Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 6, Number 1, pp. 45–56. http://www.scpe.org

IMPACT OF REALISTIC WORKLOAD IN PEER-TO-PEER SYSTEMS A CASE STUDY: FREENET

DA COSTA GEORGES* AND OLIVIER RICHARD*

Abstract. This article addresses the problem of the study of the performance evaluation and behavior of the large scale Peer-to-Peer file sharing systems. In particular the impact of realistic workload is considered by evaluating the Freenet system. This evaluation is achieved by a simulation approach. A set of inputs is determined as well as their distribution law in order to generate a more realistic workload. One of them is an original characterization of user's requests. An other contribution is to show the impact of these more realistic inputs on the overall system performances. Notably new abrupt behaviors in the learning process are described.

Key words. Freenet, Stockage, Pair á Pair, Simulation, Zipf Law

1. Introduction. Large peer-to-peer file-sharing systems had became widespread thanks to the use of projects like Napster [27] and Gnutella [16]. Their simplicity for exchanging music file explains their success as well as the copyright law transgression.

The first generation systems were based on rudimentary architecture and protocols: a central index for Napster and an inundation search protocol for Gnutella. Those systems developed fast thanks to their match of the user's demands. On the other hand, the scalability challenge they bring has raised a great interest. Numerous researches address this issue. Amongst the projects, one can enumerate academic ones like Oceanstore [4], CAN [26], Past [18], Chord [29] as well as some free or commercial softwares [24, 12].

A general problem encountered is the performance evaluation of these systems and the analyze of their statistical properties. Relatively few studies are made in this area [23, 13]. The difficulty to experiment those systems in real conditions is due to the great number of computers involved. In this context it is very difficult to evaluate a prototype. Usually studies approach the evaluation problem with restricted simulations, notably little realistic workload [8, 32].

Our main motivation is to evaluate the impact of a more realistic workload on Peer-to-Peer system's performances and behavior. As it's nearly impossible to use test such systems in a real environment (from some thousand to a few millions units) and to make an evaluation fully controlled we needed to choose an alternative solution. In this first step we have chosen to focus on the Freenet system [8] and to adopt a simulation approach.

The Freenet system is a distributed file sharing system focusing on the anonymity of it's users which aims to provide an alternative uncensored Web.

The paper is organized as follows. The next section is a fast overview of the Freenet system. Section 3 describes the simulation methodology we used. Section 4 consists in a characterization of the inputs for the simulation. The section 5 presents results obtained, and the Section 6 concludes.

2. Freenet System. The Freenet Project [7] has been created in the end of the years 90 in the academic environment and is beside the first of such systems which has been the topic of a study [8].

It's aim is to provide a service of data sharing in a cooperative way while keeping a total anonymity for the author and the reader. In the same way it is aimed to prevent any kind of censorship and to become a *free web*. Users of this system are even protected from legal attacks because the data they keep on the space they share is encrypted to prevent them from being able to read them.

Freenet has a Peer-to-Peer architecture. The users share some resources to store the files and manage the routing messages. Files are referenced by keys which only depend of their content. They are spread thanks to a cache system that react dynamically according to the load.

2.1. Routing mechanism. Each node has a dynamic table which keeps the informations it possesses on the other nodes, the key they own for example. It's by providing the key associated with an object that the system can find this object. To do so, the key is forwarded from neighbor to neighbor. Each node locally compares the key with the key it knows it's neighbor own. Then it forwards the request to the one which possesses the nearest key. The requests have a limited time to live (like the *Time To Live* of IP), the HTL for *Hop To Live* which is decreased each time the requests go through a new node. The number of node a request has go through is called the request path length.

*Laboratoire ID-IMAG (UMR 5132), Projet APACHE (CNRS/INPG/INRIA/UJF) Georges.Da-Costa@imag.fr, Olivier.Richard@imag.fr)

When an object is found, it follows backward the path the request followed to find it. Each time it passes on a node, the node adds a reference to the node which had the object and mirrors the object.

This routing algorithm has some side effects. It makes the routing becoming better and better for two reasons:

- Nodes should eventually specialize in an interval of key. If a node is associated with a key in a routing table, it will tend to treat similar key.
- Nodes should specialize in stocking objects with similar keys. As forwarding a key leads to take in the cache the object, routing similar key leads to own objects with similar keys.

Those two effects should lead to improve the quality of the system by a learning process. It behaves like a cache system. The most popular objects will be more often copied and thus will spread quickly in the network and be easily reachable.

The use of caches doesn't prevent from the disappearance of files in the network. As the size of caches are finite, when a node needs to stock a new object which would leads to exceed the size granted to the cache, the system must cleans it's cache. The policy used in Freenet (we are not based on the latest version) is to destroy the oldest used object (LRU policy). It's the same algorithm which is used for the dynamical routing table. So it's not really a cache system because there is no original version of each file. Yet all the files will remain in the network as long as they will be requested.

3. Methodology of simulation. The simulation problem is difficult in the context of Peer-to-Peer systems on account of the great number of nodes in those networks and the complexity of their behavior. Moreover we aim to study the application protocol as well as it's physical impact on the network. The main constrains encountered in this domain are the time and memory. The usual approach by using the event driven simulator like NS2 [14] is not adapted to go beyond several thousand nodes in account of the cost of a packet-level simulation. To resolve those issues we did several hypotheses in order to achieve an acceptable simulation method:

- There is no temporal consideration during the simulation: each event is treated sequentially without dating.
- The application protocol traces are sufficient to obtain inputs for a simulation of the physical network.

The first hypothesis simplifies the detail level.

The second lead to decreases the complexity of the simulator by splitting it into parts: An application-level simulator and a physical-level one.

All the facets of the system behavior can't be deeply studied. Notably all the results achieved by this approach are to be considered in a statistical point of view. By example the congestion phenomena in the physical network can't be explored without temporal considerations. But we can estimate the overall workload on the physical network.

3.1. The simulator description. The resulting simulator has three parts. A *request generator*, a *logical simulator* and a *physical simulator*. For a simulation, the first part describes the inputs of the system like the file size distribution which will be explained in section 4. The two other parts simulate the core system.

The logical simulator takes care of the application-level part. It has been obtained by expanding the simulator used by the Freenet authors [8]. Some of the modifications were instrumentation aimed to extract information needed for the physical simulator part.

The physical simulator consists in routing the application level packets into the physical network. The routing algorithm used is the classical A-Star shortest path algorithm [17]. It's complexity is exponential in time according to the size of the network.

4. Realistic inputs generation. The goal of this study is to take into account the main characteristics of real inputs and their implication on the performances and the behavior of the system. In this section we select some relevant inputs and analyze their distribution law.

As the Freenet system is to provide an anonymous alternative to the Web, the chosen inputs are those relative to the HTTP traffic. The most important components are the way the requests are done and the files they aim. If those Peer-to-Peer system spread in a wide way, we think the users will use them like they actually use the web. Moreover the underlying physical structure of the Internet is needed to evaluate their performances.

To determine those input parameters we used some studies on the protocols TCP [25], HTTP [2, 28], and on the web itself [3, 1, 11, 10, 6]. Thus we used web logs [31] and specific traces extracted from the Gnutella Peer-to-Peer system.

The following deals with the requests and files characteristics, and the network structure.

TABLE 4.1
Distribution law of requests parameters. The distribution law of the file requested *is a representation of the popularity of the files, the* user's activity *represents the distribution of the number of requests send by the users, and the* temporal distance *is the time between two requests are done (this variable is not used in this study).*

	File requested	User's activity	Temporal distance
Distribution law	Extended Zipf law $P(t) = \frac{C}{t^\alpha}$, $\alpha = 0.8$	Exponential $P(t) = \frac{1}{\beta} exp(-\beta t)$	Pareto $P(t) = \alpha k^\alpha t^{-(\alpha+1)}$

4.1. Request characterization. To define a request, three pieces of information are needed: who made it, which object is requested, and when the request is made. The table 4.1 presents the retained distribution laws.

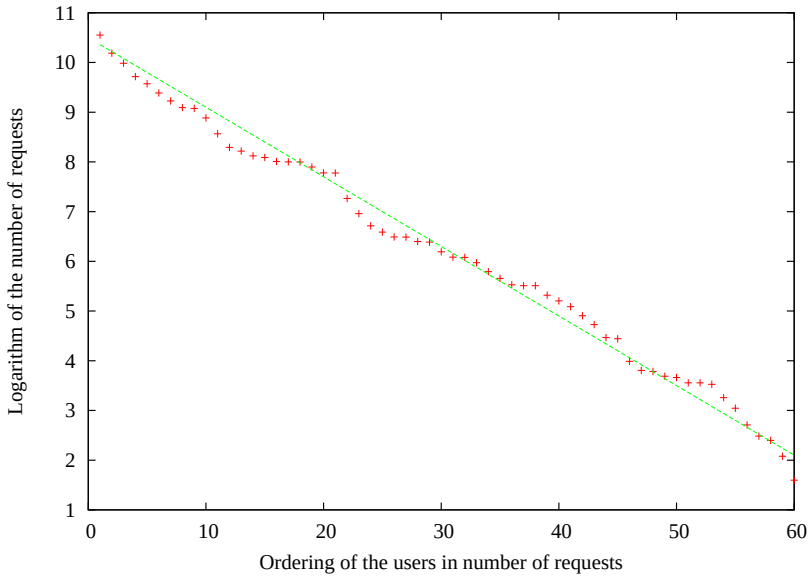

FIG. 4.1. *Distribution law of the activity of the users. The users are sorted in relation to their activity. The logarithm of the number of the request they send and a line are shown*

Requested object. According to studies of traces [3], the distribution law of the objects aimed by the requests only depends on the popularity of the files. If you suppose you can sort the files by their popularity, the request distribution follow the Zipf law [11] $P(t) = C/t$ where C is a normalization constant (used to have a total probability of one), and t is the index of popularity of the object. This law is often expressed by recalling the *90/10* law: 10% of the most requested files generate 90% of the requests. According to [6] there are no interrelationship between objects size and their popularity.

Some finer studies [6] of the web logs generated by proxies and routers show the requests repartition follows a slightly different law: $P(t) = C/t^\alpha$ where $\alpha = 0.8$. This law is called extended Zipf law.

The user who sent the request. It's quite hard to find studies based on the repartition of the user who send the requests. Most of the studies on the web load only take into account the impact of the requests on the server without taking care of who send them. There is no need to know who send a request to study the behavior of a server under stress. Some proxies logs [31] have been analyzed in order to have those information.

The distribution of the users in number of requests followed a decreasing exponential law (figure 4.1): $P(t) = \frac{e^{-\beta t}}{\beta}$ give the probability that the t^{th} user (in number of request sent) send a request at a given time with $\beta = 0.14$. This law has been obtained by analyzing Boeing's traces [5] Ircache [21] and Clarknet [9]. Those constants were obtained by making a linear regression in the log domain. The table 4.2 summarize the

characteristics of those traces, the obtained constants characterizing the distribution law, and the standard deviation.

TABLE 4.2
Summarize of the traces characteristics and the constants of the exponential law for users activities. Those traces have been made during Dates. *Each log consists in* Events number *requests. The β represents the value of the constant in the table 4.1. The* standard deviation *represent how accurate the values are.*

	Dates	Events number	β	Standard deviation
Boeing	Mar 1-5 1999	113926063	0.13	0.27
Ircache	Jan 24-30 2002	7084366	0.14	0.12
Clarknet	Aug-Sep 1995	3328632	0.18	0.4

Time between requests. Our simulator doesn't manage the time, so we didn't look at this variable. A complete study is done in [3]. In this study they found that the time between two requests follows a Pareto law: The probability that there is t seconds between two requests is $P(t) = \alpha k^\alpha t^{-(\alpha+1)}$ with $k = 1$ and $\alpha = 1.5$.

4.2. Characterization of the files: size and insertion. In this section we only take care of the object itself without taking into account it's treatment by the system. There are two free variables: It's size, and the way it has been inserted in the system (table 4.3).

TABLE 4.3
Distribution law of the parameters used to simulate the objects. Their Size. *And their* Initial position *which represents the number of files each node shares.*

	Size	Initial position
Distribution law	Lognormal $\frac{1}{t\sigma\sqrt{2\pi}}e^{-(\ln t-\mu)^2/2\sigma^2}$	Zipf law $P(t) = \frac{C}{t}$

Size of the shared files. Several studies [3, 2] show that the repartition of the files size transfered over the Internet follows a lognormal distribution. The probability that a file size is t is equal to $\frac{1}{t\sigma\sqrt{2\pi}}e^{-(\ln t-\mu)^2/2\sigma^2}$. σ and μ are experimental constants which depend of the file type. For the files over the Internet (mainly HTML pages, but some large files too), [3] gives $\mu = 9.3$ and $\sigma = 1.3$. All size are in kilo-bytes. During the measurement, the most often encountered file size was around $9ko$. Most of the files were of comparable size, and mainly of large size.

To extend this result, a Gnutella client has been modified. It gave data on the file size exchanged on the Gnutella network. In this network most of those files are multimedia ones. It showed that the file distribution of the files followed a lognormal law too. The only difference between this and the previous one concerns the values of the constants: μ and σ which become respectively 3580 and 4.9. Multimedia file are mostly around $3.6Mo$. Their size distribution scatters less that those generally found in the Internet.

Initial position of the files in the system. In our knowledge there are no studies of the initial file position in the Internet. But contrarily to the Internet, the way the files are inserted in the system may be important. In several Peer-to-Peer systems, inserting a file in the system generate some information that spread in the network.

Thus we need to know where the files where inserted in the system. To answer this question, we used web logs [31] on which Internet servers the files were acceded. It give us on which server the files were pushed.

Figure 4.2 has been produced from the informations from a proxy from Boeing. It shows the inverse of the file distribution according to the number of computers ordered by the number of files they possess. The plot of the files position is then a line, characteristic of a Zipf law.

The value of the coefficient has been calculated by doing a linear regression on the data. We don't have used all the data in order to calculate it: The trays visible on the figure 4.2 are provoked by the passage to the inverse of an integer function. to calculate this law of probability we have processed only the first value of each tray.

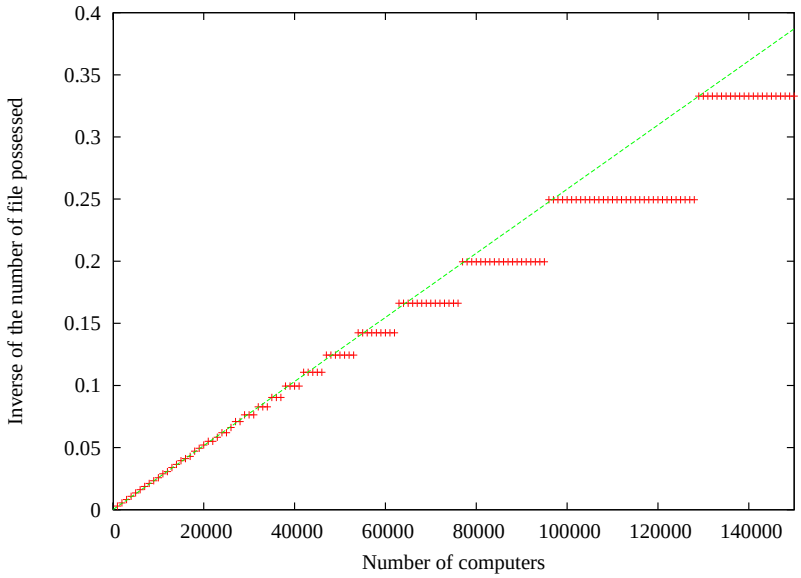

FIG. 4.2. *Inverse of the number of files possessed by the nodes. The nodes are sorted from the node that possess the most files to the node that possess the least.*

4.3. Model of physical network: Internet structure. In the context of this study, the underlying network is to be an Internet one. The modeling of the organization of the Internet network is something complex [20]. Moreover a realistic simulation should use more physical nodes than the logical ones. Everybody won't use those system. All the routers of the Internet will not participate to those systems, but they participate to the Internet structure.

Some studies have tried to characterize the Internet topology. Until now, no one found the exact nature of this structure [20]. Some characteristics have been found until now. It seems the Internet has a self-similar [10] structure. To be sharper, the Internet structure may follow some power law [22] (two functions f and g of t are link by a power law if it exists a constant α with $\forall t\ f(t) = g(t)^{\alpha}$).

We achieved a generator of network of this type by implementing the algorithm studied in [15]. It is also possible to use a more generic network generator: Brite [19]. It generate more sharp network (hierarchical one for example).

5. Results. In this section we present two kind of results. First the impact of realistic workload on the behavior of a Freenet system. It represents a deepening of the study [8]. Second an overview of the workload generated on the physical network.

5.1. Summarize of previous work. Results presented in [8] mainly show the quality of the routing algorithm and the fault tolerance of the system. They conclude that when requests are uniform, and when only the application level is taken into account, then:

- The system converges quickly (a few thousands of requests) to a stable state. At this time, more than half of the objects are under 6 hops from all the nodes.
- The system is able to grow until several hundreds thousands computers while providing a good quality of service for more than half of the objects.
- The system is fault-tolerant.
- The system evolves toward an Small World [30] architecture. In this architecture, the nodes are highly clustered yet the path length between them is small.

5.2. General experimental setup. To facilitate the comparison against [8] we use the same initial relationship network at application-level. This initial topology consists in a ring where each node is in relation with it's 4 nearest neighbors. This is a static network where all the nodes are always available.

5.3. Impact of realistic workload at the application-level. The figure 5.1 shows a direct comparison between the results of [8] which is based on the use of uniform distribution and the more realistic distributions

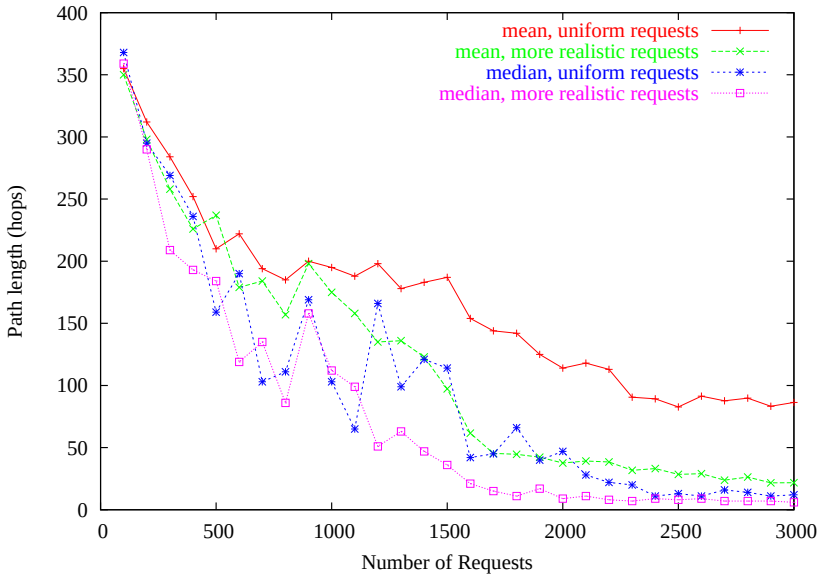

FIG. 5.1. *Evolution of the mean (or median) node number that a requests pass through to find a sought object (1000 computers) during the simulation. This evolution is evaluated with the uniform inputs, as well as with the more realistic set of inputs*

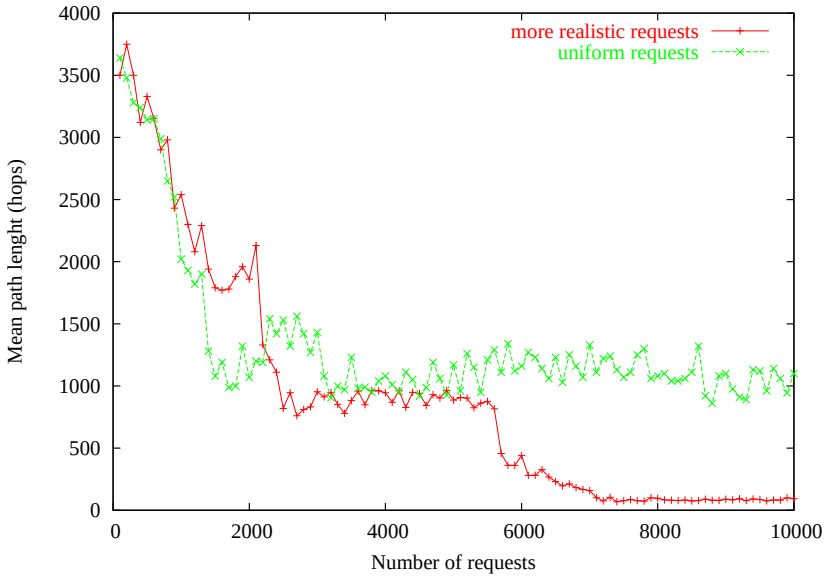

FIG. 5.2. *Evolution of the mean node number of nodes that a requests pass through to find a sought object (10000 computers) during the simulation.*

previously selected.

Those tests were done with a network of 1000 nodes, a reference cache size at 200 entries, and a file cache with 50 entries. The HTL is set at 20.

The plots on the figure 5.1 depicts the mean or median number of hops needed to access a file in function of the number of requests. Intuitively they show the evolution of the learning process. At the beginning of the learning process the system is unable to reach nearly all the files, and the more requests are treated, the more reachable the files are. There is a self organisation of the system that leads to a decrease of the distance to the objects.

At the end of the learning phase (around the 3000^{th} request) the realistic distribution leads to a significant

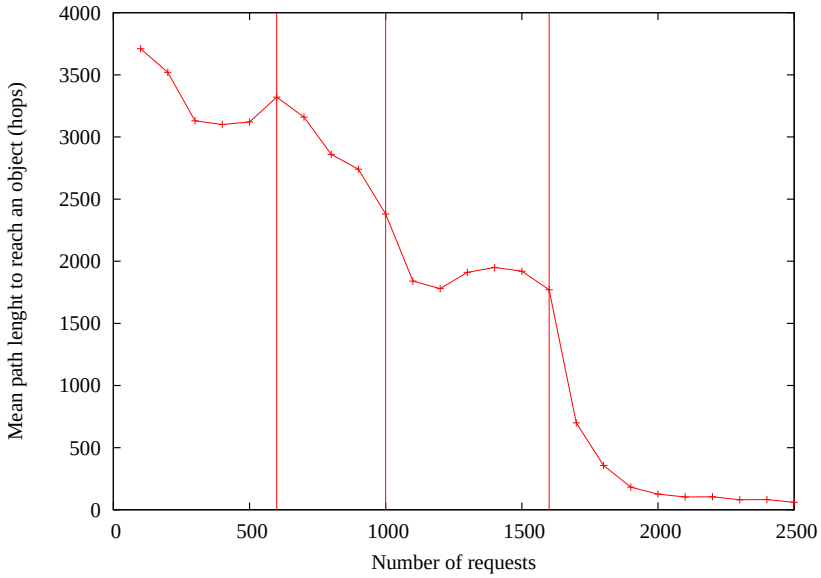

FIG. 5.3. *Evolution of the mean (or median) node number that a requests pass through to find a sought object (10000 computers) during the simulation with realistic inputs.*

improvement of the mean value of the request path length (hops) against the uniform model. For the median, there is still a slightly advantage for the realistic model (4 hops against 6).

The popularity property explains this improvement. As the most popular files are very often requested, they are wildly spread and thus the number of their access minimizes the weight of requests made on unpopular and hardly reachable files.

Contrarily to homogeneous requests that lead to a scattering of all data since the beginning. No data are penalized in this case but cost to reach data decreases little by little for all the data.

5.4. Abrupt behavior in the learning process. From this point on, we mainly focus on a larger system. The network is a 10000 nodes one. The internal value are set at 80 for the HTL, average reference cache size at 500 entries, file cache size at 2Go. The distribution used are the one of a HTTP context.

The figure 5.2 shows the mean request path length (hops) in function of the number of requests for the two approaches: the uniform and the more realistic point of view.

As on the previous experiment there is a great gap between the values at the end of the learning phase of the two approaches. But there are abrupt behaviors in the plot of the realistic requests. Those phenomena correspond to a significant decrease of the request path length. They occur after around the 2000^{th} and the 5500^{th} requests.

The behavior of the most connected nodes might give an explication of this abrupt behavior. Due to the small-world architecture of the application-level network, those nodes own most of the knowledge of the network, what corresponds to the most populated routing table.

The figures 5.3 and 5.4 underline the significant contribution of the most connected node in the learning process. The first figure is a plot of the evolution of the request path length. There are still 10000 nodes but the cache size is increased to 2500. The immediate effect of this increase is to speed up the learning phenomena which correspond of an horizontal compression of the plot.

The figure 5.4 depicts the evolution of the size of the clusters of distinctly reachable nodes by the three most connected nodes. To generate those plots, at each requests, the most connected node is selected. The number of node it can reach is plotted. Then those nodes are taken out of the network. This operation is repeated for the two next more connected nodes. In this figure the vertical lines localize the fusion of cluster of distinctly reachable nodes. Those verticals correspond in the figure 5.3 to the abrupt improvement of the request path length.

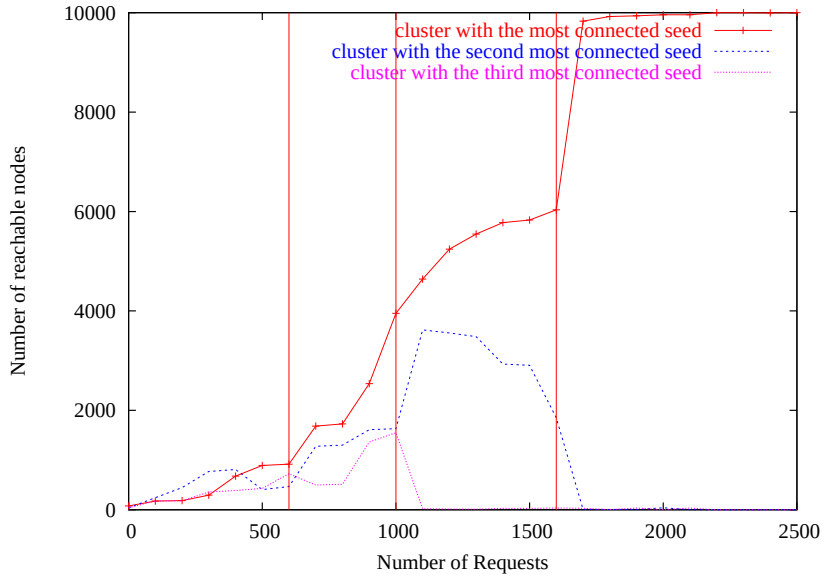

FIG. 5.4. *Clustering evolution. Evolution of the parts of the system which are distinctly reachable by the most connected nodes.*

5.5. System behavior after the learning process. The behavior of a Freenet system can be split in two parts. The part that has been approached until now: the learning process, and the stable behavior. Indeed there is a phase where the mean distance between users and files are quite constant. This is true when the file popularity is constant. Thus to verify the real quality of the learning process, it is necessary to test the system with others data than the one used for the learning phase.

To make this test several data set have been used. The firsts test are done by changing only the popularity of present files. In this category there are the full randomize test which just randomize the popularity of the file, and the reverse test which reverse the popularity of files (the most popular file become the least, and so on).

The second test is done by creating a completely new set of data.

Each test has been with 10 different runs, 100 times for each run (except for reverse which has been done only 1 time for each run).

TABLE 5.1
Impact of the popularity changes on the quality of the system

	Mean value	Standard deviation	Median value	standard deviation
Stable phase	10		4	
Reverse	10	0.008	4	0.005
Randomize	10	0.005	4	0.003
New set	10	0.013	4	0.008

The table 5.1 shows the impact of the popularity changes on the mean and median path length between the users and the data. It shows that when the learning phase is done, the network is able to handle any repartition of popularity for the files.

This property comes from a property of SHA-1 which is used to generate the keys. When it is used on a file to generate a key, the key is anywhere in the key-space. So the first keys which are the most used are randomly dispersed. Thus as the system learn by routing the keys, it can eventually handle any key with the same efficiency.

5.6. Fault tolerance of the system. The Freenet system relies on the links between nodes to find the files. This knowledge is contained in the nodes. A qualities of such system is not to be broken if some nodes

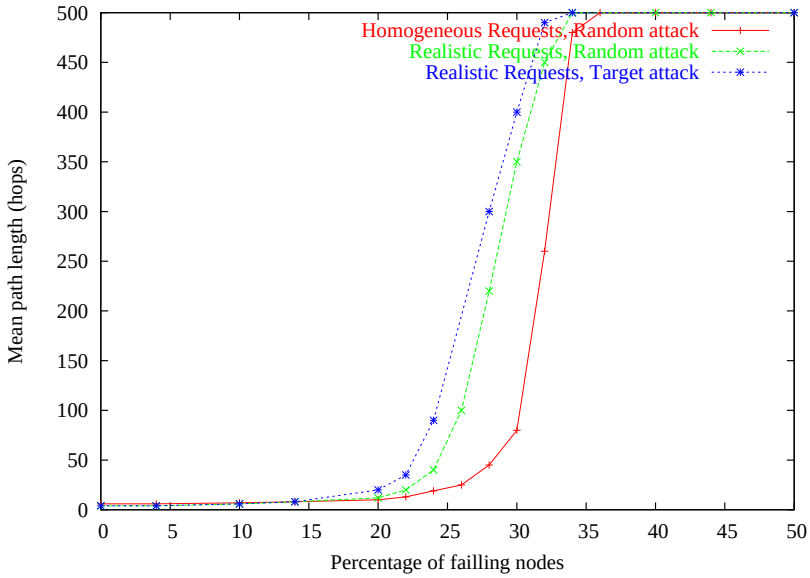

FIG. 5.5. *Comparison of the capacity of Freenet system to work when some node definitively disappear.*

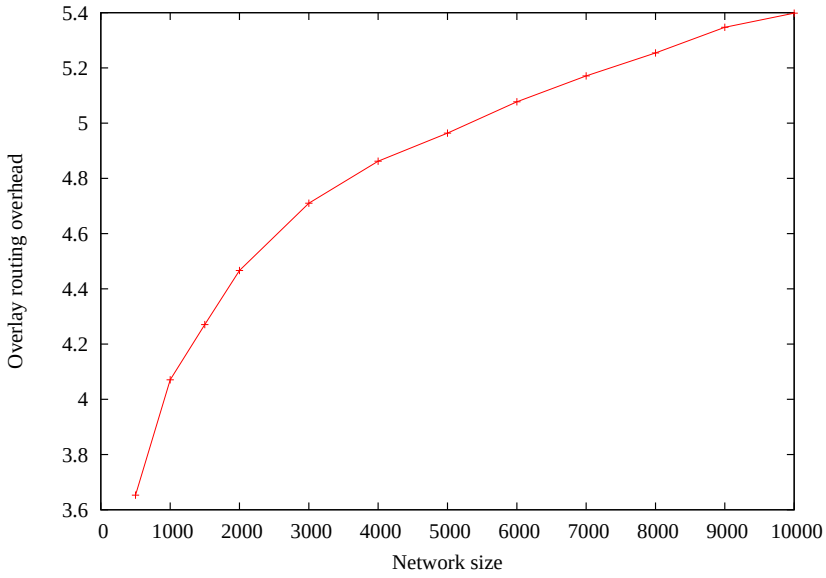

FIG. 5.6. *Evolution of the communication overhead in function of the size of the network.*

are disconnected. The case of simultaneous node failure is not just a theoretical possibility. It occurs sometime when a LAN is disconnected from the Internet, or when there is a major network partition.

There are two case: random failure and target failure. The random failure consist in choosing randomly some nodes and removing them. The target attack consist in choosing the most connected nodes (those which owns most of the knowledge of the network).

The figure 5.5 shows how the network respond for this two cases and with a network that has been created with homogeneous requests.

The system is less sensitive when it has learned with homogeneous requests. It is less sensitive too when the node are not chosen randomly. It proves that the most connected nodes are more important than others. Thus the realistic input concentrate the importance in less nodes than homogeneous input.

5.7. Workload generated on the physical network. The figure 5.6 shows the relation between the size of the network and the overlay routing overhead. It compares the length of the path followed by the files at the application level with the path they really go through at a physical level. It shows the evolution of this overhead for a network from 1000 nodes to a 1000 nodes one.

This plot show that the most node a network has, the most significant will be the overhead of the physical routing compared to the application-level view. This overhead is quite important in the Freenet case compared to some other systems (Tapestry has an overhead of less than 2 [32]).

6. Future outlook. In order to generate the requests we had to create a model of the behavior of the system's users. The behavior of users sharing multimedia files is not the same as those who share literature. There are several different populations that tend to comunicate far more in their community than with the outer world.

7. Conclusion and discussion. This paper addresses the impact of realistic workload in the Freenet Peer-to-Peer system.

Our first main contribution is to have determined a set of inputs and their distribution law in order to generate more realistic workload. This is an advance compared to the hypothesis made in previous works. Among other, we propose an original characterization of the requests generated by the users.

The second contribution is to have shown the impact of this new more realistic inputs on the overall system performance. In particular we note a great decrease of the mean request path length with more realistic inputs. An other noteworthy point is the presence of an abrupt behavior into the learning process in this context.

We have also present the cost of the overlay routing overhead in Freenet systems. This overhead is in part due to the Freenet protocol which doesn't take care of the quality of the connections it uses.

Those are preliminary results which still have to be consolidate. First of all, we have used a simulation hypothesis (time independence) to make possible the study with several thousand nodes. This hypothesis is to be validated. Two approaches might be considered: a more detailed simulation with less nodes, and an execution driven by traces. Then the physical approach is to be extended to a more realistic network.

Another important point would be to consider the intrinsic dynamic characters of Peer-to-Peer systems. We think the most important are availability of nodes and the evolution of the files popularity which will be one of our main research field in the near future.

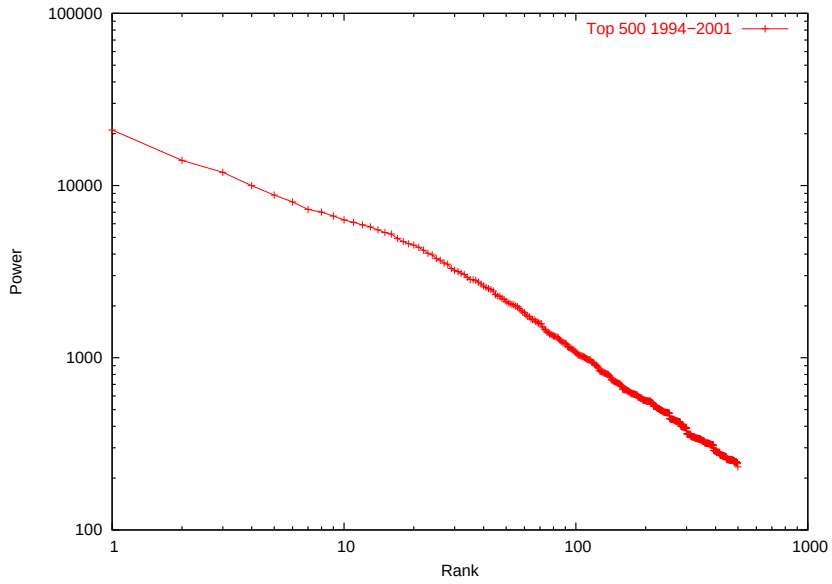

FIG. 8.1. *Logarithmic view of the rank of computers of a given power in the TOP500 between the years 1994 to 2001*

8. The omnipresent Zipf Law. In this article we have encountered several time the zipf law, or sometimes the extended zipf law. It occurred when we had to model the network, or the popularity of files. The figures 8.1 and 8.2 shows that the power of nodes are following the same law. This law is omnipresent when

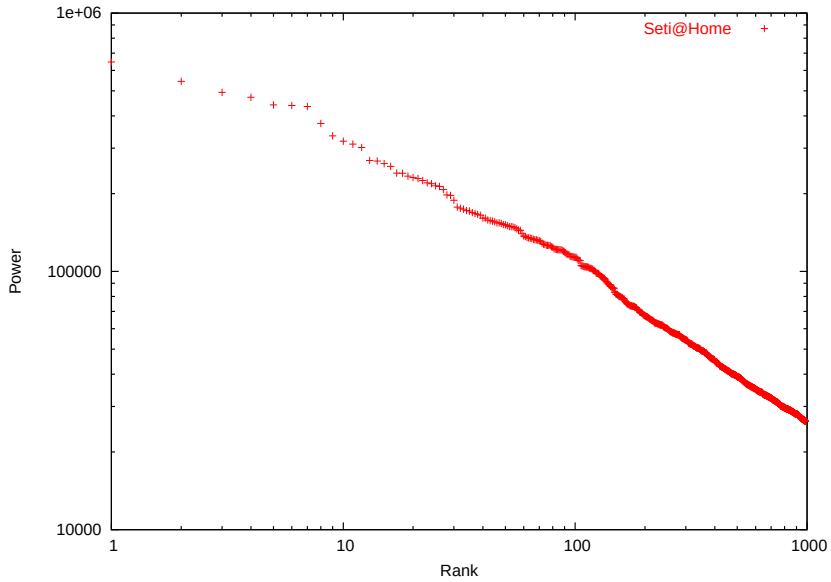

FIG. 8.2. *Logarithmic view of the most powerful participants during the 2002 summer*

there is a need to dimension hardware characteristics such as the power, network bandwidth, and so on. This law often occurs in some other fields like natural language [33].

Acknowledgments. We would like to thank the ID laboratory for granting access to its Cluster Computing Center *http://www-id.imag.fr/grappes.html*. This work utilized the ID/HP i-cluster. We would like to thank *ircache*, *Boeing* and *clarknet* for allowing us to use their web logs.

REFERENCES

[1] V. ALMEIDA, A. BESTAVROS, M. CROVELLA, AND A. DEOLIVEIRA, *Characterizing reference locality in the WWW*, Tech. Report 1996-011, Boston University Computer Science Department, 21, 1996.
[2] M. F. ARLITT AND C. L. WILLIAMSON, *Web server workload characterization: The search for invariants*, in Measurement and Modeling of Computer Systems, 1996, pp. 126–137.
[3] P. BARFORD AND M. CROVELLA, *Generating representative web workloads for network and server performance evaluation*, in Measurement and Modeling of Computer Systems, 1998, pp. 151–160.
[4] H. C. P. BEN Y. ZHAO JOHN KUBIATOWICZ, *Silverback: A global-scale archival system*, tech. report, Computer Science Division University of California Berkeley, 2001.
[5] `http://repository.cs.vt.edu/`
[6] L. BRESLAU, P. CAO, L. FAN, G. PHILLIPS, AND S. SHENKER, *Web caching and zipf-like distributions: Evidence and implications*, 1998.
[7] I. CLARKE, *A distributed decentralised information storage and retrieval system*, 1999.
[8] I. CLARKE, O. SANDBERG, B. WILEY, AND T. W. HONG, *Freenet: A distributed anonymous information storage and retrieval system*, in Workshop on Design Issues in Anonymity and Unobservability, 2000, pp. 46–66.
[9] `http://ita.ee.lbl.gov/html/contrib/clarknet-http.html`.
[10] M. E. CROVELLA AND A. BESTAVROS, *Self-similarity in World Wide Web traffic: evidence and possible causes*, IEEE/ACM Transactions on Networking, 5 (1997), pp. 835–846.
[11] C. R. CUNHA, A. BESTAVROS, AND M. E. CROVELLA, *Characteristics of www client-based traces*, Tech. Report TR-95-010, Boston University Computer Science Department, 1995.
[12] `http://www.fasttrack.nu/`.
[13] I. FOSTER, *Peer-to-peer architecture case study: Gnutella network*, tech. report, Computer Science Dep of the University of Chicago, 2001. TR-2001-26.
[14] FOURTH IEEE SYMPOSIUM ON COMPUTERS AND COMMUNICATIONS (ISCC'99), *Network Emulation in the Vint/NS Simulator*, July 1999.
[15] GLOBECOM '2000, *Generating network topologies that obey power laws*, November 2000.
[16] `http://gnutella.wego.com/`.
[17] P. HART, N. NILSSON, AND B. RAPHAEL, *A formal basis for the heuristic determination of minimum cost paths*, IEEE Transactions on Systems Science and Cybernetics, 2 (1968), pp. 100–107.
[18] G. HEIDELBERG, ed., *Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems*, IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), November 2001.

[19] IEEE MASCOTS'01, *BRITE: An Approach to Universal Topology Generation*, 2001.
[20] IEEE/ACM Transactions on Networking, *Difficulties in Simulating the Internet*, vol. 9, August 2001.
[21] `ftp://ftp.ircache.net/traces/`.
[22] A. Medina, I. Matta, and J. Byers, *On the origin of power laws in internet topologies*, Tech. Report 2000-004, Boston University Computer Science Department, January 2000.
[23] Multimedia Computing and Networking 2002 (MMCN'02), *A Measurement Study of Peer-to-Peer File Sharing Systems*, 2002.
[24] `http://www.opencola.com/`.
[25] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe, *Modeling TCP throughput: A simple model and its empirical validation*, in ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication, Vancouver, CA, 1998, pp. 303–314.
[26] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, *A scalable content addressable network*, Tech. Report TR-00-010, Berkeley, Berkeley, CA, 2000.
[27] D. Scholl, *Nap protocol specification*, tech. report, Sourceforge, 2000. `http://opennap.sourceforge.net/napster.txt`
[28] C. Silverstein, M. Henzinger, H. Marais, and M. Moricz, *Analysis of a very large altavista query log*, tech. report, DEC, 1998. On-line at `http://gatekeeper.dec.com/pub/DEC/SRC/technical-notes/abstracts/src-tn-1998-014.html`
[29] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, *Chord: A scalable peer-to-peer lookup service for internet applications*, Tech. Report TR-819, MIT, March 2001.
[30] T. Walsh, *Search in a small world*, in IJCAI, 1999, pp. 1172–1177.
[31] `http://www.web-caching.com/traces-logs.html`
[32] B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph, *Tapestry: An infrastructure for fault-tolerant wide-area location and routing*, tech. report, Computer Science Division University of California Berkeley, 2001.
[33] G. Zipf, *Human behavior and the principle of least effort*, 1949.

Edited by: Dan Grigoras, John P. Morrison, Marcin Paprzycki
Received: October 02, 2002
Accepted: December 02, 2002