Transparent Distributed Web Caching with Minimum Expected Response Time

Qing Zou, Patrick Martin and Hossam Hassanein

School of Computing
Queen's University
Kingston, Ontario, Canada K7L 3N6
{zou, martin, Hossam}@cs.queensu.ca

Abstract

Web caching is a standard approach to improving the performance and quality of Web services. The effectiveness of a single cache in this environment, however, is relatively low. Caches hit rates of 40% or lower are typical in the Web. Distributed caching seeks to improve the effectiveness of Web caching by supporting the sharing of data across multiple cache servers.

In this paper we describe the Minimum expected Response Time (MRT) distributed Web caching scheme. MRT uses a Layer 5 switch to transparently redirect cacheable HTTP requests to the cache server with the minimum expected response time. The response time estimate is produced based on information about cache server content, cache server workload, Web server workload and network latency. We present simulation experiments to show that MRT outperforms existing distributed Web caching schemes in terms of average HTTP request response times.

1 Introduction

HTTP Web traffic has grown to account to 75-80% of all Internet traffic [15] and this popularity is raising an urgent need for ways to improve the quality of the Web services. Web caching is one of the most popular approaches to this problem [15]. The effectiveness of a single cache, however, remains poor as it is, in general, no higher than 40% [1].

This research is supported by Communications and Information Technology Ontario (CITO) and the Kingston Software Factory.

One way to improve Web caching performance is to expand from a single cache to a set of cooperative caches that share data objects [9,17]. The most popular types of cooperative cache systems are the hierarchical and the distributed systems. Hierarchical cache server architectures [9] define parent and sibling relationships among the cache servers. If a cache server cannot service an HTTP request it first checks if any of its siblings have the required page. If none of them do then the cache server passes the request to its parent, which repeats the process. Eventually, if the root cache server does not have the requested page then it passes the request to the Web server. Distributed cache server architectures [13,15], on the other hand, allow all cache servers to share data in a peer-to-peer relationship. Recently, a new type of Web caching technique, namely switching-based transparent Web caching [16], can use content-aware Layer 5 switches in a distributed Web caching system to further improve performance by providing enhanced cache cooperation [6].

To the best of our knowledge, there are no effective approaches that optimize the performance of distributed transparent Web caching systems in terms of HTTP request response time. In this paper, we describe the Minimum expected Response Time (MRT) switching-based Web caching scheme, which attempts to minimize the HTTP request response time and to balance the workload among the caches based on a combination of request content, cache server content, network latency and server workload.

The rest of the paper is organized as follows. Section 2 surveys related work on distributed Web caching schemes. Section 3 describes the MRT scheme and Section 4 discusses simulation results on the performance of MRT. Finally, Section 5 presents our conclusions.

0-7803-7893-8/03/$17.00 © 2003 IEEE

2 Related Work

The design of efficient server selection algorithms is critical for distributed Web caching schemes. From the perspective of the Web clients, the request response time is an essential component of quality of the Web caching systems. It is related to the network load, network propagation delays, server (cache and Web) load and server speed. Fluctuations in network congestion and server load, however, make it difficult to collect the information.

Some existing Web caching schemes, such as the Internet Cache Protocol (ICP) [14], use the round trip time (RTT) to approximate the HTTP request response time and route the request to the server with the minimum round trip time. The RTT reflects the actual network load on the route between the client and the server. One drawback of this approach is that cache servers do the server selection, which adds load to cache servers such that they cannot efficiently process and forward packets. Another drawback is that the ping round trip time does not provide any indication of the cache server load and the speed of the cache server. The correlation between the round trip time and the HTTP request response time has been found to be low and not indicative of the request response time [12].

Switching-based transparent Web caching schemes have several advantages [2,3,8]. First, a switch is optimized for examining and processing packets. Second, removing the packet examination, network address translation and routing functions from the cache server frees up resources for serving Web pages. Switches perform the redirection more rapidly and efficiently than cache servers do. Moreover, layer 5 switches can route requests based on application level information such as the URL. The integration of layer 5 switching-based transparent caching with distributed systems provides enhanced performance. However, among all existing switched-based Web schemes there are no effective methods to simultaneously estimate the actual HTTP request response time and balance the workload among the cache servers in a distributed web caching system.

Liang et. al. [6] propose a fully distributed Web caching scheme, called Load-Balancing L5 (LB_L5), that extends the capabilities of Layer 5 switches to improve the response time and balance cache server workload. In LB_L5, a Layer 5 switch selects the best server based on cache content, cache server workload, network load and the HTTP header information. If the network latency between a cache server that stores a requested object and the Layer 5 switch is smaller than some threshold, then that cache server is considered as a candidate for access. The Layer 5 switch then uses load balancing algorithms to choose the best server from which to retrieve the object. The drawbacks of this approach are that it is difficult to set the threshold value and it cannot guarantee the minimum request response time.

3 Minimum Expected Response Time Scheme

The Minimum expected Response Time (MRT) scheme is a transparent distributed Web caching scheme that uses the client request header, cache server content, cache server workload, Web server workload and network latency to redirect cacheable requests to the most appropriate cache server. The goal of MRT is to optimize the performance of distributed Web caching systems by achieving minimum average response time and balanced load. The structure of MRT is shown in Figure 1.

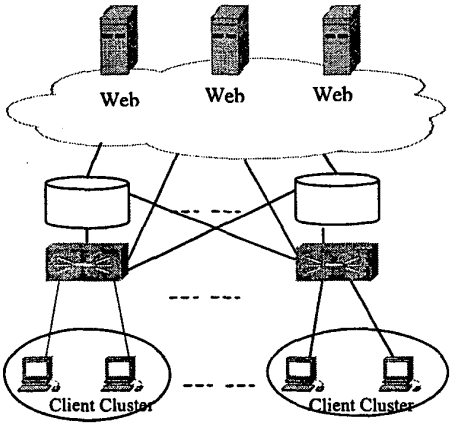


Figure 1: Distributed Switching-Based Transparent Web Caching System

3.1 Cache Server Selection Algorithm

A switch determines the cacheability of a requested object using the URL in the request's HTTP header. Non-cacheable requests are sent directly to the Web servers. A switch selects the best cache server to receive a cacheable request based on the switch's knowledge of the cache servers' content and workload, Web servers' workload and network latency.

Cache servers periodically send information about their cache content to the switches using a Bloom Filter, which is a bit string representation of the contents [10]. An object in the cache is represented by a collection of bits set to 1 in the Bloom Filter. Several independent hash functions are computed for the object's key (URL) to determine the bits set to 1. A switch uses the Bloom Filter to predict whether or not a requested object is stored in a server's cache. The same hash functions are

applied to the requested object's URL and, if all of the corresponding bits are set to 1, it predicts that the object is in the cache.

A Bloom Filter, because it has a limited number of bits, always has a non-zero number of false hits, that is predictions an object is in a cache when it is not. For a Bloom Filter of F bits with W hash functions and that represents D objects, the false hit probability Fp is given by [4]

$$Fp = (1 - e^{-WD/F})^W$$

Since a switch may find a requested object in more than one cache server, MRT selects one server from among them based on their expected response times to the request. It estimates the expected response time for a request to cache server CS, RT_{cs}, as

$$RT_{cs} = P\text{-}miss_{cs} * T\text{-}miss_{cs} + (1\text{-}P\text{-}miss_{cs}) * T\text{-}hit_{cs}$$

where $P\text{-}miss_{cs}$ is the false hit probability Fp for cache sever CS, $T\text{-}miss_{cs}$ is the time required to fetch an object in the case of a false prediction and $T\text{-}hit_{cs}$ is the time to fetch an object in the case of a true prediction.

When there is a true cache hit for a requested object, The cache-hit request response time, $T\text{-}hit_{cs}$, is the sum of the times for the individual steps, that is

$$T\text{-}hit_{cs} = 2*RTT\text{-}SW_{cs} + CacheServerProcessingTime$$

After a switch receives a HTTP request from the Web client the following steps occur:

(1) The switch sends a TCP-SYN signal to a Proxy Cache Server for a connection request. The Cache Server sends back a TCP_ACK to accept the connection. The time spent is the round trip time between the switch and the cache server ($RTT\text{-}SW_{cs}$).
(2) The switch relays the HTTP request to the cache server. The time required is half of $RTT\text{-}SW_{cs}$.
(3) The cache server processes the request. The processing time, *CacheServerProcessingTime,* is proportional to the cache server's workload. This assumption is supported by the data collected by Rousskov [11] and also used by Liang [6].
(4) Since the request is a cache hit, the cache server immediately relays the requested objects to the switch. The time spent is half of $RTT\text{-}SW_{cs}$.

When there is a false cache hit for a requested object, The cache-miss request response time, $T\text{-}miss_{cs}$, is the sum of the times for the individual steps, that is

$$T\text{-}miss_{cs} = 2*RTT\text{-}SW_{cs} + CacheServerProcessingTime + 2*RTT\text{-}WS_{cs} + WebServerProcessingTime$$

After a switch receives a HTTP request from the Web client the following steps occur:

(1) The switch sends a TCP-SYN signal to a Proxy Cache Server for connection request. The Cache Server sends back TCP_ACK to accept the connection. The time spent is the round trip time between the switch and the cache server ($RTT\text{-}SW_{cs}$).
(2) The switch relays the HTTP request to the cache server. The time required is half of $RTT\text{-}SW_{cs}$.
(3) The cache server processes the request. The time, *CacheServerProcessingTime,* is proportional to the cache server's workload.
(4) Since the request is a cache-miss, the cache server makes a TCP connection request to the original Web server. The Web server sends back TCP_ACK to accept the connection. The time spent is the round trip time between the cache server and the Web server ($RTT\text{-}WS_{cs}$).
(5) The cache server relays the HTTP request to the Web server. The time spent is half of $RTT\text{-}WS_{cs}$.
(6) The Web server processes the request. The time, *WebServerProcessingTime,* is proportional to the Web server workload.
(7) The Web server sends back the requested object to the cache server. The cache server receives the object and stores a copy. The time spent is half of $RTT\text{-}WS_{cs}$.
(8) The cache server immediately relays the object to the switch. The time spent is half of $RTT\text{-}SW_{cs}$.

Upon receiving a HTTP request, MRT-capable switches estimate the actual response time RT_{cs} for every cache server cs, then redirect the request to the cache server resulting in the minimum response time estimate.

3.2 Extended ICP Messages

A switch in MRT obtains information on cache server contents, cache server workload, and Web server load via the following four extended ICP messages [19]:

- A cache server periodically informs the switches about changes in its cache content using an **ICP_UPDATE_CONTENT** message.
- A switch acknowledges an ICP_UPDATE_CONTENT message with an **ICP_UPDATE_CONTENT_ACK** message.
- A switch periodically queries the cache servers for workload information using an **ICP_QUERY_WORKLOAD** message.
- A cache server sends its workload information to the switch in response to an ICP_QUERY_WORKLOAD message using an **ICP_UPDATE_WORKLOAD** message. A Web server can also use an ICP_UPDATE_WORKLOAD message to send its workload information to the switches.

Network latency information is obtained by measuring the round trip delay between a switch and a cache server or between a cache server and a Web server.

4 Performance Evaluation

We compare MRT, via simulation, with other approaches used to redirect requests in distributed switching-based transparent Web caching systems. We show that MRT optimizes the performance of distributed switching-based transparent Web caching systems in terms of HTTP request response time.

4.1 Simulation Model

Our simulation's network model is based on a symmetric architecture with n client clusters, n switches and n cache servers, as shown in Figure 1. n varies from 2 to 8 in our simulation runs. Each client cluster is connected to one of the switches. Each switch can communicate with all cache servers and all Web servers. We assume that the cost of transferring data between two nodes is proportional to the distance between that pair of nodes. We calculate the *network latency* between node i and node j, which is the cost of transferring data between them in milliseconds, as

$$NetworkLatency\ (i,j) = Distance\ (i,j) * LatencyFactor$$

where $Distance\ (i, j) = |i-j|$ and *LatencyFactor* is the time (in milliseconds) spent on the network when data are transferred for one unit of distance. We vary *LatencyFactor* in the simulations.

We use publicly available proxy traces from the NLANR to generate HTTP requests [7]. Client IP addresses are randomized daily and are consistent within a trace but not between traces. Each trace spans 24 hours and contains from 100,000 to 400,000 total requests. We measure the average workload of a server (cache server or Web server) in our simulation as

$$AverageWorkload = \frac{AvgTCPNum}{MaxTCPNum}$$

where *AvgTCPNum* is the average number of TCP connections per second and *MaxTCPNum* is the maximum number of TCP connections per second. *AvgTCPNum* is calculated as

$$AvgTCPNum = (1-W_q) * AvgTCPNum + W_q * TCPNum$$

where *TCPNum* is the active number of TCP connections and W_q is a weight factor.

If W_q is too large, then the averaging procedure may not filter out transient congestion or busty traffic. If W_q is too low, then the average number of TCP connections responds too slowly to changes in the actual number of TCP connections. We ran our simulation with W_q as 0.001, 0.002 and 0.25 and found that $W_q = 0.002$ is a reasonable choice.

All known Web caching systems use validation check mechanisms to maintain cache consistency [7]. Some widely used proxy servers, such as the World Wide Web Consortium's httpd [18], use an *expiration mechanism* to keep their pages up to date. Expiration-based cache servers use a variety of mechanisms to assign expiration times for cached pages. Our simulation uses a simple expiration time algorithm in which each object is assigned an expiration time equal to ¼ to ½ of the expected lifetime of that type of object. Zou [19] provides further details of the simulation program. The parameter settings used in our simulations are similar to those used in Liang [6] and Rousskov [11].

4.2 Cache Server Selection Schemes

We compare MRT with the following cache server selection schemes:

- **Content**: The Content scheme represents a class of selection algorithms that redirect HTTP requests based on the contents of cache servers. If more than one cache server may hold a requested object then the scheme chooses one of them at random.
- **Workload**: The Workload scheme represents a class of selection algorithms that redirect HTTP requests based on both the contents and the workloads of cache servers. If more than one cache server may hold a requested object then the scheme selects the one with the minimum workload.
- **RTT**: The RTT scheme represents a class of selection algorithms that redirect HTTP requests based on both contents of the cache servers and network latency. If more than one cache server may hold a requested object then RTT chooses the one with the minimum communication time.
- **LB_L5**: The LB_L5 scheme, as described earlier, redirects HTTP requests based on cache server content and workload, and network latency.

4.3 Simulation Results

We present two sets of experiments that investigate the effect of network latency and HTTP request intensity on the performance of the cache server selection algorithms. In the set of experiments involving network latency, we use raw-trace simulations in which the HTTP requests are generated directly from the NLANR trace files. In the set of experiments involving HTTP request intensity, we use controlled-trace simulations in which the request inter-arrival times in the raw trace files are modified to vary the HTTP request intensity. A

statistical analysis of the experiments results reveals that the performance is quite stable. The experiments were run with a 90% confidence level with 5% confidence intervals.

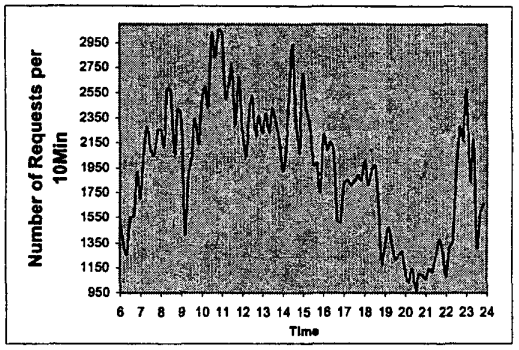


Figure 2: HTTP Request Intensity

4.3.1 Effect of Network Latency

We ran the raw-trace simulations with two NLANR 24-hour trace files. Figure 2 shows the HTTP request intensity in 10-minute periods from one of the files. The request intensity is relatively high from 8 am to 4 pm. The peak intensity is 2950 requests per 10 minutes around 11 am, and significantly decreases after 5 pm. The minimum intensity is 950 requests per 10 minutes around 8 pm. The first six hours are used as a warm-up period in our experiments and so are not included in any of the results.

We compared the response times of MRT to the other schemes under different network latencies. When the network latency is low the workload of cache servers is the main factor affecting the response time. When the workload of the cache server is light, the network latency becomes the main factor affecting the response time. The number of HTTP requests generated by client clusters determines the workload of a cache server.

The response times for the workload in Figure 2 with network latency factors of 5 ms and 125 ms are shown in Figures 3 and 4, respectively. Similar results wereobtained for other network latency factors and for the other trace file. The simulation results show that MRT outperforms the other four schemes and adapts better to high HTTP request intensities and large network latencies.

Figure 3 shows that, when the network latency is small, the response times of the five schemes follow the HTTP request intensity. The Workload scheme, LB_L5 scheme and MRT scheme consider the workload at the cache servers so they have better performance than the Content and the RTT schemes when the request intensity is high. At peak request intensity (10:50 am) MRT outperforms Content by 22% and RTT by 25%. MRT has similar performance to the Workload and LB_L5 schemes.

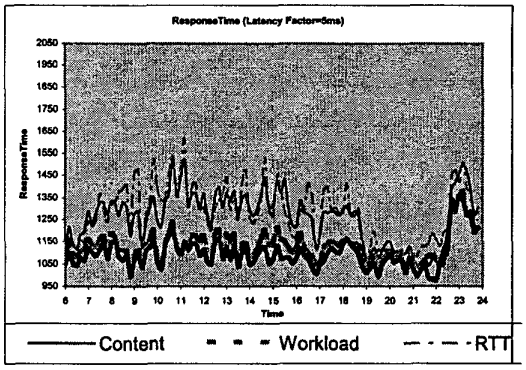


Figure 3: Response Time At Latency Factor = 5ms

Figure 4 shows that, when the network latency is large, both LB_L5 and MRT avoid redirecting requests to remote cache servers. MRT outperforms LB_L5 by 11%. It outperforms Content by 44% and Workload by 34% since neither of them considers network latency when making their routing decisions. MRT outperforms RTT by 30%. Even if all the cache servers storing the requested object have large network latency RTT has to redirect the request to one of them so the average response time of RTT increases as the network latency increases.

4.3.2 Effect of HTTP Request Intensity

The controlled-trace driven simulation experiments investigate the effect of the HTTP request intensity on the response time of MRT and the other schemes. Each client cluster sends the same number of HTTP requests per 10-minute period in all the experiments. We simulate different HTTP request intensities by varying the request inter-arrival times in the input traces. For example, 50% request intensity lengthens the mean inter-arrival time by 2 and 200% request intensity shortens the mean inter-arrival time by half.

Figure 5 illustrates that MRT outperforms the other schemes under a variety of request intensities and network latency factors. We see that, in general, the average HTTP request response times of all the schemes increase as the HTTP request intensity increases for all network latency factors. As the HTTP intensity increases, the workload of each cache server increases which in turn increases the processing time per request at the cache servers. MRT and LB_L5, however, are

best able to adapt to higher request intensities under all network latencies because they consider both cache server workload and network latency. MRT achieves better average response times than LB_L5 because it considers them in an integrated fashion.

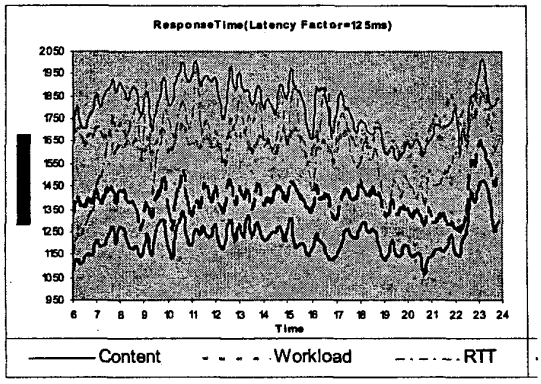


Figure 4: Response Time At Latency Factor = 125 ms

Content and RTT have the largest increases in average response times as HTTP request intensity increases, under all network latencies, because neither scheme uses cache server workload information when making their routing decisions. They both may distribute load unevenly on the cache servers even though requests are balanced across the client-clusters. The load imbalance on the cache servers becomes more pronounced as HTTP request intensity increases.

The improvement in response time for the workload-aware schemes, for all network latencies, becomes more significant as the request intensity increases. The average increase in response time across all network latencies is around 1400 milliseconds for both Content and RTT when the HTTP request intensity increases from 50% to 250%. The average increase of response time for Workload, LB_L5 and MRT in the same circumstances is only around 700 milliseconds. Workload's rate of increase in response time becomes larger, however, for larger network latencies because Workload fails to take into account the fact that communication costs are a larger portion of the response times.

MRT has knowledge of the network latency and server workload and uses this knowledge to predict the request response time by estimating the impact of workload and network latency. It chooses the server that can service the request fastest. MRT has good performance when the Web caching system has high request intensity or large network latency. When the request intensity is 50% and the network latency factor is 5 milliseconds the average response time of MRT is similar to the Content, Workload, RTT and LB_L5 schemes. However, when the request intensity is 250% and the network latency factor is 125 milliseconds, the average response time of MRT is lower than that of Content, Workload, RTT and LB_L5 by 70%, 34%, 49% and 9%, respectively.

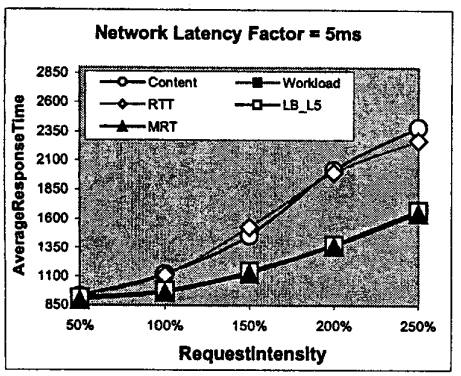
5 Conclusions

In this paper we study the problem of optimizing HTTP request response time performance in distributed switching-based transparent Web caching systems. We present a heuristic solution to the problem called the Minimum expected Response Time (MRT) scheme. MRT uses a Layer 5 switch to transparently redirect cacheable HTTP requests to the cache server with the minimum expected response time. The response time estimate is produced based on information about cache server content, cache server workload, Web server workload and network latency.

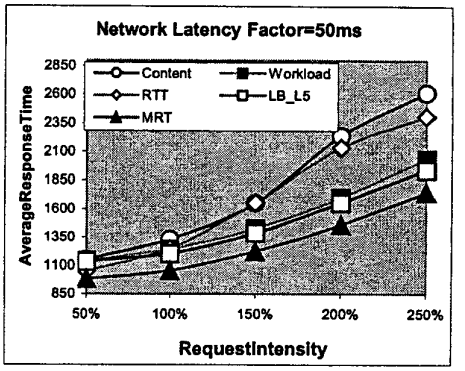
We describe a simulation model that was developed to study the performance of MRT and evaluate the performance of MRT using the model. We compare MRT with three generic schemes based on Content, Workload and RTT information, respectively, and with the LB_L5 Web caching scheme. Our experiments show that MRT outperforms all the other schemes in terms of HTTP request response time. They also indicate that MRT is more scalable than the other schemes since it is better able to adapt to high HTTP request intensity and large network latency.

We plan to expand our work with MRT into two main areas. The first area is the impact of large objects on cache server performance. The size of an object requested affects the time due to server delay, which is incurred retrieving the object from the disk, and the time due to transmission latency. These times may have a significant effect on the request response time if most of the requested objects are large objects, such as Video or Audio documents. Request response times cannot be compared directly if objects are of different size.

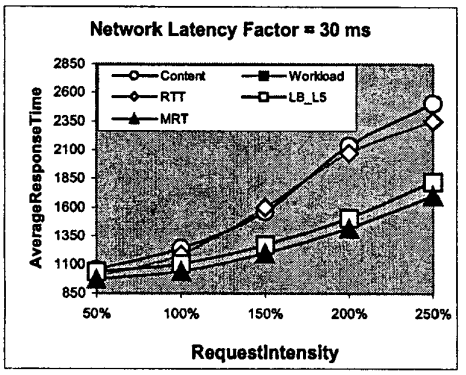
The second area is different request priorities, or differentiated caching. The Layer 5 switches used in MRT can distinguish different types of requested objects, such as multimedia objects or image objects, and can assign priorities to requests for different types of objects. We plan to investigate the impact prioritized service can have on response times and on meeting Quality of Service demands.



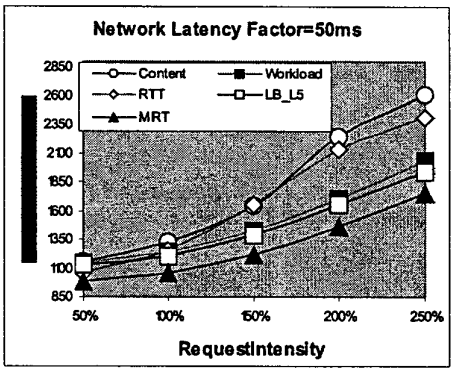
(a)



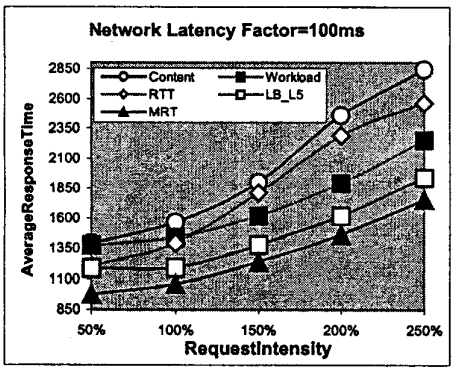
(d)



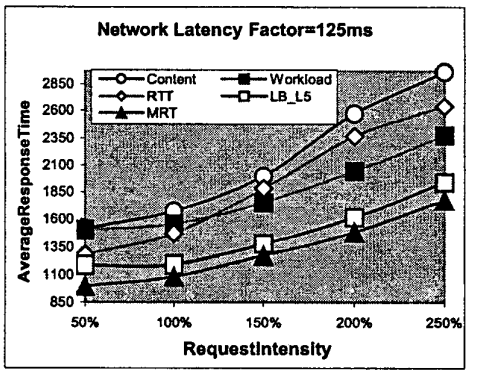
(b)



(e)



(c)



(f)

Figure 5: Average Response Time Versus HTTP Request Intensity

References

[1] M. Abrams, C.R. Standridge, G. Abdulla, S. Williams, and E.A. Fox, "Caching Proxies: Limitations and Potentials," *Proceedings of the 4th International World-Wide Web Conference.* 1995.

[2] G. Apostolopoulos, V. Peris, P. Pradhan, and D. Saha, *L5: A Self Learning Layer 5 Switch*, Technical Report RC21461, IBM T.J.Watson Research Center, 1999.

[3] Cisco Systems Inc. *Cisco CSS-11150 - Content Services Switch*, Available at http://www.cisco.com/warp/public/cc/pd/si/11000/index.shtml, January 2002.

[4] C. Faloutsos and S. Christodoulakis, "Design of a Signature File Method that Accounts for Non-Uniform Occurrence and Query Frequencies," *Proceedings of 11th International Conference on Very Large Databases (VLDB)*, pp. 165-170, Stockholm, Sweden, August 1985.

[5] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *ITTT/ACM Transactions on Networking*, August 1993.

[6] Z. Liang, H. Hassanein and P.Martin, "Transparent Distributed Web Caching," *Proceedings of the IEEE Local Computer Network Conference*, pp.225-233, November 2001.

[7] National Laboratory for Applied Network Research (NLANR), *Ircache project*. Available at http://ircache.nlanr.net/, January 2002.

[8] Nortel Networks Ltd., *Alteon Websystems ACEdirector*, Available at http://www.nortelnetworks.com/products/library/collateral/intel_int/acedirector.pdf, February 2002.

[9] P. Rodriguez, C. Spanner, and E. Biersack, "Web Caching Architectures: Hierarchical and Distributed Caching," *Proceedings of the 4th International Web Caching Workshop*, April 1999.

[10] A. Rousskov and D. Wessels, "Cache Digests," *Proceedings of the Third International WWW Caching Workshop*, Manchester England, June 1998.

[11] A. Rousskov and V. Soloviev, "On the Performance of Caching Proxies," *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'98/PERFORMANCE '98)*, pp. 272-273, Madison, WI, June 1998.

[12] M. Sayal, Y. Breitbart, P. Scheuermann, and R. Vingralek, "Selection Algorithms for Replicated Web Servers," *Performance Evaluation Review* 26(3), pp. 44-50, December 1998.

[13] R. Tewari, M. Dahlin, H.M.Vin, and J.S. Kay, "Design Considerations for Distributed Caching on the Internet," *Proceedings of the 17th International Conference on Distributed Computing Systems (ICDS'99)*, 1999.

[14] D. Wessels and K. Claffy, *Internet Cache Protocol (ICP), version2*, IETF RFC 2186, September 1997.

[15] D. Wessels and K. Claffy, "ICP and the Squid Web Cache," *IEEE Journal on Selected Areas in Communication* 16(3), pp. 345-357, April 1998.

[16] B.Williams, "Transparent Web Caching Solutions," *Proceedings of the 3rd International WWW Caching Workshop*, 1998.

[17] A. Wolman, G.Voelker, N.Sharma, N. Cardwell, A.Karlin, and H. Levy, "On the Scale and Performance of Cooperative Web Proxy Caching," *Proceedings of the 17th Symposium on Operating Systems Principles*, December 1999.

[18] World Wide Web Consortium. *CERN httpd*, at http://www.w3.org/pub/WWW/Daemon, February 2002.

[19] Q. Zou, *Transparent Web Caching with Minimum Response Time*, M.Sc. Thesis, Department of Computing and Information Science, Queen's University, Kingston ON, January 2002.