Combining the use of clustering and scale-free nature of exchanges into a simple and efficient P2P system

Pierre Fraigniaud* Philippe Gauron† Matthieu Latapy‡

Abstract

It appeared recently that user interests in a P2P system possess *clustering* properties that may be used to reduce significantly the amount of traffic of flooding-based search strategies. It was also observed that they possess *scale-free* properties that may be used for the design of efficient routing-based search strategies. In this paper, we show that the combination of these two properties make it possible to design an efficient and simple fully decentralized search strategy. Further, simulations processed on real-world traces show that other unidentified properties hidden in actual queries make our protocol even more efficient, performing searches in logarithmic expected number of steps.

1 Preliminaries

We focus here on *fully decentralized* Peer-to-Peer (P2P) systems, characterized by the fact that a (large) set of users, called *peers*, exchange information without 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 it can provide, then it symply sends it to the demander; else, p forwards the query to one or several neighboring peer(s). The way peers are connected together and the choice of the peer(s) the query is forwarded is an essential part of the P2P system architecture.

There are currently two main known ways of forwarding queries in fully decentralized P2P systems: either by flooding (as in, e.g., Gnutella), or by using Distributed Hash Tables (DHTs) and their underlying routing protocols (as in, e.g., Chord [13] or CAN [11]). 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. As a consequence, the design 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 connect the peers in an unstructured manner – which is simple but requires flooding – or connect them in a structured manner – which enables routing but is complex.

Our aim in this paper is to propose a protocol having both avantages: *simple* search in an *unstructured* P2P system. To acheive this, we will mainly use the statistical properties of real-world queries.

1.1 The central idea

Peers are nodes of a *physical* underlying network, e.g., the Internet. This network is supporting the basic communication primitives between peers. Fully decentralized P2P systems (the ones considered in this paper) are based on virtual connections between peers, which forms 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 (e.g., the IP 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 these communications are processed, 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 there-

*CNRS, Laboratoire de Recherche en Informatique (LRI), Univ. Paris Sud, 91405 Orsay, France. E-mail: pierre@lri.fr

†Laboratoire de Recherche en Informatique (LRI), Univ. Paris Sud, 91405 Orsay, France. E-mail: gauron@lri.fr

‡CNRS, Laboratoire d'Informatique Algorithmique : Fondements et Applications (LIAFA), Univ. Paris VII, 2 place Jussieu, 75005 Paris, France. E-mail: latapy@liafa.jussieu.fr

fore map to the physical network, so that neighboring peers in the overlay network would be close in the physical networks. A communication between two neighbour peers in the overlay network should 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 in it is not more than $1+\epsilon$ times the distance in the physical network, for any $\epsilon > 0$. However, this approach requires complex control procedures. Moreover, if a subnetwork disconnects from internet, the incidence in logical network (P2P) network will not be disconnection of distant nodes but probable disconnection of a large number of nodes strongly connected together, which could induces long latencies.

In a similar way, one may try to relate the overlay network to the interests of peers: if two peers have interests in common then they will probably exchange much and so they should be close in the overlay network. Indeed, peers do not exchange objects with arbitrary other peers. Instead, peers tends to group themselves into communities, with lots of exchanges inside communities, and only few exchanges between them. Several authors already noticed it and proposed some improvements of axisting systems based on it [14, 15, 5, 8]. Our aim here is to show that this idea can be pushed much further, with the advantage of keeping the system very simple.

1.2 Peer interests as a graph

On may represent peers interest by a graph in which two peers are connected iff they have some interests in common. The 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, or vectorial representations. We will here use the following definition: two peers are connected in the interest graph if they exchanged an object in the past. Notice that two such peers may actually have very different interests, but it seems clear that in general their interest are related in some way. Notice also, and this is essential in our context, that, since the P2P system processes all the queries, it has a (distributed) knowledge of the interest graph between the peers in the system, at any moment.

It has been shown recently [8, 7] that such graphs, like most social networks and real-world complex networks, have several non-trivial which make them very different from random graphs. In particular:

- they have a low density (the average degree is very low compared to the number of nodes),
- their average distance is small (it typically scales logarithmically with the size of the network),
- they have a clusterized structure (despite the fact that their global density is low, they are locally dense)
- they have a scale-free nature (degrees are very heterogeneous, most nodes having a low degree but some having a high one).

In our protocol, the overlay network will be nothing but the interest graph we have just describe. Therefore, its performances will strongly rely on these properties. To design it, we will use some previous works which we quickly describe now.

1.3 Using scale-free properties

Some propositions for the use of real-world networks scale-free nature for the design of efficient search strategies have been made in [2, 6, 12]. 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 such degree distributions.

In [2], at each step of the search process the current node scans its neighbors (or even the neighbors of its neighbors), and if none has the searched data, then the query is forwarded to the highest degree neighbor. A mean-field analysis of this process, 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 $2 < \alpha < 3$ scales sub-linearily as $n^{3(1-2/\alpha)}$.

In [6], the authors perform simulations on another model of power law networks, and compare the random walk search with the search guided by high degree nodes. They observe that the latter search strategy performs better than the former [1].

In [12], the authors propose an original approach. Every node first publishes its data on all nodes along

[1] Notice however that, despite one may understand from [6] that the search guided by high degree nodes performs polylogarithmically, this is not true: even if the obtained path is of logarithmic length, the search follows loops which give it a polynomial length.

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 clutural and social factors induce a clusterized structure of the interest graph. For example, if a peer p_1 is interested by an object $\mathcal{O}$ held by another peer p_2, then it probably will be interested by other objects held by p_2. Moreover, p_1 probably will also be interested by objects held by other peers interested in $\mathcal{O}$. This can be summarized by the following facts: peers organize themselves in communities, and two peers which exchanged data are likely to exchange other data in the future.

Based on this, [14] proposed to enhance Gnutella with an *interest-based* structure in which a link (called *shortcut*) between peers which have exchanged an object is added on top of the Gnutella network. Simulations based on real-world traces show that the shortcuts reduce the total load of the system by a factor 3 to 7. Hence, the clusterized nature of the interest graph can be used to improve search strategies. Nevertheless, this search strategy remains based on flooding the network.

Other contributions [15, 5, 8] have also shown that clustering in peer interests is indeed important, and may be used to improve current protocols.

1.5 Our contribution

Previous results give evidence for the scale-free nature of peer interest graphs and their clusterized structure. Some works show that current protocols can be improved using one of these properties. We claim here that the interest graph actually plays a central role in the performances of P2P systems, and that many of its properties may be used to design very simple though efficient protocols.

To support this claim, we present a protocol in which the overlay network is nothing but the interest graph (defined by the queries already processed).

By the somewhat greedy nature of this protocol, joining and leaving procedures are very simple. In fact, it also easily allows brutal disconnections of the users, because it does not rely on any structured overlay. On the other hand, we present a very simple search procedure which is not based on flooding, nor it requires any information on the global topology of the overlay, nor it requires sophisticated publish procedures.

To evaluate its performances, we performed intensive simulations on real-world traces. These simulations show that our search procedure locates objects in a *logarithmic* expected number of steps, which outperforms all comparable previous propositions.

2 The QRE protocol

This section is devoted to the description of the QRE (pronounce *query*) protocol. In order to illustrate its main features, we deliberately kept it as simple as possible. One might use several classical heuristics to improve it, but this would result in hiding the main characteristics of QRE. Moreover, the fact that we always make the choice of simplicity makes it possible to evaluate the direct impact of our contribution, without mixing it with other optimizations. We however insist on the fact that the protocol actually can be improved, and that an implementation should take this into account.

Connections between peers in the overlay are driven by the queries processed in the system: a peer is connected to the peers to which it provided an object of which have provided an object to it. These queries are routed by a *search* procedure (described below), and are of the form $\langle @, \mathcal{O}, k \rangle$ where @ is the address of the source peer initiating the query (e.g., its IP address), $\mathcal{O}$ 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 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 (but not necessarily frequent) communications between a peer p and its neighbors allows this (the induced overload can be kept very small, e.g., using Bloom filters). Finally, we assume that each peer knows the degree of each of its neighbors.

2.1 The search protocol

Upon reception of a query $Q = \langle @, \mathcal{O}, k \rangle$, a peer p essentially executes a deep-first search where the priority is given to highest degree nodes: if neither p nor any of its neighbors provide $\mathcal{O}$, then p forwards the query to its highest degree neighbor among the ones which have not already receive Q; if there is none, then p sends the query back to the peer from which it received it.

If $k = 1$, then the search stops as soon as a provider of $\mathcal{O}$ has been found. If $k > 1$ then p decreases it by the number of providers it has found (among itself and its neighbors) and forwards the query as before, with the new number of wanted sources.

Additionally, to avoid loops in the search, the peers must store the list of queries Q that they have processed so far, as well as the identity of the neighbors to which Q has already been forwarded. This can be handled efficiently but we do not enter in these details here. QRE don't use hashtables, which permit the use of substrings search. This functionnality is usefull for lookinf for $\mathcal{O}$ we only know some words of the description.

2.2 Dynamics of the system

Any successful search results in a modification of connections between peers in QRE: if p_1 receives an answer to 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. Their neighbors are informed of the changes in their degrees. This way, the system maintains an overlay which is nothing but the interest graph as defined above (two peers are connected if they already exchanges a data) and in which each peer knows the degree of its neighbors and the data they provide.

As in most previously proposed P2P systems, we assume that any peer which wants to join the system knows an *entry point*, i.e. a peer already in the system, whose address is publicly available. We will suppose that the joining peer always wants to provide or to get an object (else it does not need to enter the system). Therefore, it is always associated to an object (one it provides or one it looks for). The join procedure is based on such an object, say $\mathcal{O}$: the joining peer sends a query for $\mathcal{O}$ and connects, as usual, to the peer(s) that answer(s) this query. If no object giving a positive result exists for the joining peer then it connects directly to the entry point. The entry point is hopefully a friend from the same community which invited the new node (the entry point is then evidently one of the most interested in the arrival of the new node). If so, the number of steps of the connection is decreased.

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

3 Performances of QRE

There is currently no model capturing accuratly enough peers behaviors to make it possible to evaluate our protocol formally. Because of this, we focused on simulations. We used *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.

3.1 Simulation protocol

We extracted from the trace a (chronological) list of tuples $Q^{(i)} = (p_0^{(i)}, p_1^{(i)}, p_2^{(i)}, \ldots, p_{k_i}^{(i)})$, each associated to a query: $p_0^{(i)}$ is the sender of the query, k_i is the number of providers the corresponding object and $p_j^{(i)}$, $j = 1, 2, \ldots, k_i$, are the providers. We obtained $342,204$ 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 $p_1^{(i)}, p_2^{(i)}, \ldots, p_{k_i}^{(i)}$ are the providers of the object wanted by $p_0^{(i)}$. In other words, we simulated the behavior of QRE, as described in Section 2, for a query $\langle p_0^{(i)}, \mathcal{O}_i, k_i \rangle$ when $p_1^{(i)}, p_2^{(i)}, \ldots, p_{k_i}^{(i)}$ are the peers currently providing $\mathcal{O}_i$.

If $p_0^{(i)}$ is not yet in the network at step i, then the simulator performs the join procedure where the entry point is chosen uniformly at random among the peers currently in the network. For the sake of simplicity, we do not present peer departures in this version.

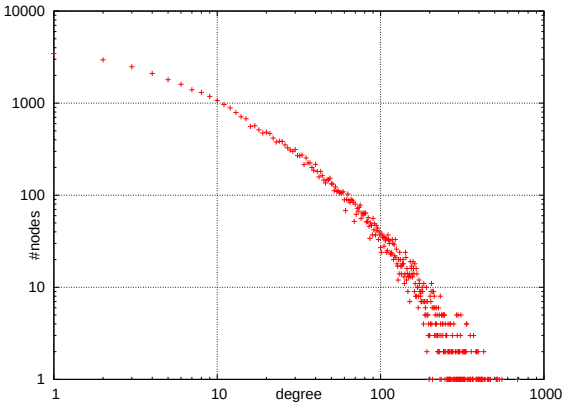

Figure 1: Degree distribution in the overlay of QRE.

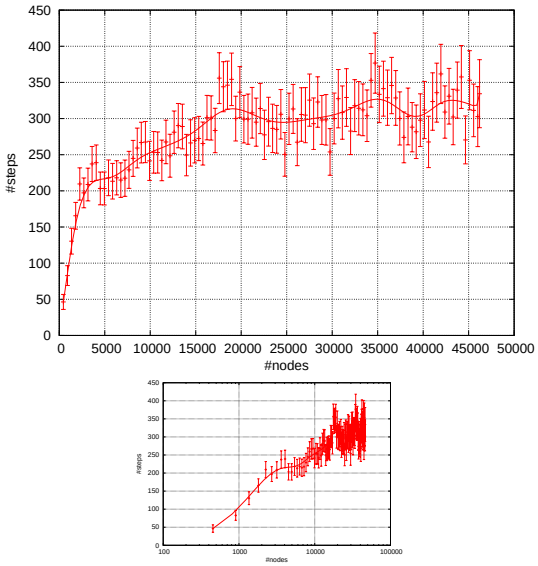

Figure 2: Average number of steps to locate an object.

3.2 Simulation Results

Figure 1 displays the degree distribution of the peers, 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), but $\delta(k)$ does not follow a clear power law. Nevertheless, we will see later 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. Conversly, 2/3 of the peers have a degree ≤ 20. The average degree is 48. These characteristics prove that QRE scales well with the number of nodes.

Figure 2 displays the average number of steps $s(n)$ required to locate one copy of the wanted 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, which shows that the search procedure performs efficiently in QRE: it is as good as DHTs like Chord [13], Viceroy [10], and those based on the binary *de Bruijn* graph ([4] and references therein).

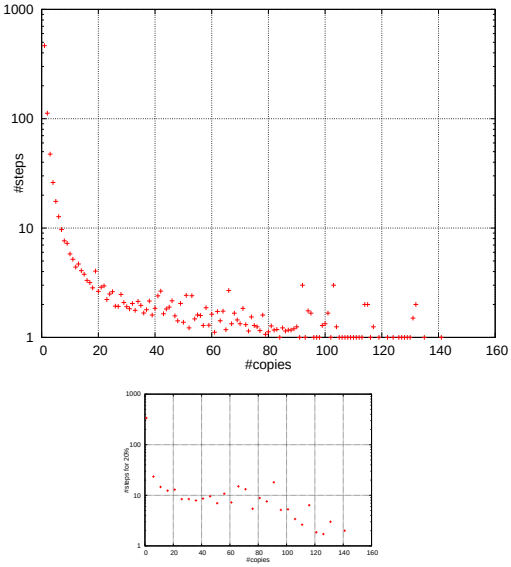

Figure 3: Impact of the number of providers on the search time.

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 present in the system. This number decreases abruptly with the number of providers. In fact, locating an object that has at least seven copies in the network requires no more than 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, note that it does not mean that $\mathcal{O}$ is at distance at most 2 from any peer, but means that $\mathcal{O}$ is at distance at most 2 from any peer *interested by* $\mathcal{O}$. This demonstrates the existence of communities in the interest graph, captured and used in the QRE protocol.

Rare objects are located by the search procedure of QRE after a relatively large number of steps, but, once this price has been payed 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 its provider are more and more connected.

Figure 3 also displays the average number of steps required to locate 20% of the total number of copies

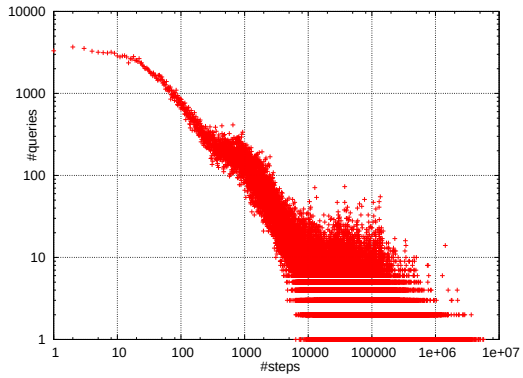

Figure 4: Distribution of the number of steps needed to locate a object.

of an object. Locating up to 21 copies of a popular object requires at most 13 steps.

Finally, Figure 4 displays, for $k \geq 1$, 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 many steps. Typically, a TTL of 100 steps would enable most queries to be satisfied.

4 Conclusion

Our aim is this work was to push the use of peer interest properties much further than before, in order to argue that they actually are among the main perspectives, if not the main, for the design of efficient P2P systems. To achieve this, we proposed the interest graph as an overlay network. We then defined very simple procedures for searching, joining and leaving the system, making it essentially self-organized.

The properties of such graphs, in particular their clusterized structure and the heterogeneity between node degrees, made our protocol very efficient: it locates objects in a logarithmic number of steps in this network, without flooding, nor using sophisticated routing or publish procedures. QRE permit also substring search which is usefull in a lot of applications of the P2P systems.

Certainly more subtile and more efficient protocols may be defined, using other properties or combining this approach with others. More work need to be done in this direction to identify the heuristics which will perform well in practice. Finally, notice that the properties of overlay networks like QRE have other interesting consequences not discussed here, like robustness of the network.

References

[1] I. Abraham, D. Malkhi, and O. Dobzinski. LAND: Stretch $(1+\epsilon)$ locality-aware networks for DHTs. In SODA 2004.

[2] L. Adamic, R. Lukose, A. Puniyani, and B. Huberman. Search in power law networks. Physical Review E, vol. 64, 046135, 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), 2003.

[5] S. Handurukande, A.-M. Kermarrec, F. Le-Fessant, and L. Massoulié. Exploiting Semantic Clustering in the eDonkey P2P Network. 2004. 11-th ACM SIGOPS European Workshop (SIGOPS)

[6] B. Kim, C. Yoon, S. Han, and H. Jeong. Path finding strategies in scale-free networks. Physical Review E, vol. 65, 027103, 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. 2004. International Workshop on Distributed Computing (IWDC), India.

[8] F. Le-Fessant, S. Handurukande, A.-M. Kermarrec, and L. Massoulié. Clustering in Peer-to-Peer File Sharing Workloads. 2004. 3-rd International Workshop on Peer-to-Peer Systems (IPTPS)

[9] Q. Lv, P. Cao, E. Cohen, K. Li, S. Shenker. Search and replication in unstructured peer-to-peer networks. In 6th Int. Conference on Supercomputing, pages 84-95, 2002.

[10] D. Malkhi, M. Naor, and D. Ratajczak. Viceroy: a scalable and dynamic lookup network. In 21st ACM Symp. on Principles of Distributed Computing (PODC), 2002.

[11] 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.

[12] N. Sarshar, P. Boykin, and V. Roychowdhury. Percolation search in power law networks: making unstructured peer-to-peer networks scalable. 2004. Submitted.

[13] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup service for Internet applications. In SIGCOMM 2001.

[14] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In INFOCOM 2003.

[15] S. Voulgaris, A.-M. Kermarrec, L. Massoulié, and M. van Steen. Exploiting Semantic Proximity in Peer-to-peer Content Searching. 2004. 10-th IEEE Int'l Workshop on Future Trends in Distributed Computing Systems (FTDCS)