Analysis of peer-to-peer systems: workload characterization and effects on traffic cacheability

Mauro Andreolini
University of Rome "Tor Vergata"
andreolini@ing.uniroma2.it

Riccardo Lancellotti
University of Modena and Reggio Emilia
lancellotti.riccardo@unimo.it

Philip S. Yu
IBM T.J. Watson Research Center
psyu@us.ibm.com

Abstract

Peer-to-peer file sharing networks have emerged as a new popular application in the Internet scenario. In this paper, we provide an analytical model of the resources size and of the contents shared at a given node. We also study the composition of the content workload hosted in the Gnutella network over time. Finally, we investigate the negative impact of oversimplified hypotheses (e.g., the use of filenames as resource identifiers) on the potentially achievable hit rate of a file sharing cache. The message coming out of our findings is clear: file sharing traffic can be reduced by using a cache to minimize download time and network usage. The design and tuning of the cache server should take into account the presence of different resources sharing the same name and should consider push-based downloads. Failing to do so can result in reduced effectiveness of the caching mechanism.

1. Introduction

In the last few years, peer-to-peer systems have experienced a huge popularity growth among the users of the Internet. This reality is witnessed by the large number of applications deployed in a collaborative fashion. Examples include distributed file systems [6, 2], Web caches [5] and, above all, file sharing systems. The strong points of peer-to-peer systems lie in their excellent scalability and fault tolerance, due to the highly distributed architecture and to the degree of resource replication. Another popular feature among file sharing protocols, *anonymity*, has contributed to the high success of this particular family of applications.

Since file sharing services have emerged as the killer application on peer-to-peer systems, many research efforts have been devoted into the characterization of the typical workload and traffic patterns involved. In particular, the hottest topics focus towards performance impact (bottleneck analysis, efficient routing schemes, resource cacheability) and characterization of the user workload (session times, file types and sizes, shared contents per node, client connectivity).

Several peer-to-peer file exchange services are available to the user community nowadays. Although a complete list of these systems is beyond the scope of our work, the most popular networks happen to be Gnutella, Kazaa and EDonkey. According to a recent study [13], the popularity of different file sharing networks changes over time. However, Gnutella and Kazaa are undoubtedly among the most widespread ones.

The contribution of this paper is twofold. First, we provide a workload analysis of the Gnutella network, mainly oriented to resource characterization. We provide an analytical model for the sizes of both the individual resources and the whole content shared at a single node. To the best of our knowledge, this is the first study providing an analytical model of content sizes hosted over a file sharing network. Moreover, we confirm the results on workload composition provided by Leibowitz *et al.* [7] for the Kazaa network also for Gnutella file-sharing.

The second contribution of our research is the analysis of the impact on download traffic caching performance of simplifying assumptions introduced by previous studies [7, 8]. In particular, the filename is considered as a unique identifier for the content, although several filenames may be associated to the same content and viceversa. We also analyze in greater detail the consequences of having nodes advertising unroutable IP addresses (also called *fake IPs*) in the network. We show how those two factors lead to an overestimation of resource cacheability. This is the first study addressing the issues of factors reducing download cacheabil-

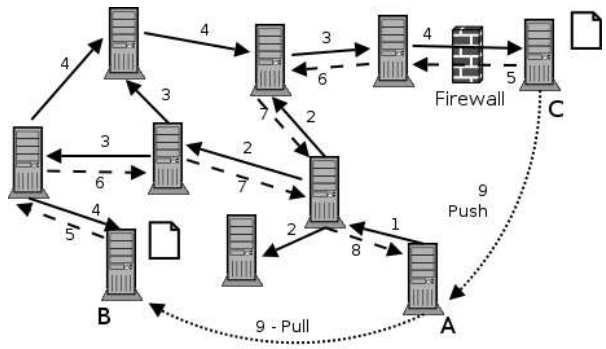

Figure 1. Sample Gnutella network

ity in a file sharing network.

The rest of this paper is structured as follows. In Section 2 we give a brief overview of the Gnutella file sharing service. Section 3 describes our approach to the problem as well as the methodology used to collect and analyze results. In Section 4 we provide an overview of Gnutella file sharing workload, while in Section 5 we provide an analytical model for resource sizes and for the amount of data shared by peers. Section 6 analyzes the impact on caching performance of the use of different resource identifier as well as of the presence of nodes advertising fake IPs. Section 7 presents the state of the art in workload analysis of peer-to-peer systems, while Section 8 shows the concluding remarks.

2. The Gnutella file sharing network

The Gnutella network [3] is a file-sharing system based on a controlled flood search mechanism. Gnutella implements an overlay network with flexible routing and advanced resource location services. When a *peer* (also called *servent* or simply *node*) joins the Gnutella overlay network, it creates a new TCP connection towards one or more peers. Network management and query messages are propagated through the overlay network, while file exchange occurs creating a new connections for the resource transfer.

Fig. 1 shows the working of the Gnutella file sharing system: let A be a peer looking for a resource (the sheet icon on peers B and C). The resource discovery phase uses a flood-based algorithm in which the query is propagated by each peer towards its neighbors (steps 1 to 4 in Fig. 1). In case of hit a query-hit message is propagated back from the peers in which the resource is found (B and C) to the origin peer A (steps 5 to 8 in Fig. 1).

Once the resource is discovered, a new connection towards the peer B is created (step 9 in Fig. 1) and resource download is carried out using the standard HTTP protocol. Peer C is not directly reachable because of a the presence of a firewall that blocks incoming connections. Often firewalls also provides NAT and IP masquerading, so peer C may have non routable IP addresses (and for communications with the rest of the Internet TCP and IP communications use the public IP of the firewall). The firewall increases the complexity of download from peer C because the connection involving the transfer of the actual resource data must be initiated by peer C (the non-addressable/non-reachable servent hosting the file), instead of peer A (the requester). In the case of a peer behind a firewall, then, the traditional *pull* scheme for file exchange is substituted by a *push* one. Obviously, if both nodes are behind a firewall, data exchange is impossible.

A resource in a Gnutella network is identified in two ways. For most operations (such as lookup and HTTP download) the filename is used. Moreover, for each shared resource, a digest of the file content is computed. The latter is typically used to detect corrupted files or identical copies of the same resource to allow parallel downloads from multiple servents. Indeed, due to the strong hash functions used for digest computation, the hash value can be considered as a unique identifier for the actual file content. In our study we take into account both identifiers to provide a more accurate characterization of resource characteristics and cacheability.

3. Experimental methodology

In this section, we describe and motivate the design of the system used for data collection and analysis and we provide a brief description of the data collection strategy of our experiments.

3.1. Architectural choices

The two principal issues addressed during the design phase regard both the mechanisms used for the collection and analysis of data, and the nature of the data itself.

In current literature, two main approaches have been deployed for the collection of data from a P2P system: *crawling* [11, 10] and *packet interception* [7, 8, 12].

Although packet-level traffic analysis represents the most straightforward way to extract information about the peer-to-peer workload, we have chosen not to go that way. By analyzing packets, it is possible to record peer-to-peer information exchange and downloads only for the portion of network under study. Hence it is difficult to decide how much the observed traffic and workload patterns are representative of the overall network and links analyzed for workload characterization must be carefully selected. On the other hand, crawling the peer-to-peer network through well-defined queries does not suffer from the aforementioned problem. Nevertheless, it must be said that both the storage capacity of the crawler and the time to

live of the queries may represent limiting factors in the resource discovery process. To reduce the impact of those problems, we made sure not to have any disk storage limitations during the collection process. Furthermore, we have verified that the amount of data collected in our experiments is comparable in size and variety to the one presented in other studies [7, 11, 12]. We believe, thus, that the above issues do not interfere with the accuracy of our results.

The use of a data crawler brings another problem, which is related to the measurement of the popularity of a given resource. Traffic-oriented studies [7, 8] use the number of downloads for a specific resource as a measure of its popularity. Instead, we consider the number of replicas as a measure of popularity. This approach has been used in other studies such as [11, 10, 9] and is based on the assumption that, once a resource has been retrieved, it becomes available also on the peer that downloaded it (at least for a short period of time). The discussion on the exact relationship between resource replicas and downloads is still ongoing [4], Nevertheless, we can assume that the number of downloads and the number of replicas for a resource are highly correlated.

Another important choice concerns the networks which have to be analyzed. Due to their popularity, we believe that a realistic peer-to-peer workload characterization should take into account at least the Kazaa or Gnutella network. Unfortunately, the Kazaa protocol specification is not publicly available and it uses cryptography to make any kind of data analysis or reverse engineering of the protocol very difficult. Therefore, a crawler-based study of the Kazaa network is not implementable in a straightforward way.

3.2. Data collection strategy

A first important decision we have made is to let the queries run until the set of obtained results reaches a steady state (that is, it does not show any significant increase). This implies that the interval between two consecutive *hit* messages is very long. We consider the query stabilized if we do not see *hit* messages for at least 15 minutes. As a result crawling runs takes typically 48 hours. Running the queries for short periods would lead to a less detailed working set representation. Moreover, we are not interested in characterizing the hourly behavior of the network nodes (a study of the time in the day each node is available has already been given in [12]), thus our time-related studies do not require a finer (possibly hour-wise) time granularity. On the other hand, longer crawler runs would reduce the accuracy of the temporal analysis. Our choice represents a reasonable trade-off between richness of the result set and time accuracy.

The results presented in this paper are obtained through

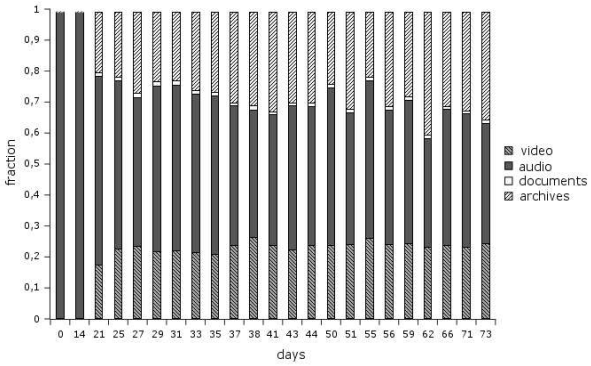

Figure 2. Percentage of resources per type

only one crawler node. However, we validated them running similar queries from nodes at different geographical network locations in two different occasions. We found that the characterization is very similar (i.e. the results from the two nodes on the number of resources found are within 5% one from each other and the workload composition breakdown is identical). This can be explained by considering that, although Gnutella is a distributed network, some very popular nodes act as *hubs* that are chosen as neighbors by the vast majority of the participating nodes, as pointed out in [11]. Indeed, as soon as the query reaches one of those hubs, it is propagated in a similar way notwithstanding the different origin node.

4. Workload overview

In this section we provide an overview of the peer-to-peer file sharing workload. In particular, we characterize resources based on their type and we study the amount of data available on the generic peer of the network.

We started data collection on August 12th, 2003 and continued our experiments for nearly three months. On each crawler run we contacted a number of nodes ranging from 30000 to more than 100000 with an average of 78900. In a similar way the average number of resources discovered for each observation oscillates between 305000 and 1500000, with an average value of 798000. The smallest number of IP addresses and resources occur in the first observations, during the summer period. This result is expected, as because many file-sharing peers are operated by private persons, the summer holidays reduce the file-sharer population.

4.1. Working set composition

We consider now the composition of the workload, in particular the working set partition into basic resource types, as a function of time.

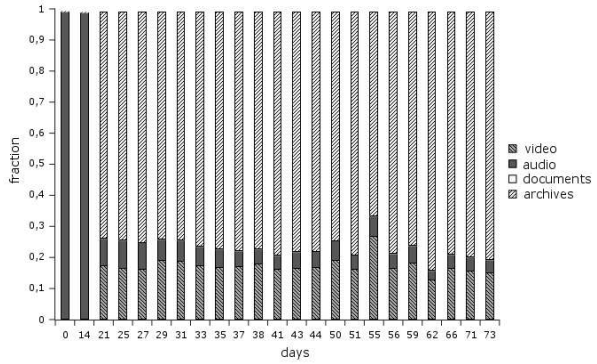

Figure 3. Percentage of bytes per type

Resource type	file extension
Video	.avi, .mpg, .mpeg
Audio	.mp3, .ogg
Documents	.doc, .pdf
Archives	.arj, .zip, .rar, .gz, .exe

Table 1. The resource partitioning scheme

We partitioned the resources into four categories: *audio* and *video* contents, *archives* (essentially software), and *documents*. Table 1 shows the chosen categories along with the file extensions used to assign a filename to the appropriate category. This separation is consistent with the one presented in [7].

Using the filename to determine the resource type does not lead to significant errors in our study. Indeed, we have downloaded nearly one hundred random files and found that, even if different contents can share the same name, their type corresponds to the filename extension (in our observations we found only two files whose mime type was different from the mime type suggested by the filename).

Let $\mathcal{R}(t) = \{\rho_1, \rho_2, \ldots, \rho_N\}$ be the global set of resources available in the file sharing network at time t. We call ρ_i a generic resource, with $\rho_i \in \mathcal{R}(t)$. Furthermore we define $\tau(\rho_i)$ as the type of the resource, where $\tau(\rho_i) \in \mathcal{T}$, and $\mathcal{T} = \{\text{audio}, \text{video}, \text{document}, \text{archive}\}$ is the set of categories. We also introduce $P(\rho_i)$, defined as the number of replicas of a given resource ρ_i in the file sharing network.

For each crawler run (identified by the time t) and for each type $\tau_j \in \mathcal{T}$, we report the number of resources of each type $N(\tau_j, t)$. We define $N(\tau_j, t) = \sum_{\rho_i \in \mathcal{R}_{\tau j}(t)} P(\rho_i)$, where $\mathcal{R}_{\tau j}(t) = \{\rho_i \in \mathcal{R}(t) : \tau(\rho_i) = \tau_j\}$. The values are normalized and plotted as bars in Fig. 2. The number below each bar represents the offset $\Delta t = t - t_0$ (in days) from the first run.

The first two bars of Fig. 2, showing data collected during the month of August, look different from the other ones: the vast majority of resources belongs to the audio category. During the same period, the number of shared resources is greatly reduced. This seems to suggest a seasonal dependency in workload composition, with the workload size shrinking during summer due to the reduction in the number of available nodes. The data collected in our experiments, being related only to three months of observations, still does not allow us to reach a definitive conclusion on this interesting point.

If we discard the summer period, we notice that the working set composition is quite regular. Audio clips are the most popular resources. Videos and archives also contribute to the workload composition (albeit in a more contained way). On the other hand, documents are much less widespread (in Fig. 2 they are placed right above the audio resources). In particular, we verified that the distribution of the resource types presents a standard deviation of 1 to 6 percent points, a value that confirms the stability over time of resource type distribution.

Fig. 3 shows the contribution of the different resource types to the entire working set size. Let $s(\rho_i)$ be the size of the resource ρ_i. We define the size of the resources of type τ_j as $S(\tau_j, t) = \sum_{\rho_i \in \mathcal{R}_{\tau j}(t)} s(\rho_i) P(\rho_i)$. The bars in the graph represent a normalized value of $S(\tau_j, t), \forall \tau_j \in \mathcal{T}$ as a function of Δt. In other words, the bars show how many bytes of shared data belong to the corresponding resource category. A comparison with Fig. 2 makes it evident that the contributions to the working set size are different from the ones to the number of shared resources. While audio files account for the majority of files, their impact on content size is much less significant. On the other hand, if we consider archives, Fig. 3 shows that they provide (at least if we discard the summer period) the largest fraction of the shared bytes. This discrepancy is due to the great differences in average resource size: while audio files are typically a few MBytes long, archival resources are generally two or more orders of magnitude larger, reaching sizes of GBytes. The same considerations also hold for video resources. The number of files belonging to the video category is, overall, nearly the half with respect to audio clips, but the contribution of video resources to working set size is more than double with respect to .mp3 and .ogg files. Our findings confirm the results of Leibowitz *et al.* [7], in a study of Kazaa downloads carried out by means of a network traffic analyzer. Moreover, when considering resource size for the various types, we confirm the stability observed for the number of resources. Indeed, standard deviation ranges from 4 to less than 1 percent points depending on the resource type.

From the data plotted in Fig. 2 and 3, we derive also the average resource size as a function of the resource type $\overline{s}(\tau_j, t) = \frac{S(\tau_j, t)}{N(\tau_j, t)}, \forall \tau_j \in \mathcal{T}$. We evaluate the mean size

	Mean resource size [MB]			
	Min	Average	Max	Stddev
Video	30	32	34	2.5
Audio	4.7	4.8	5	0.3
Document	8	8.4	8.9	0.8
Archive	110	120	130	4.5

Table 2. Mean size of resources according to type over time

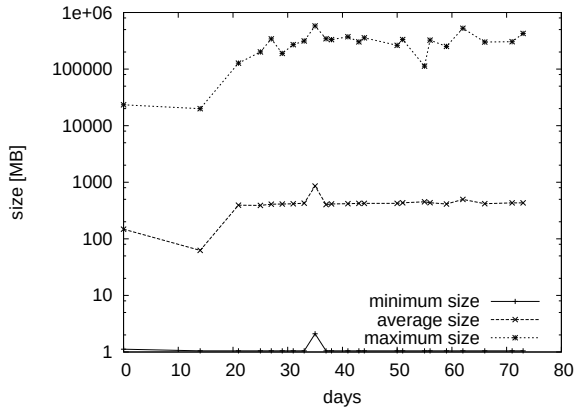

Figure 4. Contents shared each peer over time

of resources belonging to a given category at each crawling interval t. Table 2 shows the minimum, average and maximum values as well as the standard deviation of the samples over the whole period. The Table confirms the previous findings: archives tend to be much larger than the other resources, while audio clips are the smallest files. The small difference between minimum and maximum values in Table 2, as well as the standard deviation confirms that the average resource size $\overline{s}(\tau_j, t)$ is fairly stable over time.

4.2. Node contents

We now study how resources are distributed among nodes in the network. Let $\mathcal{N}(t)$ be the set of nodes of the peer-to-peer network at a given instant t and $n_k \in \mathcal{N}(t)$ a generic node hosting a file sharing peer. $\mathcal{R}_{nk}(t)$ is the set of resources hosted on node n_k. We define $S(n_k, t) = \sum_{\rho_i \in \mathcal{R}_{nk}(t)} s(\rho_i)$ as the size of contents hosted on the node n_k at time t. Fig. 4 shows the minimum, average and maximum size of contents hosted in a single node as a function of the time elapsed since the first crawler run. A first interesting remark which can be drawn from the graph is the high variance of the amount of content hosted on the file sharing network. The size of resources in each node varies from 1Mb to nearly 1 TB, ranging over six orders of magnitude.

Another interesting remark is that, despite the high variability, the average size of the resources hosted on a single node is pretty stable over time, oscillating around a value of 400-500 MB (the standard deviation is of 30 MB). The leftmost part of the graph (in particular the data related to the first 30 days) tends to show slightly different values from those seen in other periods. However, we already observed that, during the summer, the workload composition is different with respect to the other observation intervals, with a reduced presence of video resources. Since videos and archives account for a significant portion of the whole working set, the content size measurements corresponding to this period are affected.

5. Analytical models

In this section we provide an analytical fitting for the data studied in the previous section. In particular we focus on resource size and on the shared resource volume for each peer.

We start by providing an analytical model of the resource size distribution. We generate a histogram of the resource size. We show results for a single day of observations, since we verified that the resource size histograms do not change significantly over time.

The development of an analytical model of the histogram of resource sizes $s(\rho_i)$ is not feasible. Thus, we have performed separate fits pertaining to each resource type (i.e. modeling the size of resources in the sets $\mathcal{R}_{\tau j}, \forall \tau_j \in \mathcal{T}$). For the fitting we used the nonlinear least-squares (NLLS) Marquardt-Levenberg algorithm.

The sizes of resources belonging to the audio and document categories can be modeled through a lognormal distribution, while distribution of file size for videos and documents may be better fitted through a combination of lognormal and Pareto distribution. The use of a lognormal curve for the body of a distribution and Pareto for its tail has been already successfully used in other workload characterizations with heavy-tailed distributions such as in [1]. When considering the size histogram of video resources we also notice an irregular popularity spike between 700 and 750 MB. Indeed sharing a movie in a CD-sized file has been observed in many peers. The tip of the spike is nearly one order of magnitude higher than the expected value (according to the distribution). However, this anomaly is negligible, being two order of magnitude lower than the values shown in the body of the distribution.

To provide a clear reference for the parameters used in our model, we recall that the lognormal probability distribution function is defined as: $p(x) = \frac{1}{x\sqrt{2\pi\sigma^2}} e^{\frac{-(\ln(x)-\mu)^2}{2\sigma^2}}$ for $x > 0$, while the Pareto distribution is defined as

Video	
Distribution	Lognormal if $x < 6$ MB, Pareto otherwise
Lognormal param.	$\sigma^2 = 1.23$, $\mu = 1.55$
Pareto param.	$a = 6.0$, $b = 0.12$
Audio	
Distribution	Lognormal
Lognormal param.	$\sigma^2 = 0.12$, $\mu = 1.42$
Document	
Distribution	Lognormal
Lognormal param.	$\sigma^2 = 2.38$, $\mu = 1.23$
Archive	
Distribution	Lognormal if $x < 10$ MB, Pareto otherwise
Lognormal param.	$\sigma^2 = 0.31$, $\mu = 1.00$
Pareto param.	$a = 5.98$, $b = 0.1$

Table 3. Size of resources according to type

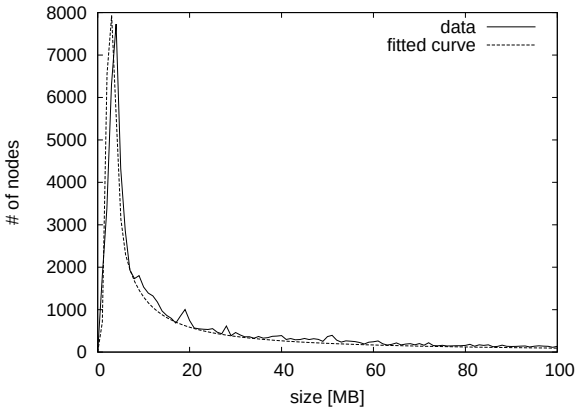

Figure 5. Content size for each node

$p(x) = \frac{b}{x}\left(\frac{a}{x}\right)^b$, for $x > a$. Table 3 provides the parameters used for fitting the resource size histogram data with analytical distributions.

The second set of data for which we provide an analytical mode is the content size $S(n_k, t)$ shared by a peer $\forall n_k \in \mathcal{N}(t)$ during a given day. Fig. 5 illustrates a histogram of the volume of resource shared by peers. For each possible content size, the number of nodes sharing the corresponding amount of data is plotted. As for the resource size histogram, the choice of focusing on a particular day does not invalidate our conclusions, since we verified that the graph does not change significantly over time.

As expected, the resulting curve is clearly heavy-tailed. It shows that many nodes share only a few MB of data (typically less than 10 MB). On the other hand, a non-negligible number of nodes shares massive amounts of data.

The content size histogram is difficult to fit with a simple closed-form probability distribution. As for the resource size, we obtained a good fitting by modeling the body with a lognormal distribution and the tail with a Pareto distribution, where the body is composed by sizes below 5MB. By referring to the formulas provided previously in the section, we found that the parameters providing the best fit are respectively $\mu = 1.2$ and $\sigma^2 = 0.2$ for the lognormal distribution and $a = 4.96$ and $b = 0.14$ for the Pareto distribution. Fig. 5 shows the collected data and the analytical curve that best fits them. Although not shown in the graph, we also notice an irregular popularity spike between 700 and 750 MB. Taking into account resource size histograms we conclude that the spike is due to the nodes sharing one large video file. The tip of the spike reaches a popularity value of 70 instances, which is four times higher than the expected value (according to the distribution). However, the anomalous spike is negligible, being nearly three order of magnitude lower than the values shown in the leftmost part of Fig. 5.

6. Impact on caching

6.1. Correlation between resource names and actual content

In this section we evaluate the goodness of the filename as a unique identifier for a given resource. The relationship between a resource filename and a resource hash value is many-to-many. It is possible for different resources to share the same filename; for example, an audio clip can be encoded in different ways, resulting in several files with distinct hash values and the same name. On the other hand, due to the not-so-strict naming conventions adopted in the file sharing world, the same content can be hosted in different nodes under different names. We believe that, for popular resources, the discrepancy between hashes and filenames can introduce a non-negligible error in capacity planning studies.

In the following, we will refer to the *resource name popularity* as to the number of occurrences of a name on the network nodes. Similarly, the term *resource hash popularity* will be used to denote the number of times an hash code appears on different nodes of the network.

Computing the number of distinct filenames and hash values observed during the entire crawling period, we find a difference of only 2%. The reason behind this small difference lies in the extremely high number of resources which occur only once, yielding an almost one-to-one mapping between filenames and hash values.

Things change dramatically when we focus on popular resources. Fig. 6 displays the average number of distinct hash values, computed over all resources having a given name popularity. We avoided a scatter graph because the resulting figure would be hardly readable. The figure shows that, in the average, the number of distinct hash values associated with a filename is greater than one. This means that, when dealing with popular resources, multiple hash values

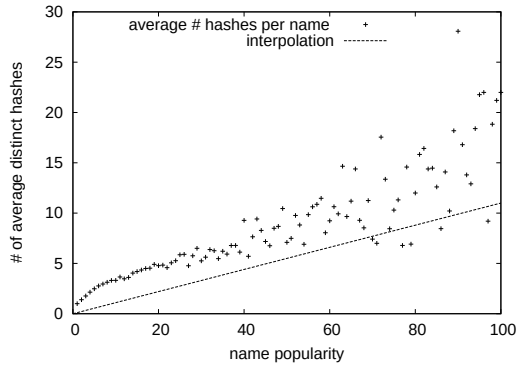

Figure 6. name popularity vs. hashes

are associated to the same filename. As a consequence, the use of filenames as unique resource identifiers is inappropriate and may lead to evaluation errors. We downloaded some resources and found that, in most cases, different files with the same name are just different versions (e.g., different encoding of different metadata) of the same resource. However, the two files are different for download purposes because download integrity is checked against the hash code.

The data interpolation in Fig. 6 shows a linear correlation between the number of hashes and the name popularity. The number of distinct hash values follows roughly the law: $y = mx$, where $m = 0.11$. The fluctuations in the average value of distinct hashes on the right side of the graph are due to the reduced number of samples for highly popular resource names (for example, the number of resource filenames with name popularity equal to 100 is much lower than the number of resource filenames with name popularity equal to 2).

Let us now focus on resource popularity and cacheability issues. We show that using filenames (instead of hash values) as resource identifiers can lead to an overestimation of the potential for caching. Our study confirms that the number of replicas of a resource follows a Zipf distribution. Zipf distribution is widely used in computer science literature to model the concept of popularity. We already defined $\mathcal{R}(t)$ as the set of resources at the instant t. Let $P(\rho_i)$ is the resource popularity. We recall that in this study we use the number of replicas as a measure of the resource popularity. In the following, we assume that the resources are ordered according to their popularity, such that $P(\rho_i) > P(\rho_{i+1})$, $\forall i < N$. $P(\rho_i)$ is inversely proportional to its popularity rank i, $P \propto \frac{1}{i^\alpha}$. From the remarks presented earlier in this section, we expect a different skewness factor α depending on whether we use hash values or filenames to identify distinct resources.

For every run of the crawler (identified with the time t at which it occurred), we compute the number of replicas of filenames and hash values as a function of the popularity rank. We fitted those popularity plots into Zipf curves and retrieved the corresponding $\alpha(t)$ values. A summary of the obtained results is reported in Table 4. The Zipf coefficients have proven to be fairly stable over time. In particular, if we exclude the summer period, where α tends to be lower for both resource identifiers, the use of hash values yields an estimation of $\alpha(t)$ between 0.54 and 0.64, with an average value of 0.58 and a standard deviation of 0.03. If filenames are used, instead, the Zipf coefficient ranges between 0.58 and 0.71, with an average of 0.64 and a standard deviation of 0.04. This confirms that, when it comes to popular resources, the use of filenames as resource identifiers tends to aggregate different contents under the same name, thus leading to an overestimation of the popularity skewness.

	Zipf α parameter			
	Min	Average	Max	Stddev
Hash	0.54	0.58	0.64	0.03
Name	0.58	0.64	0.71	0.04

Table 4. Zipf α values

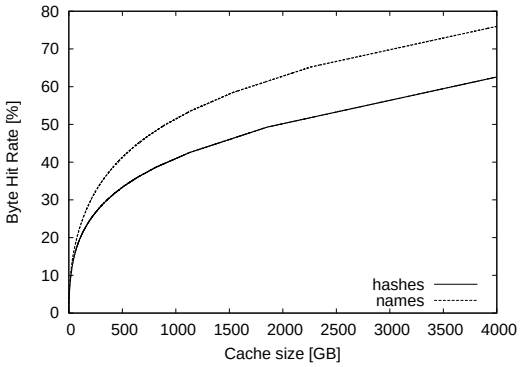

Figure 7. Potential cache byte hit rate

An incorrect estimation of resource popularity leads to erroneous conclusions on the potential advantages of resource caching, as we show next.

To express the relationship between workload parameters and potential cache hit rate we have developed a simple model. We omit the explicit indication of the dependency from time in the formulas and we refer to the stationary case, which represents an upper bound to cacheability. Under stationary assumption, a file-sharing cache will contain the most popular resources, i.e. if S_C is the size of the cache, the storage will host the i^* most popular resources, where i^* is defined as $i^* = max\{i : \sum_{j=0}^{i} s(\rho_j) < S_C\}$. We now introduce the traffic generated by the first i resources as: $S(i) = \sum_{j=0}^{i} P(\rho_j)s(\rho_j)$. It is possible to express the potential cache byte hit rate as $HR = \frac{S(i^*)}{S(N)}$. As $P(\rho_i) \propto \frac{1}{i^\alpha}$,

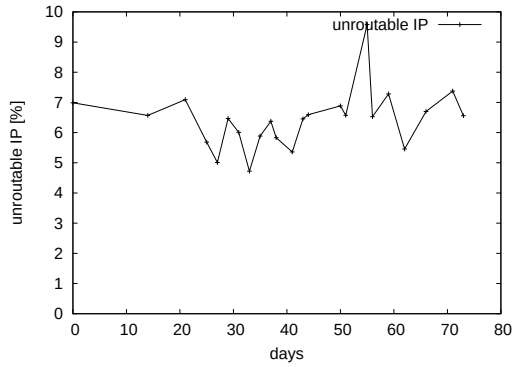

Figure 8. percentage of fake IPs

HR can further simplified as

$$HR = \frac{\sum_{i=0}^{i^*} \frac{s(\rho_i)}{i^\alpha}}{\sum_{i=0}^{N} \frac{s(\rho_i)}{i^\alpha}} \quad (1)$$

From this expression it is clear that the byte hit rate depends on the values of N and α.

Fig. 7 illustrates the theoretically achievable cache hit rate as a function of the cache size. We verified that changing the crawler logs does not affect the validity of our conclusions. We present two curves, obtained considering both the filename and the hash value as resource identifiers, respectively. The use of the filename leads to overestimation of the achievable hit rate up to more than 15 percent (with consequent underprovisioning of the cache). In particular the use of filenames leads to:

- overestimation of the value of α.
- underestimation the working set size (N is 25% less with respect to the correct value obtained by using hash values).

It is also worth noting that, when deploying a cache, the use of filenames as resource identifiers can also introduce serious consistency problems. The hash values are very useful in the task of data corruption detection. If filenames are adopted as unique resource identifiers, a cache can only store one resource with that filename. If a peer node issues to a hypothetical peer-to-peer cache a request for a resource with the same filename and different hash value, an inconsistency occurs. Most likely, this results in the rejection of the download.

6.2. Impact of non-routable IP addresses

We now focus on nodes advertising non-routable IP addresses. Furthermore, we provide an insight of the implications of the amount of such nodes on the node content sizes and on the potential cacheability of resources.

	working set reduction		
	Min	Average	Max
workload size	10%	11.1%	13%
# of resources	10%	11.7%	13%

Table 5. Effect of non considering unroutable IP in workload size

Fig. 8 shows the percentage of nodes advertising IP addresses belonging to non-routable subnets (such as 192.168.0.0 or 172.16.0.0). We see that the presence of non-routable IP addresses in the file-sharing network is not negligible. The graph also shows a significant variance in the number of non-routable IP addresses over time, ranging from 4.7% to 9.6%, with an average of 6.4%.

Now we turn to the possible implications of fake IP addresses neglection on the working set analysis. To this purpose, in Table 5 we show the working set reduction caused by discarding fake IP addresses. The reduction is computed both in terms of the number of resources and their sizes. The first interesting remark is that, by eliminating non-routable IP addresses, an overall underestimation of the working set size occurs, which is fairly stable over time, ranging from 87% to 90% for workload size and number of resources.

It is interesting to note that the impact on working set size is, on the average, nearly doubled with respect to the one on the number of nodes. In other words, discarding 6% of the nodes (with unroutable IPs) brings a reduction of more than 10% in the working set. This is due to the higher volume of resources hosted by those nodes. Indeed the average amount of resources shared by those nodes (790 MB) is nearly double with respect to other nodes (410 MB).

As previously described in Section 2, the mechanism introduced in the Gnutella protocol to handle unroutable IPs has an important effect on cacheability. Traditional caches work by intercepting requests and responding in place of the remote node in case of a hit. This mechanism is suitable for a pull paradigm, where requests are issued by the clients. However, for unroutable IP addresses where a push mechanism is used, traditional caching cannot be applied to resources hosted by unreachable nodes. We can assume that those files are uncacheable, which brings a further reduction in the potential achievable hit rate. As a consequence, the hit rate formula in equation 1 changes as follows:

$$HR' = \frac{\sum_{i=0}^{i^*} \frac{s(\rho_i)}{i^\alpha} c(\rho_i)}{\sum_{i=0}^{N} \frac{s(\rho_i)}{i^\alpha}} \quad (2)$$

and $c(\rho_i) = 1$ if ρ_i is cacheable, 0 otherwise.

The overall working set used to calculate the hit rate takes every resource into account (including the files stored

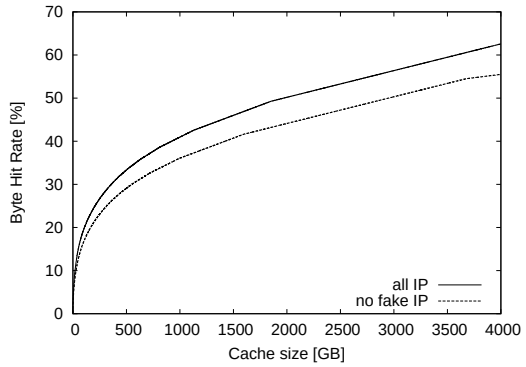

Figure 9. percentage of fake IPs

on unreachable nodes), because they all contribute to file-sharing traffic.

We quantified the penalty in hit rate resulting from resource uncacheability. Fig. 9 shows the achievable cache hit rate as a function of the cache size: the upper curve shows HR, while the lower curve represents $HR^{'}$. In both curves we consider the hash value as the resource identifier, to avoid the errors described in the previous section. The graph shows that taking into account non-routable addresses results in a further drop of nearly 8% in the expected hit rate.

7. Related Work

Peer-to-peer networks have proven a better alternative to traditional client-server systems when it comes to resource sharing. The enormous growth in popularity of peer-to-peer systems has led researchers to study the performance of the overlay routing mechanisms and to characterize the workload and the behavior of users. We find that the latter point, in particular, has not yet been throughly investigated.

Two main techniques have been used for data collection from peer-to-peer systems, namely crawling [11, 10] and packet interception [7, 8, 12].

In [7], Leibowitz *et al.* study the characteristics of traffic related to file downloads in the Kazaa network. The flow of data packets is logged at the central Internet junction of a major Israeli ISP for later inspection. The paper overviews the composition of the traffic for distinct file types (recognized by their name extensions), and evaluates the potential gains due to cacheability of peer-to-peer resources. In particular, the authors study the popularity distribution of file names and calculate the theoretical byte hit rate as a function of the cache size. Their conclusion is that peer-to-peer traffic exhibits high potential for caching, although the expected hit rates could be biased by their choice of using the name of resources as identifier.

A similar analysis is carried out in [8]. In particular the paper provides some discussion on how popularity ranking of resources varies over time, concluding that 15% of the most popular files do not show rank changes over long periods of time.

A packet sniffer-based approach is also used in [12]. Sen and Wang compare the user characteristics of three peer-to-peer file sharing systems (Gnutella, FastTrack and DirectConnect), pointing out connectivity capabilities, user behavior and traffic patterns. To this purpose, the authors use logs collected from several Internet border routers.

Other studies, like [11, 10] use a crawler for collecting data among the participating nodes. In [11], a user characterization of Napster and Gnutella file sharing services is presented, with a focus on network aspects (bottleneck bandwidth, node latency) and node availability. In [10], the analysis focuses on the differences between the structure of the overlay Gnutella network and the physical network topologies, evaluating the performance losses due to the discrepancies between the two.

The discussion is still ongoing on whether resource popularity in peer-to-peer file sharing network can be modeled using Zipf distributions. For example, in [4], Gummadi *et al.* found that download popularity follows a truncated-Zipf distribution due to the download-at-most-once behavior of users. In our study, on the other hand, we found that resource popularity can be described using a Zipf distribution. It is important to note that our study focuses on maximum achievable hit rate (upper bound), hence our conclusions on resource cacheability is referred to an upper bound on cache hit rate. Moreover, our observations on resource hashes and filenames as well as on non-routable IPs have general validity as they are not strictly dependent on the resource/download popularity distributions.

8. Conclusions

In this paper, we have studied the workload of the Gnutella peer-to-peer file sharing system. The first contribution of our research is a characterization of the resources shared by the users. In particular, we evaluate the impact of each resource type to the whole set of files shared over the network. Moreover, we provide an analytical model for both resource sizes and for the size of content shared at each node.

As a further contribution we provide an insight to the factors that can reduce resource cacheability. Our findings are that the use of file names as resource identifiers (instead of the hash values) can lead to an overestimation of achievable hit rate of up to 15%. We also quantified the impact of nodes with non-routable IP addresses to the theoretical cache performance and found that considering cacheable resources hosted by nodes with fake IP addresses brings a fur-

ther overestimation of 8% of hit rate. The message coming out of these considerations is clear: file sharing traffic can be reduced by using a cache to minimize download time and network usage. The task of designing and tuning the cache server, however, should take into account the presence of different resources sharing the same name and should be able to handle push-based downloads. Failing to do so can result in reduced effectiveness of the caching mechanism.

References

[1] P. Barford and M. Crovella. An architecture for a WWW workload generator. In *Proc. of SIGMETRICS, 1998*, 1998.

[2] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proc. of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, Chateau Lake Louise, Banff, Canada, Oct. 2001.

[3] The gnutella protocol specification v0.4. White papers, 2003.

[4] K. P. Gummadi, R. J. Dunn, S. Soriu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, modeling and analysis of peer-to-peer file-sharing workload. In *Proc. of the 19th ACM Symposium on Operating Systems Principles (SOSP-19)*, 2003.

[5] S. Iyer, A. Rowstron, and P. Druschel. Squirrel: A decentralized, peer-to-peer web cache. In *Proc. of 21st ACM Symposium on Principles of Distributed Computing (PODC 2002)*, Monterey, CA, Jul. 2002.

[6] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An architecture for global-scale persistent storage. In *Proc. of ACM ASPLOS*. ACM, Nov. 2000.

[7] N. Leibowitz, A. Bergman, R. Ben-Shaul, and A. Shavit. Are file swapping networks cacheable? characterizing p2p traffic. In *Proc. of 7th Int'l Workshop on Web Content Caching and Distribution (WCW '02)*, Boulder, CO, USA, Aug. 2002.

[8] N. Leibowitz, M. Ripeanu, and A. Wierzbicki. Deconstructiong the kazaa network. In *Proc. of 3rd IEEE Workshop on Internet Applications (WIAPP '03)*, San Jose, CA, USA, Jun. 2003.

[9] D. Nogueira, L. Rocha, J. Santos, P. Araujo, V. Almeida, and W. M. Jr. A methodology for workload characterization of file-sharing peer-to-peer networks. In *Proc. of IEEE 5th Workshop on Workload Characterization*, Austin, TX, Nov. 2002.

[10] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing Journal*, 6(1), 2002.

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

[12] S. Sen and J. Wang. Analyzing peer-to-peer traffic across large networks. In *Proc. of the second ACM SIGCOMM Workshop on Internet measurment*, Marseille, France, Nov. 2002.

[13] D. Towsley. Peer-to-peer and application level networking. In *Proc. of IFIP WG7.3 Int'l Symposium on Computer Performance Modeling, Measurement and Evaluation*, Rome, Italy, Sep. 2002.