The UK National Web Cache
A State of the Art Report

Neil G. Smith,
HENSA Unix,
The University of Kent at Canterbury.

Abstract

Two years after its introduction at the First International World-Wide Web Conference at CERN, Geneva, the use of caching technology to improve the efficiency of network utilisation has become a hot topic. With relatively poor international connectivity, if was through necessity that UK academia was one of the first communities to make widespread use of this technology on a large scale. The implementation of a national strategy proposed by HENSA Unix in June 1995 has led an experimental project to become what is probably the most mature caching facility in the world today. In this paper we present a brief history of the project, a discussion of the evolution of the hardware, software and networking systems involved, and take a look to the future of the project within the framework of the UKs networking strategy. It is hoped that some of our experiences may be of use to other large bodies of users who are tired of waiting for their Web pages to arrive.

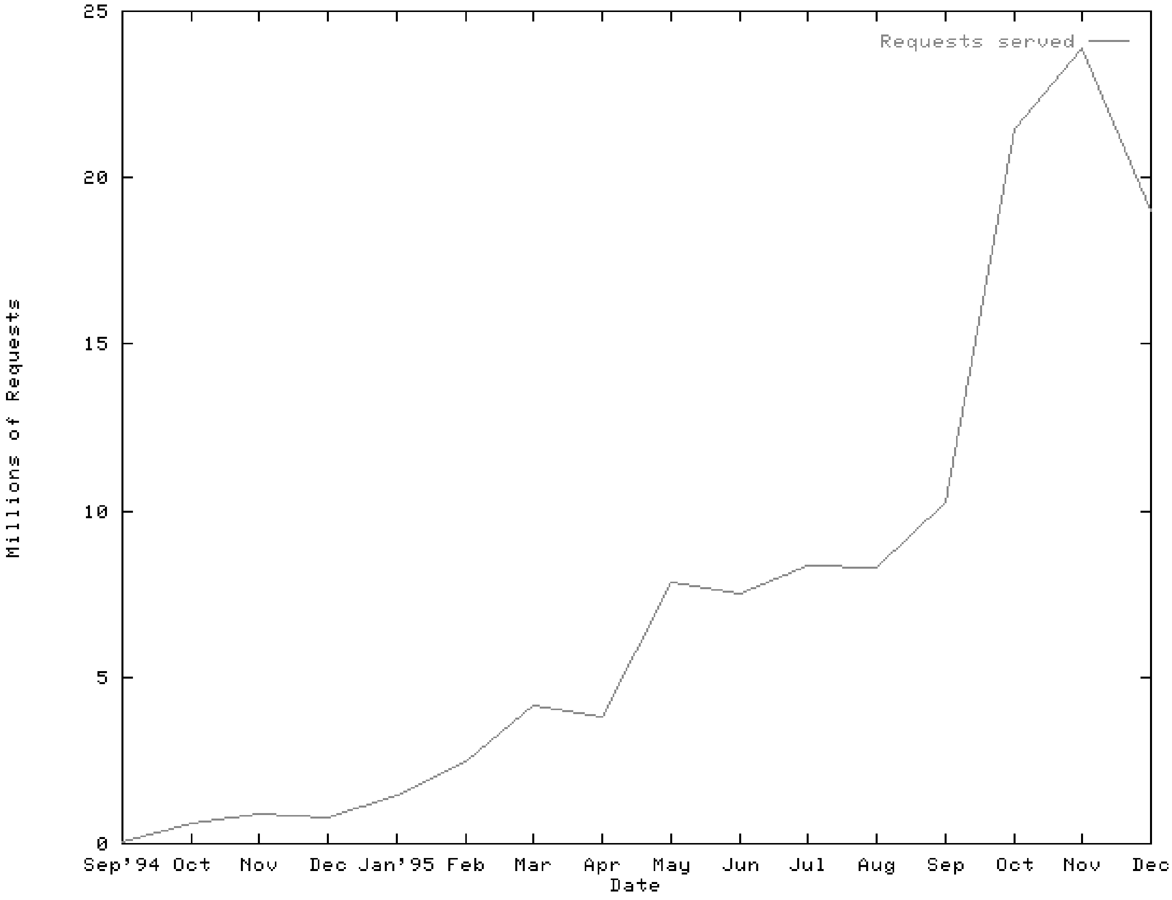
Contents

Introduction

The World-Wide Web has long suffered as a result of its own popularity. The combination of the ease with which large video and audio data types can be incorporated into documents, and the model of a single publisher serving countless clients places great demands on bandwidth. While this may not present a problem on a local area network, or even within a national context, as soon as information passes across international networks the lack of bandwidth and the resulting congestion is immediately apparent. (This problem is very obvious in the UK. Nationally we have good connectivity with a 150Mbps backbone, but our international links are relatively slow: 4Mbps to the United States, 4Mbps to Europe and 2Mbps to Scandinavia.) The problem was already recognised by the time of the First International World-Wide Web Conference [1] in May 1994 and has received a steady stream of attention since then. The consensus of opinion suggests that distributing the publication responsibility through the deployment of Web proxy caches gives us the quickest route to a medium term solution. In future more sophisticated schemes may allow for more flexible publication mechanisms that avoid some of the problems that proxies introduce. However, for the moment they are all that we have and their role is now central in many users' access to the Web. This means that all new protocol developments must take account of these intermediate servers and work with them. This makes protocol development more complicated, but the impact that proxy caches can have is so great that we cannot afford to ignore them.

System Evolution

The evolution of the systems in use at HENSA Unix has been forced by the great demand on the service. Until recently this demand always out-stripped the resources available. At points in the services history the demand has been so great that queues on the servers resulted in using the cache actually being slower than going direct. The graph below shows how the service has grown (December's dip is, of course, the seasonal norm).



Lagoon - CERN - Netscape

In November 1993, after initial experiments confirmed that the wholesale mirroring of Web pages was not an effective way to reduce the latency seen on the networks, HENSA Unix adopted Lagoon[2] as an experimental proxy cache. At the time, the necessary protocol extensions to support proxying were not in place and early versions of Lagoon had to make use of a CGI script that rewrote HTML on the fly in order to direct clients back to the cache for each subsequent page that they retrieved. Despite some innovative features (cache co-operation was already being discussed) this HTML rewriting and other performance related problems meant that the client base being supported by HENSA Unix was becoming too large for Lagoon.

At about the time of the First International Conference, a proxy mechanism was introduced into the the CERN HTTP server[3]. New versions of Mosaic made the use of this facility transparent to the user and proxying started to become a viable proposition. HENSA Unix continued to use the CERN server for almost a year but with the increasing popularity of the cache, the forking process model used by the CERN server started to place a higher and higher load on our hardware. At this point the caching service was still experimental and not receiving its own funding.

It was because the CERN server forked for each connection it received that the service eventually started to fail. The incoming connections could not be accepted fast enough and users were being turned away. Hacks to increase the priority of the parent process while decreasing that of the child processes only helped for a very short time.

At the beginning of 1995 Netscape started beta testing their own proxy server and HENSA Unix was asked to act as a test site. The Netscape server[4] relies on a non-forking process model and thereby places a significantly lower strain on the hardware. The Netscape proxy server is still used to provide the main caching service; a service currently responding to over 1,100,000 requests every day.

Alternative Servers

While the evolution from Lagoon to the CERN server and finally to the Netscape proxy server represents a considerable improvement in stability, configurability and performance, the fundamental principles involved have not changed a great deal. Each of these servers still merely acts a simple proxy with a cache of pages to improve performance. Other projects have developed proxy servers that attempt to go further. The most notable of these being the Harvest Object Cache [5].

Harvest allows a single cache to interact with neighbour and parent caches in a co-operating hierarchy. These neighbours will normally be on networks that the cache has good access to. This model improves performance in the case of a cache miss by allowing other close-by caches to say whether they have the requested page. If another local cache has the page then it will be retrieved from that cache rather than the remote site. This means that any cache that is part of a large co-operative hierarchy benefits from the pages stored in all the other caches in that hierarchy.

While Harvest's approach goes one better than other simpler proxies it, unfortunately, relies on a single process model. This process uses non-blocking I/O and this results in relatively good performance. However, the question remains as to whether this model can ever be fast enough to serve the size of community currently using HENSA Unix. This community currently averages 28 connections a second at peak times in the afternoons, with peaks in activity of more than 100 connections per second.

Hardware Demands

For the first eighteen months of service, the HENSA Unix cache was placed on the same single processor Sparc 10 serving the HENSA Unix FTP archive. As the popularity of both services increased an upgrade was required and a Silicon Graphics Challenge S was deployed. It was anticipated that the cache would remain on the Sparc 10 while the FTP archive moved to the Silicon Graphics machine. Unfortunately the demand on the service increased to fill all the spare capacity on the Sparc and with the Challenge S being the fastest machine available both the FTP and the Web caching service were moved to this machine.

The very high connection rate being experienced on this server and on other conventional HTTP servers at other sites, stressed operating systems in ways in which they had never been stressed before. Now it was not the hardware that was insufficient, but bugs in TCP code implementations that made the service unstable. With the obvious demand for fixes from all quarters of the community, the vendors were quick to patch up the problems and demand could continue to grow. At this point another problem struck the HENSA Unix service. The Silicon Graphics machine had always been intended to support the FTP archive and as a result it was ordered with a small number of very large disks (three 9GB disks). Even with the cached files spread across three disks, the I/O bottleneck was great enough to mean that in some cases going via the cache was actually slower than going to a site directly. The impact this had on the FTP archive (the service for which we received our funding) was that FTP users were unable to connect to the machine at all. At this point the cache was serving about 300,000 requests each day.

The solution to the problem came when an emergency equipment purchase expanded the service with a dual processor Silicon Graphics Challenge DM. This machine was ordered with six 2GB disks to ensure that bandwidth within the machine would not be a constraint on the service. The service was migrated to this machine in June 1995. The start of the UK academic year in October 1995 saw this machine responding to over 900,000 requests each day. Once again, the demand had expanded to fill the available capacity.

A proposal by HENSA Unix to take the experience gained through operating this experimental service and deploy a scalable and reliable national service was accepted in July 1995 and for the first time ever, enabled us to invest in equipment that would be capable of keeping up with the demand. In order to provide resilience in the case of hardware failure a number of machines would be used. Based on previous experience, each of these machines would have the optimum balance of processor power, disk bandwidth and system memory.

Hardware Resource Balancing

A busy Web cache tests all the sub-systems in a machine. Surprisingly, network bandwidth is not always the most important concern or the first bottleneck hit. This reflects the disparity between transfer rates on local and on international networks. Instead, a lack of disk bandwidth, processor speed or real memory can bring a cache server to a grinding halt.

In the case of the Netscape Proxy server it is the combination of the speed of the processor and the amount of real memory that determines how many concurrent users you may support. Each of our 175MHz R4400 based servers, with 128MB of memory can support approximately 650 concurrent connections.

Disk bandwidth is a more serious concern than disk space once a minimum level has been passed.

Simulations based on real cache activity show the hit-rates being achieved by larger and larger caches stabalizing at approximately 55%. The growth in hit-rate is quite rapid, and with very large disk drives now available at a fraction of their cost even two years ago, there is no reason why all caches could not achieve this hit-rate. While the size of the disk determines the hit-rate, bandwidth to the disks is most likely to be the first bottleneck after the international networks. Making use of a large number of disks, and distributing the cache data across these disks is a facility now offered by both Harvest and the Netscape Proxy server.

It remains to be seen whether the continual growth of both server and client populations on the Web makes a significant difference to the hit-rates attained by, and the disk space demanded by, caching proxies. It is true to say that with more servers there will be more potentially cachable data, but on the other hand, with more clients there are a greater number of hits on the popular pages. This may lead to caches that are as effective without any increase in disk capacity.

Networking and Machine Load Balancing

Throughout the first two years of service, the network structure surrounding the HENSA Unix cache did not change. It was only with the acceptance of the proposal for a national strategy that changes were made to the operation of the cache on the network.

Having multiple machines provides resilience in the case of hardware failure. If these machines are distributed across several sites then resilience against network failure is also gained. Currently the HENSA Unix cache is implemented with machines at two sites, the University of Kent and the University of Leeds. This distribution also ensures that the bandwidth into or, more importantly, as the caches are bandwidth magnifiers, out of any particular site does not become the bottleneck in the whole scheme.

In order to evenly distribute the load across the machines supporting the cache we anticipated having to modify a DNS name server to return the name of the most lightly loaded machine. In fact, this proved unnecessary as more recent versions of BIND provide a round-robin facility that rotates the list of addresses corresponding to a single name. With a five minute Time-To-Live on the name this is sufficient to ensure that, over a 24 hour period, the load across all six machines is even. It also gives us the ability to quickly reconfigure the group of machines supporting the service in the event of a hardware failure.

Further distribution of the caching facility is envisaged in the UK’s overall strategy. This distribution consists of local caches operated by an institution or even a department within an institution. We are encouraging these local caches to then make use of the national cache to minimise redundant transfers across the international network links. Simulations based on the log files collected at HENSA Unix show us that an institution, even with only a relatively small cache, 500MB of disk, can reduce the load placed on the national facility by as much as 40%. Institutions without the specialist knowledge to operate a WWW proxy cache are being encouraged to approach their closest Metropolitan Area Network to make use of a cache at this point.

Through the study of server log files from sites outside the UK a number of institutions were found who were not making use of the national caching facility. When questioned, the most common response was that they intended to install a local cache and did not want to have to go through the user education procedure twice, first they would be telling their users to make use of the national cache at HENSA, and shortly afterwards redirecting them to the local cache.

We are sympathetic to their problem as educating a population that is increasingly unaware that it is even using a cache is exceedingly difficult. In order to provide a solution and encourage the early use of the caches available the "virtual local cache" was created. This technique allows an institution to give its users the impression that they already have a local cache. The education programme can start without investment in hardware, software, or time. The technique makes use of the DNS to direct clients to the national cache through the use of a Canonical Name (an alias for another machine). Once a local cache has been installed this Canonical Name can be changed to point at the new machine without the users seeing a break in service.

The Users

As the service offered by the HENSA Unix cache has evolved, so has the user community that it serves. In 1993 and 1994 virtually all of the users were conscious of the fact that they were using a cache, and all of them understood the function that it served. With the wider use of institutional caches installed by computing service departments a larger population is now making unconscious use of caching technology.

These users are obviously aware of the congestion on the international network links, they experience it every day. Unfortunately they are not aware of the techniques that can be used to maximise throughput on a congested link, and they are not prepared to accept these techniques when they are forced upon them.

The Netscape proxy server’s process model allows a cache administrator to exactly specify the number of simultaneous connections that the proxy can hold open at one time. At HENSA Unix this fact was used to restrict the number of connections to between 400 and 600 in a vain hope that this restriction would reduce network congestion and result in better throughput. In order to ensure that all the processes in the process pool were not in use at the same time, which would result in users being turned away, the timeout placed on each connection was kept deliberately short, 90 seconds.

Unfortunately this scheme did not prove popular. The cache administrator had not understood the way in which the vast majority of clients use the Web, that is, in batch fashion. The author had assumed that a user who had not received their page within 90 seconds would probably have given up and moved on. In reality is seems that large numbers of users are prepared to wait much longer for their pages. Often this waiting time is spent doing other things with the client visiting their browser every so often to see how much longer the transfer is estimated to take. In order to accommodate these users the timeout on cache connections was increased to 15 minutes. This is not 15 minutes for the complete transfer, but rather 15 minutes between individual packets. It is probably safe to assume that if packets are more than 15 minutes apart the network is not worth using.

This significantly increased timeout means that a much larger proportion of the Netscape proxy server processes are in use simply waiting for packets. These processes are no longer available to other users. In order to ensure that users are not turned away, a much large number of processes must be made available. Currently each of the servers in use at HENSA Unix runs 650 processes, meaning that the whole cache can support nearly 4,000 simultaneous connections.

This very large number of connections almost certainly results in far greater congestion on the international links. However, at least the users never have to wait for a process to become available, and they never see a time out as a result of the cache. Perversely, throughput is down, but the customers are happy.

Future Developments

In terms of efficiency, and as a method of saving bandwidth, caching has a lot of potential still to be developed. Stable, and well understood cache co-operation is one goal, client resilience in the event of cache or network failure another. These issues are currently under development, while others, such as "missed hit" reporting (reporting hits that a server would have seen if the cache had not been there), and user identification on the far side of a cache, or chain of caches, receive less coverage. For a cache to be totally acceptable its use has to be completely transparent to both clients and servers. There is still some way to go, but some of the most recent developments representing the state of the art are discussed here.

Proxy Auto-configuration [6]

The problems of user education and resilience are addressed by version 2 of the Netscape Navigator, currently available in a beta test version. This client has the ability to run arbitrary pieces of Javascript to determine which proxy or proxies it should use. This script can be downloaded from anywhere on the Web, thereby giving cache maintainers or site administrators a single point at which they can define the configuration of all their users' clients.

In addition to choosing whether or not to use a cache, or which cache to use, this Javascript can return a list of caches that should be tried. In the event of one proxy becoming unavailable the next in the list is tried. If no proxy responds to the client then the final option is for a direct connection. In the worst case the client will simply behave as if it were not configured to use a proxy at all.

By encouraging users of the national cache to upgrade to a client that understands this proxy auto-configuration it is hoped that we will be able to make significantly more efficient use of the caches. Currently each of the six cache machines operates independently and as a result the caches have a large number of pages in common. Disk space is wasted with duplicated pages, and can result in a client having to wait for a remote transfer when another cache may already have the page locally.

By using proxy auto-configuration we hope to dedicate each of the cache machines to a specific list of domains. These domains would be chosen to balance the load across all the machines, and would result in the same machine always being used for pages from a specific domain. This should result in higher hit-rates as fewer duplicate pages will give more efficient use of disk space, and every client accessing a particular domain will use the same cache.

Cache co-operation

As far as cache co-operation is concerned Harvest is leading the field. Unfortunately this co-operation is limited to networks of Harvest caches as no other server currently understands the inter-cache communication protocol used. In order to make cache co-operation the norm rather than the exception it is necessary to extend and standardize this protocol.

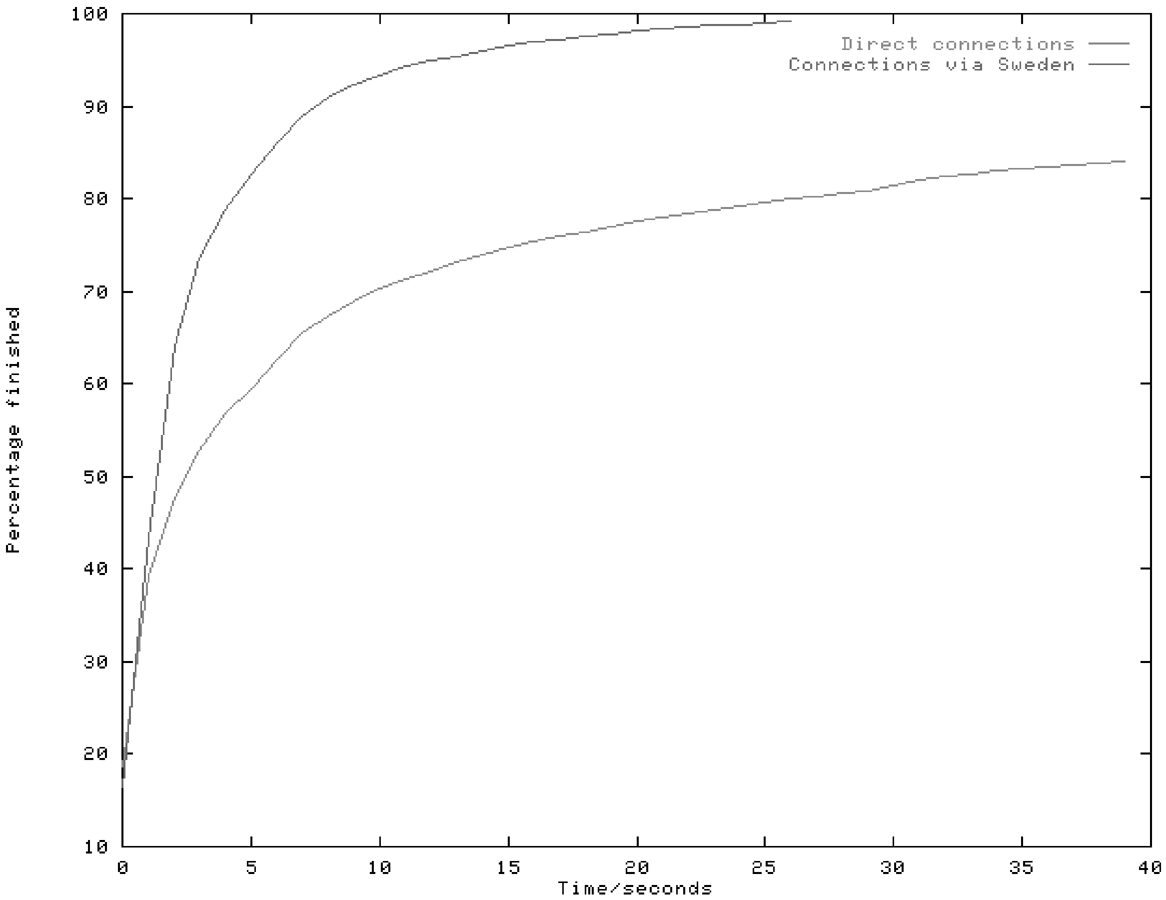
Currently the protocol talks in terms of cache hits or cache misses. Both the CERN server and the Netscape proxy server have an additional state which is a cache hit based on the result of a conditional request to the remote server. This means that the CERN or Netscape server may physically have the appropriate file in its cache, but before releasing it, the server would like to ensure that the file is up to date. This up-to-date check consists of a conditional request to the remote server. If the cache's copy is still up to date then no further transfer is required. If the cached copy has gone out of date then the remote server sends a fresh copy. In any case, a conditional request will take significantly longer than a pure cache hit. However, in the case where a transfer of the whole file does not occur, the conditional request will very often be significantly faster than a cache miss.

Ultimately, what is required is a server with the co-operative model implemented by Harvest and the speed, ease of configuration and administrative features of Netscape.

Networks for Caches

The Web caching service at HENSA Unix provides real benefits. However these benefits are only appreciated in the case of a cache hit. Cache misses, or checks to ensure that documents are up to date, have to make use of the same shared and highly congested bandwidth as all other international traffic. While this shared bandwidth is soon to be augmented, past experience has shown that demand will quickly rise to fill that capacity, and we will quickly return to the same congested situation. With the cache’s relatively low demand for and extremely efficient use of bandwidth it would make sense to provide dedicated international network links for use by caches alone. In this case, cache hits are as fast as they always were, but cache misses are accelerated to a point where they are significantly better than if the shared bandwidth were used. It is with this property that the cache starts to become a very much more attractive service. As the UK’s shared international links are upgraded it is hoped that the old, now redundant, links will become dedicated to the caching service. With hit-rates at their current levels, a dedicated 4Mbps link could mean the cache delivering as much as a conventional, shared 10Mbps link.

In order to demonstrate the effectiveness of the cache in a situation where there is spare bandwidth, HENSA Unix asked for help from Lulea University in Sweden. The UK has a 2Mbps connection to Scandinavia which typically runs at 50% capacity. Scandinavia has a 34Mbps link to the US which is running at well below full capacity. By directing the HENSA Unix cache to make use of another proxy at Lulea University we were able to test its performance with, what was virtually, a dedicated 1Mbps line. In order not to place too great a load on the Lulea server, and in order to be able to compare the two routes, direct and via Sweden, only two out of the five machines operating the cache at HENSA Unix were configured to go via Sweden. In addition it was only the .gov domain that would be fetched by that route. This resulted in about 17,000 requests to the Swedish proxy over a two day period. The results are clear from the graph below which shows the percentage of all requests that have been successfully serviced within a specific number of seconds.



New HTTP Protocols

Dedicated bandwidth will offer the UK National Cache a lot more than just faster access to pages in the United States. Extensions to, or developments of the HTTP protocol protocol such at Keep-Alive [7] and HTTP-NG [7], will make use of connections persistent across many URL requests.

On a congested line these protocols are liable to result in poorer performance as the TCP protocol slows down the transfer rates as the connection ages in order to attempt to reduce the congestion. This effect can be clearly seen when retrieving large files across congested network. The transfer rate starts relatively high, but gradually degrades as the connection ages.

On a non-congested link the cache will be able to take full advantage of these protocols. This advantage can be gained whether or not the user's browser makes use of the new protocols as the cache will be able to translate from plain HTTP to the new protocols and back again.

It is possible to imagine a situation where a particularly busy cache holds open a connection to a popular server permanently. This would completely eliminate the costs associated with making a new connection for every URL requested.

Conclusion

"Necessity is the mother of invention". In the summer of 1993 it was clear that the bandwidth demands placed on the Internet by the World-Wide Web made the dream of global hypermedia communication very difficult to achieve. The problems faced by the United Kingdom (good national bandwidth allowing individuals easy access to very limited international bandwidth) will sooner or later be faced by all other countries as they embrace the revolution. We hope that the description of the problems that we have faced and overcome will help these other communities deploy bandwidth saving measures that are also ‘state of the art’.

References

[1] The First International World-Wide Web Conference,
http://www.elsevier.nl/cgi-bin/ID/WWW94

- World-Wide Web Proxies, *Ari Luotonen and Kevin Altis, CERN, CH*
- A Caching Relay for the World Wide Web, *Steve Glassman, SRC, DEC, US*
- What can Archives offer the World Wide Web, *Neil Smith, Unix Hensa, The University of Kent at Canterbury, UK*

[2] Lagoon, http://www.win.tue.nl/lagoon/

[3] CERN Server, http://www.w3.org/pub/WWW/Daemon/

[4] Netscape Proxy Server, http://home.netscape.com/comprod/proxy_server.html

[5] Harvest Object Cache, http://excalibur.usc.edu/

[6] Netscape’s Proxy Auto-Config,
http://home.netscape.com/eng/mozilla/2.0/relnotes/demo/proxy-live.html

[7] HTTP issues such as Keep-Alive and HTTP-NG,
http://www.w3.org/hypertext/WWW/Protocols/
