce the number of redundant query messages generated by flooding-based algorithms. This allows to improve the performance of P2P systems, irrespective of the employment of techniques relying on data semantics at the application layer (i.e. semantic indexing or clustering). In reality, the flooding approach is still a fundamental building block of unstructured P2P systems. It represents the natural way for exchanging messages between nodes which are connected in an ad hoc manner.

This paper proposes a *Cluster-Based Search Technique* (CBST) for P2P systems, which is implemented over a connectivity-based clustering protocol. A connectivity-based clustering protocol aims to discover the natural organization of nodes, based on their connectivity. Thus, it delimits the boundaries of sub-graphs (i.e. clusters) which are loosely connected and in which nodes are highly connected. In the P2P literature, two main protocols have been proposed, i.e.

the Connectivity-based Distributed node Clustering (CDC) [12], and the SCM-based Distributed Clustering (SDC) [9]. These works have been introduced by arguing that efficient routing protocols could be defined by taking advantage of the clustering features of the underlying network. However, none of them has proposed an appropriate routing protocol, or provided analytical models or experimental results on how their clustering schemes contribute to reduce the bandwidth consumption. The main focus was on the clustering scheme accuracy, i.e. using the Scale Coverage Measure (SCM), and its maintenance against node dynamicity.

The CBST works as follows. Based on a local knowledge about the network clustering in its neighborhood, each node maintains information about the *local links* to nodes in its cluster (intra-cluster information), as well as about *global links* connecting its cluster to other reachable clusters (inter-cluster information). The intra-cluster routing information is equivalent to a spanning tree, rooted at that node and covering its partners (i.e. the nodes that belong to its cluster). These information are efficiently gathered and maintained in a cluster-based routing table. The benefits in query routing are two folds. First, a query Q is efficiently disseminated in a given cluster, using the spanning tree of the first node contacted in that cluster. Second, the query messages between clusters are restricted to those traversing the global links specified by the querying node. Extensive simulations have demonstrated the efficiency of the *CBST* technique compared to the pure flooding and random walk routing techniques.

The rest of this paper is organized as follows. Section 2 gives an overview of the existing connectivity-based clustering schemes. In section 3 we describe our cluster-based search technique *CBST*. Section 4 discusses the trade-off between search accuracy and bandwidth consumption that can be achieved by *CBST*. Section 5 presents performance evaluation through simulation. In section 6, a comparison to related works is provided. Section 7 concludes.

2 Connectivity-based clustering schemes

A connectivity-based clustering protocol should fulfill the following requirements in order to be considered as appropriate for P2P systems.

- A natural requirement is that it should achieve a good *clustering accuracy*, i.e. it should organize the network such that nodes are highly connected in the same cluster and less connected between clusters.
- It should well control the cluster size (or cluster diameter). Due to the lack of knowledge about network structure, it is expensive to maintain expanded clusters in large P2P networks.
- It should be fully distributed. Nodes should form clusters automatically without the knowledge of the complete network topology.
- It should recover from node dynamics, with small overhead in term of the number of messages exchanged between nodes.

Centralized clustering algorithms, like MCL (Markov Cluster) [13], can achieve high clustering accuracy. However, such algorithms assume that the complete network topology is available at a central point, and thus cannot be used in

P2P networks. The CDC scheme [10] is a distributed approach that discovers connectivity-based clusters in P2P networks. A set of nodes are selected as “originators” and clusters are formed around them using flow simulation. The quality of the clustering scheme depends on how well the “originators” are distributed in the network. Furthermore, the main issue with the CDC algorithm is that it cannot handle node dynamicity in a decent way. The whole network has to be re-clustered at each node join or leave, which incurs a large traffic overhead. The SCM-based Distributed Clustering (SDC) protocol [9] proposes to satisfy all the design criteria discussed above. The protocol performs in a fully distributed way, which is briefly described in Section 2.2. The clustering accuracy is dynamically adjusted using the Scaled Coverage Measure (*SCM*), a practical clustering accuracy measure, which is presented in Section 2.1. Besides, the cluster size is well controlled, and the node arrival/departure is locally handled with a small number of messages, while keeping a good quality of clustering.

2.1 SCM: Accuracy Measure for Graph Clustering

The *Scaled Coverage Measure* (SCM) has been proposed by S. Van Dagon [13] to evaluate the accuracy of a clustering scheme. The key idea behind this performance measure is that an optimal clustering of a given graph should minimize both the number of inter-cluster edges and the number of non-neighbor vertices in each cluster. Let $G = (V, E)$ be a graph, where V is the set of nodes corresponding to the set of peers in a P2P system, and E is the set of links, which are the logical connections between peers. We assume that $C = \{C_1, C_2, \dots, C_l\}$ is a given clustering on graph G . Each cluster C_i is a non-empty subset of V , and $\cup_{i=1}^l C_i = V$. Given a node $n_i \in V$, we have the following notations:

- **Nbr**(n_i): the set of neighbors of node n_i ;
- **Clust**(n_i): the set of nodes in the same cluster as node n_i (excluding n_i);
- **FalsePos**(n_i): the set of nodes in the same cluster as n_i but not neighbors of n_i ;
- **FalseNeg**(n_i): the set of neighbors of n_i but not in the same cluster as n_i .

Then the Scaled Coverage Measure of node n_i with respect to the clustering C , $SCM(n_i)$, is defined as:

$$SCM(n_i) = 1 - \frac{|FalsePos(n_i)| + |FalseNeg(n_i)|}{|Nbr(n_i) \cup Clust(n_i)|} \quad (1)$$

The *SCM* value of the graph G , $SCM(G)$, is defined as the average of the *SCM* values of all of the nodes: $SCM(G) = \sum_{n_i} SCM(n_i)/N$, where N is the network size (i.e. $|V| = N$).

2.2 The SDC Protocol

Given a network, each node n_o is initialized as an orphan node with its own *clust_id* (any unique id is sufficient) and *clust_size* (1 in this case). For *SCM* computation, node n_o maintains two variables a_o and b_o such that:

$a_o = |Nbr(n_o) \cup Clust(n_o)|$, $b_o = |FalsePos(n_o, C)| + |FalseNeg(n_o, C)|$. Initially, $a_o = b_o = |Nbr(n_o)|$, and according to Equation 1: $SCM = 1 - b_o/a_o$.

When running the SDC protocol, all nodes start to exchange messages with their neighbors, conduct some simple computation, and form clusters in a greedy manner. After a number of rounds of communication, the clustering procedure becomes stable without further message exchange and the network is finally clustered. The clustering procedure executed when node n_o attempts to join a new cluster can be briefly described as follows.

- First, node n_o probes its neighborhood (i.e. *Clust_Probe* message) to discover “candidate clusters” to which it may join. Then, a request message is flooded in each candidate cluster to indicate the intention of node n_o to join that cluster.
- Second, each node n_j in a candidate cluster C_i computes the gain $\Delta SCM(n_j)$ assuming that node n_o joins C_i . Note that this computation only requires the information of whether n_o is a n_j 's neighbor or not. According to equation 1, if $n_o \in Nbr(n_j)$,

$$\Delta SCM(n_j) = 1/a_{n_j}.$$

Otherwise, the gain in SCM is given by,

$$\Delta SCM(n_j) = b_{n_j}/a_{n_j} - (b_{n_j} + 1)/(a_{n_j} + 1).$$

Similarly, each node in $Clust(n_o)$ has to compute its gain as if n_o leaves its current cluster. After gain computation, node n_j sends back its gain value to node n_o (i.e. *Clust_Reply* message).

- Upon receiving all the reply messages from all the nodes in its cluster and a candidate cluster C_i , node n_o computes the overall gain $\Delta SCM(G)$ assuming it leaves its original cluster and joins C_i . If $\Delta SCM(G) > 0$, n_o should join C_i . There might be multiple candidate clusters of which ΔSCM are positive, n_o should join the one with the maximum SCM gain. Once n_o determines which cluster to join, a *Clust_Update* message is flooded in its original cluster and the new cluster. This allows to update the clustering information of its old and new partners.
- The cluster diameter is bounded by a predefined threshold D , which is used to limit the flooding of request and update messages within clusters. If the arrival of node n_o to a new cluster implies that the new cluster diameter will exceed the D value, then the join request of node n_o will be rejected. This allows to control the cluster size as well. Simulation results in [9] have shown that the average cluster size is very stable, for topologies with different scales.

Note that, after node n_o joins the new cluster, its neighbors in the original cluster are affected and should check whether they should join other clusters, in the same way as node n_o has done. The whole procedure will end if no node can join any cluster based on $\Delta SCM(G)$, and the cluster diameter control.

Neighbor	Cost(nb of hops)	Destination
Partner entries		
n_1	h_{11}	p_1
n_2	h_{22}	p_2
\dots	\dots	\dots
n_n	h_{nk}	p_k
Cluster entries		
n_1	h_{11}	C_1
n_2	h_{22}	C_2
\dots	\dots	\dots
n_m	h_{ml}	C_l

Table 1: Routing table of node n_o

3 CBST for unstructured P2P systems

In our work, we aim at defining a cluster-based search technique on the top of a connectivity-based clustering protocol (i.e. the SDC protocol). In this section, we first define a mechanism for building *cluster-based* routing tables, and updating their entries against node dynamicity. Second, we propose a routing mechanism that allows a query to travel across clusters such that a maximum number of nodes is visited (for a given value of *TTL*), and redundant query messages are quasi-eliminated.

3.1 Cluster-based Routing Table

Let $G = (V, E)$ be the graph corresponding to an unstructured P2P network, and C the clustering obtained from running the SDC protocol on that graph. Suppose that node $n_o \in V$ belongs to the cluster $C_o \in C$. According to the notations in 2.1, $Nbr(n_o)$ is the set of n_o 's neighbors, and $Clust(n_o)$ is the set of n_o 's partners, i.e. the set of nodes that belong to the same cluster C_o . Note that $C_o = Clust(n_o) \cup \{n_o\}$.

3.1.1 Routing Table Description

Table 1 describes the routing table maintained at node n_o . We distinguish between intra-cluster routing information, which is represented by the first set of *partner* entries, and inter-cluster routing information, which is represented by the second set of *cluster* entries. The former provides information about paths to each partner $p \in Clust(n_o)$, while the latter provides information about paths to a subset of reachable clusters.

A partner entry e_p (respectively cluster entry e_c) in Table 1 is read as follows. Going through neighbor n_i , node n_o can reach a partner p_j (respectively a new cluster C_j), with a minimal number of hops h_{ij} . Thus, the set of partner entries is equivalent to a spanning tree of the subgraph C_o , rooted at node n_o . The number of these entries is limited to the cluster size, which is well controlled by the underlying clustering protocol. On the other hand, to control the number of cluster entries, node n_o chooses to keep information only about clusters that contain, at least, one node locating at a maximum distance of D .

Before presenting how the routing table RT at node n_o is maintained, we list the set of properties that should be satisfied in order to keep RT staleness.

Property 1 $Pa = Clust(n_o)$, where Pa is the set of destinations (peers) of partner entries in the n_o 's RT ($Pa := \{n_i \in V / \exists e_p \in RT, e_p.destination = n_i\}$).

Property 2 $\forall e_p \in RT, e_p.cost = \min(distance(n_o, n_i))$ over all the intra-cluster paths (n_o, n_i) , where $n_i = e_p.destination$.

These two first properties guarantee that the set of partner entries in n_o 's RT represents the spanning tree of the subgraph C_o , rooted at node n_o itself.

Property 3 $\forall e_c \in RT, \exists n_i \in D-Nbr(n_o)$ such that $n_i.clust_id = e_c.destination$, where $D-Nbr(n_o)$ is the D -neighborhood of node n_o , i.e. $D-Nbr(n_o) = \{n_i \in V / distance(n_o, n_i) \leq D\}$.

This property guarantees that each cluster referred by a cluster entry in n_o 's RT is still existing in the network, and contains at least one node belonging to the $D-Nbr(n_o)$.

Property 4 $\forall e \in RT, e.dist \leq D$.

This final property guarantees that the cost values in n_o 's RT are all bounded by the predefined threshold D , which is used by the SDC protocol to control the cluster diameter.

3.1.2 Table Maintenance

When running the SDC protocol on a P2P network, nodes exchange some messages and after a round of communications, they determine their respective clusters (Section 2.2). In our work, we define an efficient algorithm which allows to create/update our routing tables, while the clustering scheme is establishing. The main idea is to only use the SDC clustering messages in order to gather and maintain cluster-based routing tables, without requiring additional messages. Due to space limitations, we do not present the details of our algorithm. However, for illustration, we consider the network of Figure 1, which contains 8 nodes and 4 different clusters (the orphan node 7 is considered as a separate cluster). All nodes maintain their initial routing tables, which are in a valid state, before node 7 attempts to join a new cluster. Figure 2 shows the network after node 7 has joined cluster B . The routing tables are updated accordingly, and thus are in a new valid state.

At node 7, the cluster entry that corresponds to the newly joined cluster is removed. That is, the second entry of its initial table is removed. Then, entries that describe paths to the new partners 1 and 5 are added. New partners is identified thanks to the reply messages which have been sent back to node 7.

The clustering change made by node 7 will certainly incur modifications on other nodes' routing tables. According to SDC , node 7 sends a $Clust_Update$ message to be flooded in both old (if still exist) and new clusters (e.g. cluster B). Such a message can be used to update the routing tables of visited nodes. To this end, we assume that it is flooded in all candidate clusters, and is associated with its routing table. Hence, the modifications are propagated from a node to

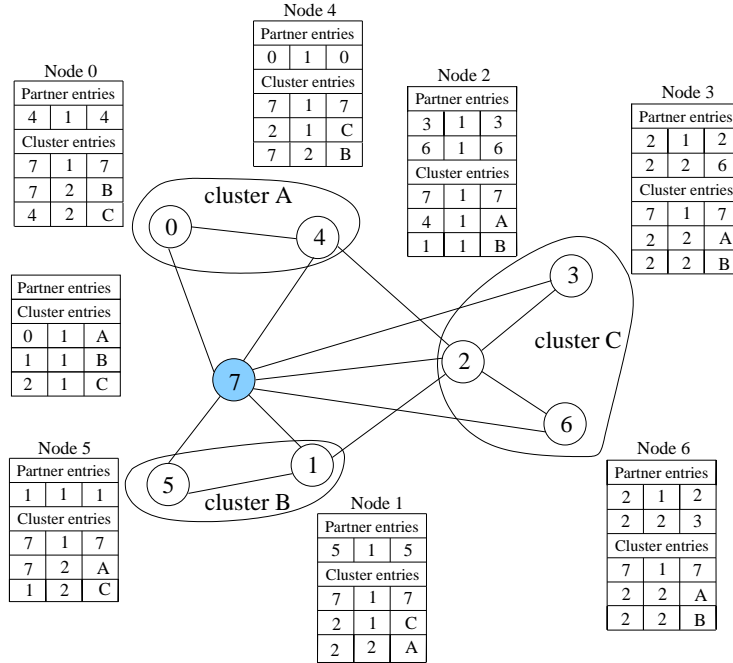


Figure 1: Initial routing tables

another. Remind that the flooding of *SDC* clustering messages is limited to the maximum cluster diameter value D . Thanks to the Property 4, such a limited flooding is sufficient for reaching all affected nodes.

In our example, each cluster entry that supposes that node 7 belongs to its old cluster should be removed, e.g. the third entry in the initial table of node 6. Besides, the clustering change made by node 7 may either introduce a new cluster into the D -neighborhood of a given node, or provide a shorter path to a cluster which is already known. In the first case, a cluster entry for the new cluster should be added. However, in the second case, the corresponding entry should be appropriately updated. For instance, the last entry in the initial table of node 6, which describes the path to cluster B , is replaced by the fourth entry in its final table, since it provides a smaller cost value.

In the case where an alternative path to a given cluster is provided with a same cost, then the current node chooses the path in which the next hop node is a partner. For instance, in the routing table of node 1, the last cluster entry describes the path to cluster A via node 2, with a cost of 2 hops. This entry in the final table by an entry that describes another path, which goes through the new partner 7 with a same cost. The benefit of such a choice is discussed while presenting our query propagation mechanism.

Once the clustering scheme is established, and the routing tables are in a valid state, the only events that can disturb this network stability is node connection/disconnection. Suppose now that node n_o enters/leaves the P2P system. Figure 3 shows the decomposition of the network into subgraphs according to the relative position to n_o . Here, C_o is the cluster of node n_o before its leaves,

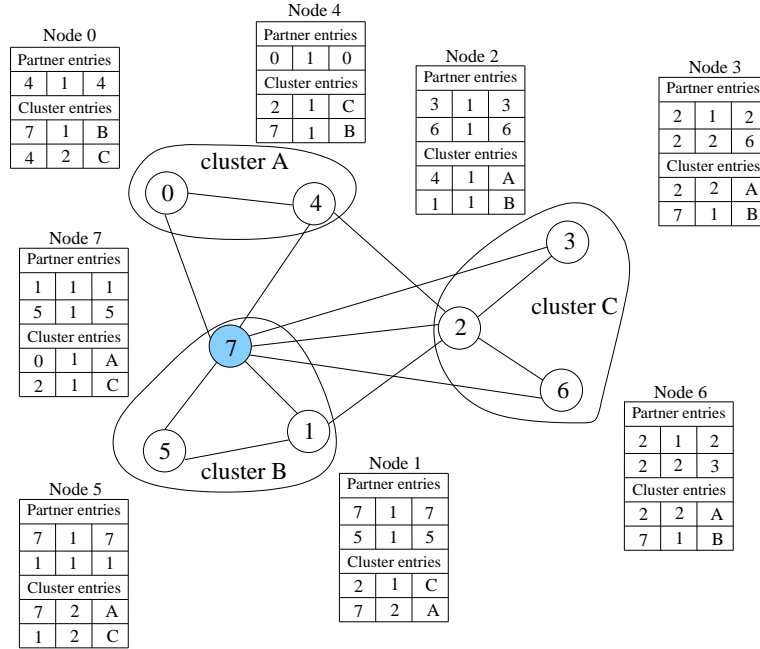
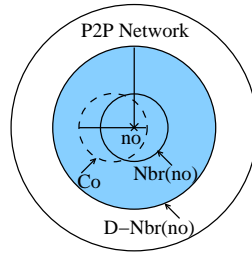


Figure 2: Final routing tables

or the new cluster it joins upon its arrival into the network. The diameter d of cluster C_o is bounded by the predefined threshold D .

Figure 3: Network around node n_o

First, consider a node n_i which is out of the D -neighborhood of node n_o (i.e. $n_i \notin D\text{-Nbr}(n_o)$). Its routing table RT_i is not affected and remains in its valid state. In fact, according to Property 4, each entry e in RT_i describes a path to a given destination such that $e.cost \leq D$. Hence, each node n_j locating on this path should satisfy the following condition: $distance(n_i, n_j) \leq D$. Since $distance(n_i, n_o) > D$, node n_o does not affect any path information maintained in RT_i . **Therefore, a first conclusion is that the node arrival/departure are localized events and their impacts on routing tables are limited to the D -neighborhood of the arriving/leaving node (i.e. the gray zone in Figure 3).**

When node n_o leaves or enters the P2P network, the topology structure in its neighborhood is changed, and thus existing clusters may be affected. In [9],

the *SDC* protocol handles node dynamics in a very efficient way. It ensures a good clustering accuracy, with a low cost in term of the number of exchanged messages. Once again, we aim here at exploiting the clustering messages in order to update our cluster-based routing tables in $D\text{-}Nbr(n_o)$.

According to *SDC*, before leaving the network, node n_o sends a LEAVE message that contains its identifier to all nodes in $Nbr(n_o)$, as well as in C_o through flooding. As such, each affected node in $Nbr(n_o) \cup C_o$ can update its clustering information, i.e. cluster size and SCM value. Concerning the routing tables, all *stale* entries that describe paths traversing node n_o should be removed. To this end, we assume in our work that the LEAVE message sent by a node n_i has the following structure: $leave = \langle n_o, SE_i, TTL_i \rangle$, where SE_i is the set of stale entries in its routing table RT_i , relative to node n_o 's exit. While the TTL_i value controls the message flooding, and is dynamically adjusted at node n_i . In the following, we describe how SE_i and TTL_i are determined.

a) Set of Stale Entries SE :

At a neighbor node n_i in $Nbr(n_o)$, the set SE_i of stale entries is given by: $SE_i = \{e \in RT_i \mid e.\text{neighbor} = n_o\}$. For instance, if we consider that node 0 is leaving from the network of figure 2, the sets of stale entries at nodes 4 and 7 are: $SE_4 = \{ep_1\}$, and $SE_7 = \{ec_1\}$.

At a non-neighbor node n_j which has received the leave message from a neighbor n_i , the set SE_j of stale entries is given by:

$$SE_j = \{e \in RT_j \mid \exists e' \in SE_i; (e.\text{neighbor} = n_i) \\ \text{and } (e.\text{destination} = e'.\text{destination})\}$$

In our example, the set of stale entries at node 5 which receives the message from node 7 is: $SE_5 = \{ec_1\}$.

b) Dynamic TTL value:

In order to reach all the stale entries in $D\text{-}Nbr(n_o)$, a solution could be to flood the LEAVE message with a TTL value, initialized by n_o to D , and decremented by one after each message hop. However, to reduce the number of exchanged messages, we adopt an alternative in which each node n_i locally adjusts the TTL value.

First, if the set SE_i of stale entries is empty at node n_i , the TTL value is set to 0 and the LEAVE message is stopped. Otherwise, the TTL value is set as follows. Let e be a stale entry found at node n_i . This entry describes the path to a given destination $dest$. Suppose now that there is a node n_j who is reaching the same destination $dest$ through node n_i . Hence, there is a stale entry e' that should be also removed from the routing table of node n_j . However, according to Property 4:

$$e'.\text{cost} \leq D \Rightarrow \text{distance}(n_j, n_i) + e.\text{cost} \leq D \\ \Rightarrow \text{distance}(n_j, n_i) \leq D - e.\text{cost} \quad (2)$$

The value of TTL_i should be greater than $\text{distance}(n_j, n_i)$ in order to reach n_j and remove its stale entry e' . According to 2, TTL_i could be set to $D - e.\text{cost}$. By applying this condition to all stale entries at node n_i , we conclude that

the TTL value is given by: $TTL_i = (D - \min_{e \in SE_i}(e.cost))$. This alternative allows to remove all the stale entries at affected nodes in $D-Nbr(n_o)$, with a minimal number of LEAVE messages. Node arrival is treated in a similar way as node departure. As stated before, the key idea is to limit the strategy of routing table updates to affected regions, with a dynamically controlled flooding.

3.2 Query Propagation

We describe now how a query Q is propagated in the network, using cluster-based routing tables. The query message is in the following form.

$$Q_Msg = \langle Q_id, cluster_id, TTL, dest_List \rangle$$

- Q_id : the query identifier. It mainly allows to avoid processing the same query multiple times.
- $cluster_id$: the cluster identifier of the node sending the query. It mainly allows to avoid serving the same query within the same cluster multiple times.
- TTL : the TTL value which is the maximum number of hops a query Q can make in the network.
- $dest_List$: the destination list which contains routing information provided by the tables of visited nodes.

Recall that our routing tables do not provide information about the requested objects. The destination list $dest_List$ serves as a memory that keeps trace of the query all along its path. Thus, at each forwarding step, the query is able to remember all the routes it has already traversed. In other terms, the query propagation mechanism is still *blind* from a data location point of view, but is supported with memory which allows avoiding unnecessary messages. An $dest_List$'s element is represented by: $\langle neighbor, dest \rangle$, where $neighbor$ identifies which neighbor of the current node should be chosen for the next query hop, in order to reach destination $dest$ with a minimal cost. A neighbor field value of -1 indicates that the corresponding destination is either reached or targeted by another path that does not include the current node. To illustrate our mechanism, suppose that node 0 in our network example issues a query Q_0 (Figure 4). Upon receiving the query message, a node n first decrements the TTL value. If $TTL > 0$, it proceeds to *update* the query message and *forward* it. Otherwise, the query forwarding is stopped. To update $dest_List$, node n performs the following steps.

a) Adding new partner entries: First, node n verifies if it is the first visited node in its cluster. In that case, n adds new partner elements, each of which provides information about the path to a given partner (e.g. node 7 and node 2 in Figure 4). In other terms, the query will be disseminated in a given cluster according to the spanning tree rooted at the first visited node.

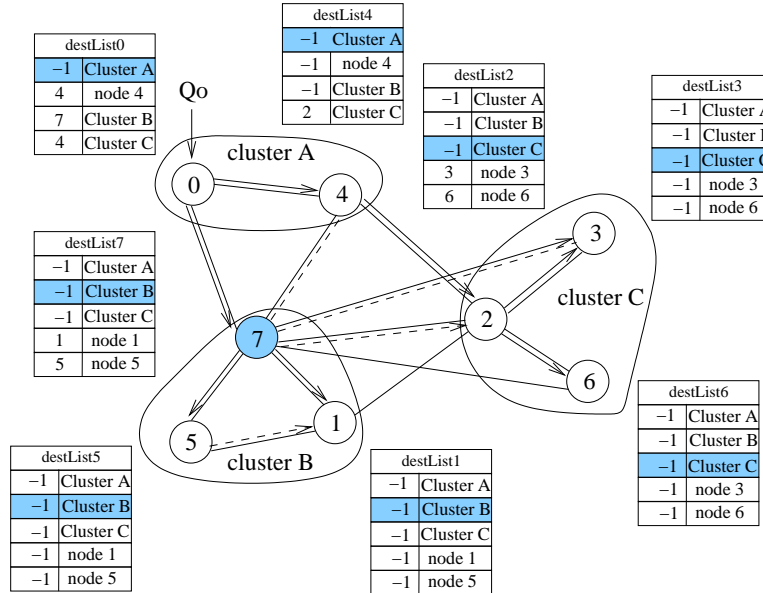


Figure 4: Cluster-based query propagation

b) Setting entries: Second, node n should update each element e in $dest_List$ as follows. If $n = e.neighbor$ which means that node n locates on the path described by element e , then n has to precise the next query hop based on its routing table. Otherwise, the neighbor field value is set to -1 to indicate that all paths that traverse node n are not supposed to reach the corresponding destination.

c) Adding new cluster entries: Third, node n checks if its routing table contains information about additional clusters that do not figure in $dest_List$. If such *cluster* entries exist then corresponding elements, which are extracted from n 's routing table, are added to $dest_List$.

To illustrate the destination list update, consider $dest_List_4$ of query Q_0 in Figure 4. Node 4 belongs to the same cluster as the query sender, which is node 0. The last element in $dest_List_4$ indicates that through node 2 the query can reach cluster C. The two first entries indicate that node 4, as well as its own cluster A, have been already visited. The third entry informs node 4 that it does not locate on the path leading to cluster B. This allows avoiding redundant messages since node 4 could also send a message to cluster B via its neighbor node 7. Once $dest_List$ updated, node n forwards the query to each neighbor in $dest_List$. For example, node 4 sends the query only to node 2. If node n does not belong to the same cluster as the query sender, it sets the *cluster_id* field of the query to its cluster identifier before forwarding it.

Now let us examine the cost of propagating query Q_0 in Figure 4. Our cluster-based propagation mechanism requires only 7 messages to cover the entire network (i.e. the continuous arrows). In comparison, a flooding mechanism would result in 4 additional, unnecessary messages (i.e. dashed arrows). Obviously, this propagation efficiency may be compromised by a smaller value of

query accuracy, which is quantified by the total number of visited nodes for a given value of TTL . For example, node 6 could be reached with a TTL value of 2 using the flooding technique, while a TTL value of 3 is required for our cluster-based technique. In fact, this trade-off between query accuracy and query propagation cost will be discussed later in this paper.

4 Discussion

In P2P networks, a search is *successful* if it discovers at least one replica of the requested object. The efficiency of a search technique could be quantified by the success rate (or *accuracy*), which is the ratio of successful to total searches made. However, search techniques are also evaluated according to bandwidth consumption.

4.1 Search Accuracy vs. BW Consumption

The performance of flooded-based search techniques is quantified by the pair (M, P) , where M is the total number of exchanged messages, and P is the total number of visited nodes. The search accuracy of blind techniques depends on the number of visited nodes P . To increase the probability of finding a replica of the requested object, these techniques tend to propagate the query to a large number of nodes, since they do not dispose of any information related to object locations. However, the main issue is to achieve a good trade-off between search accuracy and bandwidth consumption. In other terms, increasing P should not result in increasing the number of redundant messages $M - P$, which unnecessarily overload the network.

a) Intra-cluster messages: In our approach, a query is disseminated inside a given cluster based on the spanning tree rooted at the first visited node in that cluster. Thus, **the intra-cluster query propagation guarantees that all partner nodes falling into the query scope are visited only once, through the shortest intra-cluster paths. In the case where the cluster diameter $d \leq TTL$, the cluster will be entirely covered.**

b) Inter-cluster messages: Now suppose that node n maintains information about l clusters in its routing table. If $TTL \leq D$, then the set of clusters visited by the query is included among those l clusters. Otherwise, i.e. if the TTL value permits to go beyond the $D-Nbr(n)$, additional clusters could be visited by the query. Here, two different nodes locating on two different query routes might have information about a same new cluster. Thus, the latter will be reached twice, which may incur redundant messages. Another issue is that, a cluster entry maintained in n 's routing table describes the shortest path to a first contacted node in that cluster. However, it does not guarantee that all the nodes visited in the new cluster are reached with a minimal number of hops.

4.2 Clustering Scheme Accuracy vs. BW Consumption

Besides studying the benefits obtained for query processing, the cost of maintaining an accurate clustering scheme should be taken into account, especially

in the context of P2P systems. In [9], simulation results show that the cost of recovering from a given node arrival/departure is very limited, and quasi-independent of the network size. This interesting result is due to the very efficient solution adopted by *SDC* to handle node dynamicity. This cost is limited to 50 messages (the cost of node joining is slightly less than the cost of node leaving), and this for a network size ranging from 1000 to 5000. In our work, we have mainly used the clustering messages in order to update the cluster-based routing tables. However, we require some additional messages to make all the necessary modifications to all affected tables (Sections 3.1.2). According to *SDC*, the *Clust_Update* message has only to be flooded in the old and new clusters of the moving node n_o . While the *Leave* message should be flooded in $Nbr(n_o)$ and the old cluster C_o of the leaving node n_o .

For our CBST purposes, however, the *Clust_Update* and the *Leave* messages should be flooded in all neighboring clusters of node n_o . However, this incurs a very limited number of messages, and this because of the two following reasons. First, the number of neighbor clusters of node n_o is supposed to be very small since the clustering metric is node connectivity. Nodes that are neighbors are preferred to be partners in the same cluster, which is taken into consideration by the *SCM* gain computation. Second, the *SDC* protocol has a stable performance in term of controlling the cluster size. Thus the flooding within a given cluster is well controlled. Besides, the solution we have proposed for dynamically adjusting the TTL value of the leave message reduces at maximum the number of induced messages.

5 Experimental Results

We validated *CBST* through event-driven simulations, which are commonly used to evaluate the performance of large scale P2P systems. Simulations allow controlling system parameters, and thus studying their impact on overall system performance. Furthermore, our performance evaluation consists mainly in quantifying the trade-off between search accuracy and bandwidth consumption. The first is measured in terms of the number of visited nodes, while the second is measured in terms of the number of exchanged messages in the system. Network parameters such as latency and bandwidth do not interfere with these measurements. Thus, simulation results are supposed to give a rough estimation of real values that might be obtained from real implementations.

5.1 Simulation Setup

We used the SimJava package [4] and the BRITE universal topology generator [5] to generate power law P2P networks, with an average node degree of 4. Note that the *SDC* protocol has been tested over both random and power-law topologies. However, the latter yield better performance regarding the overhead traffic, which is required for maintaining good clustering accuracy, and the cluster size control. On the other hand, recent studies (e.g. [15], [11]) have shown that many real-life networks (e.g. social networks, P2P networks) have common characteristics, including power law degree distributions. In our experiments, we first generate network topologies with sizes varying between 100 and 3000. Then, we run the *SDC* protocol to establish a connectivity-based clustering

scheme over the generated networks. The value of D , which is used to control the cluster size is set to 3, as considered in [9]. Finally, we implement our search technique over these clustered networks in both static and dynamic settings.

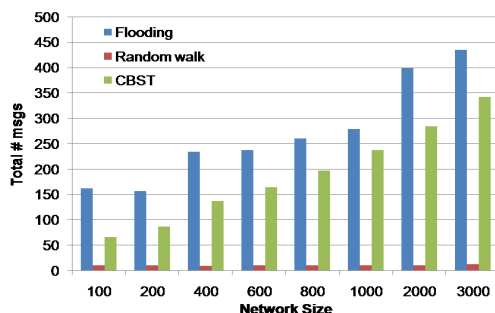


Figure 5: Query cost vs. network size

5.2 Performance in Static Systems

In this set of experiments, we assume that all nodes remain connected to the system, the clustering scheme is stable, and all routing tables are in a valid state. Under these assumptions, we quantify the trade-off between search accuracy and bandwidth consumption. We compare *CBST* against flooding and random walk [16] techniques. The comparison to flooding techniques is very relevant for evaluating search accuracy, since they provide the highest accuracy values (for a given *TTL*). However, they produce a huge traffic overhead in the network. Thus, we choose to compare *CBST* to the random walk technique, which is a good representant of the blind techniques that have been proposed to overcome the problem of bandwidth consumption. In these experiments, a query Q is associated with a *TTL* value of 3.

5.2.1 Search accuracy vs. Bandwidth Consumption

Figure 5 depicts the total number of messages exchanged for propagating a query Q , while Figure 6 depicts the number of visited nodes, in function of the network size. We can observe that the random walk approach reduces very significantly the number of exchanged messages. In fact, a requesting node n sends out k query messages to an equal number of randomly chosen neighbors. Each of these messages is forwarded to a randomly chosen neighbor at each step. In our experiments, we use a high value of k such that a query originator sends the query message to all its neighbors. However, as we can see in Figure 6, the search accuracy is also very reduced. A very limited number of nodes are visited by the query Q . For our *CBST* technique, Figure 5 shows that the number of exchanged messages per query is significantly reduced. For instance, this query cost is reduced by a factor of 2 for a network of 200 nodes. However, in return, *CBST* incurs a small decrease in search accuracy compared to flooding (Figure 6). This result is due to the fact that our approach focuses on eliminating redundant messages without affecting search accuracy.

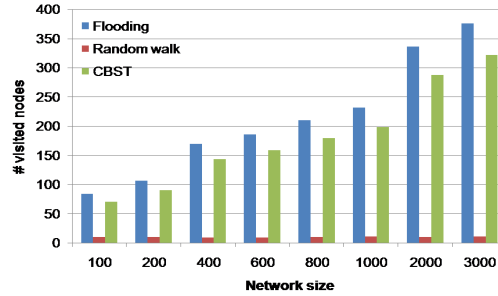
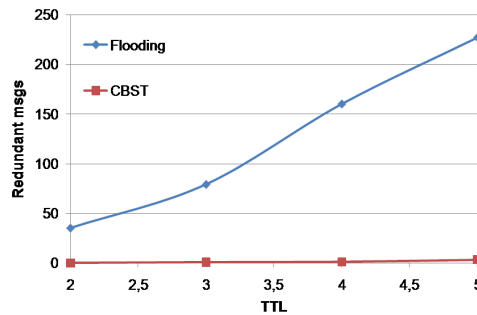


Figure 6: Search accuracy vs. network size

Figure 7: Influence of TTL

5.2.2 Influence of TTL

Here, we fix the network size to 100 and we vary the value of TTL between 2 and 5. Figure 7 gives the number of redundant messages for both flooding and $CBST$ techniques. We can see that, as long as TTL is less than the maximum value of cluster diameter D , the $CBST$ produces a very limited number of redundant messages. Moreover, we note that in our approach, a redundant message may not be discarded, and thus may contribute to improve search accuracy. To illustrate, a given node may receive a first query message as being a partner in a cluster, and another message as being a node locating on the path leading to a new cluster. Once TTL exceeds the cluster diameter, the number of $CBST$ redundant messages increases slightly. However, this number is much smaller than the one produced by flooding. For instance, the number of redundant messages is limited to 4 for a TTL value of 5.

5.3 Performance in Dynamic Systems

The previous results have shown that $CBST$ achieves a very good trade-off between search accuracy and bandwidth consumption. However, it is very important to study the performance of $CBST$ in dynamic P2P systems. In order to propagate a query Q , $CBST$ uses the cluster-based routing tables which may be in an invalid state while recovering from a node arrival/departure.

We assume that 10% of the network size are initially disconnected. Then, nodes start to leave the system at a rate τ . We consider that each node depar-

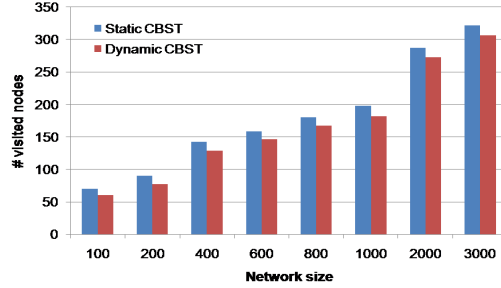


Figure 8: Search accuracy vs. network size

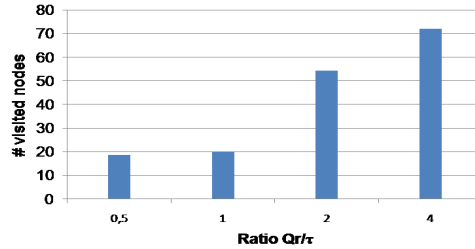


Figure 9: Search accuracy vs. rate ratio

ture is followed by the arrival of another node, such that the total number of nodes remains constant in the system. Simultaneously, users issue queries with a uniform query rate Q_r . In a first set of experiments, we set the rate τ of node connection/disconnection and the query rate Q_r to 0.1, i.e. 1 event (connection/disconnection or query) per 10 minutes. Current file-sharing P2P systems are well characterized by such rate values, as well as by the ratio τ/Q_r value of 1. Figure 8 depicts the number of visited nodes in function of the network size, in both static and dynamic settings. We can observe that node dynamicity results in small degradation of the search accuracy of *CBST*. This degradation is due to the stale entries in routing tables used while propagating a query Q . However, as discussed in Section 3.1.2, the node arrival/departure are localized events, and thus only queries that are issued in the same locality are affected. Because of that, we can see also that the number of visited nodes is reduced by a constant factor, which is independent of the network size. Since the rate ratio Q_r/τ is quite related to the nature of P2P applications, we choose to study the search accuracy of *CBST* while varying the value of that ratio. Here, the network size is fixed to 100 nodes. Figure 9 shows that, for a given query rate, the more stable is the network, the higher is the search accuracy. For instance, *CBST* can improve the search efficiency of database P2P applications, which are characterized by a higher stability.

6 Comparison with Related Work

To better understand the efficiency of *CBST* technique, we compare our performance to other blind search techniques. The flooding mechanism provides the highest accuracy values (for a given *TTL*), since it allows to cover the entire network. However, it produces a huge traffic overhead in the network.

Many works have proposed other techniques that aim to overcome the problem of bandwidth consumption. Quantitative comparisons between the well known techniques are provided by [17]. Lv *et al.* [16] have proposed replacing flooding with random walks. The most important advantage of this algorithm is the significant message reduction it achieves, since it produces $k * TTL$ messages in the worst case. It also achieves some kind of local “load balancing”, since no nodes are favored in the forwarding process over others. However, the most serious disadvantage is its highly variable performance. It has been shown in [17], that this algorithm displays low accuracy and requires large values of *TTL* in order to satisfy the query.

Adamic *et al.* [8] addressed this problem by recommending that instead of using purely random walks, the search protocol should bias its walks toward high-degree nodes. The intuition behind this is that if we arrange for neighbors to be aware of each other’s shared files, high-degree nodes will have (pointers to) a large number of files and hence will be more likely to have an answer that matches the query. However, this approach ignores the problem of overloaded nodes. In fact, by always biasing the random walk towards high-degree nodes, it can exacerbate the problem if the high-degree node does not have the capacity to handle a large number of queries. Other blind search protocols operate on hybrid topologies built upon the notion of Ultrapeers [2] or Supernodes [6]. Obviously, such protocols have to be supported with efficient ultrapeer/supernode selection mechanism.

7 Conclusion

In this paper, we studied how the search efficiency can be improved by clustering a P2P network based on node connectivity. Our solution does not impose a specific network structure or require indexes over the shared data. Instead, it simply relies on inherent clustering patterns that can be extracted from the underlying topologies.

We proposed a cluster-based search technique that is implemented over the *SDC* protocol, an appropriate connectivity-based clustering protocol for P2P systems. Each node maintains a routing table that provides information about paths to partners belonging to its cluster, as well as to other reachable clusters. Such cluster-based routing tables are created/maintained using the clustering messages involved by the *SDC* protocol, without requiring additional traffic overhead. Then, we discussed the performance of our search technique according to search accuracy, and bandwidth consumption. This performance is evaluated through simulation. An interesting result shows that the proposed technique allows eliminating redundant messages, which prevent many search techniques from scaling up. In return, only a small decrease in the number of visited nodes per query is incurred.

References

- [1] A.Crespo and H.G.Molina. Routing indices for peer-to-peer systems. In Proc. of the 28 th Conference on Distributed Computing Systems, 2002.
- [2] A.Singla and C.Rohrs. Ultrapeers: another step towards gnutella scalability. Technical report, 2002.
- [3] B.Yang and H-G.Molina. Improving search in peer-to-peer networks. In Proc of the 22 nd International Conference on Distributed Computing Systems (ICDCS), 2002.
- [4] F.Howell and R.McNab. Simjava: a discrete event simulation package for java with the applications in computer systems modeling. In Int. Conf on Web-based Modelling and Simulation, San Diego CA, Society for Computer Simulation, 1998.
- [5] <http://www.cs.bu.edu/brite/>.
- [6] <http://www.gnutella2.com>.
- [7] I.Stoica and *et al.* Chord: A scalable peer-to-peer lookup service for internet applications. In Proc ACM SIGCOMM, 2001.
- [8] L.Adamic and *et al.* Search in power law networks. Physical Review E, 64:46135–46143, 2001.
- [9] Y. Li, L. Lao, and J.-H. Cui. Sdc: A distributed clustering protocol for peer-to-peer networks. In Networking, pages 1234–1239, 2006.
- [10] L.Ramaswamy, B.Gedik, and L.Liu. A distributed approach to node clustering in decentralized peer-to-peer networks. IEEE Transactions on Parallel and Distributed Systems, 16(9):814–829, 2005.
- [11] M.Ripeanu, I.Foster, and A.Iamnitchi. Mapping the gnutella network. IEEE Internet Computing Journal, 6(1), 2002.
- [12] L. Ramaswamy, B. Gedik, and L. Liu. Connectivity based node clustering in decentralized peer-to-peer networks. In P2P '03: Proceedings of the 3rd International Conference on Peer-to-Peer Computing, page 66, 2003.
- [13] S.Dongen. A new cluster algorithm for graphs. Technical report, Amsterdam, 1998.
- [14] S.Ratnasamy, P.Francis, M.Handley, R.M.Karp, and S.Shenker. A scalable content-addressable network. In SIGCOMM, 2001.
- [15] S.Saroiu, P.Gummadi, and S.Gribble. A measurement study of peer-to-peer file sharing systems. In Proc of Multimedia Computing and Networking (MMCN), 2002.
- [16] Q. *et al.* Search and replication in unstructured peer-to-peer networks. In ACM Int. conference on Supercomputing, 2002.
- [17] D. Tsoumakos and N. Roussopoulos. A comparison of peer-to-peer search methods. In Int.Workshop on the Web and Databases (WebDB), pages 61–66, 2003.

Contents

1	Introduction	3
2	Connectivity-based clustering schemes	4
2.1	SCM: Accuracy Measure for Graph Clustering	5
2.2	The SDC Protocol	5
3	CBST for unstructured P2P systems	7
3.1	Cluster-based Routing Table	7
3.1.1	Routing Table Description	7
3.1.2	Table Maintenance	8
3.2	Query Propagation	12
4	Discussion	14
4.1	Search Accuracy vs. BW Consumption	14
4.2	Clustering Scheme Accuracy vs. BW Consumption	14
5	Experimental Results	15
5.1	Simulation Setup	15
5.2	Performance in Static Systems	16
5.2.1	Search accuracy vs. Bandwidth Consumption	16
5.2.2	Influence of TTL	17
5.3	Performance in Dynamic Systems	17
6	Comparison with Related Work	19
7	Conclusion	19



Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
<http://www.inria.fr>
ISSN 0249-6399