Performance Evaluation of Redirection Schemes in Content Distribution Networks

Jussi Kangasharju Keith W. Ross
Institut Eurécom, B.P. 193
Sophia Antipolis, France

James W. Roberts
France Télécom R&D
Issy les Moulineaux, France

Abstract

Content distribution on the Web is moving from an architecture where objects are placed on a single, designated server to an architecture where objects are replicated on geographically distributed servers and clients transparently access a nearby copy of an object. In this paper we study how the different redirection schemes used in modern content distribution networks affect the user-perceived performance in normal Web page viewing. Using both simulations and experiments with real Web servers we show that redirection schemes that require clients to retrieve different parts of a Web page from different servers yield sub-optimal performance compared to schemes where a client accesses only one server for all the parts of a Web page. This implies that when replicating Web pages, we should treat the whole page (HTML and images) as a single entity.

1 Introduction

Content distribution on the Web is moving from an architecture where objects are placed on a single, designated server to an architecture where objects are replicated on geographically distributed servers and clients transparently access a nearby copy of an object [1, 2, 6, 11, 12]. The new architectures are constructed from a set of servers, which we call content servers, that contain copies of the objects. These copies can be created statically using some pre-determined rules, or dynamically on-demand depending on the load and client request patterns. When a client wants to retrieve an object, it contacts a mapping service that provides the client with an address of a content server that has a copy of the requested object. There have already been some proposals for such architectures [4, 10, 15] and several companies have started to offer dynamic content distribution services over their own networks [1, 2, 6, 11].

A vital component of a content distribution architecture is a method for redirecting clients to the content servers. What is common to most of the proposed architectures is that the client is redirected to the content server *by the system*. This means that the system must contain mechanisms for determining what is the best content server for each client. On the other hand, the proposed architectures are transparent to the client, i.e., they do not require modifying the clients or installing new software at client-side. We will discuss the details of different redirection methods in Section 2.

In this paper we study how different redirection schemes affect the user-perceived performance. As the measure for performance we use the total time to download all objects on a Web page, i.e., both the HTML and embedded images. Some redirection schemes require that the client retrieves some part of a Web page (e.g., the HTML-part)

from one server, and other parts (e.g., embedded images) from another server. If the client is using persistent connections of HTTP/1.1 [7], this means that the client cannot benefit from previous requests that have opened the underlying TCP-congestion window; instead the client must open a new connection to another server and this connection will initially suffer from a small congestion window. Of course, if the new server is significantly closer than the old server, the client can retrieve the remaining objects faster from the new server.

Using simulations and experiments on the Web we will evaluate the performance of different redirection strategies and how they affect the download time of the whole Web page in different situations. We will evaluate the performance of redirection strategies using both multiple parallel connections and persistent connections with pipelining.

1.1 Related Work

Nielsen et al. [14] studied the performance of persistent connections and pipelining and their results show that pipelining is essential to make persistent connections perform better than multiple, non-persistent connections in parallel. Although modern browsers implement persistent connections, they do not implement pipelining [18]. For this reason the popular browsers open several persistent connections in parallel to a server.

Recently several companies [1, 2, 6, 11] have begun to offer content distribution services. In their services, the content is distributed over several, geographically dispersed servers and clients are directed to one of these servers using DNS redirection. We will discuss DNS redirection in more detail in Section 2.

Rodriguez et al. [16] study parallel access schemes where the client requests different parts of one object from different servers. Their scheme is designed for large objects and is not well suited for typical web page viewing; also it requires modifications to client software. In our work we study the performance of currently employed redirection schemes which redirect the client to a single server but require no modifications to client software.

This paper is organized as follows. Section 2 presents the different redirection schemes used in real world systems. Section 3 describes the model used in our simulations and Section 4 presents the results obtained in the simulations. Section 5 presents the results obtained in experiments on the real network. Section 6 discusses the implications of our results. Finally, Section 7 concludes the paper and presents directions for future work.

2 Redirection to Servers

Clients can be redirected to servers with several different methods. For example, the origin server could redirect clients using the appropriate HTTP-reply codes, the client could be given a list of alternative content servers, or the system could use other mechanisms, such as DNS redirection. These different mechanisms have all different overheads on the user-perceived performance which we will discuss in Section 6. For the remainder of this paper we assume that the system uses DNS redirection (or a similar method) because of its wide-spread use in the real world.

Currently the content distribution companies redirect clients using DNS redirection in two different ways. In both redirection schemes the client sends a DNS query to the authoritative DNS server of the content distribution company which replies with an IP-address for a content server that the authoritative DNS server deems to be the best for the client. (The reply can include IP-addresses of multiple servers but modern clients use only

one of them.) The client then contacts the content server and requests the object from it. The advantage of using DNS redirection is that it does not require any modifications to the client software because typically URLs identify hosts by their names.

The two different schemes are as follows. In the first scheme, which we call *full redirection*, the content distributor has complete control over the DNS mapping of the origin server. When a client requests any object from the origin server, it will get redirected to a content server. This scheme requires that either all content servers replicate all the content from the origin server, or that the content servers act as surrogate proxies for the origin server. A major advantage of full redirection is that it adapts dynamically to new hot-spots because all client requests are redirected to geographically dispersed content servers.

The second scheme, which we call *selective redirection*, goes as follows. The references to replicated objects are changed to point to a server in the content distribution network. When the client wants to retrieve a replicated object, it resolves the hostname which redirects it to a content server. In this scheme, the replicated objects appear to be simply objects that are served from another server. An advantage of this scheme is that the content servers only need to have the content that has been replicated. In modern content distribution networks that use selective redirection, the burden of deciding which objects to replicate is placed on the content provider. A system using selective redirection is slower to adapt to hot-spots because it must first identify them, change the references to the new hot objects, and possibly replicate the objects to content servers. In addition, clients that have cached references to the new hot objects (e.g., caching the HTML page referencing a hot image) would not be redirected, but would instead go to the origin server thus negating the benefits of using a content distribution network.

3 Simulation Model

For the simulations we used the NS network simulator [13]. We used a very simple network topology and it is shown in Figure 1. In Figure 1, C is the client, S is the server, and L is the link between the two. To represent different network conditions we varied the delay, bandwidth, and loss rate on the link L. We used FullTCP-agents at both the client and server and we set the MSS to 1460 bytes. This is the MSS that an Ethernet-connected machine would obtain and we have obtained the same MSS on real connections to distant servers from our local Ethernet. As suggested in [14], we disabled Nagle's algorithm on the server's TCP agent.

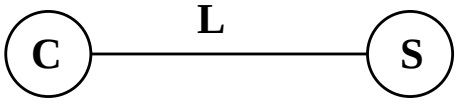
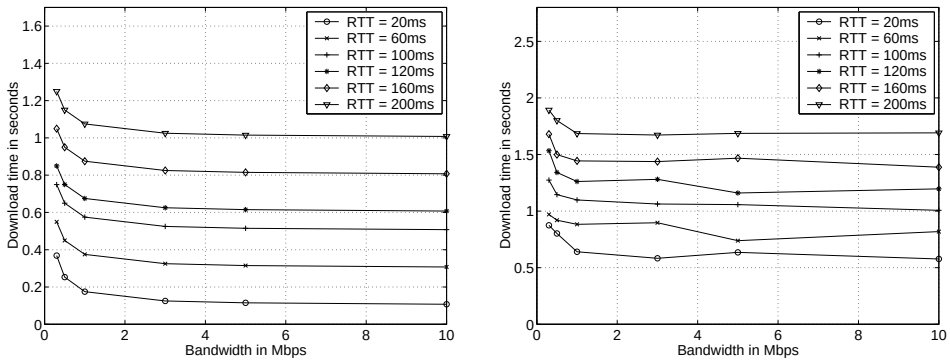


Figure 1: Simulation model

In all of our simulations, the client first sent a request to the server, the server replied with one file (the HTML-file). When the client had received all of this file, it requested the images from the server.

We observed that because all our HTTP-connections were short, the underlying TCP connection never progressed beyond slow-start. Therefore the bandwidth of the link had only minimal effect on the overall download time. This is also shown by the graphs

in Figure 2 which show that the total download time stays almost constant once the bandwidth is greater than 1 Mbit/s. This is also true in the case when the loss rate on the link is high (Figure 2b, averaged over 3000 simulation runs).



(a) No loss

(b) 5% loss rate

Figure 2: Effect of bandwidth on download time

Because of the negligible effect of the bandwidth, we compared the different redirection schemes only by varying the round-trip times and loss rates. Our simulation model does not account for server loads or how the client obtains the redirection; we will discuss these issues in Section 6.

3.1 Redirection Schemes

We compared the redirection schemes for two different clients. The first one was a client that does not implement pipelining and opens several persistent connections in parallel and the second client implemented pipelining. Given the typical number of embedded images on a page (see Section 3.2) and the number of persistent connections opened by popular browsers (2 or 6, see [18]), we assumed that the client opening parallel connections cannot retrieve all images with one set of connections. Instead, it first sends one batch of requests and when it receives the replies, it requests the second batch; this matches the behavior of a client implementing persistent connections without pipelining.

The baseline for our comparisons was a scheme where the client retrieves all object from the origin server. We compared this baseline scheme to other schemes where the client retrieves the HTML from the origin server and is redirected to another server for the images. To model the second server, we simply used two models from Figure 1 in parallel and the only connection between them was when the first model had completed its download and it triggered the second model. We used six different round-trip time values for the servers, 10, 20, 60, 100, 120, and 160 ms. We have observed that the value of 160 ms is quite typical from Europe to popular web sites in the US, and the smaller values reflect conditions within the US.

Our model does not account for the delay caused by a potential DNS lookup to get the address of the second server. We also assume that the parallel connections do not interfere with one another and represent them by one connection which retrieves one fifth of the image data on the page. In reality, the download time of the parallel client would

be dictated by the largest image because the client could not divide the images equally between all the parallel connections. Modern clients also start requesting embedded images as soon as they have seen the references to them instead of waiting for the whole HTML page to download. Our model does not take this fully into account, but the behavior of our model is appropriate for images that are referenced near the end of the HTML page.

3.2 Files

To estimate the size of a Web page, we downloaded all the homepages of the most popular sites from Hot100.com [8]. We found that the mean file size is 20 KB. This only includes the actual HTML for the page and does not include any embedded objects. The mean amount of embedded image data on a page is 40 KB, the mean number of embedded images on a page is 15.5, and the mean size of a single embedded image is 2.5 KB. Most of the HTML pages are at least 5 KB and almost none of the HTML pages are larger than 45 KB. To retrieve a typical homepage with all the embedded images we need to transfer around 50–60 KB and the total amount of data (HTML and images) can be as high as 250 KB. We constructed four different sized pages in order to cover as many different real pages as possible. Table 1 shows the different pages and the amount of HTML and image data on each of them. We also show the amount of image data retrieved by the parallel client which was equal to one fifth of the total image data.

Page	**HTML**	**Images**	**Parallel**
Small	5 KB	10 KB	2 KB
Medium	10 KB	20 KB	4 KB
Large	20 KB	40 KB	8 KB
X-Large	40 KB	80 KB	16 KB

Table 1: Different pages used in simulations

4 Simulation Results

In this section we will present the results from our simulations. We ran simulations for all the different parameter values (RTT and loss rate) but because of space limitations we only show some of the combinations here. The results for the simulation runs not shown here were similar.

4.1 Loss Free Conditions

We first simulated retrievals under completely loss free conditions. In Figure 3 we show the performance of the baseline scheme, i.e., retrieving everything from one server. We plot one curve for each different origin server (RTT_o), and a point on a curve shows the download time relative to the download time from the origin server if we had used a server with RTT given by the x-axis value for all the objects. The plot shows the download times for a client using parallel connections. We observed only very small differences between different file sizes.

In Figure 4 we show the relative download time of the selective redirection scheme for a client using parallel connections and for the Medium and Large pages. We plot one

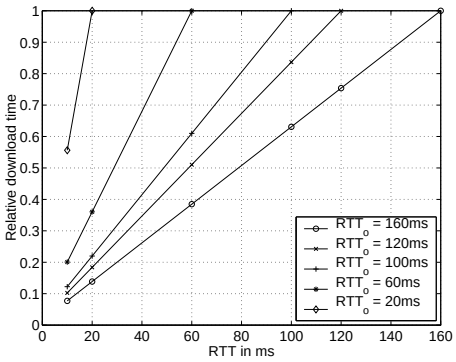
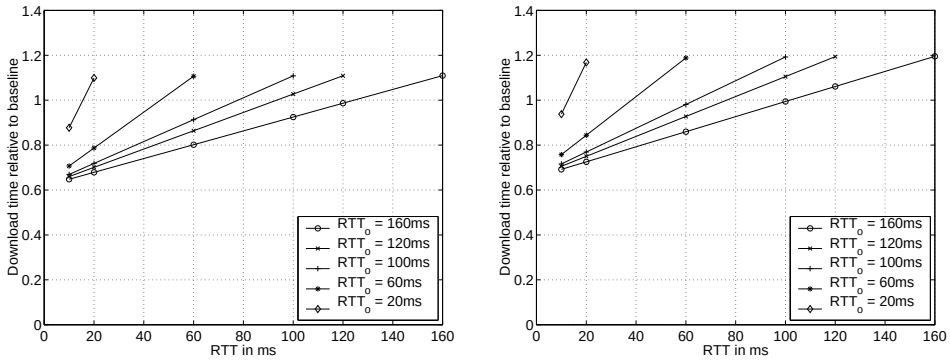


Figure 3: Performance of the baseline scheme

curve for each different origin server (RTT_o). A point on a curve shows the download time relative to the baseline that a client would obtain if the origin server had a round-trip time of RTT_o and the new server had the RTT on the x-axis. The baseline in these graphs refers to retrieving all object from the origin server with round-trip time RTT_o. As we can see, typically we need the RTT to the new server to be less than 75% of RTT_o in order for it to worth it to switch to the new server. In Figure 5 we show the results for a client using pipelining. We can see that in this case, the RTT to the new server should be less than 50% of RTT_o.



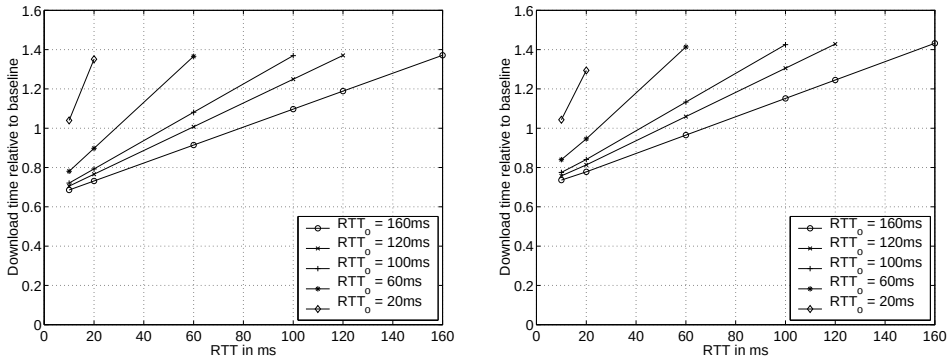
(a) Medium page

(b) Large page

Figure 4: Download times with parallel connections

Comparing Figures 4 and 5 we see that for parallel connections the slope of the curves is smaller, meaning that a small reduction in RTT only gets small reductions in total download time. For pipelining the slope is larger meaning larger gains for small reductions in RTT. Overall, we see that the maximum gain in download time is around 30% of the baseline. This is achieved when the new server is extremely close (RTT around 10 ms).

As we can see in Figure 3, if the origin server is slow (100 ms or more), we can get impressive gains if we were able to access a nearby server for all of the page. By choosing to go to a nearby server instead of the origin server for all the of the page we can download



(a) Medium page

(b) Large page

Figure 5: Download times with pipelining

it in 10% of the time it would have taken to get the page from the origin server. The possible gains of going to a nearby fast server are much greater than the achievable gains obtained with schemes where the client must switch servers during the download. This holds for both parallel persistent connections and pipelining connections. We also see that the schemes which switch servers (Figures 4 and 5) can obtain at maximum only a 30% reduction in download time. In the same situation, by going directly to the nearby fast server, the client would reduce the download time by 90%. In other words, even if the new server would be fast enough to warrant switching to it, the client would be much better off by going to that fast server already for all the objects. We can conclude that under good network conditions, switching servers during download *will give sub-optimal performance* compared to a scheme which redirects the clients to a good server for all the objects.

4.2 Simulations with Loss in Network

We then ran the same simulations using a 2% loss rate on all the links in the simulation. The results in all cases were similar to the ones obtained under loss free conditions. Figure 6 shows the performance of the baseline scheme, i.e., retrieving everything from the origin server with round-trip time RTT_o. As with the no-loss situation, the differences between different file sizes were very small.

When we compare Figures 3 and 6, we see that in the no-loss situation, the maximum gains are larger than in the situation where there is loss on the link. We believe this is because the underlying TCP connection is still in slow-start and therefore its RTT-estimate has not yet adapted well enough to the link RTT. Hence, the RTT-estimates for the links with small RTTs are too high and discovering a lost packet takes more time than it would if the TCP connection knew the RTT better.

In Figure 7 we show the relative download times of the selective redirection scheme for a client using parallel connections and Medium and Large pages averaged over 3000 simulation runs.

As we can see, the general form of the curves matches those obtained in loss free conditions (Figures 4 and 5). The only difference is that the point where switching servers would become useful is lower than in the loss free case. Furthermore, the maximum gain in

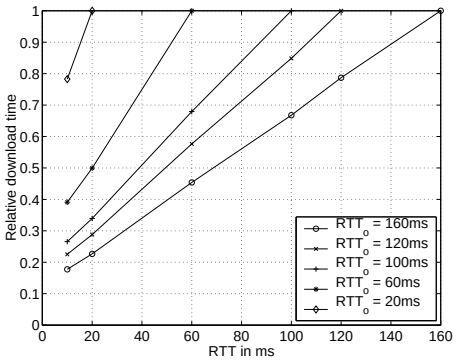
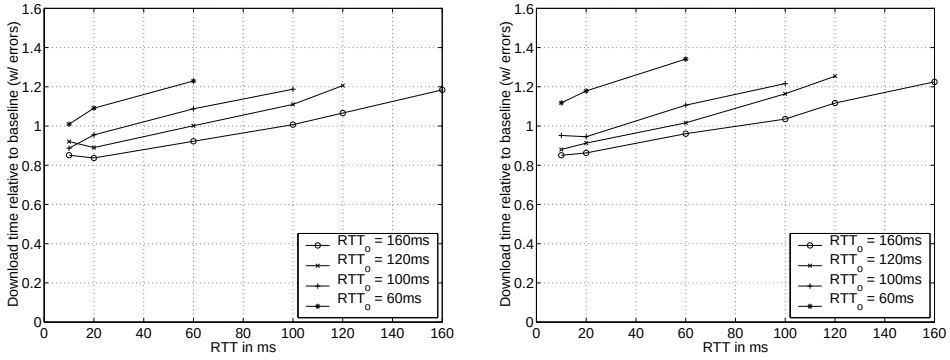


Figure 6: Performance of the baseline scheme with loss



(a) Medium page

(b) Large page

Figure 7: Download times with parallel connections with loss

download time is less than 20% and for Small pages switching servers always resulted in a slower download. We believe that the reason for the stricter RTT requirement for the new server is due to the possibility of losing TCP SYN packets when opening the connection to the new server. Most of the downloads took less than 2 seconds from start to finish, but a lost SYN packet caused a 6 second timeout. This slows down the connection to the new server considerably and gives a substantial advantage to using the persistent connection to the old server. Also, the TCP connections are in slow-start and do not therefore have an accurate estimate of the RTT and discovering lost packets will take longer on the new connection. If a packet on the persistent connection is lost, it will be discovered faster, either because of duplicate ACKs or because the TCP agents have an idea of the connection round-trip time and know when to expect packets.

The results confirm our conclusions from the loss free case. It is preferable *not to switch servers* during download of a single page. Switching servers greatly limits the gains in performance obtained from using a nearby server for all the objects. This means that *a system which forces clients to switch servers during the download of a single page will provide clients with sub-optimal service*; either the new server is not fast enough, or even if it is, the client should have been redirected to the fast server for all the objects.

5 Experiments

To validate the results obtained from our simulations, we ran several experiments in which we retrieved objects from real, replicated web sites. We used three different sites, Apache [3], Debian [5], and Squid [17], and chose several mirror servers of those sites for our experiments. From each of the three main sites we selected some files that matched the files in our simulations as closely as possible. For Apache and Squid our files were close to the Small page in Table 1, and for Debian they were similar to the Medium page.

Our goal was to have our client run like a modern browser, i.e., no pipelining. We chose one HTML file and one image file from each site and divided the servers in pairs. The client would first request the both files from both servers in the pair and then request the HTML from the first server the image from the other server in the pair. Before requesting the objects we performed a DNS query on the hostnames in order to eliminate the effects of long DNS lookups. We ran our experiments several hundred times during different times of day and on several days.

We show the results from 7 of our experiments in Table 2. In each experiment we used a different pair of servers to get as many different combinations as possible. The RTTs to the two servers shown in columns RTT_A and RTT_B reflect typical RTT values from our client machine to the servers in the experiment. We used server A as the baseline and show the relative download times for two other download schemes in columns AB and BB. Column AB refers to experiments where the client retrieved the HTML from server A and the image data from server B. In column BB we show the relative download time when the client retrieved everything from server B.

Experiment	$\mathbf{RTT_A}$	$\mathbf{RTT_B}$	**AB**	**BB**
Apache-1	80 ms	25 ms	0.77	0.14
Apache-2	60 ms	50 ms	1.34	0.58
Debian-1	100 ms	40 ms	0.98	0.25
Debian-2	180 ms	90 ms	0.97	0.72
Debian-3	80 ms	65 ms	1.26	0.79
Squid-1	200 ms	45 ms	0.73	0.20
Squid-2	70 ms	45 ms	1.03	0.59

Table 2: Results from experiments

The results we obtained in our experiments closely match those we obtained in our simulations. In fact, for all the experiments shown in Table 2, our simulations correctly estimated whether switching servers would result in a gain in relative download time or not.

6 Discussion

Our results show that the client can download a whole web page fastest if it is using persistent connections to a nearby content server (possibly using parallel persistent connections for embedded images). Of the two redirection schemes, full and selective redirection, full redirection achieves this goal easily since all requests to the origin server are redirected to a nearby content server.

With selective redirection it is possible to achieve the same effect by ensuring that all objects on a web page are replicated in the same way and thus client requests for the objects would be redirected to the same content server. This puts the burden of ensuring efficiency on the party deciding the replicated objects. Some modern systems put this responsibility on the *content provider* by allowing the content provider to tag individual objects for replication. We feel that ensuring efficient delivery of content to the clients should be the *responsibility of the content distributor*; in fact, efficient delivery of content is exactly the reason why content providers enlist the services of content distributors.

Our work is based on the assumption that the client is able to open persistent connections to all servers, although we do not assume pipelining of requests. Even though the content provider may run a web server that does not implement persistent connections, the content distributor can implement persistent connections in its own content servers. If the content provider's web server does not implement persistent connections then the RTT threshold for switching servers would be higher (because new connections to the origin server would go through slow-start again). If a suitable content server exists, the client would be better off using that server for all objects.

Our simulation model does not account for two important factors, namely server loads and redirection costs. A major reason for distributing content on several servers is to take off load from the origin server and distribute it among the content servers. Increasing the load on the origin server in our model would have the effect of making it more attractive to switch servers, i.e., it would make the RTT-threshold for switching lower. On the other hand, our model assumes that the client knows the address of the new content server. In reality, the client would have to obtain this address somehow before contacting the server. This would make switching servers less attractive, i.e., raise the RTT-threshold.

The cost of getting the redirection depends on the redirection technique used. If the system is using DNS redirection, then the client can expect to spend at least 200 ms for getting the address of the content server [9]. In the worst case, the DNS lookup can take several seconds to resolve. In our simulations and experiments all the downloads took only a few seconds at maximum and a long DNS lookup would have caused a significant slow-down for switching servers. If the client needs to contact the origin server to get the redirection, this would add at least one round-trip time to the origin server.

7 Conclusion

In this paper we have evaluated the performance of the different client redirection schemes used in modern content distribution networks. Using both simulations and experiments on the real network we have found that redirection schemes, which force clients to retrieve objects on a web page from multiple servers, always yields sub-optimal performance in terms of the overall client download time compared to schemes which allow the client to retrieve all objects from one, good server. This implies that when replicating web pages, we must treat the HTML-page and the embedded images as a single entity and replicate either all or none of them. Full redirection achieves this goal and yields superior performance compared to selective redirection which may split the web page between several servers.

In our future work we will expand our simulation models to include several clients and explore the effects of different network topologies on the results. We will improve our simulation models by including other parameters, such as server load. We will also do a more extensive set of experiments to validate our conclusions.

References

[1] Adero.<URL:http://www.adero.com/>.

[2] Akamai. <URL:http://www.akamai.com>.

[3] The Apache Software Foundation. <URL:http://www.apache.org/>.

[4] S. M. Baker and B. Moon. Distributed cooperative web servers. In *Proceedings of The Eighth International WWW Conference*, Toronto, Canada, May 11–14, 1999.

[5] Debian GNU/Linux web site. <URL:http://www.debian.org>.

[6] Digital Island inc. <URL:http://www.digisle.net>.

[7] R. T. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. *RFC 2616: Hypertext Transfer Protocol – HTTP/1.1*, June 1999.

[8] 100hot sites. <URL:http://www.hot100.com/>.

[9] J. Kangasharju and K. W. Ross. A replicated architecture for the domain name system. In *Proceedings of IEEE Infocom*, Tel Aviv, Israel, March 26–30 2000.

[10] J. Kangasharju, K. W. Ross, and J. W. Roberts. Locating copies of objects using the domain name system. In *Proceedings of the 4th Web Caching Workshop*, San Diego, CA, March 31 – April 2, 1999.

[11] Mirror Image Internet. <URL:http://www.mirror-image.com>.

[12] A distributed testbed for national information provisioning. <URL:http://ircache.nlanr.net/>.

[13] UCB/LBNL/VINT network simulator - ns (version 2). <URL:http://www-mash.cs.berkeley.edu/ns/>.

[14] H. F. Nielsen, J. Gettys, A. Baird-Smith, E. Prud'hommeaux, H. W. Lie, and C. Lilley. Network performance effects of HTTP/1.1, CSS1, and PNG. In *Proceedings of ACM SIGCOMM*, Cannes, France, September 14–18 1997.

[15] M. Rabinovich and A. Aggarwal. RaDaR: A scalable architecture for a global web hosting service. In *Proceedings of The Eighth International WWW Conference*, Toronto, Canada, May 11–14, 1999.

[16] P. Rodriguez, A. Kirpal, and E. W. Biersack. Parallel-access for mirror sites in the internet. In *Proceedings of IEEE Infocom*, Tel Aviv, Israel, March 26–30 2000.

[17] Squid Internet Object Cache. <URL:http://squid.nlanr.net/>.

[18] Z. Wang and P. Cao. Persistent connection behavior of popular browsers. <URL:http://www.cs.wisc.edu/~cao/papers/persistent-connection.html>.

Vitae

Jussi Kangasharju received his Master of Science in Technology from the Department of Computer Science and Engineering in Helsinki University of Technology, Finland, in 1998. He received his DEA from Ecole Superieure des Sciences Informatiques (ESSI), France in 1998. He is currently pursuing PhD studies at University of Nice (Sophia Antipolis) on web content distribution in the Multimedia Communications Department of Eurecom. His research interests include content distribution, web caching, and Internet protocols.

Keith Ross received his Ph.D. from the University of Michigan in 1985 (Program in Computer, Information and Control Engineering). He was a professor at the University of Pennsylvania from 1985 through 1997. At the University of Pennsylvania, his primary appointment was in the Department of Systems Engineering and his secondary appointment was in the Wharton School. He joined the Multimedia Communications Dept. at Institut Eurecom in January 1998, and became department chairman in October 1998. In Fall 1999, while remaining a professor at Institut Eurecom, he co-founded and became CEO of Wimba.com. Keith Ross has published over 50 papers and written two books. He has served on editorial boards of five major journals, and has served on the program committees of major networking conferences, including Infocom and Sigcomm. He has supervised more than ten Ph.D. theses. His research and teaching interests include multimedia networking, asynchronous learning, Web caching, streaming audio and video, and traffic modeling.

Jim Roberts has a BSc in mathematics from the University of Surrey, UK and a PhD from the University of Paris. Since graduating in 1970, he has worked in the field of teletraffic engineering and performance evaluation. He is currently with France Telecom R&D. His research has been mainly in the field of multiservice networks, ISDN, ATM and IP. He is author of many papers in this field and editor of the book "Broadband Network Teletraffic", the final report of the European COST 257 project which he chaired. He received the best paper award for Infocom 99. He is a member of a several journal editorial boards and conference programme committees in the networking field and participates in standardization activities at the ITU.