Statistical analysis of a P2P query graph based on degrees and their time-evolution

Jean-Loup Guillaume, Matthieu Latapy, Stevens Le-Blond

Statistical analysis of a P2P query graph based on degrees and their time-evolution

Jean-Loup Guillaume, Matthieu Latapy, Stevens Le-Blond
LIAFA – CNRS – Université Paris 7
2 place Jussieu, 75005 Paris, France.
(guillaume,latapy,stevens)@liafa.jussieu.fr

Abstract

Despite their crucial impact on the performances of P2P systems, very few is known on peers behaviors in such networks. We propose here a study of these of these behaviors in a running environment using a semi-centralised P2P system (`eDonkey`). To achieve this, we use a trace of the queries made to a large server managing up to fifty thousands peers simultaneously, and a few thousands query per second. We analyse these data using complex network methods, and focus in particular on the degrees, their correlations, and their time-evolution. Results show a large variety of observed phenomena, including the variety of peers behaviors and heterogeneity of data queries, which should be taken into account when designing P2P systems.

Introduction.

In peer-to-peer (P2P) systems, all the participants, called *peers*, play similar roles in the sense that there is no central authority governing the relations between them. Most P2P systems currently running provide a way to exchange data (music files, programs, etc) in a distributed way. When a peer joins a *fully distributed* P2P system, it registers its address and the data it provides on one or several other peers. When it looks for some data, it sends the corresponding query to the peers it knows, which in turn transmit this query to other peers until it reaches a peer sharing the data. This peer then sends the data to the peer seeking it. On the contrary, in a *centralised* P2P system, each peer registers its address and the data it provides on a central server. When it looks for some data, a peer sends the corresponding query to the server, which answers with the address of another (or several other) peer providing the data. The initial peer may then retrieve the data directly from the one sharing it. See [18] for a more complete description.

In such a system, the exchanges between peers are not random: if someone has a data of interest to someone else (a music file for example) then he/she probably has other data of interest for the same person (musics from the same artist for example). Likewise, the peers behaviors (tendency to provide many or few data, for example) has crucial consequences on the way a P2P system will work, in particular on its efficiency. Despite this, very few is nowadays known on P2P exchanges properties and on how peers behave [11, 14].

The distributed aspects and the large size of P2P systems are the main reasons for this lack of knowledge. However, some of the main P2P systems currently running use a mid-term between the centralised and the fully distributed approaches [7]: a small set of servers is used for the processing of the queries and data transfers are managed directly between peers.

In this paper, we analyse a large trace of queries and exchanges processed in such a system, which gives new insight on the properties of exchanges and the peers behaviors. To achieve this, we mainly use the methods from the recent field of complex network analysis, in particular the study of node degrees,

their correlations and their time-evolution. In section 1, we describe the P2P protocol under concern, the trace we use, the statistical tools used to analyse it, and the context into which our work lies. We then study the peers point of view (how do peers behave? what is a typical peer load?) and the data point of view (are there various kinds of data in the system?) in sections 2 and 3 respectively.

1 Preliminaries.

In this section, we describe the P2P protocol we use for our analysis, namely `eDonkey`. We then describe the trace we will study and the way it has been collected. We will represent and analyse these data using graphs and statistical properties of graphs, which will prove to be very efficient tools for this purpose. We introduce and discuss them later in this section. Finally, we describe the context in which our work lies.

The `eDonkey` protocol.

As already pointed out, an `eDonkey` system relies on dedicated servers whose only purpose in to manage queries and bring peers into contact. To achieve this, a server has to process various kinds of commands:

- `login`/`logout` commands correspond to arrival/departure of peers. Once a peer is connected to the server (using a `login` command), it sends metadata to the server which describe the files it provides. The server stores these metadata into a local table for the processing of later queries.
- `filesearch` commands are entered interactively by the end-user. They generally contain one or a few words describing the wanted data, but they can be much more complex (including logical operations on words, size of the requested file, its type, etc). The server answers such commands with metadata (mainly hashcodes of the files) fitting the given description.
- `sourcesearch` commands allow a peer to know other peers providing a file (or parts of it) given its hashcode (obtained through a `filesearch` query in general). The server answers such a command with a list of peers providing the data, the list may be incomplete (it gives a bounded number of sources, not all the ones which have the data). Notice that each peer keeps its `sourcesearch` commands in a buffer and sends them every 5 minutes. Moreover, it automatically sends them again every 15 minutes to get new sources, if any.

There are various other details in the protocol (TCP vs UDP connexions, HighId vs LowId for peers, etc) but they do not play a significant role in the following, therefore we do not detail them. For more detailed information, we refer to [7].

The measurement and the trace.

In this paper, we focus on the exchanges between peers and on the properties of data. Therefore, we will mainly consider the `sourcesearch` commands. We record the answers of the server to such commands in the following form:

$$[T] \quad Q \quad H \quad "S_1, S_2, \ldots, S_n"$$

where T is a timestamp (in seconds from the beginning of the trace), Q is the peer which has emitted the query, H is the hashcode of the data queried by Q, and $(S_i)_{i=1\ldots n}$ is a list of peers which has registered the data. The server chooses these peers from its knowledge of which one provides which data, and only gives a bounded number of such peers ($n < N$ for a given N independent of the query).

In order to get these measurements, Lugdunum [19], the most popular and efficient `eDonkey` server software, has been modified to trace its answers to `sourcesearch` commands. Measurements have been made on a AMD Opteron 246 CPU with 3.2 Gb of main memory and a 64 bits Linux kernel. Such a

server can handle up to 250 000 clients simultaneously, but in practice the server "Razorback" (another Lugdunum server which can handle more than 600 000 clients, using two AMD Opteron 248 with 6 Gb of main memory) is very famous and other servers are only working at their maximal capacity when Razorback is rebooting.

We studied several traces, during up to more than two days. During such a period, the server typically processed 1.5 million `login` commands, nearly the same amount of `logout` commands, and around 210 million `sourcesearch` commands. The results presented hereafter are qualitatively the same for all these traces. We therefore chose to present them using a single typical trace (of 800 minutes).

Notice that the trace has been started simultaneously with the server reboot, therefore we can observe the startup phase of the server. Moreover, it is long enough to ensure that the server as reached a steady state, as will be shown in the following.

We argue that these traces are representative of the exchanges actually processed in a typical P2P system, mainly because of three points:

- the observed queries depend mainly on peers behaviors, not on the underlying protocol,
- the observed phenomena do not significantly vary from one trace to another,
- the huge size of the trace ensures that we capture most of the behaviors which indeed occur.

The query graph $\mathcal{Q}$.

There are several ways to conduct statistical studies on the data collected on queries as described above. In this paper, we propose to use tools from the recent field of complex network analysis. We will show that they make it possible to give strong insight on some important properties of the trace. However, this approach might be completed using tools from other fields, like signal processing for example.

A first way to encode the gathered data into a complex network is to define a labeled weighted bipartite graph $\mathcal{Q} = (P, D, E, w)$ as follows:

- P is the set of peers in the network, D is the set of data (hashcodes),
- $E \subseteq (P \times D) \cup (D \times P)$ is the set of directed edges where $(p, d) \in E$ if the peer p has made a query for the data d, and $(d, p) \in E$ if p has been cited by the server as a provider of d,
- w is a weight function over E where $w(x, y)$, for all $(x, y) \in E$, is the number of times x has requested y or the number of times y has been cited as a provider for x.

We will call this graph *the query graph*, since it mainly encodes which peer queried which data and who is cited in the answer to these queries, and denote it by $\mathcal{Q}$.

Notice that, despite this graph encodes much information on the exchanges captured by the trace, it does not contain *all* the information. However this graph is essential to put some properties into light, which would not appear without it. For example, one can immediately see on this graph which data are queried most often by looking at their number of incoming edges.

Statistical properties of graphs.

Complex network analysis is a scientific area in full development aiming at describing very large graphs met in practice and extracting some relevant information from them. Is is based on various statistical properties which can be measured on graphs, and on comparison of the ones met in practice with the ones of random graphs. It appeared recently that most real-world complex networks have some statistical properties in common which make them very different from random graphs. See [2, 6, 12] for surveys on these results.

The graph degree distribution is one of the main such properties. It is defined as $(P_k)_{k=0...K}$ where P_k is the proportion of nodes of degree k (*i.e.* with exactly k edges), and K is the maximal degree. If the

graph is directed, there are two degree distributions, namely the in-degree distribution and the out-degree distribution. Likewise, in a bipartite graph, there are two degree distributions, and in a directed bipartite graph like the query graph $\mathcal{Q}$ there are four. Moreover, in a weighted (directed) graph, the weighted (in- or out-) degree of a node is the sum of all the weight of its (incoming or outgoing) edges. The query graph $\mathcal{Q}$ induces four such weighted degree distributions.

In classical random graphs the degree distribution follows a Poisson law [5]. This implies in particular that the mean degree in such a graph is significant since all nodes will have almost this degree. On the contrary, in most real-world complex networks, the degree distribution follows a power law: $P_k \sim k^{-\alpha}$ for a given α. This implies in particular that, despite most nodes have a low degree, there is a significant number of nodes with very high degree. In such cases, the mean degree is not a relevant information. On the contrary, there are very high degree nodes which play a particular role in the graph. This has many important consequences, see for example [3, 13].

If several properties (like in- and out-degree, for example) are defined over the nodes of a given graph, one may study the *correlations* between them. This is generally done using a plot where each node induces a point at (x, y) if the value of the properties of the node (its in- and out-degree, for example) are x and y. If the two properties (here degrees) are uncorrelated, then the obtained plots display no particular structure. On the contrary, one may detect some important correlations with such plots. For example, if the points lie close to the diagonal (or any straight line) then the two properties have a linear dependence. But, of course, one may in general observe more complex correlations.

In this paper, we will in particular study the time-evolution of degrees in $\mathcal{Q}$. Notice that this cannot be done simply by plotting the time-evolution of the average degree, since, as we will see, the degree distributions in $\mathcal{Q}$ follow power laws. Therefore, the average degree is a very poor indicator. However, this power law structure gives an original way to study the evolution of degrees. The nodes may be separated into two very different classes: low degree nodes, which represent the vast majority of the whole, and the few very high degree nodes, which play a particular role in the system. One may then study these two classes with two orthogonal approaches: one may plot the time-evolution of the proportion of nodes of degree i for small values of i whereas one may plot the time-evolution of the degree of the few nodes with the highest final degree. This is what we will do in the following.

Many other statistical properties may be defined on graphs, like clustering, centrality, shortest path lengths, etc, and correlations between them. They give strong insight in the structure of the graph under concern and may be combined to give even more information. This is however beyond the scope of this paper, where we focus mainly on degree distributions which are the most simple graph properties, and we will see that they already help much in analysing the query graph $\mathcal{Q}$.

Context.

In the last few years, various measurements of P2P networks have been carried out. Some used active measurement [14, 15], others used passive measurement and flow analysis [1, 4, 9, 11, 16, 17]. We briefly present these previous works here, with a special emphasis on the ones based on passive measurements.

Adar and Huberman [1] studied the Gnutella 0.4 traffic over a 24 hours period of time from a given client. The main point of the analysis was to discriminate users between free-riders (no sharing at all), and active peers (lot of sharing). It turns out that 70% of the users are free-riders, while 1% of the clients answer to 50% of the queries.

In [4], Gnutella traffic is observed during 35 hours from a client point of view with the objective to study the amount and type of signaling traffic. In particular the authors observed the distribution of TTL for the search queries, in order to evaluate the distance from their client to others peers. Similarly, [11, 17] studied Gnutella traffic from various traces with different duration (or geographic location) in order to study queries caching strategies to decrease signaling traffic.

Data from various P2P systems (Fastrack, Gnutella, Directconnect) have also been collected directly from routers of some Internet Service Providers [9, 16]. These study are mainly aimed at finding strategies for ISP to reduce the P2P traffic. In particular [9] showed that P2P data contain enough redundancy to use efficient caching strategies.

More recently, [8, 10] used passive measurement based on KaZaA and KaZaA Lite (Fastrack). In [10] it is confirmed that KaZaA traffic is mainly composed of a few very popular files. Their results show that this tendency is even more pronounced than supposed before, which increase the advantages of caching strategies. Finally, [8] innovated using 3 KaZaA clients within NY Polytechnic network to get strong insight on KaZaA protocol (which is not open source). Their measurements allowed to estimate the number of privileged clients in KaZaA to 30 000, each client having from 40 to 60 connections to others privileged clients and from 100 to 200 connections to ordinary clients.

All these studies, and others, have therefore been based on the use of clients or routers to understand the traffic itself, the main conclusion being that, independently of the P2P system in concern, most of the traffic is concentrated on few very popular files. In this context, the use of caching strategies might therefore be very efficient.

Our approach is quite different since it is based on measurements from the server side. This makes it possible to confirm and improve some of the previous results, but our main aim is to deepen the study of peers behaviors. Indeed, one may consider that the queries we observe are weakly influenced by the underlying protocol (we will discuss this in the following where it may not be true): these queries mainly depend on the users interests, culture, etc. Our study is therefore directed towards the analysis of peer behaviors, which certainly play an important role in the efficiency of P2P systems and their design. Moreover, the use of tools from the recent field of complex network analysis provides some new and original insights on the topic.

2 The peers point of view

In this section, we study the query graph $\mathcal{Q}$ from the peers point of view. More precisely, we will focus on the degrees of the nodes in P, the set of peers. Recall that the peer p in P has an outgoing edge to the data d if and only if p has made a query for d, and that p has an incoming edge from d if and only if p has been pointed out as a provider of d. Therefore, we may look at the following values:

- *the out-degree* of a peer is the number of (distinct) data it has looked for,
- *the weighted out-degree* of a peer is the number of queries sent by the peer (including several queries for the same data),
- *the in-degree* of a peer is the number of data for which it has been pointed out as a provider,
- *the weighted in-degree* of a peer is the number of times it has been pointed out
- *the in- and out- weight* express the number of times a given peer is cited for a given file, or request a given file.

All these degrees play an important role in describing a given peer. For example, a peer with a high out-degree is a peer seeking many data, a peer with a high in-degree certainly has many data to provide, and a peer with high weighted in-degree is a peer which is solicited very often (it may shares many data or very popular ones), and therefore may be overloaded. We will use all these notions in the section to describe peer behaviors.

Let us first consider the unweighted in- and out-degree distributions of peers, and the weight distribution (Figure 1). As expected, they show that the node degrees are very heterogeneous, with a power law shape: whereas most peers have a small degree, some have a high one (several orders of magnitude more). This means that there is no *typical* behavior of peers. In other words, they cannot be properly modeled

nor simulated using a notion of *mean* peer, which could not capture the variety observed in practice. Let us emphasize on the fact that this has significant importance for the design of P2P systems, which should certainly take this heterogeneity into account.

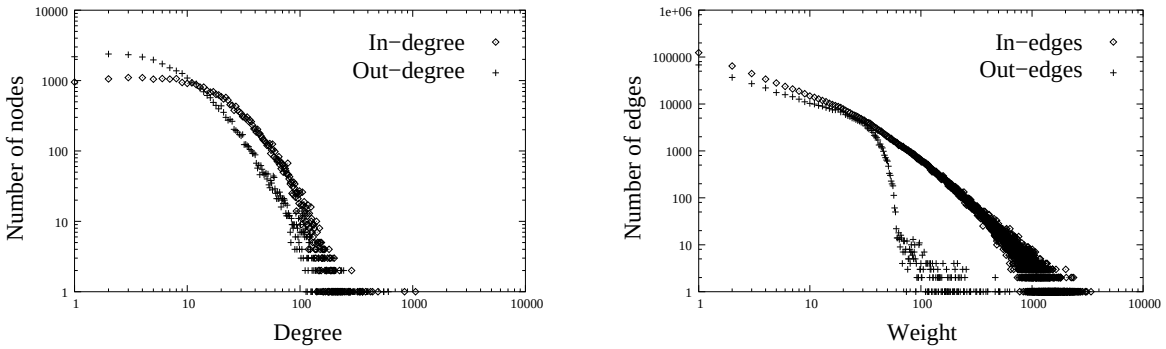

Figure 1: Left: the in- and out-degree distributions of peers. Right: the in- and out-degree weight distributions on the edges.

One may also notice that the out-weight distribution has a cutoff (Figure 1, right). This can be interpreted as a consequence of the fact that a peer should not request a given file more often than once every 15 minutes. Therefore, there is a maximal number of queries a peer may send, which corresponds to the cutoff. The peers which have a degree lower than the cutoff are the ones which send few queries or have joined the server later. The ones which reach the cutoff are the ones with classical client software and use it at its maximal rate. The ones which are above the cutoff certainly correspond to peers with modified client software, which allows them to query the server more frequently. Therefore, this plot makes it possible to evaluate the number of unfair peers which may endangered the system by overloading it, and even detect them.

A natural continuation of the analysis is to study the correlations between in- and out-degrees, as displayed in Figure 2 (left). These plots do not display strong correlations, but they show several things. First, it appears clearly that high in-degree nodes are not the ones with high out-degree, and conversely. This means an important thing: peers which provide many data do not in general send many queries, whereas the peers which make most queries provide only few data. This indicates that some peers are installed to serve mainly as data repositories from which other peers get data, whereas some peers do not behave fairly in the system since they get many data while providing very few. Finally, this tendency, despite it is not strongly pronounced, is general: if a peer has a high in-degree (out-degree), then it tends to have a low out-degree (in-degree, respectively).

One may observe different phenomena on the weighted in- and out-degree correlations (Figure 2, right). For example, the nodes with highest weighted out-degree tend to send very few queries. This seems to confirm our hypothesis that there are peers which mainly play the role of data providers.

Let us now turn to the time-evolution of degrees. The first natural ideas certainly are to plot the in- and out-degree distributions at several dates, as well as the time-evolution of mean in- and out-degrees, see Figure 3. The plots of degree distributions show that it is very stable, which may be surprising but is quite typical of large complex networks. The time-evolution of the average degree shows that on average the nodes have a higher in-degree than their out-degree, which means that on average they are more often contacted than they make requests. This is a consequence of the fact that, in most client software, a data retrieved from the P2P system is automatically made available on the corresponding peer for others.

However, as already pointed out, this mean behavior is not very meaningful. This is particularly clear here, since we have shown above that peers have very different natures. Therefore, this average over all

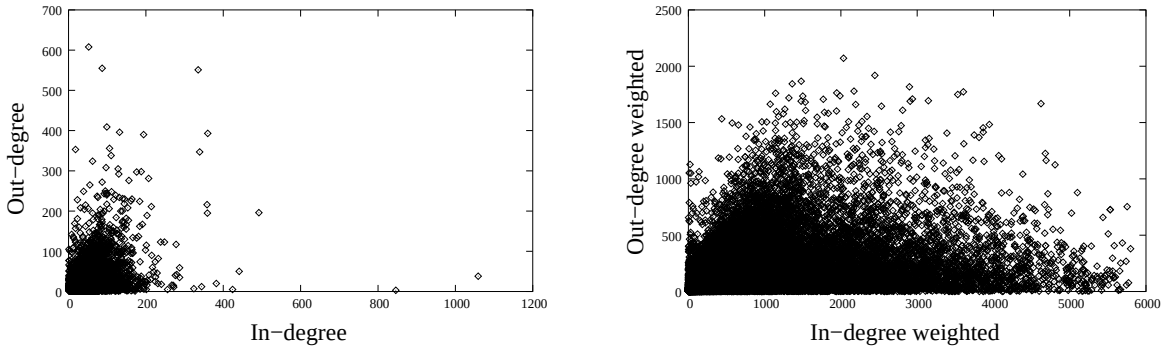

Figure 2: Left: correlation between in- and out-degrees. Right: correlation between weighted in- and out-degrees. The in- and out-degrees are on the x- and y-axis respectively.

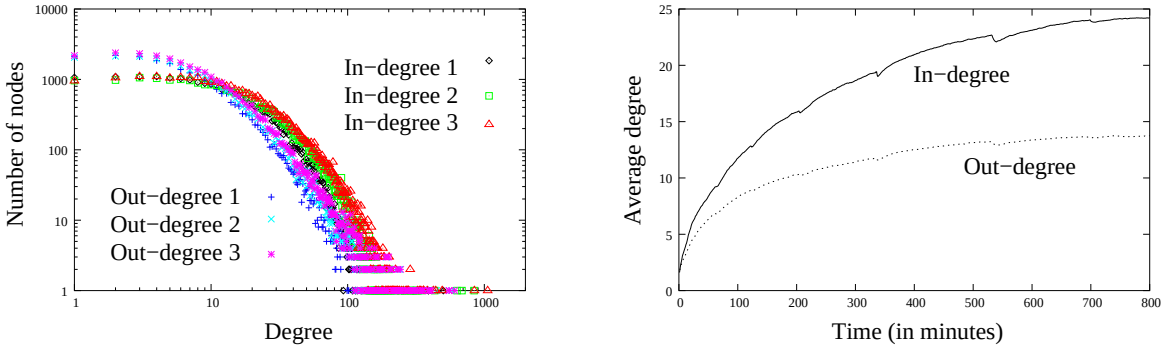

Figure 3: Left: the in- and out-degree distributions at various dates. Right: time-evolution of the average in- and out-degrees (the irregularities in the plots correspond to peaks in the arrival and departure of peers, due to reboot of other large servers in the system).

the peers has to be taken very carefully, and certainly does not mean that there is a notion of *average peer* which would have the average in-degree and the average out-degree: most peers do not have the average in-degree nor the average out-degree, and if their in- or out-degree is high then the other tends to be low. The evolution of the average in- and out-degrees must be viewed as a property of the whole system, not of its components.

To obtain more precise information, we now study separately low and high degree nodes. To achieve this, we plotted the proportion of nodes of (weighted or not, in- or out-) degree i for small values of i (typically $i \leq 10$). This proportion is very stable, as one may have guessed from Figure 3 (left), therefore we do not reproduce these plots here. On the contrary, we will focus on high degree nodes, for which it is possible to plot the time-evolution of their degree. This is what we do in Figure 4.

Let us first notice that these plots confirm that nodes with highest in-degree tend to have a low out-degree, and conversely. Moreover, these plots show that the high degree nodes are not nodes which have a low degree during a long time and then experience an abrupt grow up in their degree. On the contrary, they are among the highest degree nodes for a long time, and their degree grows quite regularly. Moreover, they all behave in a similar fashion, therefore one may consider that there is a *typical* behavior for high degree nodes, or, equivalently, for very active peers. This gives hints for their accurate modeling and simulation.

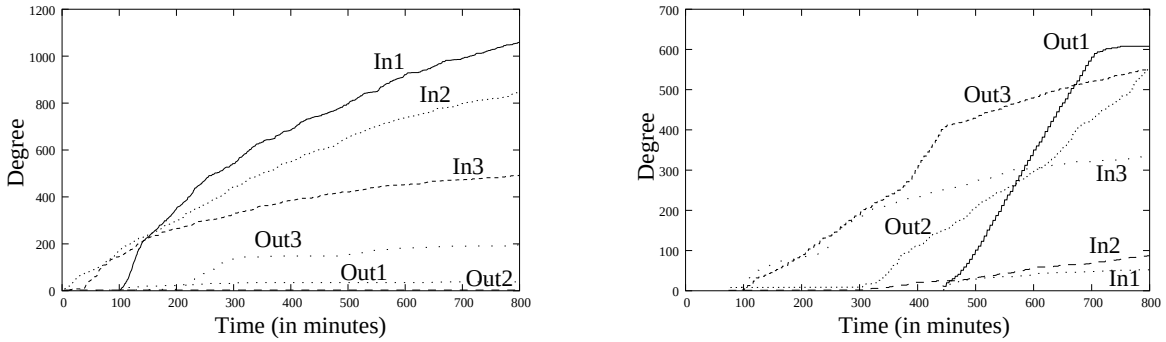

Figure 4: Time-evolution of in- and out-degrees for the three peers with maximal final in-degree (left) and out-degree (right).

If we now consider the time-evolution of the *weighted* in-degrees of the nodes with the maximal final in-degrees, we obtain Figure 5 (left). We observe several things on this plots. First, as before, the nodes with the highest weighted in-degrees have in general a low weighted out-degree. Moreover, the time-evolution of the weighted in-degree is very regular and homogeneous, which shows that it is possible to introduce a relevant model.

One may also consider that the weighted in-degree of a peer is a measure of its load. We can then use this plot to try to understand why the server fails in distributing fairly the load among peers (which is proved by the weighted in-degree distribution). The protocol ensures that the queries for given data will be distributed fairly among the peers providing them. Two causes may make this fail: the corresponding peers may provide rare data (therefore the server has no choice when these data are queried but to cite these peers), or they may provide many (different) data and are most likely to be cited.

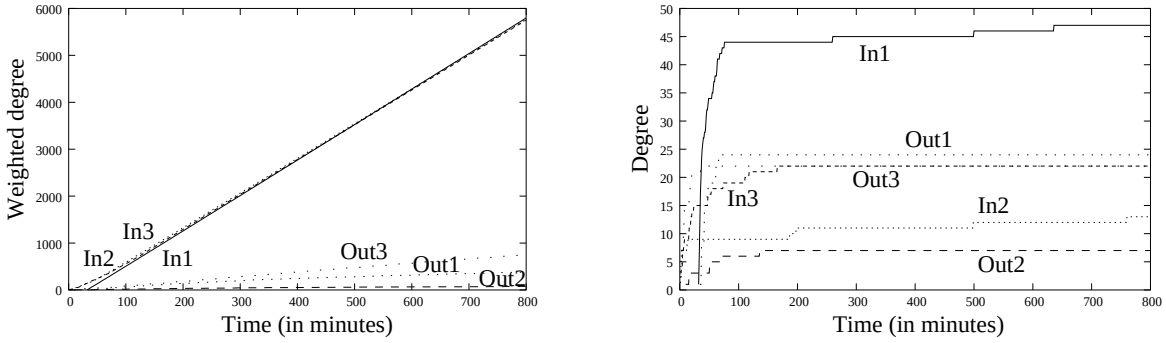

Figure 5: Left: time-evolution of weighted in-degrees for the three peers with maximal final weighted in-degree. Right: time-evolution of the (unweighted) in-degree of the *same* nodes.

In order to decide between these two hypothesis, let us observe, for the peers under concern, the time-evolution of their unweighted in-degree (Figure 5, right). This plot shows that these nodes reach very quickly their maximal in-degree. In other words, we know very quickly *all* the data they will provide; the growing in their in-degree simply means that they will be queried frequently for these data. Because of the load-balancing managed by the server, this makes us conclude that a large weighted in-degree is actually due to the fact that the corresponding peer provides rare but very popular data, which certainly correspond to newly introduced data.

We may now study the time-evolution of weighted out-degree of largest final weighted out-degree peers in a similar way, see Figure 6. The weighted out-degree of a peer is a measure of the load it induces on the server. It appears that the nodes with the highest weighted out-degree also have a high weighted in-degree, which is due to the fact that when a peer retrieves some data it also makes them available on its machine. This is a good point for the protocol, which induce that a greedy peer also has to serve the community. The stairs in the plots are due to technical specificities of the protocol which we do not detail here.

If we look at the time-evolution of the unweighted out-degrees for these nodes (Figure 6, right) we again see that either they converge quite quickly to a value which is not very large, either they converge very slowly. This means that some high weighted out-degree are due to the fact that peers send many queries for the same data, or that some peer are continuously looking for new files. As already discussed, the first point may be induced by unfair modifications of client software, and these plots show that, despite they induce a significant overload for the server, these modifications have little benefit, if any, for the unfair peers.

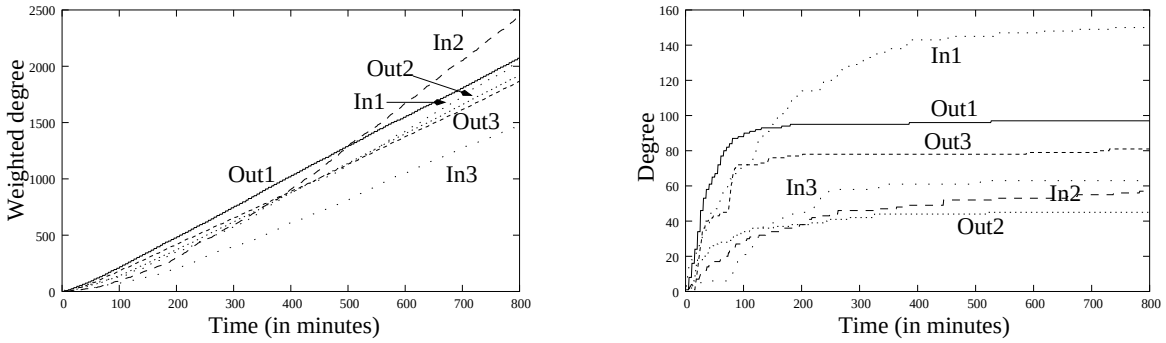

Figure 6: Left: time-evolution of weighted out-degrees for the three peers with maximal final weighted out-degree. Right: time-evolution of the (unweighted) out-degree of the *same* nodes.

3 The data point of view

In the previous section, we studied precisely the peers behaviors using their degrees in the query graph $\mathcal{Q}$. The same kind of study may be conduced with benefit from the *data* point of view, which constitute the other part of this bipartite graph. We present rapidly our main observations concerning this here.

Recall that the data d in D has an incoming edge from the peer p if and only if p has made a query for d, and that d has an outgoing edge to p if and only if p has been pointed out as a provider of d. Because of the lack of space, we will focus on the unweighted degrees here. Therefore, we may look at the following values:

- *the in-degree* of a data is the number of (distinct) peers which have looked for it,
- *the out-degree* of a data is the number of (distinct) peers which provide it,

We show in Figure 7 the in- and out-degree distributions for data at different dates (left) as well as the time-evolution of the average in- and out-degrees. As previously, the degrees display a very high heterogeneity (the plots fit surprisingly well power laws), therefore the average degrees should be considered as global properties of the system which are not relevant for its components. They show however that the average in- and out-degrees converge to a steady value, and that the average out-degree is larger than the average in-degree. This is due to the fact that when a peer is downloading some data, it generally provides them (after the download, but also during it, since the data are divided into blocs).

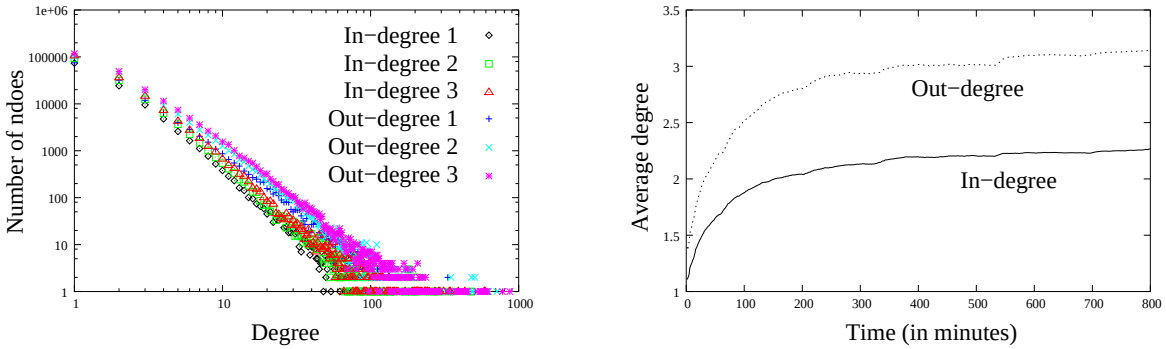

Figure 7: Left: the in- and out-degree distributions of data at different dates. Right: time-evolution of the average in- and out-degrees of data.

We can observe the effect of this strategy more precisely by plotting the correlations between in- and out-degrees, weighted or not, of data (Figure 8): it clearly appears that data which are queried often by peers are also provided by many peers. This is a very good point for the protocol, which avoids this way the overload of peers which provide data wanted by many other peers.

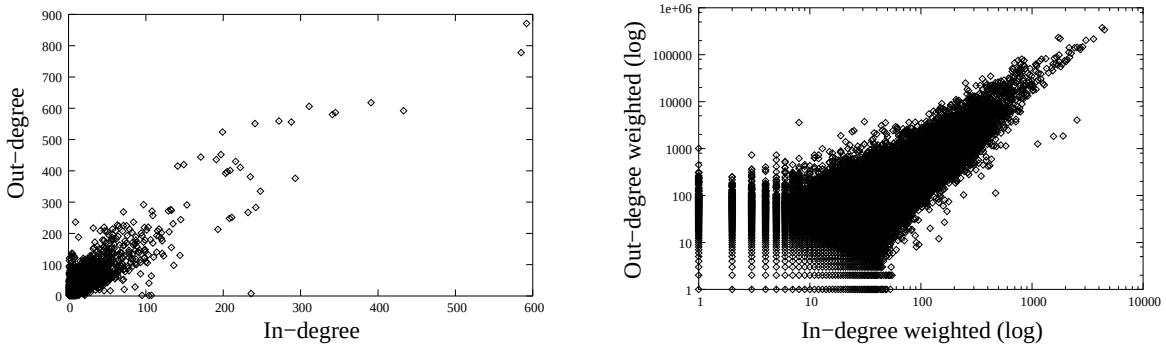

Figure 8: Left: correlations between in- and out-degrees of data. Right: correlations between weighted in- and out-degrees of data.

This plot also shows that, in almost all the cases, a data has a larger out-degree than its in-degree (most points are above the diagonal), which is another effect of this strategy. This means that the difference observed in the average in- and out-degrees (Figure 7, right) may be representative of what actually happens for most data.

We may now look more precisely at the data with highest final in- and out-degree, see Figure 9. Like in the case of peers, these plots show that the highest in-degree data clearly have a typical behavior. Therefore, in this case too, it is possible to give a general description of properties of very popular data, despite the global heterogeneity of data in general.

Conclusion and discussion

We present in this paper an in-deep study of peer behaviors using their degrees in the query graph, their correlations and their time-evolution. We also give some insight on the properties of the exchanged data using similar techniques. We use for this a representative trace of queries processed by a large `eDonkey` server during a significant period of time.

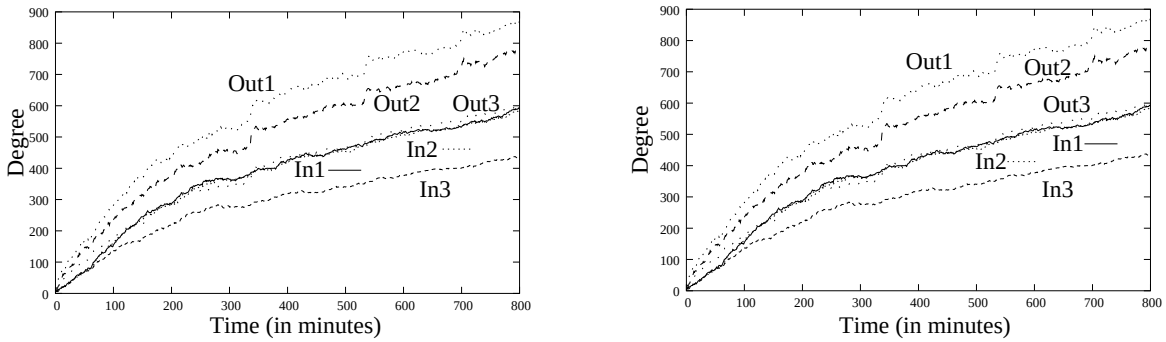

Figure 9: Time-evolution of in- and out-degrees for the three data with maximal final in-degree (left) and out-degree (right).

Our analysis gives evidence for several phenomena. Some of them are induced by the protocol properties, but most are mainly related to peer behaviors. Notice that some properties belong to both classes. For example, we pointed out a phenomenon due to the fact that unfair peers use modified client software. Evaluating the amount of such peers and their impact on the system may be crucial.

Besides the precise details of our results and the description we obtained of peers and data, two main points strongly appear in our study:

- Both peers and data are highly heterogeneous, which makes irrelevant any notion of *typical* peer having a mean behavior,
- On the contrary, peers may be separated into several classes (peers which mainly provide data, peers with rare data, peers which send many queries, etc) which correspond to well-defined behaviors.

The first point has already been noticed in previous studies, and both are also true for data. These results may certainly been deepen, but we already point out basic properties for an accurate modeling and simulation of a wide variety of peers. We believe that these facts are of high relevance for the design of efficient P2P systems, and should be taken into account in further research. They should lead to accurate description and modeling of classes of behaviors in P2P systems.

Let us also insist on the fact that we derived these results using in-deep analysis of degrees, in particular their time-evolution. Up to our knowledge, this is the first use of the approach of considering separately high and low degree nodes. This is very general and may be used with benefit in all the cases where node degrees are heterogeneous (typically with power law distribution). Indeed, whereas the time-evolution of degrees is a significant property of high degree nodes, it makes no sense for low degree ones. On the contrary, the arrival rate of low degree nodes is relevant, whereas it has no meaning for high degree ones. This separate study of low and high degree nodes is much more relevant than the time-evolution of the average degree. In the present case, it gave evidence of the fact that high degree nodes have simple typical behaviors. In other cases, it may show that they have more complex behavior, or are themselves of different kinds.

Finally, this work opens several kinds of perspective which would certainly lead to much more insight on P2P systems and peer behaviors in such environments.

First, one may look at more precise statistical properties of the query graph degrees. For example, we did not discuss the possible correlations between the degrees of peers and the ones of the data they look for and/or provide. Going further, one may wonder if the peers having the same profile provide the same data, or data with similar profiles. For example, if a peer provides rare data, then do the peers which download these data also provide (other) rare data?

Of course, one may also use other statistical properties (not based on degrees) to obtain more insight. Complex network analysis nowadays provides a wide variety of powerful such notions. Let us cite the clustering (local density), the centrality (importance of nodes in the network), the reciprocity (uni- or bi-directional exchanges), etc. More subtle properties include clique size distributions, communities detection, etc. Another direction would be to study the texts of the queries (entered by the end-users) and relate them to the peers of the data properties.

These ideas are currently under development using techniques similar to the ones presented here, and we hope that they will give more insight on the various aspects of P2P systems.

Acknowledgments. We warmly thank Timur Friedman (LIP6) and Jerome Laban (EPITECH) who made it possible to make measurements from their machines. We also thank Mathieu Bernier (eFarm) and Fabrice Le-Fessant (LIX-INRIA) for useful discussions on the `eDonkey` protocol. Finally this paper would not have existed without the devotion of Eric Dumazet who kindly modified his *Lugdunum* server so as to get the traces.

References

[1] E. Adar and B.A. Huberman. Free riding on gnutella. *First Monday*, September 2000.

[2] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics 74, 47*, 2002.

[3] R. Albert, H. Jeong, and A.-L. Barabási. Error and attack tolerance in complex networks. *Nature*, 406:378–382, 2000.

[4] K. Anderson. Analysis of the traffic on the gnutella network. 2001.

[5] B. Bollobás. *Random Graphs.* Academic Press, 1985.

[6] S.N. Dorogovtsev and J.F.F. Mendes. Evolution of networks. *Adv. Phys. 51, 1079-1187*, 2002.

[7] A.-M. Kermarrec F. Le Fessant, S. Handurukande and L. Massouli. Clustering in peer-to-peer file sharing workloads, 2004.

[8] K. W. Ross J. Liang, R. Kumar. Understanding kazaa. 2004.

[9] N. Leibowitz, A. Bergman, R. Ben-Shaul, and A. Shavit. Are file swapping cacheable? characterizing p2p traffic. In *7th International Worshop on Web Content Caching and Distribution (WCW'03)*, 2002.

[10] N. Leibowitz, M. Ripeanu, and A. Wierzbicki. Deconstructing the kazaa network. In *3rd IEEE Workshop on Internet Applications (WIAPP'03)*, 2003.

[11] E.P. Markatos. Tracing a large-scale peer to peer system : an hour in the life of gnutella. Technical Report 298, 2001.

[12] M.E.J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

[13] R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Phys. Rev. Lett.*, 86:3200–3203, 2001.

[14] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design, 2002.

[15] S. Saroiu, P. Krishna Gummadi, and S.D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)*, San Jose, CA, USA, January 2002.

[16] S. Sen and J. Wang. Analysing peer to peer traffic accross large networks. In *Internet Measurement Workshop (IMW 2002), Marseille, France*, 2002.

[17] K. Sripanidkulchai. The popularity of gnutella queries and its implications on scaling. 2001.

[18] Andrew Tanenbaum and Marteen Van Steen. *Distributed Systems: Principles and Paradigms.* 2002.

[19] Lugdunum url:. http://lugdunum2k.free.fr/kiten.html.