Article can be accessed online at http://www.publishingindia.com

Machine Learning Based Architecture for Rule Establishment of Web Proxy Server

P.S. Banerjee*, G. Sahoo, Umesh Prasad*****

Abstract

In present scenario Internet has become an integral part of every ones life, as many services like mail, news, chat are available and huge amounts of information on almost any subject is available. However, in most cases the bandwidth to connect to the Internet is limited. It needs to be used efficiently and more importantly productively. Generally, bandwidth is distributed among groups of users based on some policy constraints. However, it turns out that the users do not always use the entire allocated bandwidth at all times. Also, some times they need more bandwidth than the bandwidth allocated to them. Ideally, productive usage should be preferred over unproductive usage when bandwidth is scarce. But when it is abundant then any kind of use can be permitted provided it is in consonance with policy. The bandwidth usage patterns of users vary with time of the day, time of the year and requirements. So there is a need for dynamic allocation of bandwidth that satisfies the requirements of the users, manages variable usage and is consistent with administrative usage policy.

Internet usage is varied and in the context of an institution or organization an administrator would like to maximize productive usage. There is, therefore, a need to implement control access policies, which prevents unproductive use but at the same time does not, to the extent possible, impose censorship. Squid proxy server is a full-featured web proxy, which increases the efficiency of the Internet link by providing caching and proxy services. Squid provides many mechanisms to set access control policies. However, deciding which policies to implement requires experimentation and usage statistics that must be processed to obtain useful data. The proposed architecture elaborated in this paper is based on machine learning to determine policies depending on the content of current URLs being visited. The main component in this architecture is the Squid traffic Analyzer, which classifies the traffic and generates URL lists. These URL lists are used in formulating access policies. The concept of delay priority will also be introduced which gives more options to system administrators in setting policies for bandwidth management. As Squid allows HTTP tunneling, it forms a loophole for strict policy management. In this paper the proxy tunneling in Squid has also been considered and some possible solutions to this problem will also be suggested.

Keywords: Web Proxy, Machine-Learning, Network Traffic, Meta Data

1. Introduction

Squid proxy server is generally employed at the end of a local network and the starting point of an Internet link. Typically the allowed bandwidth on Internet links is limited, ranging from kbps to Mbps. So there is a need for better utilization of this resource and for finding better policies. In the context of limited bandwidth an organization would like to use it for productive purposes like data of academic relevance in an academic institution. However, Internet usage ranges from research,

* Assistant Professor, Department of Computer Science & Engineering, Jaypee University of Engineering & Technology, Guna, Madhya Pradesh, India. E-mail: partha1010@gmail.com

** Professor & HOD, Department of Information Technology & MCA, B.I.T. Mesra, Ranchi, Jharkhand, India. E-mail: gsahoo@bitmesra.ac.in

*** Assistant Professor, Department of Computer Science and Engineering, B.I.T. Extension Centre, Lalpur, Ranchi, Jharkhand, India. E-mail: umeshprasad_bit@rediffmail.com

Figure 1: Phases in Squid Traffic Classification

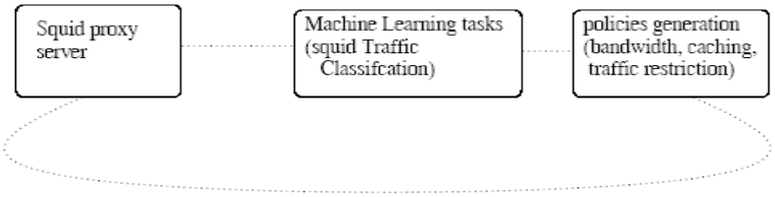
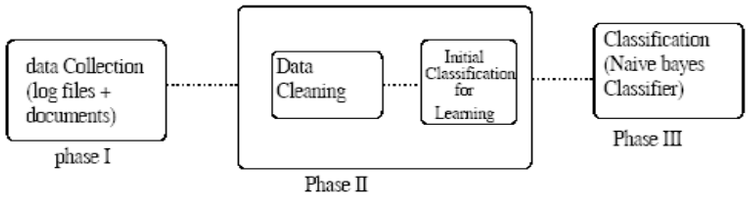


Figure 2: Architecture



news, entertainment, sports, general information to downloading movies, music and pornography. So there is a need for traffic analysis to find the shape of the traffic in terms of content and help system administrators in setting policies for shaping the traffic. Traffic analysis can categorize URLs and IP Addresses into categories like academic, sports, music etc. This categorization can be used to allocate bandwidth according to policies adopted by the organization. Squid has a set of mechanisms for implementing policies based on many configuration parameters like source, destination, domains, time, and regular expression on URLs etc. Squid allows HTTP tunneling to support SSL and TLS. But, these provide a loophole for overcoming policies.

Generally, users use mass-downloaders or some other tools, which use HTTP tunneling [6] [3] to over come access control policies. So there is a need to find a way for enforcing policy strictly. The aim of proposed architecture is to find out and implement environment and usage sensitive policies in Squid proxy server that leads to better utilization of bandwidth and other resources. In proposed architecture machine-learning techniques has been adopted to study the current environment and usage patterns. System administrators can then use this information to decide and implement policy by making use of the existing tools available in Squid to control access.

2. System Design

From an abstract point of view the problem can be seen as improving the performance of a proxy server (Squid) in a given environment. For the measure of performance two parameters will be used-

a. Overall better utilization of bandwidth within the constraints set for each delay pool

b. Traffic control based on content (to model policy decisions).

This problem will be treated in two parts. To improve bandwidth utilization, the concept of priority in delay pools will be introduced and to control traffic based on content the problem will be formulated as a machine learning task and learn the appropriate class for a URL from given set of classes. The log data and cache documents of Squid Proxy have been used for training purposes in the learning task (classifying the documents). The learning itself is done by using a naive Bayes Classifier on the collected data. Once classification is done, policies based on these categories are used to tune the Squid proxy server. This process is continuous and can be applied till performance is improved to the required levels.

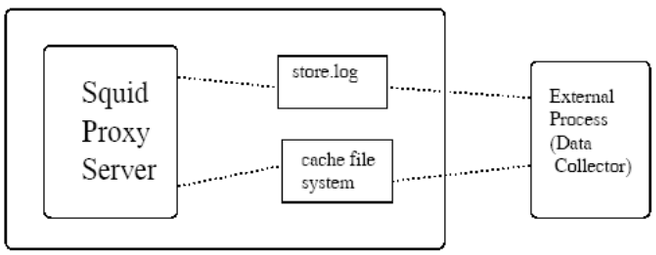
The complete classification of Squid traffic is done in three different phases: data collection, learning and classification. In the data collection phase the log data

and documents from Squid proxy, which characterize the traffic, have been collected. The learning phase includes some user/manual interaction. It includes two steps namely *data cleaning* and *learning*.

2.1 Data Collection

The log files of Squid and documents from the cache file system have been collected. The collection of documents was done with an external process. A Squid proxy server has been configured and enabled store.log, access.log and cache.log. As the sizes of the log files are in the range of 250MB - 500 MB, the log each day has been rotated. Squid renames the old log files with integer extensions. The log files have been zipped to solve the problem of space on the system.When users access Internet pages/data, Squid caches it based on the caching strategy and lifetime of the pages/data. Squid stores such cached pages in the *cachefs* and writes a log entry in *store.log*. An external process ("Data Collector") have been designed which uses the store.log and *cachefs* to collect the documents. The store. log record gives information about the "location" of the pages/data in the *cachefs*. The external process, Data Collector copies the pages/data to the local directory or disk space. The schematic diagram of data collector is as shown in following figure:-

Figure 3: Data Collector



2.2. Classification of Traffic

The traffic through Squid proxy can be classified into different categories: news, sports, entertainment, music, science and technology, chat, movies, academic and pornography. The traffic is characterized by the log data and documents collected through the external process - Data Collector. The naive Bayesian Classification algorithms have been used on these documents and classify them. The category of a document is identified using the content in the document or page. For example, if a document contains research/educational information it is considered to be in the academic category.

2.3. Bayesian Classification Algorithm

The Bayesian Classification Algorithm performs better than almost all other competing algorithms [1] in document classification. The naive Bayes classifier can be applied to learning tasks where each instance *x* is described by a conjunction of attribute values and where the target function *f (x)* can take on any value from some finite set *V.* A set of training examples of the target function is provided by the tuple of attribute values $< a_1, a_2, ..., a_n >$. In this case the most probable category for an unknown URL (document) also called the *maximum a posteriori* hypothesis is expected, which we represent by V_{MAP}. The Bayesian approach to classifying a new instance is to assign the most probable target value, V_{MAP}, given the attribute values $< a_1, a_2, .., a_n >$ that describe the instance.

Bayes theorem is the basis of Bayesian learning methods because it provides a way to calculate the posterior probability *P*(h|D), from the prior probability *P*(h), together with *P*(D) and *P*(D|h). Here *P*(h|D*)* is the probability *h* holds given training data D (similarly, *P*(D|h). *P(h)* is the *a priori* probability that *h* holds and *P(D)* the a priori probability that data D will be observed. Bayes theorem gives a formula to calculate *P*(h|D*)* provided we know the other three probabilities.

Bayes Theorem

$$P(h|D) = \frac{P(D|h)p(h)}{P(D)} \quad (1)$$

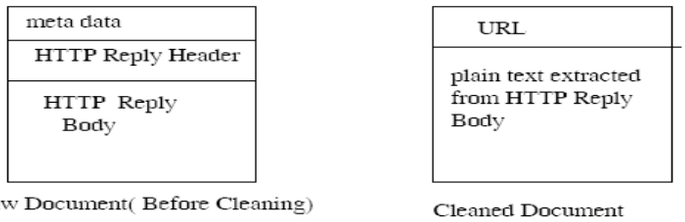
With the help of Bayes theorem
V_{MAP} can be rewritten as

$$V_{MAP} = arg\max_{v_j \in V} \frac{P(a_1, a_2, ..., a_n|v_j)P(v_j)}{P(a_1, a_2, ..., a_n)}$$

$$= arg\max_{v_j \in v} P(a_1, a_2, ..., a_n|v_j)P(v_j)$$

In above equation the two terms can be evaluated based on the training data. It is easy to estimate each of *P (Vj)* simply by counting the frequency with which each target value *Vj* occurs in the training data. However, estimating

Figure 4: Document Structure



the different $P(a_1, a_2, ..., a_n)$ terms in this fashion is not feasible unless a very large set of training data is available. The problem is that the number of possible instances times the number of possible target values. Therefore, there is a need to see every instance in the instance space many times in order to obtain reliable estimates. The naive Bayes classifier makes one simplifying assumption: the attribute values are conditionally independent given the target value.

V_{MAP}

$$= arg \max_{v_j \in V} P(v_j | a_1, a_2, ..., a_n) \quad (3)$$

In other words, the assumption is that given the target value of the instance, the probability of observing the conjunction *al, a2, ..., an* is just the product of the probabilities for the individual attributes: $P(a_1, a_2, ..., a_n) = \Pi_i P(a_i|v_j)$. Substituting this in the above equation, we get the following equation.

Naive Bayes classifier:

$$V_{NB} = arg \max_{v_j \in V} P(v_j) \prod_i P(a_i|v_j) \quad (4)$$

where V_{NB} denotes the target value output by the naive Bayes Classifier.

To summarize, the naive Bayes learning method involves a learning step in which the various $P(V_j)$ and $P(a_i|v_j)$ are estimated, based on their frequencies over training data. The set of these estimates corresponds to the learned hypothesis. This hypothesis is then used to classify the new instance by applying the equation of naive Bayes classifier.

2.4. Document Cleaning

As the documents collected are not directly amenable for learning, the documents need to be cleaned. Currently, only the documents with the following content types will be handled: text/html, text/plain, Application/*, etc. - essentially documents whose content has text in it. The structure of a typical document is as follows: meta data, HTTP Reply Headers, HTTP Reply Body. After that the plain text from the *HTTP Reply Body* will be extracted. The information of interest is the URL and plain text. This plain text forms the content of our cleaned document. A cleaned document consists of the URL and the plain text extracted from *HTTP Reply Body.*

2.5. Initial Categorization for Learning

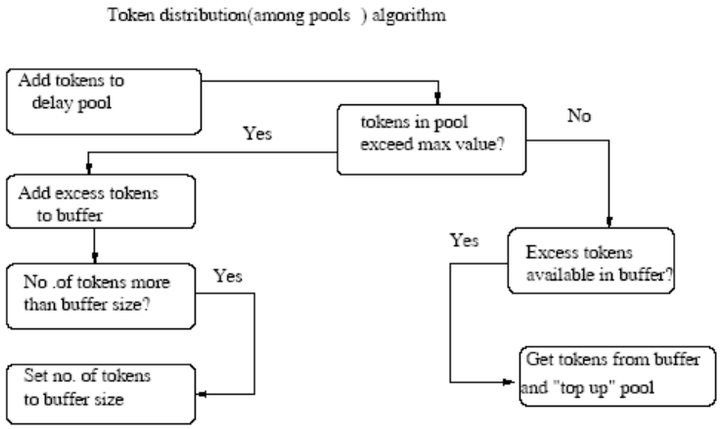
In a Bayes classifier the value of *P* (D|h) needs to be calculated. This means all the data used in learning must be pre-classified. Since the volume of training data is quite large, instead of classifying all documents manually, following technique has been used to come up with class labels for the training documents. For each class or category, keywords for the URL and for the content will be identified. The logic is if a keyword is present in the URL or if a subset of keywords is present in the document the category of the document is identified as the category associated with the keyword(s). Initial categorization is done using URL, document content and keywords. For verification purposes some classified documents have been randomly selected and verified it manually.

Algorithm for Initial Categorization

For each category (i) find some key words which potentially represents that category. For example, for category education,"edu", "research" etc forms the set of key words.

For each document D if (keyword of category C, present in the URL) category of document D is C. Else if (maximum subset of keywords of category C belongs to the document D) category of document D is C. Else category of document D is *not known.* Verify categories of some randomly chosen documents.

Figure 5: Transferring Bandwidth among Pools



2.6. Classification Algorithm

The next phase in Squid traffic analysis is classification. This is accomplished by the simple *naive Bayes classifier* [1]. The *naive Bayes classifier* performs well on many document classification problems and produces good results. The cleaned documents along with their category are taken as training data for the naive Bayes classifier. For classifying other (that is documents not in the training set), this Bayes classifier is used to get the most likely category for the document. The words in a document are treated as the attributes for that document.

2.7. Delay Pools

The delay pools concept, which is a bandwidth management scheme, has some drawbacks. The major one is that the unused bandwidth of a Delay pool is wasted and cannot be utilized by other Delay pools. Dynamic Delaypools [18] solve this problem and allow utilization of the full bandwidth.The dynamic delay pools implementation is speedy and simple. The algorithm for bandwidth adjustment works in two stages.

- Allocation of Bandwidth to users per delaypool.
- Transfer of bandwidth from one pool to another.

The algorithm uses the *aggregate* number of tokens in the delay pool as the criterion for whether the user's link bandwidth is fully utilized or it skips the networks settings entirely. If number of tokens exceeds a cutoff value (presently 10% the aggregate delay pool size), indicating that there is some capacity being under-utilized, each individual user's bandwidth allocation is increased by a fraction. If the number of tokens is less than the cutoff value, indicating that the requests are backing up and link bandwidth is fully utilized, an individual user's bandwidth allocation is reduced by a fraction.

2.8. Transferring Bandwidth Among Delay Pools

Using a single delay pool allows bandwidth to be shared fairly among users. However, in many instances, the bandwidth needs to be distributed unequally among different classes of users. This can be easily achieved by having several delay pools with different parameters. However, when several delay pools operate on the same server, some pools do not use all their allocated bandwidth while others are saturated, so the bandwidth, which is not being utilized, is allowed to be transferred to another delay pool as depicted in figure 6.

The above algorithm checks the level of tokens in each delay pool after adding the token allocation available to that particular pool. If level of tokens is greater than the maximum allowed value, then excess tokens are transferred to a buffer and if a delay pool is encountered whose tokens are less than maximum permissible value, the tokens in the buffer are used to "top up" the pool to its maximum value or until the tokens in the buffer are exhausted. Accordingly, excess tokens that would be utilized by lightly loaded delay pools are distributed among the pools that are using their token allocation fully. The maximum size of the buffer is set to a few seconds worth of tokens.

2.9. Delay Priority

A simple modification to the above dynamic delaypools has been proposed and introduces the concept of priority among the delay pools. As the unused bandwidth of delay pools is distributed among other delay pools, the priority imposes an order on the distribution of unused bandwidth among the other delay pools. The delay pool with highest priority gets the unused bandwidth before the lesser priority delay pools.The usage of such a priority is illustrated with a simple example. Suppose we have defined delay pools which accept different types of traffic based on the category say C_1, C_2,...,C_n. If we want the delay pool associated with C_4 to get maximum bandwidth, then the priority setting of C_4 to a high value can solve the problem.

2.10. Policy Generation

The classification of traffic results in categories and a set of documents that belong to that category. As documents are related to the URLs, one can directly link the URLs with the categories.

(category, Set of Documents)→ (category, set of URLs), as each Doc(i) is related to URL(i).

The set of URLs along with the category forms the basis for the formation of policies. An administrator can regulate traffic by suitably changing the policies. In Educational Institutes, administrators want most bandwidth should be used for academic purposes rather than movies, music other unrelated stuff. So this category names, URL lists help in forming the policies.

2.11. Bandwidth

The traffic analysis helps in formulating bandwidth policies. The steps for creating policies based on URL lists of category is as mentioned below-

1. Design some named ACL elements based on a regular expression of URLs using A CL types of form *dstdomain _ regex, dstdomain, url_reg ex, urlpath _ regex.*
2. Design ACL lists like *http_ access, delay_ access* based on these named ACL elements.
3. Define Delay Pool parameters based on the categories and define the Maximum Bandwidth and restore rate of each Delay Pool.
4. Use our *Delay Priority* concept to define the priority for each category, so that the unused bandwidth by that category is distributed according to the priority of other categories.
5. The ACL List *http_ access* can be used to allow/ deny the traffic.

2.12. Caching

The traffic analysis results also help in tuning the Squid to cache hot pages and deny caching of unwanted traffic. The ACL Lists like *no_cache* can be used along with the named ACL elements defined above to tune the caching. For categories like *news* and *academic* the pages should be always cached and the pages of porn content should be denied.

3. Results & Performance Evaluation

The selection of performance generally depends on the problem, its understanding, and variables to be optimized. Here the aim was to improve the percentage of bandwidth for certain classes and reduce the bandwidth available to other classes. More formally (category, bandwidthusage) tuples form the measure. However, it should be noted that in this case breaks down into two independent measures since categorization of URLs is handled by the naive Bayes classifier and bandwidth utilization is managed independently by priority based dynamic delay pools. So first the traffic has been classified, and then formulates policies based on the classification results and finally set these policies in the Squid proxy. Performance is measured independently for classification and for bandwidth management.

Certain keywords has been taken for each category to identify the category of the document in initial categorization. Some of these keywords are as mentioned below-News: news, vaartha, timesofindia, cnn, eenadu, hindu ; Sports: sport, cricket, football, tennis, chess, athletic; Entertainment: greetings, dance, hero , celebrity, marriage, jokes; Music: music, song, audio, mp3, ram; Science & technology: Science, technology, breakthrough. For this category, manually some documents have been collected and added to this category for learning; Chat:

Table 1: Performance Measure (Two Categories)

Content	files downloaded			bandwidth(KBytes)		
	before	after	change	before	after	change
porncontent	184204	48737	-73.54%	981596	1,86905	80.9%
other documents	919923	979920	6%	11082749	12192349	1.5%

chat, msg, messenger, notify, pager; Movie: movie, video, dat, asf, asx, mplayer, realplayer; Academic: research, .pdf, .ps, .gz,.bz, download, package, edu ,university,exam , academic; Pornography: nude, porn, sex, adult, xxx, lesbian, blowjob, fuck, naked.The classification of traffic results in URL lists for each category. A simple method is to use the http _ access, url_ regex _path to design policy. ac1 url_path_regex porn_url porn_url_lists # porn_ ul_ lists contains the URLs found in the classification. http_ access deny par _url. # deny traffic for this category. These rules provide simple policies to deny some kinds of porn content through the proxy. But one should find more reasonable access controls.The keywords with maximum frequency in the URL lists has been found out and form another acl element to complement the above.

The policies have been set based on the URL lists for categories. The log files and documents has been collected after setting the policies. Some of results in terms of performance measure for a day by taking average of a period of 7 days in as mentioned below.

Deny Pornography Content

The policies have been set to deny pornographic content using keywords and list of URLs. The bandwidth usage and number of files downloaded that belong to this category before and after setting the policies has been compared Using Access.log:

The statistics *(average values per day) for* a period of a week, before and after setting policies. Downloading a page like www.google.com results in sending several requests for text, for all images in that page, banners etc. A file download corresponds to a single file, so for go ogle it will result in multiple downloads.The URL entries in the access.log have been classified as follows:

Table 2: Performance Measure (Documents)

Content	files downloaded		
	before	after	change
porncontent	4126	837	-79.7%
other documents	25874	29163	12.7%

Table 3: Bandwidth Distribution (All categories)

Category	files downloaded			bandwidth		
	before	after	change	before	after	change
news	121888	189773	55.5%	15,45,548	1856768	20.1%
academic	245656	235877	-3.08%	28,87,556	27,67,778	-4.14%
chat	211839	200754	-5.2%	70,566	72,567	2.83%
music	36785	25785	-29%	10,45,334	856777	18.03%
movie	7622	10476	37.4%	3,55,765	378992	6.03%
sport	103023	230445	123.68%	13,53,467	15,67844	15.86%
entertainment	36056	31546	-12.50%	5,77,694	7,46654	29.32%
science technology	7554	8973	18.74%	1,07,772	1,87556	74.03%
pornography	184204	48737	-78.76%	9,81,596	1,86,905	80.9%
unclassifed	21446	18335	-	85,798	96552	-

- The frequent keywords in URL lists(output of naive Bayes classifier)for pornography.
- Keywords used in the initial categorization.

Thirty thousand documents have been collected for a period of a day before and after setting policies. The naive Bayesian classifier have been applied the on these documents. The characteristic of the documents is that; the content of the file is understandable like text/ html or text/plain & the size of the document is at least 8Kbytes(before cleaning). So the bandwidth here will not signify much, as the images, song files like .mp3, .rm files etc. has not been considered.

All Categories: Here the data collected from the access.log file has been classified into all nine categories. The high change in sports category is due to frequent visits to cricket and tennis. The cache hit rate for sport category is high.

Bayesian Learning Classification

Initially 30,000 cleaned documents have been collected. 5000 documents have been trained using initial categorization. For each category we used around 200 - 400 documents in training. For pornography, chat and academic categories we used around 800 -1000 documents for training. Then 25000 documents have been classified using the Bayes classifier and found the frequency of URL along with documents. A different document of the same URL which belongs to the same category to trained data has also been added. These way exactly 1000 training documents were obtained for each category. 20,000 more documents have been collected and classified them using the naive Bayesian classifier. The accuracy of classification for the categories is as follows: The verification of results done with the keywords and manually. When the number of domains is less the verification has been performed mannualy. For the category like academic, the presence of the keyword .edu is good enough.

- Chat - 97% . Verification of this category is done with keywords in URL.
- Pornography - 98%. The total number of domains we got around 150 and these sites are frequently accessed. But some home pages in geocites and yahoo groups also found.
- Aacademic: 98%. The verification of this category is also done with the keywords in URL and documents.

For other categories the accuracy was as follows:

- Music: 92%. In the remaining 8% documents, the naive Bayes classifier identifying the category as music, but the probability of V_{MAP} is not greater than 0.5 value.
- Movie: 88.4%.
- Entertainment: 86%.
- News: 94%. The number of vocabulary words for this category is high.

HTTP Tunneling Policies

The HTTP Tunneling Policies results are in the form of entries in access.log and not in terms of percentages since it is not possible to measure such percentages.

4. Conclusion

Through this paper an approach has been presented for better management of access and bandwidth to an internet link through a proxy server. Access control is done by categorizing URLs into different static categories using a naive Bayes classifier. Bandwidth control is done using priority based dynamic delay pools. Some methods to inhibit HTTP tunneling have also been tested. The initial results are encouraging. For example pornographic content reduced significantly and the bandwidth was well utilized for other categories. Policies using priority based dynamic delay were implemented and bandwidth utilization improved. Log traces show that the methods to inhibit HTTP tunnelling they do manage to stop tunneling requests.

References

1. Mitchell, T. M. (1997). *Machine Learning,* McGraw-Hill.
2. Tanenbaum, S. (1999). *Computer Networks.* (3rd ed.) Delhi: Prentice-Hall of India Pvt. Ltd.
3. Luotonen, A. (1999). Tunneling TCP Based Protocols through Web Proxy Servers. Retrieved from www.webcache.Com/Writings/lnternet- Drafts/ draft-luotonen-web-proxy-tunneling-01.txt
4. Squid Proxy Server. Retrieved from www.squid-cache.org
5. Squid Frequently asked Questions. Retrieved from www.squid-cache.org/Doc/FAQj*

6. Squid Configuration File: Squid Configuration. Retrieved from http://www.squid-cache.org/Doc/config/
7. Squid configuration Manual. Retrieved from www.visolve.comjsquid24s1/contents.html
8. david@luyer.net
9. Upgrading to TLS within HTTP/1.1. (1997). Retrieved from *www.ietf.org/rfc2817.txt*
10. HTTP-Tunnel Corporation- Networking Products for Corporate Communications. Retrieved from *www.http-tunnel.com*
11. Cutting Edge Web Applications. Retrieved from *www.totalrc.com*
12. Squid Cache Logfile Analysis Scripts. Retrieved from *www.squid-cache.org/Scripts*
13. Calamaris: Log Analyzer. Retrieved from *http://cord.de/tools/calamaris/*
14. Webalizer: Log Analyzer. Retrieved from *http://mrunix.netjwebalizr*
15. Cache Digest Specification- Version 5. Retrieved from www.sequid-cache.*org/ CacheDigest/cachedigest-v5. Txt*
16. Hyper Text Transfer Protocol - HTTP/1.1. Retrieved from *www.ietf org/rfc/rfc2616.txt*
17. Rousskov, A. & Soloviev, V. (1999). A Performance Study of the Squid Proxy on HTTP/l.0. *World Wide Web*, June, 2(1-2), 47-67.
18. Chamara Gunaratne, Gihan Dias (University of Moratuwa) *Using DynamicDelay Pools for Bandwidth Management* URL: *www.2002.iwcw.org/*
19. Squid Programmers Guide. Retrieved from *www.squid-cache.org/Prog-Guide/prog-guide.html*
20. Lang, K. (1995). *NewsWeeder: Learning to Filter Netnews*. In Priedits & Russel (eds.), Proceedings of 12th International conference on machine learning (pp. 331-339). San Francisco: Morgann Kaufmann Publishers.
21. Rish, I. (2001). *An Empirical Study of the Naive Bayes Classifier.* IJCAI-01 workshop on Empirical Methods in AI.
22. Rish, I., Hellerstein, J. & Jayram, T. S. (2001). *An Analysis of Data Characteristics that Affect Naive Bayes Performance*. IBM Technical Report RC21993, 2001.
23. Rish, I. (2000). *Advances in Bayesian Learning.* A short tutorial presented at ICAI'2000. Las Vegas.