

## 1.0 INTRODUCTION

The size of content on the Internet is interesting for a variety of reasons. Previous studies [5] have shown that the sizes of files can explain some basic properties of Internet traffic like self-similarity. Several studies have been performed to measure the sizes of files that a user requests. These studies have been implemented by profiling several parts of the network, the server [2], the client [3] and caches [4] for file access patterns. In this paper we present *Arachne*, a suite of measurement tools to quantify content on the Internet. We step forward with two approaches to measuring sizes of content. First, we sample equal amounts of files based on different MIME types like images, text, application etc. This gives us insight into the typical sizes for the different MIME types. Second, we employ Google's Zeitgeist [7], a lexicon of popular searches on the search engine, to represent popular queries on the Internet. Generating searches of these keywords, we retrieve websites that are on the first page that is indexed by Google and measure the sizes of files on those websites. This exercise aims to characterize the distribution of content on the Internet, both popular and MIME-type agnostic.

The statistical properties of traffic on the WWW are well documented in previous studies. Mathematical explanations [1] and physical explanations [5] (via file sizes) have been offered. Understanding these properties goes a long way in engineering the Internet to live upto the desirable properties of generality, reliability, congestion control and fairness in some cases. The trouble however, is that the Internet has grown by several orders of magnitude in the past decade and learning whether the assumptions still hold, is an interesting exercise. Apart from the Internet growing, content distribution has also found special channels like peer-to-peer systems[11]. This segregation might yield a different view of the world upon closer scrutiny. The ubiquity of the WWW has also spawned the need to better architect software architectures[10]. Benchmarking efforts like SpecWEB99 [9] base their load generation on previous results about file sizes. We propose that a fresh look at content sizes warrants to enable three things:

- Understanding traffic characteristics and verifying long-held assumptions
- Building better software architectures
- Building more realistic benchmarks for evaluation

This paper makes a set of contributions. First we observe that the *average* size of content is growing when compared to previously measured averages. Second, we observe that as far as the *popular content* is concerned, our method of measuring client generated access yields a distribution that is not heavy-tailed like previous distributions have been. We hypothesize that this is because traffic on the Internet is segregated. What we mean by this is, content like music downloads, which are still pretty popular, have found channels like peer-to-peer networks and this might explain why bigger files are infrequent in our set of measured files.

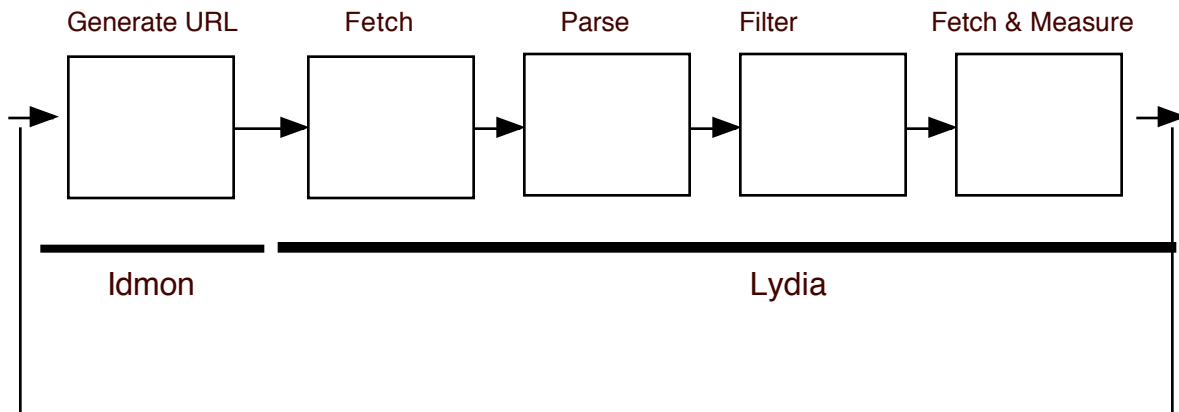
The rest of the paper is organized as follows: Section 2 presented related work. This is followed by an overview of the design of Arachne and lessons learned while building it. Section 4 goes on to talk about evaluation and the results we gather. Section 5 presents some concluding remarks followed by a generous overview of future work in section 6.

## 2.0 RELATED WORK

Several studies have been performed in the area of measuring the size of content on the Internet. Two sets of studies were conducted at Boston University that traced the client access behavior for popular content [3,6] in 1995 and 1998. Sedan et. al instrumented Mosaic browsers at Intel to study the client access patterns again[2]. Arlitt et. al. [4] studied server access patterns at four locations to get an insight into sizes of files that were popularly accessed. Leland et. al [1] were the first to point out that the nature of traffic in LAN's did not follow the Poisson distribution that was assumed for telephone networks. This was followed up by a number of studies that looked at the nature of traffic on various network configurations. Crovella [5] offered the first explanation of self-similarity in network traffic based on file sizes. We are not aware of studies past 1998 that have looked at file sizes on the Internet.

## 3.0 DESIGN OVERVIEW

Arachne has two major components, Idmon and Lydia. Figure 1 illustrates Arachne's pipeline. The design goals included modularity and separation based on function.



**Figure 1 -- Arachne's Pipeline**

Arachne currently uses two modes to generate URLs to sample content from, can parse forty different kinds of files and is a suite of nearly one hundred scripts. The scripts for Arachne were

---

written using a combination of C, PERL and MATLAB. The tool is almost automatic and works by active probing. For the purposes of this study, we limit ourselves to client-side studies.

### 3.1 Idmon, the list generator

When deciding which pages to sample for content, a number of methods can be used. The method that is used is chosen based on the answer to the following questions:

- Which part are we monitoring? The client or the server?
- What do we want to measure? Average sizes of different types of files or popular content?
- Are we exhaustively measuring anything? Exhaustive IP-based searches or measuring the sizes of all files served up on a single website?

In this study, Idmon was written to do client side measurements. We measure average sizes of different MIME types and also sizes of popular content based on Google's Zeitgeist. We do not pick websites at random to sample. We also do not measure how many files, *on the whole*, are being served up on a particular server. Therefore, we are not exhaustively measuring anything at this point.

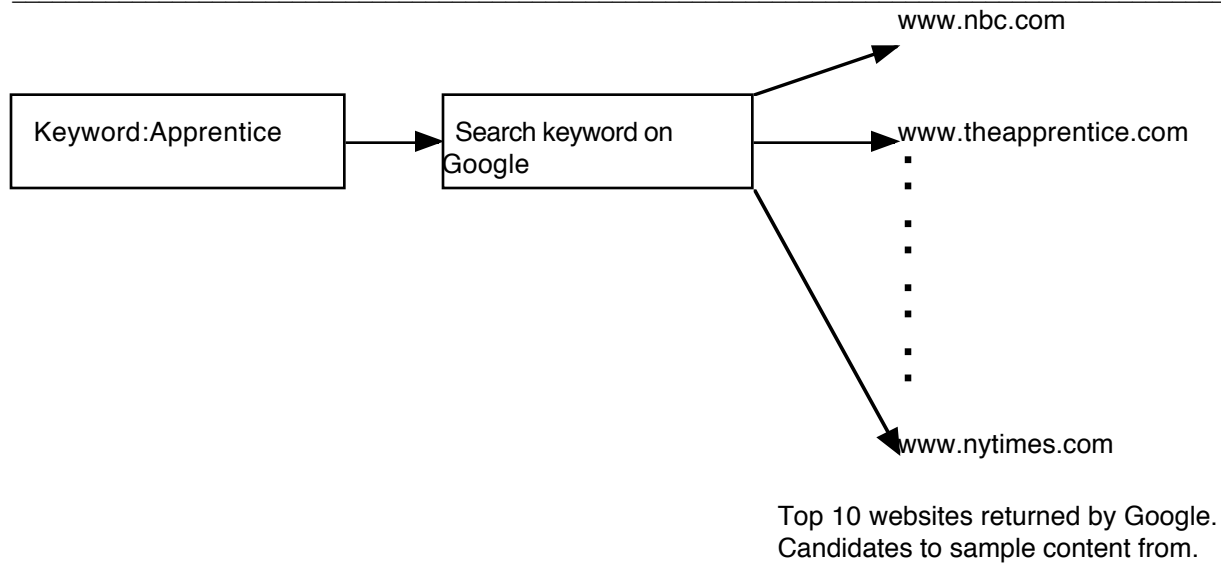
In order to measure *average file sizes* of different MIME types, Idmon is responsible for generating searches on a search engine, www.ask.com, looking for specific kinds of files. For example, to search for the MIME type, *audio*, Idmon generates a search for *Free ring tones*. To find sizes of MIME type, *video*, Idmon generates a search for *animations, Apple movie trailers*, and, *visualization*. We list the association of categories and searches launched for those categories in Table 1.

Category	Query
HTML	All
Image	April Fool, Movie Trailers
Sound	Free Ring tones, Gnutella file-size greps
Video	Apple movie trailers, animations, visualization
Application/Formatted Document	Citeseer keywords:processor, rendering

**Table 1 -- Association of categories and queries generated**

The goal was to generate about 1000 sample files in each category. Due to time constraints, we were unable to generate more than 250 samples for the category, applications. Therefore, we performed our averaging over nearly 250 samples in each category even though in some cases (like images and sound) we were able to generate all the 1000 samples. From the queries, we pick as many pages as we need to meet this goal, from the results of the query based on these keywords. Our choice of search engines for this step was [www.ask.com](http://www.ask.com) because, Google had some problems with firewalls which we hope to explain at a later date. Using the other search engine avoided some time overhead in generating the initial list of URLs because we were able to automate it better.

In order to measure *popular file sizes*, Idmon uses Google's Zeitgeist [8] to generate queries based on popular keywords over a particular time period. Please refer to Appendix A.1 for a list of all the keywords that we generated queries for. After generating the queries, Idmon picks the top ten websites as the location to profile. In some cases, some websites in the top ten list (basically, the first index page) were not responding or the links were broken. In these cases, Idmon goes beyond the first index page. Figure 2 shows the procedure for the keyword *Apprentice*.



**Figure 2 -- popular content searches from Google's Zeitgeist**

### 3.2 Lydia, the analyst

Lydia has a couple of interesting functions. This component is responsible for fetching the pages from the list that Idmon generated. Besides this, Lydia also parses the webpages for interesting files, filters the sample to get the available files and measures their sizes. To fetch pages, we simply use *wget()*, a standard UNIX utility to fetch websites and all other remote files. To parse the websites, there are a set of regular expressions that are defined for the various extensions (html,shtml,gif,jpg,mp3,mov). Our scripts simply pick these file types, record their locations (absolute or relative) and create sample lists. When we say that Lydia *filters* the samples, what this means is that we only fetch the files whose locations are easily resolved. There are times when the file names of various types of content is specified as a relative path. In these cases, figuring out the actual remote address of the file, is sometimes a nontrivial task. The most obvious remote path for absolute files would be the the name of the URL that the remote file was found on, appended with the name of the remote file. But, we have observed some cases where this is not so. Given that we do not know the directory structure of these web servers, we cannot postulate what the exact location might be. We just leave the problematic cases out. The final step of the analyst is to fetch the content instances (using *wget()* again) and measuring the size of the file. These measurements are stored for later mathematical characterization. This has been discussed in section 4. Table 2 shows the MIME types that Lydia parses for.

Category	File Extensions
HTML	HTML,html,HTM,htm,shtml
Image	gif,jpg,xbm,GIF,jpeg,gif89,tif,tiff
Sound	au,wav,snd,lha,mid,midi,mp3,WAV,rmj,RMJ
Video	mpg,mpeg,mov,MOV,avi,mp2
App/Formatted Doc.	ps,pdf,ps.gz,ps.Z,dvi

**Table 2 -- File types parsed by Lydia**

### 3.3 Properties and lessons

The properties that are desirable for a suite of tools that are used to measure content on the Internet are as follows. First, the tool must be automatic. Most of the functions of Idmon and Lydia (Sections 3.2, 3.3) lend themselves easily to implementations in C and PERL scripts. Second, the tool must be capable of logging errors encountered in the phases where the actual fetch occurs. This is important because it gives us the actual numbers of files that we are dealing with. Third, the tool itself must be nonintrusive. At the current time, this is not a concern because our user requests that fetch URLs are limited to a couple of thousand requests. If this number increases however, we would like to investigate how we can keep the probing to a minimum. Another important property of such a tool is that it should sample the web accurately for the category of *popular content*. In our case, we used a widely used search engine to help us pick what might be popular sites and conduct measurements on those sites. This reduces to client-side measurement because we assume that millions of Google users are generating these queries. We have seen however that these queries are limited to queries for popular media figures, top news items and a few other categories discussed in section 4. An interesting result is that most queries in the 2004 timeframe were not for music downloads. This probably implies that this traffic can be better observed on peer-to-peer networks. What this also means is that our method of generating popular queries is not impervious to faults (because audio is certainly popular). Server-side measurements may ameliorate some of these problems, but the question there is: *what collection of servers serve popular content?*

Some lessons that fall out of writing this set of tools are that true automation is certainly a good

goal but, perhaps slight overkill for the timeframe we were working in. Most of Idmon is manual. Most of Lydia is automated but, when the tool crashes, some manual operator has to go in and restart the tool.

## 4.0 EVALUATION

There are a variety of characteristics about the content on the Internet that might be interesting to us. This study focuses on the one of two aspects that were pointed out in [6], the size distribution of unique files. The set of unique files is that in which a file appears exactly once. In this section, we first layout the experimental parameters. Second, we tabulate a comparison of our study with previous studies conducted in 1995 and 1998. We then go on to list the methods that we used to infer various statistical properties of our samples. Fourth, we present graphs that arise from using these methods.

### 4.1 EXPERIMENTAL PARAMETERS

Our experiments were conducted on a Linux box running RedHat 9. We were running on machines with a Pentium III, 800 Mhz processor with 256 KB L1s. The system's buffer size was around 13MB and it had a 157MB cache.

### 4.2 METHODS USED

The output that the analyst produces is a set of file sizes for the various types of files. We would like to determine the following things about the data that we have at hand.

Parameters: maximum, minimum, median, mean and standard deviation of the sizes that were observed.

Distributional characteristics: Is the distribution still heavy tailed?

To determine the parameters, we simply used MATLAB on the various size sets. To determine distributional characteristics, we again borrow from [6]. We describe two methods here.

1. We used a log-log complementary distribution plot (LLCD) to figure out whether or not the size set has a heavy tail.
2. We used simple histograms or CDF plots to pare down the set of candidate models that we could use to fit the size set.

### 4.3 RESULTS

We start off by presenting the samples that we used for our measurements. We go on to present the distributional characteristics for unique files transferred in our studies.

### 4.3.1 SAMPLE COLLECTION

For the *average file sizes* of various MIME types, we measured an average of 250 samples of each MIME type that we were interested in. We refer the reader to Table 2 that talked about the various queries that were used to generate the pages from where this content was measured. Table 3 presents the label and the query it stands for.

Label	Keyword
AF	April Fool
TR	Apple Movie Trailers
a/V	Animations/Visualization
GT	Gnutella file greps
CS(P)	Citeseer query for keyword <i>processor</i>
CS(R)	Citeseer query for keyword <i>rendering</i>
RT	Ring tones

**Table 3 -- Labels and Keywords**

The reason we chose these keywords was mostly to get a good collection of samples across different file extensions. For example, a/V yields video clips that are mostly of the extensions mpeg, which are rarely found on movie trailer sites. We chose keywords *processor*, and *rendering*, for documents because rendering papers tend to have more dense figures in them than do processor papers. We chose *ring tones* because those are usually encoded as midi files. Table 4 presents exact sample sizes that we used to average sizes of files labeled by the query that generated the sample.

MIME type: extensions	Label: Number of Samples
Sound: mp3, ogg	GT:218 RT: 63
Sound: mid, midi, wav	AF: 1 RT: 9
Video: mov	TR: 265 a/V:0
Video: mpg, mpeg	TR:0 a/V:25
Image: gif	AF:75 TR:75
Image: jpg	AF:40 TR:110
App/FD: pdf	CS(P):0 CS(R):130
App/FD:ps	CS(P):28 CS(R):10
App/FD: ps.Z	CS((P):18 CS(R):1
App/FD: ps.gz	CS(P):39 CS(R):9

**Table 4 -- Labelled sample sizes for average size study**

For the *popular file sizes* study, we simply present the sample sizes for HTML and images. Table 5 simply states the week for which the data was gathered and the HTML and image sample sizes. In the cases where we did find other classes of content, they have not been mentioned because they were terribly infrequent. They were taken into account in our measurements however.

Week	HTML (Unique)	HTML(Repeats)	Images(Unique)	Images(Repeats)
April 18	1825	643	1010	1698
April 12	756	699	981	1043
April 5	600	780	734	1321

**Table 5 -- Sample sizes for the popular file size study**

### 4.3.2 UNIQUE FILES

In this section, we present the statistical parameters for the unique files that were transferred for the two kinds of measurements that we performed.

Statistic	AV04
Sample Size	1000
Minimum	42
Maximum	25322066
Mean	604746.5
Median	8294.5
Standard Deviation	282950

**Table 6 -- Parameters for average size study**

File Type	Number of files	Avg. Size (in bytes)
Image	290	6295.5
Sound	290	2161376.2
Video	290	273333.8
App/Formatted	235	829557.5

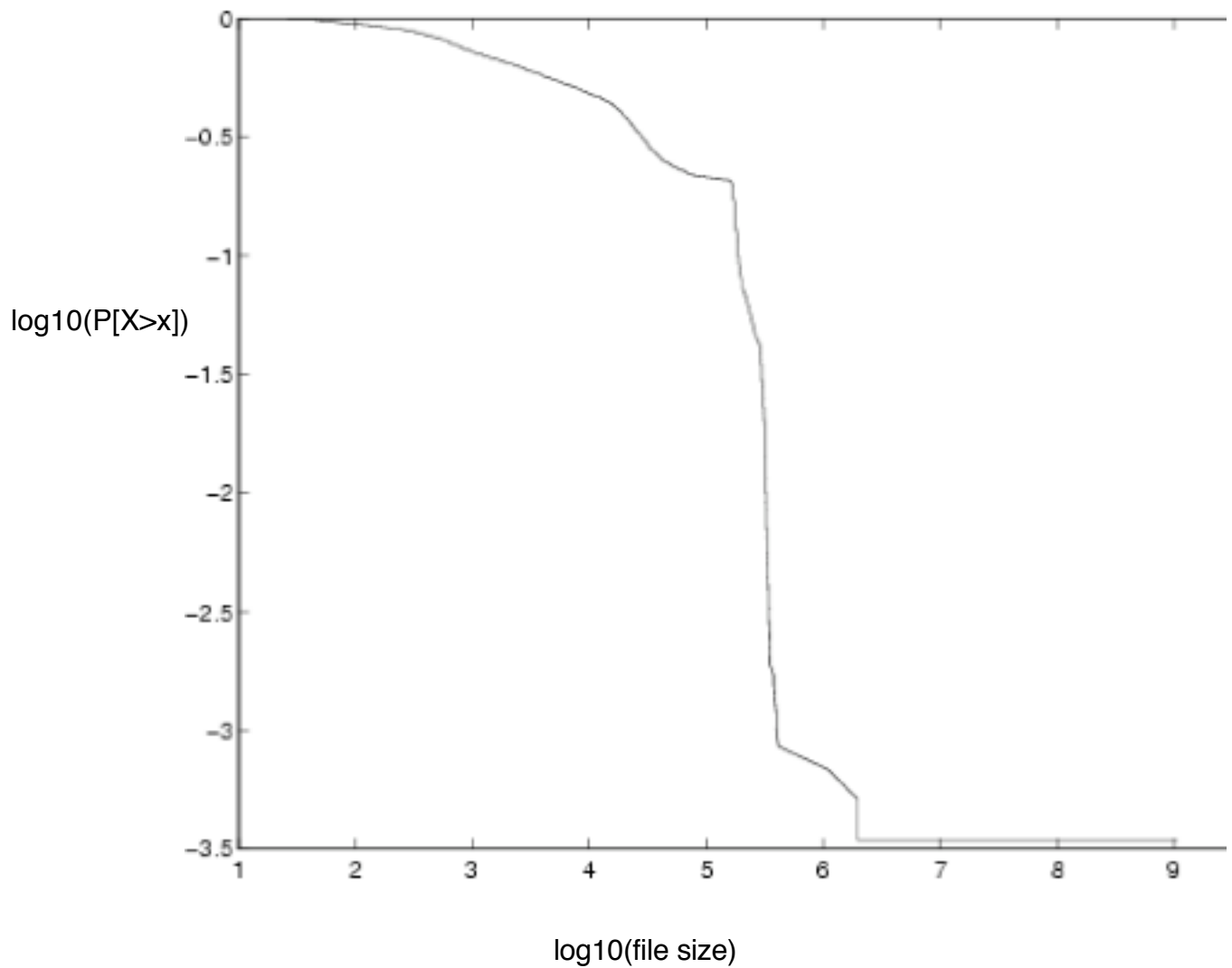
**Table 7 -- Average file sizes for different MIME types for average size study**

Statistic	AV04(P)
Sample Size	5855
Minimum	26
Maximum	1989388
Mean	54818.3
Median	8484
Standard Deviation	95705.2

**Table 8 -- Parameters for popular size study**

## LLCD plot

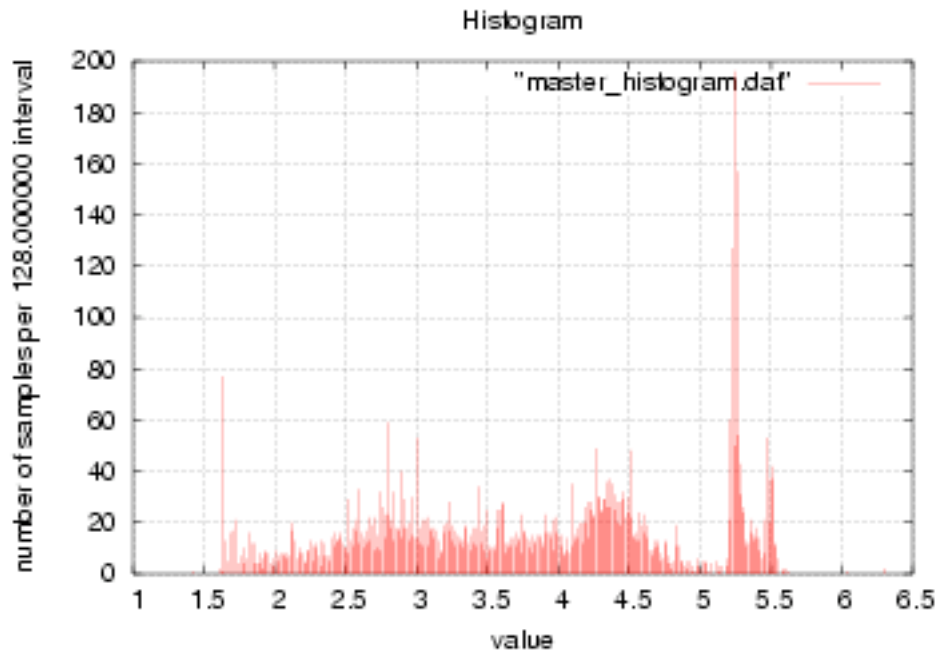
Figure 3 shows a log-log complementary distribution plot for the file sizes we gathered from the popular sizes study. For heavy-tailed distributions, we expect to see a gradually decreasing curve, like the one presented in [3]. Instead, we see a sharp drop in the plot. This indicates that for the samples observed, the file sizes may not follow a heavy-tailed distribution.



**Figure 3 -- LLCD of popular size study**

## Histogram

The following is a histogram (in 128 byte bins) of file sizes. This might indicate what model to use to characterize the distribution. We leave the actual modeling for future work.



## 4.4 COMPARISON

In this section we compare our results to previous studies. Table 9 compares average file sizes with the study done in 1995. We see that average file sizes are on a rise.

MIME Type	Avg. Size (W95)	Avg. Size (AV04)
Image	13900	6293.5
Sound	551000	2161376.2
Video	792000	1119600
Application	358000	829557.5

**Table 9 -- Comparison of average sizes**

Statistic	W98	AV04(P)
Sample Size	41,049	<b>5855</b>
Minimum	1	26
Maximum	4,092,928	1989388
Mean	7,609	54818.3
Median	2,769	8484
Std. Deviation	33,306	95705.2

**Table 10 -- Comparison of statistical parameters in client-based analysis of popular content**

## 5.0 CONCLUSIONS

Based on our results presented in the previous section, we draw a couple of conclusions.

1. Average sizes of files agnostic of MIME type is on the rise.
2. Based on the LLCDD we generated, the file sizes for the popular content do not seem to adhere as strictly to the heavy-tailed distributions previously observed.
3. Our sample set is pretty small compared to the previous studies but, given that we are using a popular search engine to do the sampling of popular content for us, this should be a reasonable set.
4. It seems like while large files certainly exist in the popular file size study, the tail drops off quickly. We hypothesize that the traffic on the Internet is segregated. What we mean by this is, popular content like music, full length movies etc. have found different channels to go through. While we are certain that these kinds of files are still popular, they don't seem to form the majority of requests performed on Google.

## 6.0 SOPHISMATA

There are several exciting avenues for future work. First, extending the sample set beyond a month of Google's popular hits would verify that the LLCDD produced is not premature. Second, figuring out the transfer times of the files would mean a trivial extension to Lydia. Third, measuring the ratio of files served up on a particular server to the actual files popularly requested was done in [3]. In order to repeat this study in our environment, we propose using htdig [7] an open-source crawler. What this tool will allow us to do is to crawl any arbitrary site for content and index all of it. Using this index, we could come up with a ratio of popular content to content served. Analyzing what models best describe the new file size distribution would also be a good exercise.

## BIBLIOGRAPHY

1. Will Leland, Murad Taqqu, Walter Willinger, and Daniel Wilson, *On the Self-Similar Nature of Ethernet Traffic (Extended Version)*, IEEE/ACM Transactions on Networking, Vol. 2, No. 1, pp. 1-15, February 1994.
2. Jeff Sedayao. "Mosaic Will Kill My Network!" -- Studying Network Traffic Patterns of Mosaic Use. In *Electronic Proceedings of the Second World Wide Web Conference '94: Mosaic and the Web*, Chicago, Illinois, October 1994.
3. Carlos Cunha, Azer Bestavros, and Mark Crovella, "Characteristics of WWW Client-based Traces". Technical Report TR-95-010, Boston University, CS Dept, Boston, MA 02215, April, 1995.
4. M. Arlitt and C. Williamson. *Web Server Workload Characterization: The Search for*

- 
- Invariants*. In Proceedings of the International Conference on Measurement and Modeling of Computer Systems, May 1996.
5. Mark E. Crovella and Azer Bestavros, "`Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes," in *IEEE/ACM Transactions on Networking*, 5(6):835--846, December 1997.
  6. P. Barford, A. Bestavros, A. Bradley, and M. E. Crovella, "*Changes in Web Client Access Patterns: Characteristics and Caching Implications*," in *World Wide Web*, Special Issue on Characterization and Performance Evaluation, Vol. 2, pp. 15-28, 1999.
  7. <http://www.htdig.org/>
  8. <http://www.google.com/press/zeitgeist.html>
  9. [www.specbench.org](http://www.specbench.org)
  10. James Larus and Michael Parkes. "*Using Cohort Scheduling to Enhance Server Performance*" Microsoft Research Technical Report MSR-TR-2001-39, March 2001.
  11. [www.gnutella.com](http://www.gnutella.com)