

ARE FILE SWAPPING NETWORKS CACHEABLE?

CHARACTERIZING P2P TRAFFIC

Nathaniel Leibowitz, Aviv Bergman, Roy Ben-Shaul, Aviv Shavit

Expand Networks, Tel-Aviv, Israel

Abstract

Peer-to-Peer (P2P) file-sharing traffic on the Internet has grown to rival that of traditional web surfing (HTTP). This paper measures and analyzes the characteristics of this relatively new type of traffic and investigates the feasibility of caching it.

1 INTRODUCTION

1.1 Interest

A significant trend has emerged in recent years whereby use of the Internet for Peer-to-Peer file sharing is growing far more rapidly than traditional HTTP traffic; consequently, many Internet Service Providers (ISPs) experience P2P-traffic volumes that now exceed traditional web-surfing traffic [1]. Just as the characteristics of HTTP-based web surfing have been widely analyzed and recorded [2] – mainly in order to determine and hopefully ameliorate their impact through techniques such as caching frequently-requested files – similar studies are now necessary in order to understand and cope with the huge bandwidth drain created by the P2P phenomenon. This paper presents the results of one such study, whose objectives were twofold:

- Analyzing P2P traffic to research methods that may (as has been done in the case of HTTP traffic) lead to reduced bandwidth consumption (i.e. caching)
- Analyzing the content that flows over P2P networks as a behavioral analysis of the tastes of computer users. Because files shared via P2P file sharing systems encompass all types of digital media consumed by computer users, these files draw a singular picture of society's craving for digital data.

1.2 Challenges

Because P2P traffic differs integrally from HTTP traffic, it necessitates novel monitoring and caching techniques. Consider a typical file download over the dominant FastTrack[3] P2P protocol: The downloader types keywords associated with the file of interest, initiating a search request. This query is sent to a supernode which is a FastTrack client which has been nominated to serve as a local file directory, maintaining lists of file names and

where they can be found. The downloader receives a reply from the supernode specifying the details of all files matching the query search request. The downloader chooses the file of interest from this list, initiating the actual download session. The FastTrack client will then attempt to locate as many FastTrack servers as possible that possess this file, and request a different subportion of the file from each. As more providers of the file are located, some of the ongoing download sessions are terminated in order to split the subpart being downloaded further between the newly located providers.

The following table highlights the unique characteristics of P2P in relation to HTTP traffic, as exemplified in the above download procedure:

HTTP	P2P
Small file size range (0-1 MB)	Huge file size range (100 KB-1 GB)
Objects transferred completely in single or few session(s)	Typical file download opens dozens of simultaneous sessions
Typical session completes within seconds	Session may take hours to complete
Relatively reliable sessions (content transmitted by dedicated servers) → number of sessions reflect popularity	Sessions constantly abort prematurely (content transmitted by PCs) → number of sessions does not indicate popularity
Gradual changes in object popularity	Steep changes in object popularity as songs, movies and applications become hits overnight
Standard protocol	Numerous proprietary protocols
Single port (80)	Numerous ports for each network
Unique content identification by URL ¹	Same content has different names

¹ Recent HTTP traffic analysis [4][5] suggests some portion of HTTP objects are associated with multiple URLs, degrading web caching performance which assumes unique content identification by URL.

The disparity between these traffic types, specifically the large sizes of P2P files, long duration of sessions, the multiple TCP sessions required for a single P2P download and the fact the majority of P2P sessions abort prematurely, demands formulating a meaningful approach to monitoring P2P traffic and devising a unique caching algorithm.

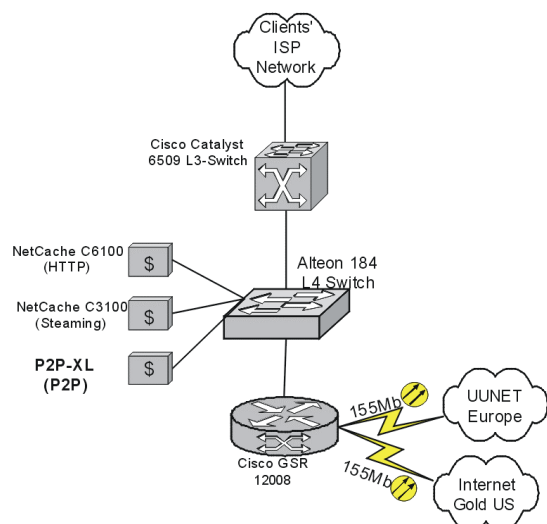
2 METHODOLOGY

This study was divided into two primary categories:

- Measuring, analyzing and characterizing P2P traffic
- Researching the ability to save P2P bandwidth via caching.

The study was facilitated by the design and construction of a special-purpose server, the P2P-XL, specifically designed to monitor P2P traffic in a transparent fashion. By placing this server at a central Internet junction, we were able to collect an exact image of the P2P traffic flowing through the junction. We then applied various analysis tools in order to approach the key issues that were mentioned, namely characterizing digital media consumption habits, and determining the extent to which caching can reduce P2P traffic. This theoretical caching analysis is verified by performing a special caching algorithm on an ISP's live connections and measuring the bandwidth savings the ISP experiences on this network.

The server was installed at a major Israeli ISP and collected data from the line that connects some 10,000 local ADSL and cable users to the USA. The following diagram describes the installation:



The Alteon Layer-4 switch directs all P2P traffic to our P2P-XL server, based on the destination port number of the connection. These connections are accepted by our P2P-XL server, which then extracts the destination IP address of this connection and tries to connect to this address. If successful, this P2P session will consist of two TCP

connections, the first being from the originator to our server, the second from our server to the destination address. By relaying the traffic received on each leg of this double connection to the other leg, the two sides of the connection can complete a full P2P session transparently. The purpose of caching is to minimize the traffic on the expensive links going from the Cisco router to the US and Europe. When a session is completed from the cache, our server transmits the requested data from the disk, thus reducing the usage of the expensive link.

The P2P-XL server runs on a dual 1 GHz Pentium III processor with 2 GB RAM, using a 3COM 100 Mbps NIC and 3 disks that provide a total of 200 GB for storing the cache. The server runs the Linux kernel 2.4.18.aal.

The pie chart in *Figure 1* depicts the traffic breakdown into distinct protocols according to bandwidth consumption^{II}.

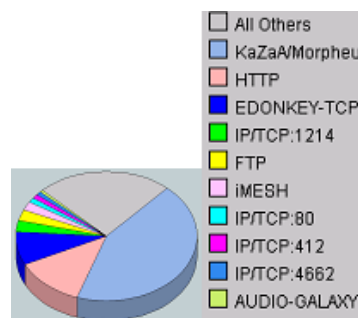


Figure 1. Most Active Protocols

We chose to intercept FastTrack based traffic (KaZaA, Morpheus and Grokster) [6], which, at 45 Mbps, accounts for the plurality of the traffic on the line. The data was collected over a period of one month.

3 RESULTS

This section summarizes our main findings by quantifying the amount of traffic to which our server was exposed and characterizing the composition of the traffic traversing the network. Then a theoretical analysis is performed to determine how much data could have been cached and its effect on bandwidth savings. Finally we present the empirical results of bandwidth saved on the ISP's network by performing caching.

3.1 Network Traffic

Our server intercepts P2P sessions, extracts the IP addresses to which the sessions was destined for, and attempts to connect to them. If the process is successful, the server holds two separate connections for each session: from the originator to the server, and from server to destination. The following table summarizes the connections and downloads

^{II} The well-known Gnutella protocol, not appearing on this chart, was measured separately as approximately 5% of the traffic on this link.

that were intercepted by our server over a course of one month:

Intercepted P2P connections	24,000,000
Successfully connected to destination	12,000,000
Destination was unreachable	8,000,000
Connection to destination timed out	4,000,000
Download sessions	3,700,000
Control sessions (searches, management, etc.)	8,700,000
Average simultaneous downloads	2,000
Peak simultaneous downloads	5,000
Total GB of content transferred in downloads	2,500

3.2 P2P Traffic Characteristics

In this subsection we analyze the composition of P2P traffic in order to obtain a better understanding of its nature and characteristics, and form a picture of user interest in digital data.

We begin by summarizing the distribution of the popularity. The number of times each file was transmitted over the network was computed^{III}, and ordered from high to low. We then computed the accumulated percent of downloads that are generated by accumulated percent of the sorted set of files.

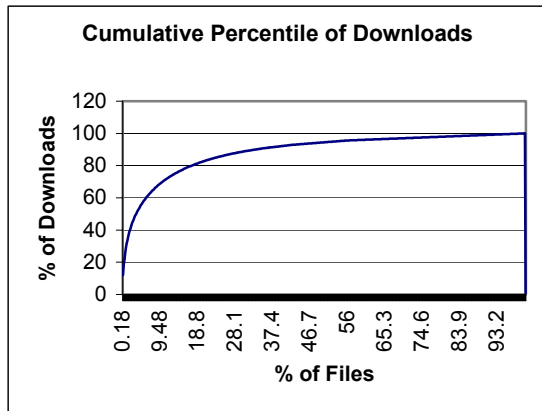


Figure 2. Cumulative Percentile of Downloads

It can be seen that less than 20% of the files account for more than 80% of the downloads. (The exact effect of this behavior on the caching potential of P2P traffic depends on the distribution of the file sizes. Actual caching results are detailed section 3.3, *Caching*.)

The following chart analyzes the breakdown of traffic into common file categories, by dividing the total traffic volume per file category:

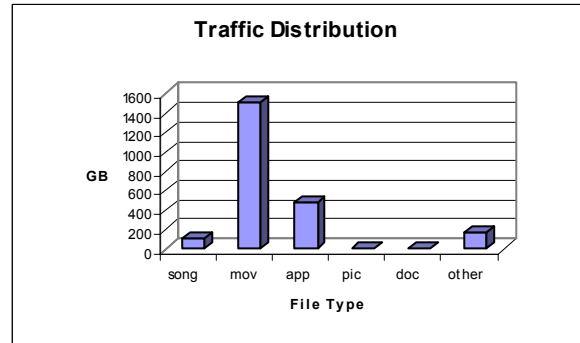


Figure 3. Traffic Distribution According to File Types

Interestingly, most of the P2P traffic on this link consisted of movie files.

The following figure shows a histogram of the file sizes of all files transmitted over the network:

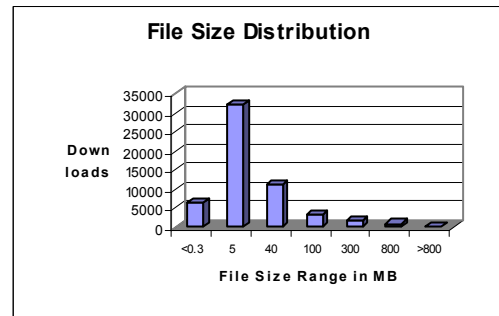


Figure 4. File Size Distribution

The bins for this histogram were chosen to reflect typical sizes of the various file types. For instance, the smallest file size range covers typical picture sizes, the next bin covers average song file sizes; short video clips would enter the third bin. Applications tend to be in the 40-100 MB range, slightly smaller than games, which may reach 300 MB. The last two bins match the sizes of full-length movies. While it is apparent in *Figure 3* that the majority of **traffic** was generated by movie downloads, *Figure 4* indicates that the sizes of the majority of **files** are within the range of typical song files. This apparent contradiction is explained by the extremely large size of movie files and their relative popularity, which accounts for their large share in the overall traffic.

Finally, we analyzed the composition of the 100 files that recurred most frequently in the network by breaking them down into file-type categories. The following charts show the percentage of each file type, and the relative traffic each of these categories generated.

^{III} In P2P networks, this type of calculation is tricky. A typical download of a file is formed by tens of sessions in which sub parts of the file are transmitted. Our approach was to accumulate the number of bytes transmitted for each file and divide by file size.

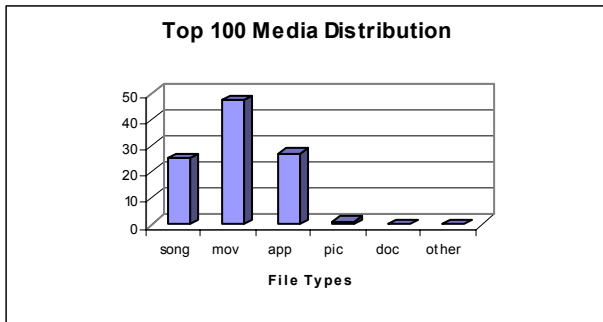


Figure 5. Distribution of Top 100 Files according to File Type

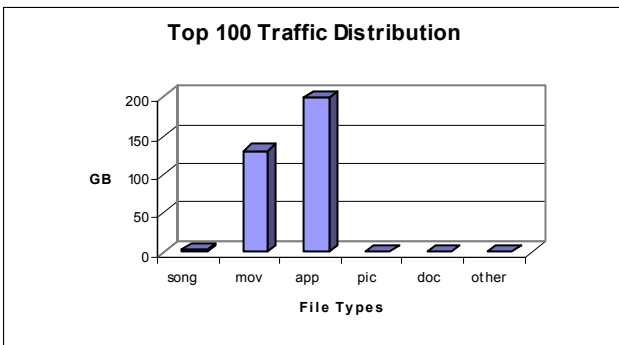


Figure 6. Distribution of Top 100 Files according to Traffic They Generated

A significant increase in traffic volume is perceived when comparing the application traffic in *Figure 3* and *Figure 6* (from 21% to 60%). And comparing *Figure 5* and *Figure 6*, it is clear that although 27% of the top ranking files are music files, they account for very little of the traffic generated by the top ranking files (1%).

3.3 Caching

The pivotal caching issue relates to what percentage of the traffic would be served from cache, had a caching mechanism been used. As mentioned above, HTTP caching mechanisms are not applicable to P2P traffic because of the distinct properties of this traffic. To analyze the extent to which P2P traffic can be cached, we focused on two aspects: theoretical caching potential and empirical caching. For the former, we developed a program that reads the traces of the P2P traffic collected by our server, and applies analytical methods for computing the optimal, static, set of files that should have been stored on a given size cache, in order to generate the largest byte-hit-rate on the traffic defined by the trace^{IV}. In order to empirically verify our

^{IV} This analytical computation does not allow a dynamic cache in which objects are replaced as their popularity fades away, and therefore is a sub-optimal limit on the attainable byte-hit-rate. However over a 4-week period

analytical computations, we implemented a caching algorithm that maximizes byte-hit-rate taking into consideration the special nature of P2P traffic, as noted above.

3.3.1 Theoretical Caching Potential

Analysis of the traffic trace shows that a potential 67% byte-hit-rate could have been generated via caching. This is especially dramatic since it exceeds the high end of HTTP caching systems, which are known to achieve values between 30% and 60% [7].

Even more significant is the surprisingly small disk space required for achieving considerable caching results. Recalling that the traffic is composed of extremely large files (songs contain few MB while movies contain up to 1 GB) caching could have been presumed impractical due to the extremely large disks that are needed for effective caching. In reality, 200 GB of disk space would suffice to achieve close to maximal caching of the traffic we monitored.

The graph in *Figure 7* plots the byte-hit-rate that can be achieved for each disk size.

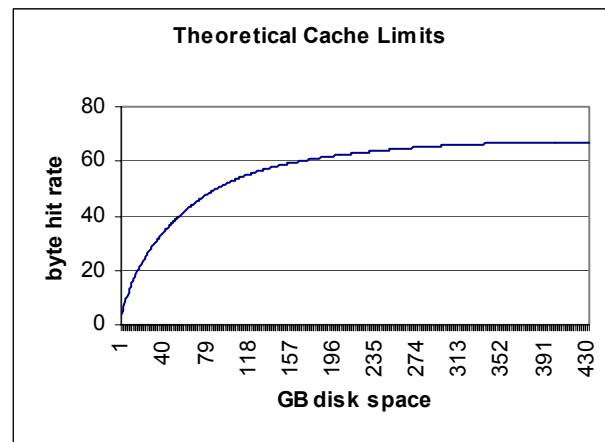


Figure 7. Theoretical Cache Limits

An important question relates to how the byte-hit-rate is affected by the volume of traffic. Our installation was exposed to 2.5 terabyte of content and the above graph relates to this content volume. What byte-hit-rate can be expected on a less crowded network? What can be expected from an installation handling heavier traffic?

Ideally, this issue would be addressed by filtering the trace data for increasingly large subsets of clients. However in a dynamic IP environment it is difficult to keep track of client identity, in addition to ethical questions of user privacy violation raised by such practices. Therefore an alternative approach was chosen.

it is reasonable that this limitation has little effect, and that this computation defines the theoretical limit on savings for the given traffic.

A graph similar to the one presented above was generated at various intervals of our testing. Each graph relates to the volume of traffic accumulated from startup to a progressively advanced phase of testing. The graph in *Figure 8* combines them all.

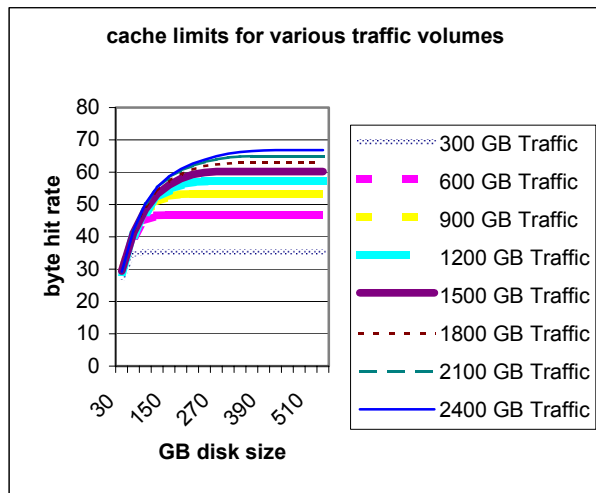


Figure 8. Cache Limits for Various Traffic Volumes

If a vertical line is run through all the graphs, we form a new graph that, for a given disk size, (the size being the location of the horizontal line on the x-axis) plots the byte-hit-rate as a function of traffic volume. The graph in *Figure 9* was produced by running the line at a large x-axis value, reflecting an infinite disk size. Hence the graph specifies the maximum hit rate achievable for each value of traffic volume.

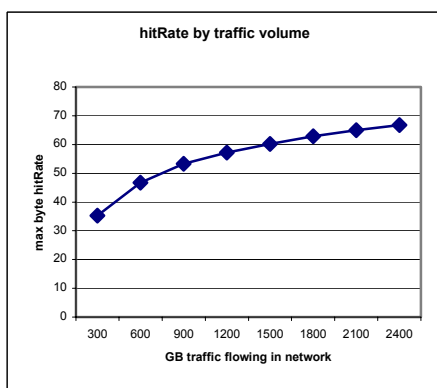


Figure 9. Hit Rate by Traffic Volume

A relatively high traffic volume is required in order to generate a high hit rate. The graph also indicates that cache steady state has not yet been reached, even after experiencing a 2.4 terabytes of traffic. If we assume the graph to be logarithmic in first order, we may perform an extrapolation as to how the caching would increase in a more central installation. The graph in *Figure 10*

extrapolates the original graph 16 steps forward using a logarithmic function of first order.

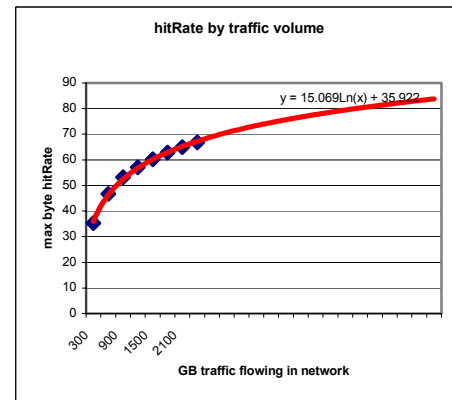


Figure 10. Hit Rate by Traffic Volume

This graph forecasts that over a more congested line experiencing three times more traffic, a byte-hit-rate of above 80% might be measured^V.

3.3.2 Empirical Verification

We implemented a caching algorithm designed to fit the unique properties of P2P traffic. Our server implemented this caching on live traffic at the same installation. In order to verify the ability to save bandwidth by caching P2P traffic, we analyzed the bandwidth measurements performed routinely by the ISP provider. Specifically, the ISP routinely measures the bandwidth entering our server and the bandwidth transmitted by it. The difference between these measurements consists of the data that our server transmitted from the cache as ‘hits’, and therefore defines the bandwidth savings generated by caching. The graph in *Figure 11*^{VI}, produced by the ISP, shows the incoming (plotted as a line) and outgoing (plotted as an area) traffic as measured during a three-week phase of our installation. The graph measures a savings of approximately 50% throughout this period. Interestingly, the hit rate, which measures the percent of downloads that were served from cache, irrespective of the file size, was measured in the range of 32% - 36%. The significantly higher byte-hit-rate compared with the hit rate indicates that the sizes of highly popular files tend to be larger than the sizes of less-popular files.

^V Alternatively, the graph may indicate that the byte-hit-rate of our installation will increase over time, as the traffic our server is exposed to triples. However, the fact that files lose their popularity as public consumer tastes shift, might down play the increase of traffic if spread out over long duration.

^{VI} The peaks charted here represent daytime and nighttime habits of usage.

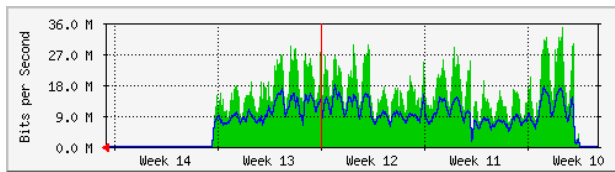


Figure 11. Traffic during Installation Period

Focusing on a unique two-hour duration later on in the installation period in which bandwidth increased to almost 60 Mbps, the ISP measured over 60% savings as seen in the graph in *Figure 12*.

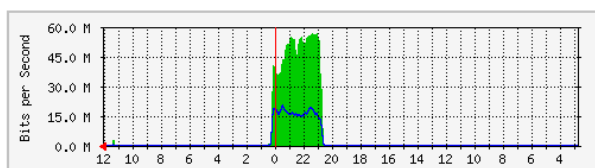


Figure 12. Unique Focus

The improvement in the savings seen during this traffic surge validates the findings summarized in Figures 8 and 9, which predict an increase in bandwidth savings as traffic volume increases.

Finally, in an additional testing session conducted after the one described here, the impact of adding disk space on the caching performance was tested. A fourth disk was added to the server, enlarging the space dedicated for caching from 200 GB to 300 GB. An approximate 10% increase in the byte-hit-rate was measured, reaching a stable 67%.

4 CONCLUSIONS

This paper derives the following understanding of the nature of P2P traffic: FastTrack-based programs form the majority of P2P traffic. The majority of the files transmitted over the P2P network are audio files. Yet, the audio files lose a certain amount of significance when focusing on the 100 most popular files of which the majority are video and application files. Furthermore, the portion of the total traffic that is generated by audio files is clearly smaller than the traffic generated by video and application files.

This paper concludes that P2P traffic transmitted over an ISP link is highly repetitive and consequently responds well to caching. Our analysis of the traffic computed a 67% byte-hit-rate which compares favorably with web caching hit rates known to be in the range of 30% to 60%. Further, it was shown that the disk space required for effective caching of P2P traffic is small enough to be practical – close to maximal caching is attained with 200 GB disk space. Finally our analysis concludes that the byte-hit-rate computed at our installation correlated with the traffic

volume, indicating that a higher byte-hit-rate may be expected on links with a heavier traffic load.

Taken together, these findings indicate that the caching of P2P traffic is an effective and desired means for coping with the bandwidth drain generated by the increase of P2P traffic.

5 ACKNOWLEDGMENTS

We wish to mention the significant QA help offered by Gili Orkaby in testing our server.

We are very grateful to Rachel Kuhr for her dedicated help in phrasing and editing this paper.

We are indebted to Don Rahmlow for his encouragement and suggestions.

Many thanks to Amos Rosenboim of Internet Gold, for assisting in the installation of the server.

6 REFERENCES

- [1] Gwendolyn Mariano, “Schools Declare File-Swapping Truce”, CNET News.com, March 14 2002, <http://news.com.com/2100-1023-859705.html>
- [2] Bradley M. Duska, David Marwood, Michael J. Feeley, “The Measured Access Characteristics of World-Wide-Web Client Proxy Caches”, Proceedings of the USENIX Symposium on Internet Technologies and Systems, Monterey CA, December 1997
- [3] Kelly Truelove, Andrew Chasin, “Morpheus Out of the Underworld”, The O’Reilly Network, 2001, <http://www.oreillynet.com/pub/a/p2p/2001/07/02/morpheus.html>
- [4] Terence Kelly, “Thin-Client Web Access Patterns: Measurements from a Cache-Busting Proxy”, Proceedings of the Sixth International Workshop on Web Caching and Content Distribution, 2001
- [5] Terence Kelly, Jeffrey Mogul, “Aliasing on the World Wide Web: Prevalence and Performance Implications”, WWW2002
- [6] Cade Metz, “Swap Meet”, PC Magazine, March 12 2002, http://www.pcmag.com/print_article/0,3048,a=23848,00.asp
- [7] Michael Rabinovich and Oliver Spatscheck, “WEB Caching and Replication”, page 90, Addison-Wesley, 2002