Working Paper Series
ISSN 1170-487X

The LRU*WWW Proxy Cache Document Replacement Algorithm

by Chung-yi Chang, Tony McGregor and Geoffrey Holmes

Working Paper 99/9
June 1999

Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand

The LRU* WWW proxy cache document replacement algorithm

Chung-yi Chang, Department of Information Technology, The Waikato Polytechnic, Hamilton, New Zealand, itjlc@twp.ac.nz
Tony McGregor, Department of Computer Science, The University of Waikato, Hamilton, New Zealand, tonym@cs.waikato.ac.nz
Geoffrey Holmes, Department of Computer Science, The University of Waikato, Hamilton, New Zealand, geoff@cs.waikato.ac.nz

Abstract

Obtaining good performance from WWW proxy caches is critically dependent on the document replacement policy used by the proxy. This paper validates the work of other authors by reproducing their studies of proxy cache document replacement algorithms. From this basis a cross-trace study is mounted. This demonstrates that the performance of most document replacement algorithms is dependent on the type of workload that they are presented with. Finally we propose a new algorithm, LRU, that consistently performs well across all our traces.*

Topics: Networking and Communication

Keywords: WWW, caching, document replacement algorithm.

1. Introduction

Because of the rapid and sustained growth of the world wide web there is stress on many of the components of the web. Proxy caches improve the web's infrastructure by reducing the load on communications links, reducing latencies for web users and by helping to reduce the traffic to the most popular servers, many of which are heavily overloaded.

Web users in the Asia/Pacific region are impacted more strongly by these factors than those in the United States and Europe and also suffer the additional burden of funding the high cost of the very long distance communications channels required to connect their networks to those in the United States, which is at the heart of the world wide web. For these reasons Asia/Pacific communities have been particularly pro-active in the deployment of proxy caches.

As an example, consider the WWW usage of the University of Waikato. Departments at the University paid approximately $1.2m for Internet traffic in 1998. Most departments employ WWW proxy caches and although exact figures are not available, typical behaviour would indicate these departments saved approximately $600,000 through the use of proxy caches. These savings are in addition to greatly reduced latencies for documents served from the proxy cache. Typical NZ/US round trip times (RTTs) are in the order of 250-300ms[AMP99]. The

smallest web pages require five round trip times [DARPA81] which results in a minimum document fetch time of 1250-1500ms[AMP99]. Documents served locally from a proxy cache face around 3-5ms RTT giving a minimum document fetch time of 15-25ms. Further latency reductions occur if the document is served by a proxy cache instead of a heavily loaded web server.

Any real web cache has limited storage. When a new document, not currently stored at the cache, is fetched a decision must be made as to whether this new document should be stored at the cache and if so which document(s) should be removed from the cache to make space for it. Given a particular cache size the effectiveness of this *document replacement policy* at selecting those documents most likely to be reused has the single biggest impact on the performance of the cache.

A number of document replacement polices have been proposed in the literature or deployed in implemented proxy caches. These include:

- Least Recently Used (LRU),
- First In First Out (FIFO),
- Least Frequently Used (LFU),
- Least Frequently Used with Document Aging (LFU-Aging),
- The largest of log (base 2) of the document size rounded down to the nearest whole number with LRU used to select between ties (⌊Log_2(Size)⌋)
- LRU with minimal number of page replacements (LRU-MIN).

These algorithms are introduced in Section 2. Further details of these algorithms can be found in [Chang98] and the references contained there in.

Most authors who propose a document replacement policy study its behaviour using trace driven simulation. In most cases one or more real cache workloads are applied to a simulator that is executed with different document replacement polices.

This paper makes three contributions to the understanding of document replacement policies. First, we independently reproduce previous studies. This process, while tedious and time consuming, reveals that there are a large number of details that must be resolved to measure the performance of a document replacement policy. Different authors resolve these details in different ways reducing the comparability of their results. Reproducing earlier results also gives us confidence to proceed to the second and third goals described below.

Second, we perform a cross-trace study, using the traces from each study (where available) and our own traces with each document replacement policy. This reveals the high sensitivity of the algorithms to the nature of the workload. Some algorithms that perform well on one workload do not perform well on another.

Finally, we propose a new algorithm LRU* which performs well across a wide range of traces.

The remainder of this paper is organised as follows: Section 2 describes the document replacement polices studied here including LRU*. Section 3 describes the traces used to generate

the workload for the simulations and outlines their main characteristics. Section 4 presents the results of the simulations. The paper ends with a discussion of the results and some suggestions for further work.

2. Document replacement polices

The document replacement algorithms used in this study are FIFO, LRU, LRU-MIN, LFU, LFU-Aging, Size, Log2(Size) and LRU*, random and infinite. The details of implementing these algorithms in the simulator are described below. Each algorithm is based on sorting the documents stored in the cache into an abstract list of documents.

FIFO: The FIFO algorithm maintains a sorted list of cached documents based on document entry times. When there is insufficient space in the cache, the document(s) located at the tail of the sorted list is replaced. More than one document will be removed if the new document is larger than the document at the tail of the list.

LRU: The LRU algorithm sorts cached documents by the latest access time. When a cache hit occurs the access time of the requested document is updated and it is moved to the head of the list. The least recently used document (located at the tail of the list) is the next to be replaced.

LRU-MIN: The LRU-MIN algorithm is similar to LRU. Like LRU, LRU-MIN maintains a sorted list of cached documents based on the time the document was last used. The difference between LRU and LRU-MIN is the method of selecting the candidate for replacement. When the cache needs to replace a document it searches from the tail of the sorted list. The first document whose size is larger than or equal to the size of the new document is removed. If all cached documents are smaller than the new document, the search is repeated looking for the first two documents greater than half the size of the new document. This process (of halving the size and doubling the number of documents to be removed) is repeated if large enough documents can still not be found for replacement.

LFU: This algorithm maintains a reference count for each cached document. All cached documents are sorted by reference count. Documents with the same reference count are sorted by recency. When a cache hit occurs, the reference count of the hit document is incremented by one and the documents are sorted using the updated reference count. When document replacement is needed, the document located at the tail of the sorted list (i.e. the document that has the smallest reference count and least recency) is removed.

LFU-Aging: The LFU-Aging algorithm sorts cached documents in the same way as LFU. The difference is that LFU-Aging additionally monitors the average of the reference counts of all cached documents. When the average is given over a given maximum number, the cache starts again. In our simulations, the maximum is set to 10 as suggested in [Arlit96].

Size: The SIZE algorithm sorts cached documents by size. Documents with the same size are sorted by recency. When there is insufficient space for caching the most recently requested document, the least recently used of the document with the largest size is replaced. More than one document will be replaced if necessary.

|_Log2(Size)_|: Floor(log2(Size)) sorts cached documents by the logarithm base 2 of the

document size. Documents with the same logarithm value are sorted by recency. When there is insufficient space for caching the most recently used document(s) with the largest logarithm value is replaced.

LRU*: Our algorithm, LRU*, is a mixture of LRU and LFU. In LRU*, cached documents are maintained in a sorted list based on document recency. When a cached document is hit, it is moved to the start of the list and its hit count is incremented by one.

When there is insufficient space for the most recently requested document, the hit count of the least recently used document is checked. If the value of hit count is zero, the document is discarded. Otherwise, the hit count is decreased by one, and the checked document is moved to the start of the sorted list. In other words, in LRU*, the least recently used document is not removed from the cache unless its hit count is zero. The cache will keep checking the hit count of the document(s) at the tail of the sorted list and removing the one(s) with hit count equal to zero until there is sufficient space for the most recently requested document.

In order to prevent a document accumulating too large a hit count, the maximum hit count of cached documents is limited. In this simulation, the maximum hit count is 5 because with a hit count of 5, a cached document can be placed at the start of the sorted list 5 times and this is considered large enough for a cached document. LRU* is intended to keep the hit document(s) in the cache longer (like LFU) and also to age cached documents more dynamically than LFU-Aging.

Random: Typical WWW cache hit rates are quite low, less than 50% in most cases, which is much lower than hit rates for most other type of caches found in computer systems. However even this low hit rate results in significant savings because there are large costs involved. We establish an upper bound on the performance of a proxy cache by simulating an infinite sized cache.

Infinite: Similarly, a working lower bound is determined by simulating a cache with a random document replacement algorithm. The boundaries give a performance envelope within which the performance of a document replacement algorithm can be judged.

Examination of Figures 2 and 3 and Table 2 will show that this envelope is normally less than 10%. This means that, under existing workloads, there is only limited room for improvement of proxy cache performance, especially given that current algorithms already claim much of this space. However the small gains possible by further improvement of cache behaviour are still valuable because a small improvement in the document replacement policy can generate significant savings in time and money.

3. The traces

Workloads used in this research were obtained from five sources. Two of them were downloaded from public FTP sites at Boston University [Cunha95] and Virginia Tech. [Williams96]. Another two were provided by the Information Technology Services at the University of Waikato which administers the cache installed on the New Zealand Internet Exchange (NZIX) [Neal96] as well as the Campus Cache of the University of Waikato. The remaining two workloads were collected from the cache of the University of Waikato Computer Science Department. The six workloads

are:

- Boston University (BU)
- Virginia Tech (VT)
- New Zealand Internet Exchange (NZIX)
- The University of Waikato - Campus Cache (UW)
- The University of Waikato, Department of Computer Science (CS)
- The University of Waikato, Department of Computer Science with duration from 1st July 1997 to 31 July 1997. (CS -JULY))

The VT workload actually contains five traces (U, C, G, BR, and BL). In this study, the trace Graduate (G), which contains the access logs of graduate students, was selected as a representative trace. The workloads NZIX, UW, CS and CS -JULY were collected from three proxy caches that form a proxy cache hierarchy as shown in Figure 1. Table 1 provides an overview of the six workloads by listing the collection period, size of log files, number of accesses and averaged number of accesses per day. The background of each workload is described below.

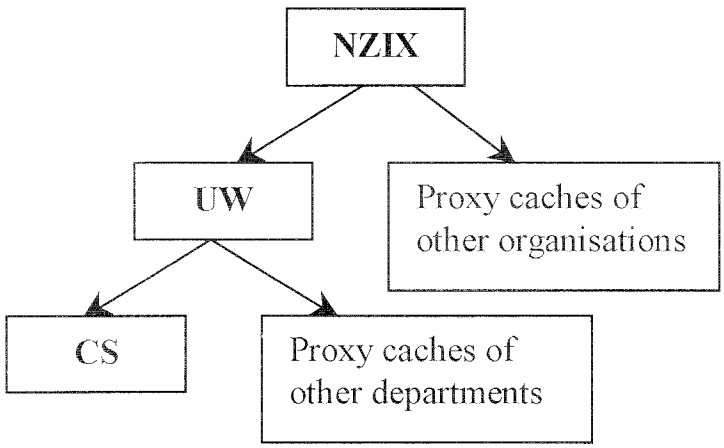

Figure 1. University of Waikato Cache Hierarchy

Boston University (BU): This trace was collected directly from end users through a modified Mosaic browser implemented by Cunha et al. [Cunah95]. Logs from each browser were written to local disk and subsequently transferred to a central repository. During the data collection period requests sent by 591 different users were traced. Users were primarily undergraduate students (558) and graduate students (42).

Virginia Tech. (VT): Five workloads were obtained from Virginia Tech: Undergraduate(U), Classroom(C), ,Graduate(G), Remote Client Backbone Accesses (BR), and Local Client Backbone Accesses (BL) [Williams96]. Workloads U, C and G were collected using a CERN proxy server. The other two were collected from backbones of local networks. The trace G collected from graduate computer science students, representing at least 25 users was selected as the representative VT workload because it was also used in [Abrams95] (in which the LRU-MIN

replacement algorithm was proposed).

New Zealand Internet Exchange (NZIX): The NZIX cache ran the NetCache3.x proxy server during the period when these traces were collected. Clients of the NZIX cache include seven education/research organisations and two Internet Server Providers (ISPs).

The University of Waikato - Campus Cache (UW): This trace was obtained from the internal campus cache that works as the parent of caches scattered through three different schools at the University of Waikato. The campus cache also runs NetCache3.x proxy server during data collection. Users of this campus cache include staff/faculty and both graduate and undergraduate students.

The University of Waikato, Department of Computer Science (CS): This workload was collected from the Computer Science Department cache which ran the Squid proxy server during data collection. Users of this cache include faculty/staff, graduate students and students who access the WWW from Unix laboratories of the Computer Science Department. It is estimated that this workload represents a population of at least 100 users.

The University of Waikato, Department of Computer Science (CS -JULY): This workload was collected from the same source as CS but with a collection period from July 1st 1997 to July 31 1997; that is the same collection period as NZIX and UW.

TRACES	BU	VT	NZIX	UW	CS -JULY	CS
Collection Period	21/Nov/94 - 28/Feb/95	20/Jan/95 - 07/Apr/95	01/July/97 - 31/July/97	01/July/97 - 31/July/97	01/July/97 - 31/July/97	04/Apr/97 - 05/Jan/98
Kb of Log - Files - compressed)	6,761	536	171,212	22,785	8,516	60,401
Number of Accesses	591,172	36,415	9,735,916	1,295,656	491,178	3,624,252
Avg. Accesses/Day	591	467	313,062	41,795	15,844	13,131

Table 1 Trace summary

4. Simulation parameters

There are a number of parameters to consider when conducting a proxy cache simulation. The parameters of this study include: cache size, identification and processing of dynamically generated scripts, processing of documents whose size has changed and simulation of document last modification requests. The choices for each of these are outlined below.

4.1 Cache size

The size of the cache affects the overall hit rate so it is not possible to directly compare results from different algorithms if different cache sizes are used, which is generally the case if different authors have undertaken the studies. One of the advantages of a cross trace study is that the same choices for this and other simulation parameters may be made.

As workloads grow the resources available for the cache also grow. This is due to both changing technology and more financial resources being available. In this study the cache size is 10% of the size of the workload.

4.2 Dynamic content

Some documents are generated at the time of the request and their content varies from request to request. As a consequence they are not normally cached. There is no foolproof method of identifying a document of this type from its URL so a heuristic must be used. The most common procedure, which is also used in this study, is to classify URLs which contain a question mark (`?`) or a `/cgi-bin/` component as dynamic document which should not be cached.

4.3 Updated documents

Sometimes, while a document is stored at the proxy cache, it is changed at the web server. Proxy caches can be configured to check that the document they have stored has the same content as that at the server when a new request arrives for the document and the copy currently held by the proxy has not been checked for some time. Clients may also request that the server confirm that the current copy is the most recent one. The HTTP protocol includes a facility to support this check without fetching the document. Proxy caches do not normally record this traffic in their logs and so it is not available to a simulation of WWW traffic.

In most cases, when the content of a document changes its size also changes. This can be detected by the simulator when a document that is already stored in the proxy is included in the log with a different size to the one stored in the proxy. This requires a new fetch of the entire document. The amount of checking traffic between the proxy and the web servers is primarily dependent on the way the proxy is configured and on the number of repeated requests in the workload but is not dependent on the document replacement policy. As a consequence, it is not of particular interest in this study and no allowance is made for this traffic in the results of the simulations. It should, however, be borne in mind if the absolute values of the traffic loadings are of interest, as they might be if, for example, a fixed sized communications channel is to be deployed.

5. Results

Some authors present their results in terms of the proportion of documents supplied by the cache, that is the *document hit rate*. Figure 2 shows the document hit rate of the ten algorithms studied against simulated time for the NZIX trace.

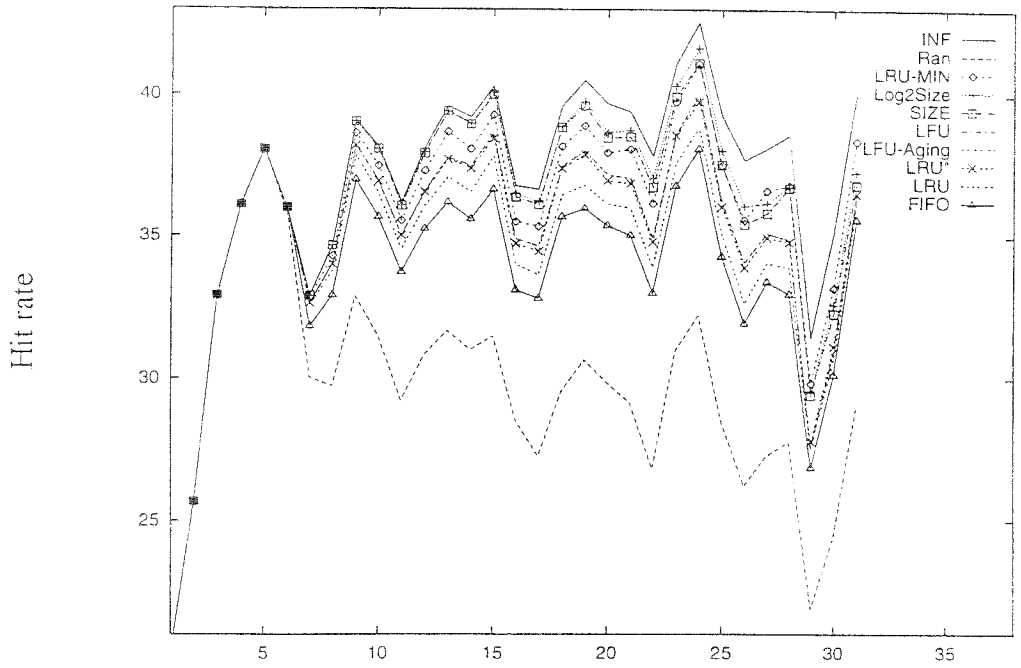

Figure 2. Daily Document Hit Rate for the NZIX Trace

If the goal of the cache is to reduce the total network traffic the document hit rate does not fully reflect the performance of the cache. Instead, a hit rate weighted by the documents size, or *byte hit rate* should be used. Figure 3 shows the byte hit rate for the NZIX trace against time.

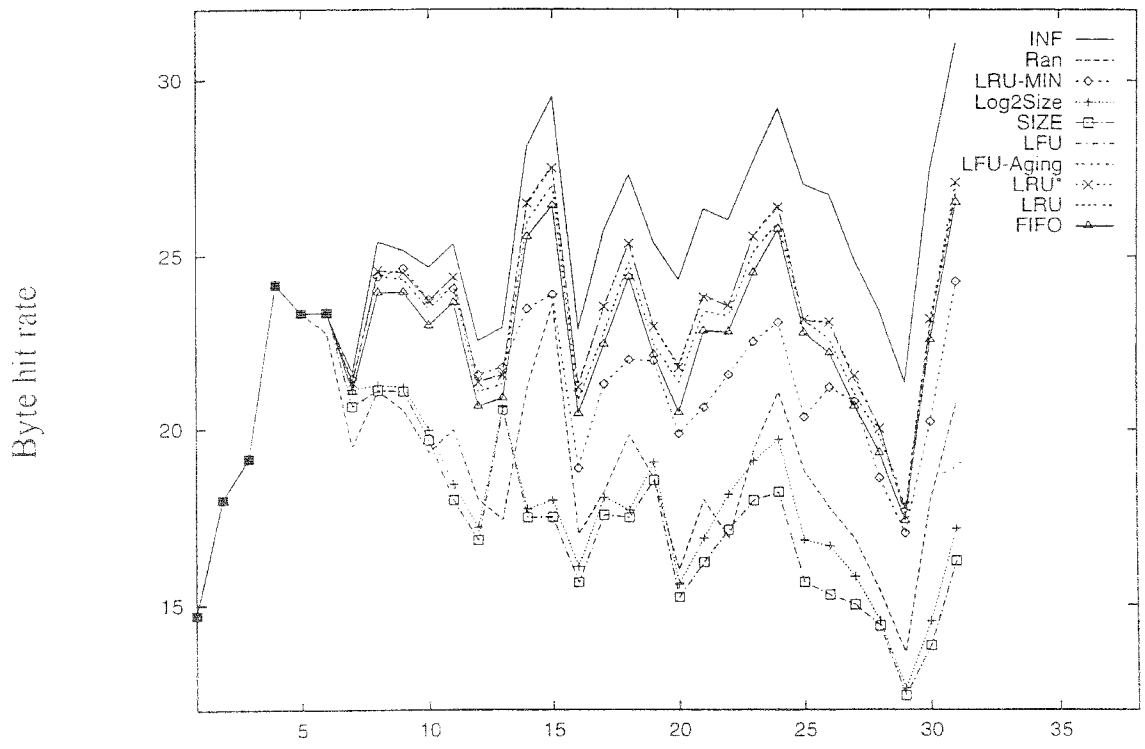

Figure 3. Daily Byte Hit Rate for the NZIX Trace

The document hit rate and weighted hit rate plots for other traces are similar in most respects to those for the NZIX trace but the relative placing of the algorithms varies. The overall performance of the ten algorithms against the six traces is shown in Table 2.

BU	Infinite	RAN	LRU	FIFO	LRU*	LFU	LFU Aging	log2 (Size)	LRU Min	Size
Avg HR	76.93	76.30	76.79	76.75	76.84	76.85	76.82	**76.91**	76.90	**76.91**
Avg BHR	47.88	46.33	47.55	47.47	47.55	**47.57**	47.56	46.99	47.45	46.92
VT	Infinite	RAN	LRU	FIFO	LRU*	LFU	LFU Aging	log2 (Size)	LRU Min	Size
Avg HR	35.59	27.26	29.25	28.00	30.51	30.67	30.67	35.05	**35.16**	35.13
Avg BHR	22.85	17.42	18.44	17.88	18.89	18.99	18.99	19.13	**20.76**	19.65
NZIX	Infinite	RAN	LRU	FIFO	LRU*	LFU	LFU Aging	log2 (Size)	LRU Min	Size
Avg HR	36.83	29.60	34.36	33.71	34.93	34.98	34.98	**36.15**	35.81	36.00
Avg BHR	24.67	19.00	22.58	22.24	**22.85**	22.83	22.83	18.29	21.49	17.82
UW	Infinite	RAN	LRU	FIFO	LRU*	LFU	LFU Aging	log2 (Size)	LRU Min	Size
Avg HR	29.39	19.38	24.99	24.00	25.65	25.92	25.92	**27.90**	27.86	27.57
Avg BHR	17.09	10.21	14.27	13.82	14.59	**14.67**	**14.67**	11.15	13.31	10.62
CS-JULY	Infinite	RAN	LRU	FIFO	LRU*	LFU	LFU Aging	log2 (Size)	LRU Min	Size
Avg BR	42.80	29.93	36.99	34.89	38.14	37.89	37.89	41.01	**41.10**	40.65
Avg BHR	26.67	17.96	22.55	21.78	**23.02**	22.75	22.75	19.04	21.94	18.29
CS	Infinite	RAN	LRU	FIFO	LRU*	LFU	LFU Aging	log2 (Size)	LRU Min	Size
Avg HR	48.92	38.04	46.84	45.77	47.24	46.65	46.65	47.83	**48.38**	47.31
Avg BHR	31.32	23.30	30.02	29.56	**30.14**	29.76	29.76	24.59	28.85	23.58

Table 2. Summary of Cross Trace Results.
The number(s) shown in **bold** type are the best daily averages under the upper bound.

From Table 2 it is evident that, with the exception of the BU trace, the upper bound for both hit rate and byte hit rate is quite low. This reflects the dynamic nature of most web traffic and the very large size of the information space. For many web caches it is byte hit rate that most accurately measures the caches performance. The LRU* algorithm is the best in the CS, CS -July and NZIX, is 0.08% from the best in the UW trace, 0.02% from the best in the BU trace and 1.87% from the best in the VT trace.

6. Conclusions and further work

The variability evident in the graphs of hit rate reveals the extent to which the performance of the algorithms is dependent on the day to day behaviour of a workload even when the overall character of the workload remains roughly the same. Table 2 shows that the performance of the algorithms is also dependent on the overall characteristics of the traces.

We suggest that, given the dependence of document replacement algorithms on the characteristics of the workload, that further research is required into the ways that workloads vary and the impact this has on proxy cache performance. Because most studies have been undertaken on traces from universities or research organisations, analysis of commercial traces is particularly important. In addition, ongoing work is required to ensure that algorithms that are shown to perform well at one time continue to do so as the workload on the web changes with time. The extent and impact of the increasing number of documents with dynamic content is an important example of the way the workload is changing and is particularly worth studying.

Finally, we suggest that algorithms like LRU*, that have good performance over a wide range of traces are particularly valuable because of the difficulty of knowing in advance what the characteristics of a workload are likely to be.

References

[Abrams95] M. Abrams, C.R. Standridge, G Abdulla, S Williams and E. Fox, Caching Proxies: Limitations and Potentials, *Proceedings of the Fourth International World Wide Web Conference*, Boston, Massachusetts, Dec 1995.

[Arlitt96] M Arlitt, A performance study of Internet Web servers, *M.Sc. Thesis, University of Saskatchewan* June 1996

[AMP99] A McGregor, The NLANR AMP active measurement program, *Project web pages: http://amp.nlanr.net/active/*

[Chang98] C-y Chang, A cross-trace simulative study of client-side web caching. *Masters Thesis, The University of Waikato, Hamilton, New Zealand*, July 1988.

[Cunha95] C Cunha, A Bestavros and M Ceovella, Characteristics of WWW client-based traces, *Technical report BU-CS-95-10, Boston University Computer Science Department*, July 1995.

[DARPA81] Transmission control protocol *RFC793, DARPA Internet Program*, September 1981, http://www.faqs.org/rfcs/rfc793.html

[Neal96] D Neal, The Harvest Object Cache in New Zealand, *Proceedings of the Fifth*

International World Wide Web Conference", Paris, France, May 1996.

[Willams96] S William, M Abrams, C R Standridge, G Abdulla and E A Fox, Removal policies in network caches for World Wide Web documents, *Proceedings of the ACM SIGCOMM Stanford, CA, Aug 1996.*