

Combining the use of clustering and scale-free nature of user exchanges into a simple and efficient P2P system^{*}

Pierre Fraigniaud¹, Philippe Gauron², and Matthieu Latapy³

¹ CNRS, LRI, Univ. Paris-Sud, 91405 Orsay, France. pierre@lri.fr

² LRI, Univ. Paris-Sud, 91405 Orsay, France. gauron@lri.fr

³ CNRS, LIAFA, Univ. Paris VII, 75005 Paris, France. latapy@liafa.jussieu.fr

Abstract. It was recently observed that the user interests in P2P systems possess *clustering* properties that can be used to reduce the amount of traffic of flooding-based search strategies. It was also observed that the user interests possess *scale-free* properties that can be used for the design of routing-based search strategies. In this paper, we show that the combination of these two properties enables the design of an efficient and simple fully decentralized search strategy. This search strategy is simple in the sense that it does not require maintaining any structured overlay network topology connecting the peers. It is efficient in the sense that simulations processed on *real-world* traces show that lookups perform in logarithmic expected number of steps.

1 Preliminaries

This paper focuses on *fully decentralized* Peer-to-Peer (P2P) systems, i.e., a (large) set of users, called *peers*, exchanging information in the absence of any central service. Such P2P systems are self-organized, and all peers play the same role. Searching for objects (files, resources, etc.) in such systems requires the use of specific algorithms: object queries are transmitted from peer to peer; if a peer p receives a query for some object that it can provide, then p simply sends the object to the demander; otherwise, p forwards the query to one or several neighboring peer(s). The way peers are connected, and the choice of the peer(s) the query is forwarded to, are essential parts of the P2P system architecture.

Two main ways of forwarding queries in fully decentralized P2P systems have been identified: either by flooding (as in, e.g., Gnutella), or by using Distributed Hash Tables and their underlying routing protocols (as in, e.g., CAN [12] or Chord [14]). Both ways present some drawbacks. In particular, the traffic induced by flooding consumes a significant portion of the bandwidth, and DHT-based protocols use ad hoc connections between peers which are generally hard to maintain and hardly support sophisticated queries. As a consequence, the design

^{*} The two first authors received supports from the INRIA project "Grand Large", and from the project PairAPair of the ACI "Grandes Masses de Données".

of simple search protocols insuring both quick answers and low control traffic is still an open problem. Roughly speaking, one is facing the following alternative: either connecting the peers in an unstructured manner – which is simple but requires flooding – or connecting the peers in a structured manner – which enables routing but is complex.

In this paper, our objective is to propose a method possessing both advantages: *simple* search in an *unstructured* P2P system. To achieve this, we will mainly use the statistical properties of real-world peer-to-peer queries and we use the interest graph as overlay.

1.1 The central idea

Peers are nodes of a *physical* underlying network, i.e., the Internet which support physical communication between peers. Fully decentralized P2P systems are based on virtual connections between peers, which form an *overlay* network on top of the physical network. Basically, a peer p_1 is connected to a peer p_2 in the overlay network if p_1 knows the physical address of p_2 , and vice versa. Communications between neighboring peers in the overlay network are routed in the physical network via its communication primitives. The P2P system has no control on the way the communications are processed by the physical network but it fully controls the overlay network, and it supports the search procedure.

As argued by various researchers (see [1] and the references therein), the overlay network should better map to the physical network, so that neighboring peers in the overlay network would be close in the physical network. A communication between two neighboring peers in the overlay network would then be processed quickly by the underlying physical network. It is shown in [1] that one can dynamically maintain an overlay network such that the distance between two peers in the overlay network is no more than $1 + \epsilon$ times their distance in the physical network, for any $\epsilon > 0$. However, this approach requires using rather complex control procedures.

Alternatively, one could relate the overlay network to the interests of peers. Indeed, if two peers have interests in common, then they will probably exchange a lot, and thus they should better be close in the overlay network. In practice, peers do not exchange objects with arbitrary other peers. Instead, they tend to group themselves into communities, with lots of exchanges inside the communities, and only few exchanges between them. Several authors already noticed this fact and, based on it, have proposed some improvements for existing systems (see, e.g., [5, 9, 15, 16, 8]). In this paper, our objective is to show that these works can be pushed further while keeping our solution very simple.

1.2 Peer interests as a graph

One can represent the peer interests as a graph in which two peers are connected if and only if they have some interests in common. Deriving a formal definition of what is meant by “having some interests in common” is a difficult task. Various propositions have been made, based on keywords, objects in common, etc. We

will here use the following simple definition: two peers are connected in the interest graph if they have exchanged an object in the past. Note that two such peers may actually have very different interests, but at least their interests are related in some way. Note also, and this is essential in our context, that, since the P2P system processes all the queries, it contains a (distributed) view of the interest graph at any time.

It has been recently shown (cf., e.g., [7, 9, 8]) that, similarly to most social networks and most real-world complex networks, the interest graph as defined above has several non-trivial statistical properties that make it very different from standard random graphs. In particular, the interest graph has:

- a low density (the average degree is very low compared to the number of nodes);
- a small average distance (it typically scales logarithmically with the size of the network);
- a clustered structure (the graph is locally dense although the (global) density is low)
- a scale-free nature (i.e., degrees are very heterogeneous, most nodes having a low degree, but some nodes having a high degree).

In our method, the overlay network will be nothing but the interest graph as described above, and the performances of our method will strongly rely on the aforementioned four properties. In fact, our method is based on both the clustering *and* the scale-free nature of the interest graph, two notions that were considered separately in several previous works, listed thereafter.

1.3 Using scale-free properties

Using the scale-free nature of real-world networks for the design of efficient search strategies have been proposed in [2, 6, 13]. In these papers, the authors approximate the heterogeneous degree distributions by power laws and study the properties of some (random or deterministic) walks in random graphs with power law degree distributions (see also [10]).

In [2], at each step of the search strategy, the current node scans its neighbors and if none has the searched data, then the query is forwarded to the highest degree neighbor. A mean-field analysis, confirmed by simulations, shows that the expected number of steps required to find an object in a random power law network with n nodes and exponent α , scales sub-linearly as $n^{3(1-2/\alpha)}$ for $2 < \alpha < 3$.

In [6], the authors perform simulations on another model of power law networks, and compare a random walk search strategy with a search strategy guided by high degree nodes. They observe that the latter search strategy performs better than the former. In particular, it returns a path of polylogarithmic length between the source and the target. Nevertheless, the search strategy performs in a polynomial number of steps due to loops in the search path.

In [13], the authors propose an original approach. Every node first publishes its data at every node along a random walk of length L . The search strategy then

proceeds along a random walk of same length, and every node traversed by the walk starts partially flooding the network (the search is sent through every edge with probability $< q$, where q is the percolation threshold of the network). It is then shown that this search efficiently locates the data by setting $L \sim n^{1-2/\alpha}$ for $2 < \alpha < 3$. The authors also present heuristics reducing the amount of traffic induced by this strategy.

1.4 Using clustering properties

Just like the heterogeneous nature of peers is captured (in part) by the degree distribution, some cultural and social factors induce a clustered structure of the interest graph. For example, if a peer p_1 is interested in an object \mathcal{O} held by another peer p_2 , then it will probably be interested in other objects held by p_2 . Moreover, p_1 will also probably be interested in objects held by other peers interested in \mathcal{O} . This can be summarized by the following facts:

- peers organize themselves in communities (dense subgraphs), and
- two peers which exchanged data are likely to exchange other data in the future.

Based on these observations, [15] proposed to enhance Gnutella with an *interest-based* structure in which a link (called *shortcut*) between peers that have exchanged an object is added on top of the Gnutella network. Simulations show that the shortcuts reduce the total load of the system by a factor 3 to 7. Hence, the clustered nature of the interest graph can be used to improve search strategies. Nevertheless, this search strategy remains based on flooding the network.

Other contributions pointed out that the clustered nature of the interest graph could be used to design efficient P2P systems [5, 8, 9, 15, 16], but no protocol has actually been proposed.

1.5 Our contribution

The previous works surveyed before gave some evidence to the fact that the scale-free nature of the interest graph, as well as its clustered structure, are two basic statistical properties that can be used for improving search strategies in P2P systems. Nevertheless, these works all used one of these two properties only. In this paper, we show that using a combination of these two properties results in even better performances.

We present the QRE (pronounced *query*) protocol, in which the overlay network is nothing but the interest graph. Despite the fact that this graph is not structured, we present a simple search procedure that is not based on flooding, and does not require any information on the global topology of the overlay network. This search procedure is simple in the sense that it does not require sophisticated publish procedures. Moreover, because of the somewhat greedy maintenance of the overlay network, joining and leaving procedures are both very simple.

To evaluate the performance of our method, we have performed intensive simulations based on *real-world traces*. These simulations demonstrate that our search procedure locates objects in a *logarithmic* expected number of steps, which outperforms all previous solutions based on only one of the two basic statistical properties of the interest graph (scale-free nature or clustered structure).

2 The QRE protocol

This section is devoted to the description of the QRE protocol. In order to illustrate the main features of QRE, we deliberately keep it as simple as possible. Moreover, keeping QRE simple enables evaluating the direct impact of our contribution, without mixing it with other optimizations.

Connections between peers in the overlay network of QRE are driven by the queries processed in the system: a peer is connected to the peers to which it has uploaded an object, or from which it has downloaded an object. Queries are routed by a *search* procedure (described below), and are of the form $\langle @, \mathcal{O}, k \rangle$ where $@$ is the physical address of the source peer initiating the query, \mathcal{O} is the description of an object, and $k \geq 1$ is the number of different providers of \mathcal{O} the source wants to get.

We assume that each peer in the system stores the objects that it provides, as well as a (compact) description of these objects in a local lookup table. We also assume that every peer stores a local copy of the lookup table of each of its neighbors. Regular communications between a peer p and its neighbors allows the system to support this facility. Finally, we assume that every peer knows the degree (number of neighbors) of each of its neighbors.

2.1 The search method

For routing a query $Q = \langle @, \mathcal{O}, k \rangle$, QRE essentially executes a (distributed) depth-first search (DFS) where the priority is given to highest degree nodes. More precisely, for every peer p that is receiving a query Q , if neither p nor any of its neighbors can positively answer to Q , then p forwards Q to its highest degree neighbor among the ones which have not yet processed Q ; if there is none, then p sends the query Q back to the peer from which it received Q . Figure 1 summarizes this simple search procedure. The search procedure proceeds this way until k copies of \mathcal{O} have been found.

To avoid loops in the search, every peer stores the list of queries Q that it has processed so far, as well as the identity of the neighbors to which Q has already been forwarded.

Note that QRE does not use hashing. As a consequence, it can support complex queries, as wild-cards searching, regular expression searching, or interval searching. Note also that the search is exhaustive in QRE.

<pre> (1) if p has \mathcal{O}, then p sends \mathcal{O} to $@$; (2) else (2.1) if p has a neighbor p' that stores \mathcal{O} then p forwards Q to p'; (2.2) else if all the neighbors of p have already received Q then p sends Q back to the neighbor from which it received Q; (2.3) else p forwards Q to its neighbor of maximum degree among the ones that have not yet received Q. </pre>

Fig. 1. The search procedure in QRE for the query $Q = \langle @, \mathcal{O}, 1 \rangle$.

2.2 Dynamics of the system

In QRE, any successful search induces a modification of the connections between the peers: if p_1 receives a positive answer for a query Q from another peer p_2 , then a link is set between p_1 and p_2 , i.e., p_1 and p_2 exchange their addresses and their lookup tables. In addition, their neighbors are informed of the changes in their degrees. This way, the system maintains an overlay network which is nothing but the interest graph as defined before.

As in most previously proposed P2P systems, we assume that any peer aiming at joining the system knows an *entry point*, i.e. a peer already in the system, whose address is publicly available. We assume that a joining peer always wants to provide or to get an object. Therefore, a joining peer is always associated to an object. The join procedure is based on such an object, say \mathcal{O} : the joining peer sends a query for \mathcal{O} and connects, as specified before, to the peer(s) that answer(s) to this query. If no peer returns a positive answer, then the joining peer connects directly to the entry point.

In QRE, when a peer wants to leave the system, it sends a leaving message to all its neighbors, and disconnects from the system. Any peer receiving a leaving message removes the sender from its lookup table, and informs its neighbors of its new degree. Note that QRE can also handle brutal departures of peers by periodically checking the presence of neighbors.

We stress the fact that QRE does not need the use of any underlying P2P system. The joining procedure is self-contained. The overlay graph grows from the entry point, based solely on the queries, and on their answers. The first peers will typically connect directly to the entry point (because the data they look for are not in the system). However, the new peers will eventually receive positive answers, and the overlay will then grow in a non-trivial manner.

3 Performance of QRE

There is currently no satisfactory model capturing the peers behavior accurately enough to enable a formal evaluation of our method (i.e., including simultaneously: clustering properties, scale-free properties, and the fact that two neighboring peers will probably exchange objects more than once). Because of this,

we performed simulations on *real-world* traces, extracted from **eDonkey** [3], and described in detail in [7]. The trace upon which we performed our simulations is 2h 53mn long and involves 46,202 peers. It contains the search requests of the users, but not the connections and disconnections.

3.1 Simulation protocol

From the trace, we extracted a (chronological) list of tuples $(p_0^i, p_1^i, p_2^i, \dots, p_{k_i}^i)$, each associated to a query Q^i . Peer p_0^i is the source of the i^{th} query, k_i is the number of requested providers, and the p_j^i , $j = 1, 2, \dots, k_i$, are the providers. We have considered 342,204 queries of that type, involving 46,202 nodes in total.

Our simulator proceeds with each tuple, step by step, as follows. Step i considers tuple Q^i , and simulates the behavior of QRE when dealing with a request Q where $p_1^i, p_2^i, \dots, p_{k_i}^i$ are the providers of the object requested by p_0^i . In other words, we simulated the behavior of QRE, as described in Section 2, for a query $\langle p_0^i, \mathcal{O}_i, k_i \rangle$ where $p_1^i, p_2^i, \dots, p_{k_i}^i$ are the peers currently holding \mathcal{O}_i .

If p_0^i is not yet in the network at step i , then the simulator performs the join procedure for p_0^i where the entry point is chosen uniformly at random among the peers currently in the network. Then, p_0^i connects to the entry point and launches the query. Its link to the entry point is removed when p_0^i receives an answer from a peer providing \mathcal{O}_i (and thus connects to it).

3.2 Simulation results

Figure 2 displays the degree distribution of the peers (after all the requests have been processed), i.e., for $k \geq 1$, the number $\delta(k)$ of peers with degree k . This distribution is heavy tailed (there are peers with large degree). However, $\delta(k)$ does not follow a strict power law. Nevertheless, we will see that the heavy tail is sufficient for our search strategy to perform efficiently. Importantly, the maximum degree is 690, but only 0.25% of the peers have a degree larger than 300. Conversely, 2/3 of the peers have a degree ≤ 20 . The average degree is 47.9. These characteristics prove that QRE scales well with the number of nodes.

Figure 3 displays the average number of steps required to locate one copy of an object, as a function of the total number of providers of this object. This number of steps decreases rapidly with the number of providers. In fact, locating an object that has at least seven copies in the network requires at most 10 steps on average, and a popular object \mathcal{O} has, on average, a copy present on a node at distance at most 2 from a peer requesting \mathcal{O} . Importantly, \mathcal{O} is not necessarily at distance at most 2 from any peer, but \mathcal{O} is at distance at most 2 from any peer *interested in* \mathcal{O} . This demonstrates the existence of communities in the interest graph, captured and used by QRE.

From Figure 3, rare objects are located by the search procedure of QRE after a relatively large number of steps. However, once this price has been paid by some peer, the next searches for the same object will require less and less steps while there are more and more copies of the object in the network, and

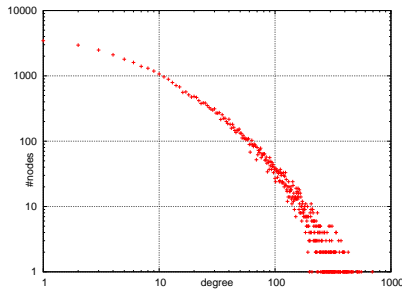


Fig. 2. Degree distribution in the overlay network of QRE.

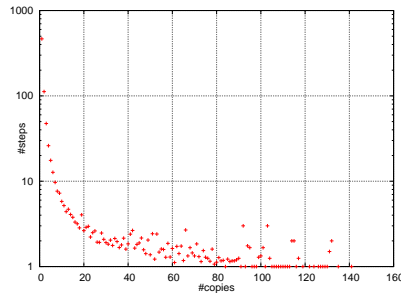


Fig. 3. Impact of the number of providers on the search time.

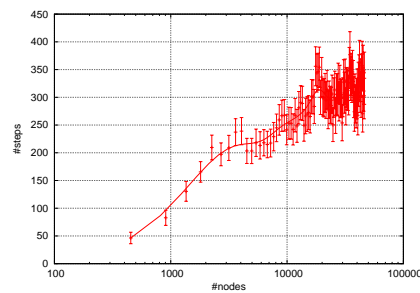
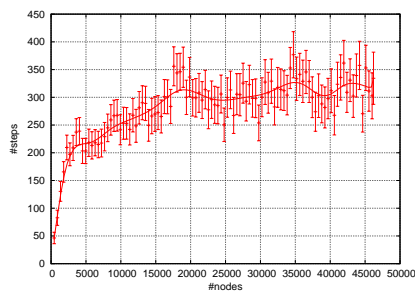


Fig. 4. Average number of steps to locate an object.

while its providers are more and more connected. Figure 4 (left) displays the average number of steps $s(n)$ required to locate one copy of an object, as a function of the number n of peers in the system. Linear regression indicates that $s(n)$ scales linearly with the logarithm of the number n of peers in the system (see Figure 4 (right)). This is the most important experimental result in our contribution: *searching in QRE requires $O(\log n)$ number of steps on average*. In particular, the search procedure of QRE performs at least as well as the search procedures of DHTs like Chord [14], Viceroy [11], or those based on the binary *de Bruijn* graph (see [4] and references therein).

Figure 5 displays the average number of steps required to locate 20% of the copies of an object currently present in the system. Locating copies of a popular object requires at most ten steps. As a consequence, QRE could be efficiently used in combination with any protocol enabling downloading large files from several providers in parallel.

Finally, Figure 6 displays the number of queries that required k steps to be performed, which is well fitted by a power law. Most queries require few steps to be performed, and only very few queries require lot of steps (this corresponds to very rare data, for which it is necessary to search a large portion of the network). Typically, a TTL of 100 steps would enable most queries to be satisfied.

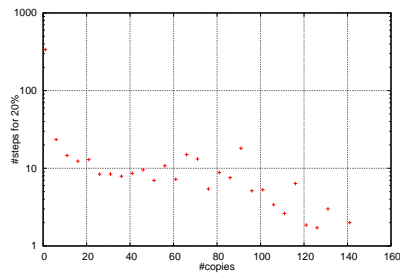


Fig. 5. Search time for 20% of the providers

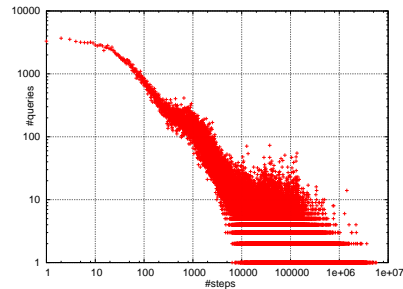


Fig. 6. Distribution of the number of hops to find an object.

4 Conclusion

In this paper, our main objective was to push further the idea of using the properties of the peer interest for the design of efficient search strategies in P2P systems. We indeed believe that these properties are among the key factors to be considered for the design of efficient and fully decentralized P2P systems. To support this belief, we demonstrate that the interest graph can be used as an overlay network supporting very simple procedures for searching, joining, and leaving the system. The main properties of the interest graph are (1) its clustered structure, and (2) the heterogeneity of the node degrees. Our search strategy uses these two properties. It locates objects in a logarithmic expected number of steps, without flooding, nor using any sophisticated routing or publish procedures.

Probably, more subtle and more efficient search strategies could be defined on the interests graph, using other properties of the graph, or combining our approach with others. In particular, it is unclear whether the DFS algorithm selecting high degree nodes first is the most appropriate search strategy for the interest graph. Further investigations are also required to measure the impact of limiting the maximum degree of the peers, as well as other issues like the the load of the high-degree nodes, the robustness of the system.

References

1. I. Abraham, D. Malkhi, and O. Dobzinski. LAND: Stretch $(1 + \epsilon)$ locality-aware networks for DHTs. In 15th ACM-SIAM Symp. on Discrete Algorithms (SODA), pages 550-559, 2004.
2. L. Adamic, R. Lukose, A. Puniyani, and B. Huberman. Search in power law networks. *Physical Review E*, vol. 64, pages 46135-46143, 2001.
3. **eDonkey**: www.edonkey2000.com/
4. P. Fraigniaud and P. Gauron. An overview of the content addressable network D2B. Brief announcement at the 22nd ACM Symp. on Principles of Distributed Computing (PODC), page 151, 2003.

5. S. Handurukande, A.-M. Kermarrec, F. Le-Fessant, and L. Massoulié. Exploiting Semantic Clustering in the eDonkey P2P Network. In 11th SIGOPS European Workshop (SIGOPS), pages 109-114, 2004.
6. B. Kim, C. Yoon, S. Han, and H. Jeong. Path finding strategies in scale-free networks. *Physical Review E*, vol. 65, pages 027103-1–027103-4, 2002.
7. S. Le-Blond, M. Latapy, and J.-L. Guillaume. Statistical analysis of a P2P query graph based on degrees and their time evolution. In 6th Int. Workshop on Distributed Computing (IWDC), 2004.
8. S. Le-Blond, M. Latapy, and J.-L. Guillaume. Clustering in P2P exchanges and consequences on performances. In 4th Int. Workshop on Peer-To-Peer Systems (IPTPS), 2005.
9. F. Le-Fessant, S. Handurukande, A.-M. Kermarrec, and L. Massoulié. Clustering in Peer-to-Peer File Sharing Workloads. In 3rd Int. Workshop on Peer-to-Peer Systems (IPTPS), 2004.
10. Q. Lv, P. Cao, E. Cohen, K. Li, S. Shenker. Search and replication in unstructured peer-to-peer networks. In 6th Int. Conf. on Supercomputing, pages 84-95, 2002.
11. D. Malkhi, M. Naor, and D. Ratajczak. Viceroy: a scalable and dynamic lookup network. In 21st Symp. on Principles of Distributed Computing (PODC), 2002.
12. S. Ratnasamy and P. Francis and M. Handley and R. Karp and S. Shenker. A scalable content-addressable network. In SIGCOMM, pages 161-172, 2001.
13. N. Sarshar, P. Boykin, and V. Roychowdhury. Percolation search in power law networks: making unstructured peer-to-peer networks scalable. In 4th International Conference on Peer-to-Peer Computing, pages 2-9, 2004.
14. I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup service for Internet applications. In SIGCOMM, pages 149-160, 2001.
15. K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In INFOCOM, pages 2166-2176, 2003.
16. S. Voulgaris, A.-M. Kermarrec, L. Massoulié, and M. van Steen. Exploiting Semantic Proximity in Peer-to-peer Content Searching. In 10th IEEE Int. Workshop on Future Trends in Distributed Computing Systems (FTDCS), pages 238-243, 2004.