Modelling and Performance Analysis of Cache Networks

George Bilchev[1], Ian Marshall[2], Chris Roadknight[3], and Sverrir Olafsson[4]

June 18, 1999

Abstract

This paper develops and implements a World Wide Web cache infrastructure model which is to be used for analysis of features that are otherwise difficult to get from existing log data or for evaluation of non-existing cache scenarios. A prominent feature of our model that differentiates it from other similar models is its dynamical aspect, which allows for the investigation of temporal features. Using the model we verify and quantify observations made from real log data and provide a more comprehensive picture of the caching processes that take place behind the scenes. We also show how the model can be used to assess the economic viability of the caching solution.

1. Introduction

Proxy caching has become an established technique for enabling effective file delivery within the World Wide Web architecture [Abr95][Bae97a]. The addition of file caching agents adds many positive features including robustness (by distributing files more widely), a possible reduction in total bandwidth requirements (by moving popular files near to the clients) and a reduction in pressure on origin servers, especially on those serving popular files. Understanding precise costs and benefits of inserting caches into the network is a highly desirable goal for network management and design.

To gain an understanding of what affects a cache's behaviour and performance it has been essential to analyse behaviour of existing WWW caches currently in operation [Roa98][Mar98][Arl96]. This analysis gives us some information about the inter-

[1] BT Laboratories, Martlesham Heath, Ispwich, Suffolk IP5 3RE.
Email: george.bilchev@bt.com
[2] BT Laboratories, Martlesham Heath, Ispwich, Suffolk IP5 3RE.
Email: marshall@drake.bt.co.uk
[3] BT Laboratories, Martlesham Heath, Ispwich, Suffolk IP5 3RE.
Email: roadknic@drake.bt.co.uk
[4] BT Laboratories, Martlesham Heath, Ispwich, Suffolk IP5 3RE.
Email: sverrir.olafsson@bt.com

relationships of cache metrics and possible causes of observed behaviour [Roa99] but only covers caches in existing locations, serving existing communities. It is therefore highly desirable to be able to model cache behaviour so that non-existing cache scenarios can be evaluated. A cache enabled WWW modelling toolbox would undoubtedly be of use to network planners but also to many Internet researchers looking for a simple, flexible model to test theories with.

In this paper we develop a WWW cache model that is easy to use, cheap to implement and fast to simulate. The model only requires a few simple input values and yet is realistic enough to verify observed data from real caches. We believe the model will be of particular interest to operator technical staff who work under short time scales.

2. The Model

A schematic representation of a single proxy cache as seen in the simulation toolbox is shown in fig. 1. The proxy can accept input from a number of user communities. Each user community is modelled by daily activity pattern (fig. 2a) and popularity statistics (fig 2b).

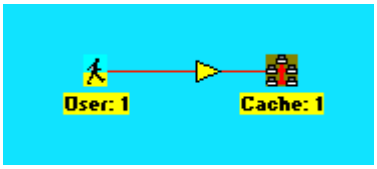


Fig. 1. A schematic representation of a single proxy cache as seen in the simulation toolbox. The cache is fed in by one user community annotated as "User: 1".

The daily activity pattern consists of an underlying trend and a stochastic component. To model the underlying trend we suggest using a 'superposition' of periodic functions:

$$y_i^{\text{trend}}(t) = \max\left\{a_i + b_i \sin(2\pi c_i \frac{t}{T} + d_i), 0\right\}$$
$$y^{\text{trend}}(t) = \max_i\left\{y_i^{\text{trend}}(t)\right\}$$

where a_i is an amplitude shift, b_i is the amplitude, c_i is the frequency, d_i is the phase and T is the period during which cyclic patterns are observed.

Once the trend has been approximated the stochastic component can be modelled as a Brownian motion:

$$y^{\text{BM}}(t) = y^{\text{BM}}(t-1) + \eta$$

Two points are worth mentioning. First, since the number of requested files is always non-negative we have to truncate a negative value of $y^{\text{BM}}(t)$ to zero. Second, bursts in positive

direction are higher than bursts in negative direction. To accommodate for this we define η as:

$$\eta = \begin{cases} \eta' & \text{if } \eta' > 0 \\ \dfrac{\eta'}{\lambda} & \text{otherwise} \end{cases}$$

where $\eta' \in \text{Norm}(0,\sigma)$ and λ is a parameter determining the ratio between the heights of the positive and negative bursts. The second modification also has the effect of reducing the number of times the series has to be truncated due to negative values.

a) b)

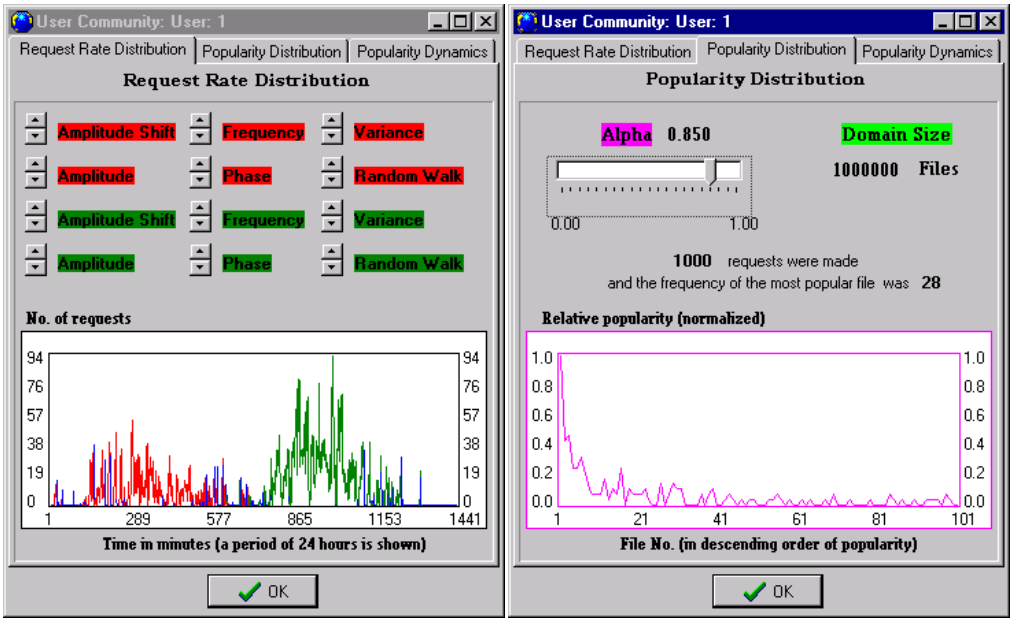


Fig. 2. Interface to the user community model. The graph on the left shows how the daily activity pattern can be tuned to approximate an observed pattern. The graph on the right shows the tuning of the file popularity as observed from the user community model.

Since the auto-correlated stochastic component (2) must be superimposed on the trend (1), a way of "guiding" the random walk of the Brownian motion towards the trend without destroying the desired properties is needed. We suggest using a sequence of non-overlapping random walks each starting from around the trend:

$$y(k\Delta t) = y^{\text{trend}}(k\Delta t) + Norm(0, \sigma^{\text{trend}})$$

i.e., at each time step $k\Delta t, k = 0,1,2,\ldots$, a Brownian motion process begins for Δt steps:

$$y^{\text{BM}}(k\Delta t + m) = y^{\text{BM}}(k\Delta t + m - 1) + \eta$$

where, $m = 1,2,...,\Delta t - 1$. Then it stops and a new process begins. This completes our model of the intensity of the http requests (i.e. the daily activity pattern). But before we can use it in our simulations of Internet caches we also need to define the popularity distribution of the requests. There is significant evidence in the literature suggesting that the popularity distribution follows a Zipf's-like law, where the relative popularity of the i^{th} most popular file is given by:

$$p_i^{relative} = \frac{1}{i^{\alpha}}$$

Therefore, we also need a random number generator that produces Zipf's distributed numbers. We define it in the following way. First the total domain size *N* and the exponent α must be specified. Then the probability of selecting file *i* is given by:

$$p_i = \frac{i^{-\alpha}}{\sum_{j=1}^{N} j^{-\alpha}}$$

A uniform random number *n* is generated in the range between 0 and 1 (most programming languages have already defined uniform random number generators) and an index *k* is found such that the following inequalities hold:

$$n \le \sum_{j=1}^{k} p_j$$

$$n > \sum_{j=1}^{k+1} p_j$$

The index *k* is the desired random number coming from the specified Zipf's-like distribution.

After developing the user community model we proceed with the cache proxy model. The cache proxy is modelled by Web content expiry statistics (fig. 3) and it also implements a simplified caching algorithm. The expiry statistics models both the rate of change of Web pages (reflecting server assigned TTL) and cache purging due to stale data (i.e., cache assigned TTL). For example, fig. 3 shows that in this particular case about 14% of the pages are not cacheable (i.e., cookies, stock quotes, etc.) and that the cache purges all files older than about two months (80640 minutes). These parameters are, of course, flexible. If the cache is of limited size, the model also implements a cache replacement algorithm. Currently only the well-known least recently used (LRU) data replacement algorithm is considered.

The simulation works as follows. The proxy cache receives requests for individual files. It checks if the requested file has already been registered in the cache model before. If not, the file is time-stamped, registered in the cache and a miss is reported (i.e., this reflects downloading the file from the origin server and caching it). If the requested file has been registered in the cache before, the proxy checks the TTL. If the file has expired the proxy verifies whether the file has changed. If so, a miss is reported and the time stamp of the file is reset (this reflects downloading of the new version of the file from the

origin server). If the file has not changed, a hit is reported and again the time stamp is reset.

Once the single proxy has been modelled, we can build meshes of interconnected proxy caches. To achieve this the simulation toolbox allows two caches to be linked together. We consider two types of links:

- *Parental* links in which a miss is propagated to the parent cache and it is responsible for providing that file back either from its cache neighbourhood or from the origin server. On its way back the file is cached at each parental level.
- *Peer* or *sibling* links, in which a peer proxy only checks if it has the file in its neighbourhood, but does not download it from the origin server in case of a miss.

Using the above-described links we can build a number of caching structures. For example, fig. 4 shows how two user communities can be "merged" together to use the same proxy cache. This can be useful to analyse file popularity, i.e., what happens to the file popularity generated by a superposition of user communities as compared to the individually generated file popularity.

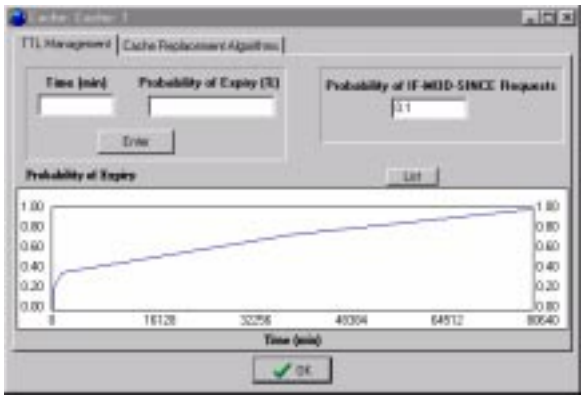


Fig. 3. Parameters defining a proxy cache model. The probability of expiry reflects both the server assigned TTL and the cache assigned TTL.

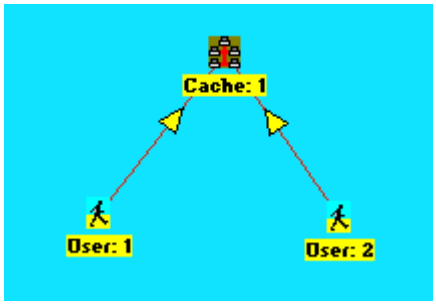


Fig. 4. Merging two user communities

As another example consider fig. 5, which shows how several first level caches can be connected to a parental higher level cache. This can be useful to analyse indirect co-operation among caches from the same level.

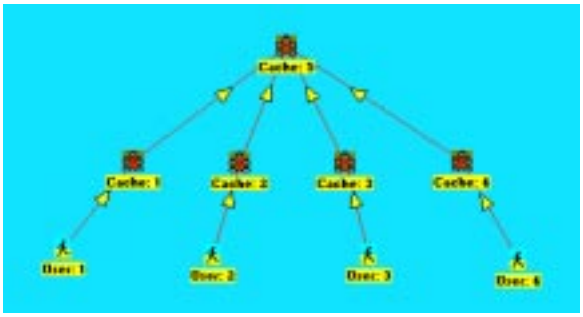


Fig. 5. A higher level cache.

3. Performance Measures

In order to analyse the performance of a cache and to reliably compare it with other caches, the following performance measures have been defined:

- *Request Hit Rate* is a measure of the efficiency of the cache. It does not correspond one-to-one with the saved bandwidth since the requests are for files with various sizes. The importance of this measure, however, stems from the fact that opening an HTTP connection is a relatively expensive process comparable to the actual time needed for smaller files to be transferred.

$$\text{Request Hit Rate} = \frac{\text{No. of Hits}}{\text{Total No. of Requests}} [\%]$$

- *Byte Hit Rate* is also a measure of the efficiency of the cache. It reflects the actual amount of saved bandwidth.

$$\text{Byte Hit Rate} = \frac{\text{Volume of Requested Data Found in the Cache}}{\text{Volume of the Total Requested Data}} [\%]$$

- *Saved Bandwidth* is a measure of particular interest to network designers. It shows the actual effect of the cache on the network.

$$\text{Saved Bandwidth} = \text{Volume of Requested Data} - \text{Volume of Data not in the Cache [MBytes]}$$

- *Saved Pounds* is a measure interesting to decision-makers. It can provide evidence of the economic feasibility of a caching solution.

$$\text{Saved Pounds} = \text{Saved Bandwidth} * 0.02 \quad [\text{Pounds}]$$

The last measure assumes that the current charging rate is 2 pence per MByte as reported in [Spa98].

4. Experiments

When users share a cache proxy they indirectly co-operate with each other by replenishing the cache with files other users might request in the near future. Since the files have certain expiry statistics, the more active the user community is, the more sharing is achieved before the files become stale. This phenomenon has been observed in real data [Dus97] and therefore, here we will verify our model by experimenting with the effect the request rate (or number of requests per unit time) has on the previously defined performance measures. Moreover, since our model allows us to monitor a number of performance characteristics including dynamic behaviour, the produced results will present a more complete picture of the caching processes than those previously published.

For the experiments we set up a user community and a proxy cache. All of the model variables are kept constant apart from the number of requests per day, for which three values (50000, 100000, and 250000 on average) are experimented with. All of the experiments show a simulation of sixty days (figures 6, 7, and 8).

Figures 6a, 7a, and 8a show the number of requests incoming into the cache proxy and the number of requests outgoing from the cache to the origin servers (or parent cache). The outgoing requests represent the cache ‘misses’ and the gap between the two graphs is a measure of the cache efficiency and defines the requests hit rate.

Figures 6b, 7b, and 8b show the data volume served by the cache proxy to the user community and the data volume that actually comes from the origin servers (or parent cache) to the cache. Again the difference between the two graphs (which in fact is the saved bandwidth as shown in figures 6c, 7c, and 8c) is a measure of the efficiency of the cache and defines the byte hit rate presented in figures 6d, 7d, and 8d.

The saved bandwidth can also be expressed in monetary terms as is done in figures 6e, 7e, and 8e. This might be of particular interest to decision-makers to assess the economic viability of the cache.

Figures 6f, 7f, and 8f show the required cache sizes in MBytes, a measure of particular interest to network designers for cache dimensioning.

5. Observations

It must be clearly understood that the 'bandwidth savings' our model predicts are typical figures (for each day) and should be used for comparative purposes only. The fact that the request rate and request locality are extremely bursty at a wide range of timescales [Mar98], indicates that there will be times when the 'saving' is minimal because the cache

is not being used, or because the current burst of requests did not generate any hits. The saving is probably best regarded as indicating a reduced frequency of congestion and increased customer satisfaction (due to reduced latency at busy periods). Unfortunately it is hard to quantify the value of this benefit. However, some observations can be made;

- Under the simulation assumptions and provided that the users are active throughout the year, the achieved caching saving for the three scenarios are approximately 3395, 9125 and 31750 pounds per year. These numbers are small compared to the cost of the large dedicated cache servers used by commercial operators (approx. 50K pounds). This indicates that most of these operators are using caches to reduce latency to their customers, and not (as is widely presumed in the academic research community) to avoid spending on network capacity. The misperception is easily explained - academic network operators are more strongly motivated by cost than customer perception, whereas for commercial operators the reverse is true.

- The larger the user community is the less space is required per user. This is simply because the space taken by popular files is shared amongst more users.

- In order to achieve the reported cache performances the three modelled caches would require disk allocations of approximately 7.5 GB, 12GB, and 20GB. The request rates correspond to the traffic typically generated by 500, 1000 and 2500 users respectively. When caches were first introduced vendors recommended around 1MB/user. Clearly this is woefully inadequate for current traffic conditions. It seems 15MB/user at a small cache and 8MB/user at a large cache would be more appropriate. Since 10GB disks are now standard on new PCs, and larger disks are relatively cheap, we would expect all new cache servers to operate with disks larger than 20GB. Operators of older caches would be weel advised to upgrade their disk rather than worry about the efficiency of their cache replacement algorithm.

In the future latency is likely to become the most important cache metric, particularly when the Internet is widely used for real time traffic as many predict. In this case the congestion reduction provided by caches will increase the quantity of potentially high value real time traffic that the network can support. We are accordingly extending our model to include the small queuing delay at each cache in the hierarchy.

6. Conclusions and Future Work

In this paper we have presented an analytic WWW cache model capable of realistically simulating some aspects of real Internet cache structures in real time, with a low overhead in terms of CPU usage and log analysis. The model only requires a few simple input values. Nevertheless, we have demonstrated that the model can be used to verify observed data from real caches. We therefore claim that the simplifying assumptions we have made appear to work. The toolbox will be of great benefit to operators needing to plan capacity

in their cache networks. The model is not yet complete, as there are some extra features to be incorporated such as user perceived latency.

References

[Abr95] M. Abrams, C.R. Standridge, G. Abdulla, S. Williams and E.A. Fox. Caching Proxies: Limitations and Potentials. *Proc. 4th Inter. World-Wide Web Conference*, Boston, MA, Dec. 1995.

[Arl96] M. Arlitt and C. Williamson. Web server workload characterization: The search for invariants. In Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, May 1996.

[Bae97a] M. Baentsch, L. Baum, G. Molter, S. Rothkugel and P. Sturm. Enhancing the web's infrastructure: From caching to replication. *IEEE Internet Computing*. March 1997. pp. 18-27.

[Dus97] Bradley M. Duska et. al., The Measured Access Characteristics of World-Wide Web Client Proxy Caches, http://www.cs.ubc.ca/spider/marwood/Projects/SPA/wwwap

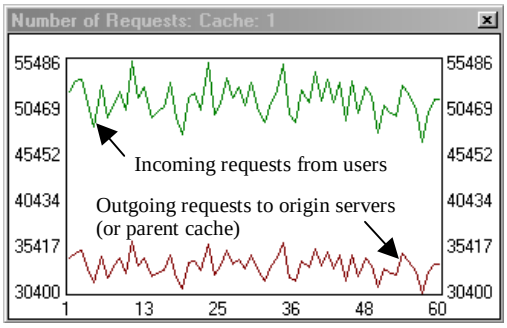
[Mar98] I Marshall, C Roadknight, Linking cache performance to user behaviour, Computer Networks and IDSN Systems, 30, pp. 2123-2131.

[Roa98] C Roadknight, I Marshall, Variations in cache behaviour, in *Computer Networks and ISDN systems* 30 (1998), pp.733-735.

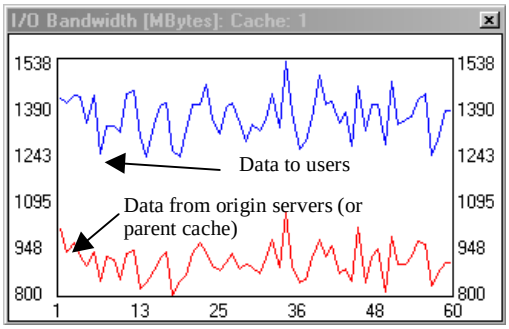
[Roa99] C. Roadknight, I. Marshall and D. Vearer. File Popularity Characterisation. Submitted to the 2nd Workshop on Internet Server Performance (WISP 99)

[Spa98] Michael Sparks et. al., Report on Statistical Analysis of the National Cache Performance, http://wwwcache.ja.net/Statistics/November_Graphs/Report.html

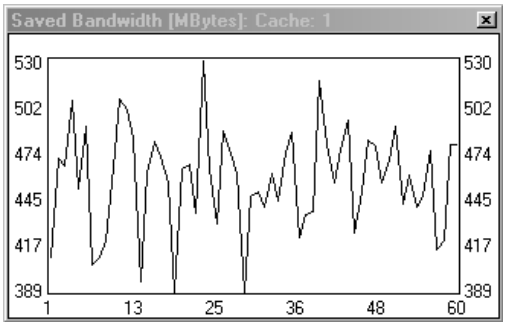
a) Number of requests.



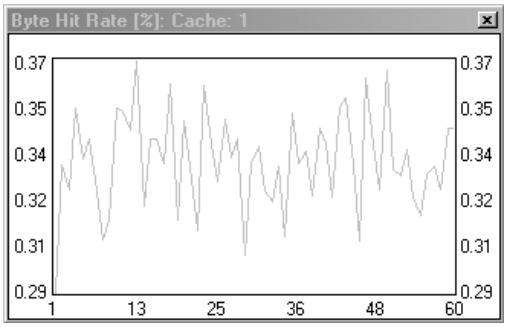
b) I/O bandwidth.



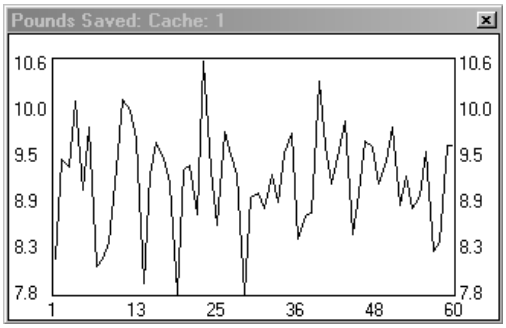
c) Saved bandwidth.



d) Byte hit rate.



e) Savings.



f) Cache size.

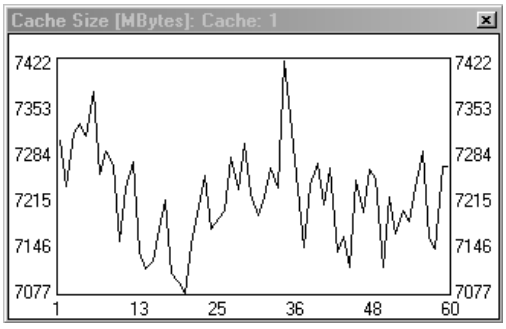
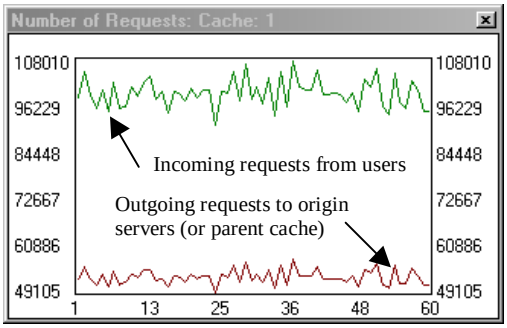
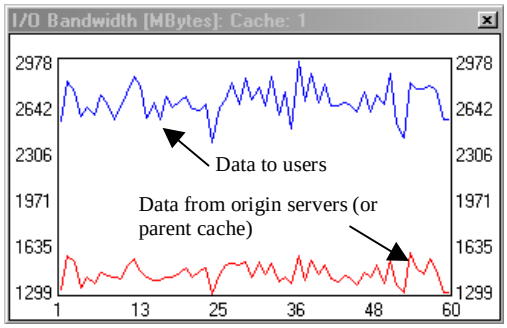


Fig. 6. Sixty-day simulations of a user community using a proxy cache to connect to the Internet. The user community generates approximately 50,000 requests per day.

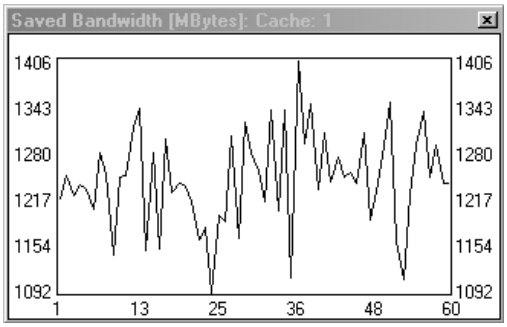
a) Number of requests.



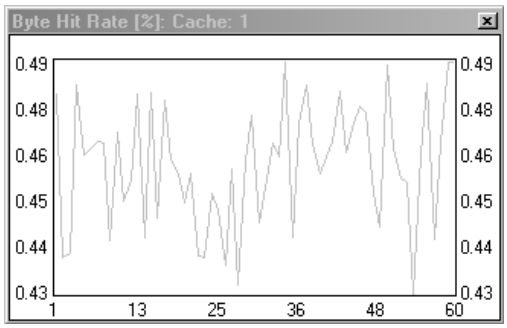
b) I/O bandwidth.



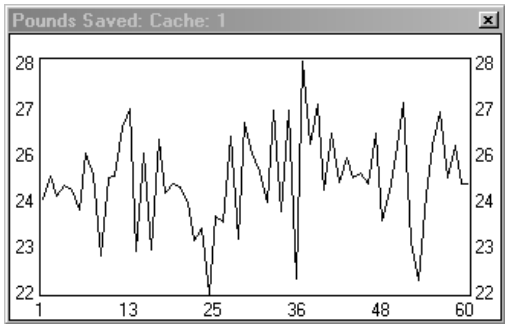
c) Saved bandwidth.



d) Byte hit rate.



e) Savings.



f) Cache size.

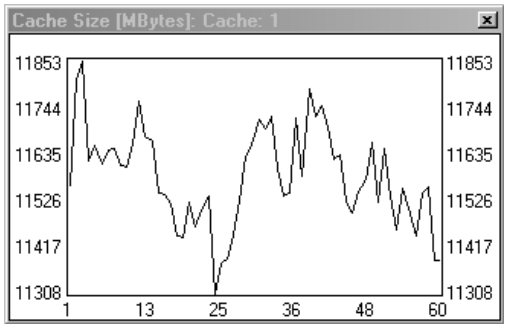
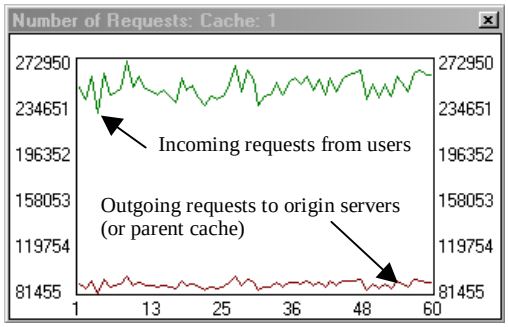
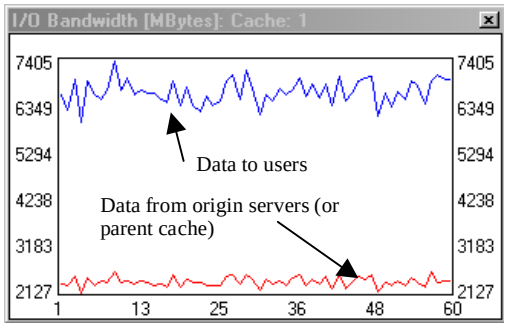


Fig. 7. Sixty-day simulations of a user community using a proxy cache to connect to the Internet. The user community generates approximately 100,000 requests per day.

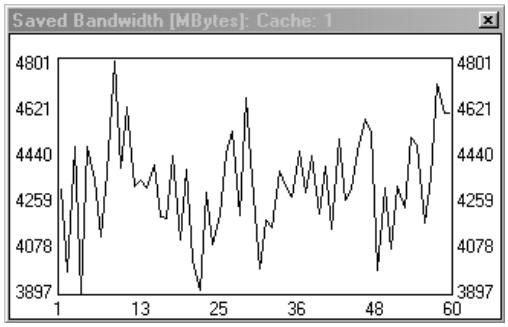
a) Number of requests.



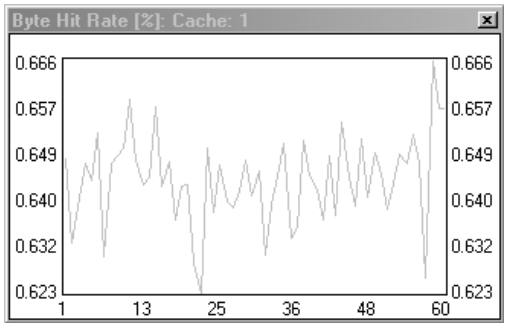
b) I/O bandwidth.



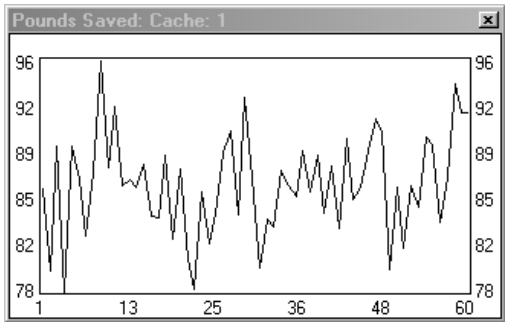
c) Saved bandwidth.



d) Byte hit rate.



e) Savings.



f) Cache size.

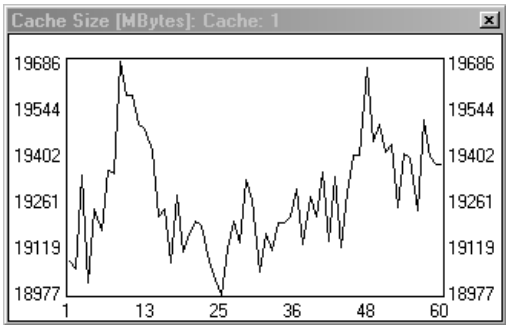


Fig. 8. Sixty-day simulations of a user community using a proxy cache to connect to the Internet. The user community generates approximately 250,000 requests per day.