One size doesn't fit all: Improving network QoS through preference-driven Web caching

Yee Man Chan
ymc@eecs.umich.edu
Electrical Engineering &
Computer Science
University of Michigan
Ann Arbor, MI 48109

Jonathan P. Womer
jwomer@umich.edu
School of Information
University of Michigan
Ann Arbor, MI 48109

Jeffrey K. MacKie-Mason
jmm@umich.edu
School of Information &
Department of Economics
University of Michigan
Ann Arbor, MI 48109

Sugih Jamin
jamin@eecs.umich.edu
Electrical Engineering &
Computer Science
University of Michigan
Ann Arbor, MI 48109

Abstract

In order to combat Internet congestion Web caches use replacement policies that attempt to keep the objects in a cache that are most likely to get requested in the future. We adopt the economic perspective that the objects with the greatest value to the users should be in a cache. Using trace driven simulations we implement an incentive compatible market-based Web cache for servers to push content into a cache. This system decentralizes the caching process as servers provide information in the form of bids for space in the cache. Truthful information from the server on valuations of objects and predictions of hit rates is obtained. This information is used in filling the cache, which can provide increased aggregate value and differential quality of service to servers when compared to LFU and LRU.

1. Introduction:

The Internet has shifted in recent years from a network dedicated to research to a network of popular culture and everyday commerce. With this shift the volume of traffic has increased dramatically. Congestion is often a problem in areas where, and times of day when bandwidth is scarce. Distributed file storage, which we refer to as "caching" is one method for improving the quality of service (QoS).

There are two resource management problems for caching: when to refresh stored objects (to keep them current with sources that might change), and which objects to replace when adding new objects to a full cache. Our research is concerned with the latter problem: replacement policy. Traditional caching has been based on Least Frequently Used (LFU) and Least Recently Used (LRU) replacement policies. Every time a request is made that the cache cannot fill the cache pulls the object from the host server and saves a copy in the cache so it may be accessed by the next request. When full, an LFU cache drops the least frequently used objects until there is enough room for the new object. Similarly, the least recently used objects are dropped first in LRU. LRU has the effect of sorting a cache by last access time and LFU has the effect of sorting by the number of requests (Williams et al.). They are simple algorithms developed, in part, to keep computational time to a minimum. Today computational cost is of less concern as processor prices continue to fall rapidly.

Traditional methods treat all requests equally. Their implicit goal is to increase the hit rate of the cache and in turn reduce the amount of bandwidth used. However, there is no compelling reason to think that aggregate hit rate is the best measure of shared cache value. Discussions in industry suggest that end user satisfaction with performance and QoS are the primary drivers for caching (Oaks). Indeed, we start from the position that a network component such as a cache is only as valuable as its users think it is. By treating all requests equally traditional caching replacement policies may not optimize this objective. Different users are likely to place substantially different values on the delay associated with retrieving Web objects. This assertion of substantial preference heterogeneity is supported by the substantial variation in the prices of network connections with different speeds. Suppose user 1 obtains greater benefit than user 2 from rapid file service. The ex ante probability of finding their objects in a cache would be unaffected by their preferences in a traditional scheme. In order to maximize the *aggregate user value* (rather than aggregate hit rate), objects that user 1 requests should have higher weights for remaining in the cache.

There is a second problem with traditional cache replacement policies. The goal of a replacement algorithm is to forecast accurately those objects that will have the highest value if they are waiting in the cache for *future* requests. Most algorithms are backward looking: they use an index based on past request streams to create the sort list for replacement priority. Such policies by their nature have difficulty anticipating shifts in users' preferences.[1] Preceding a large sporting event or the release of a new software version, pictures or software could be cached closer to users knowing that there would likely be a rise in demand for these objects.

Generally, then, we perceive two potential QoS improvements from incorporating users into the replacement policy algorithm. First, users might communicate usable information about their differential preferences for finding objects in the cache. Second, users might communicate better forecast information about future demands for objects. As a related point, it might be that distributing (part of) the forecasting function to many users might allow for much more sophisticated data mining and forecasting models than would be possible in real time from a centralized cache processor. Our first contribution is to demonstrate with simulations based on actual Web request trace data the potential value improvement possible through incorporating user-based forecasting into caching algorithms.

To design a caching replacement policy that relies on user-provided information about future object demand and relative valuations for service delay we need to solve a problem: how do we induce users to provide the valuation and forecasting information? It is well known from the theory of incentives that the cache manager cannot simply ask users to report their valuation for different service qualities and their forecasts. If the replacement policy of the cache depends on these reports, then users have an incentive to lie to obtain better performance. For example, while a cache hit may be worth only $0.01 to me, without appropriate incentives, I might report that it is worth $1.00 to me, in order to increase the likelihood that the object is waiting in the cache.

[1] There are more sophisticated ways to forecast future preferences based on past data than those embodied in the LFU and LRU algorithms. For example, one could estimate modeling time of day and day of week effects in past request streams.

To obtain the user valuation and forecasting information required for our caching algorithms, we need to provide incentives for revelation. Incentives are generally provided through the *service policy*. That is, what service is delivered to a user, and how much does that user have to pay for the service? By designing the service delivered to vary appropriately with the valuation and forecasting information the users provides, we can induce users to reveal the information we need for the replacement policy. Thus, service and replacement policies become necessarily intertwined, and we have to design them together.

To support a user-driven replacement policy, we need a service policy that obtains distributed, private information about valuations and future usage. One well-known class of decision mechanisms is specifically adapted to assigning heterogeneous preference weights to objects, to anticipating shifts in demand, and to stocking items in anticipation of their use: markets. The market mechanism we have studied is for control over the replacement policy for some portion of the space in the cache. We see two main types of agents involved in a market-based cache, servers and clients. Both act as value aggregators of the end users of web objects. Servers are interested in pushing their content into the cache to reduce the load on their network connections and to reduce the latency experienced by end users of the server's content. Clients represent institutions or ISPs that are concerned with buying real estate in the cache dedicated to saving material based on the preferences of their end users. There are several potential advantages to using a market approach. Decentralization is important not only for the reduction in communication but it allows such a mechanism to operate on a distributed network like the Internet. These agents are motivated by their own profit maximization. Their profit is directly connected to the utility of their users, ensuring the most highly demanded content is in the cache.

Thus, our second main contribution is to show how a market-based service policy could be designed to implement a user-directed replacement policy. Market mechanisms have been shown useful in the past on other similar issues when decentralized resource allocation is needed, such as reducing network congestion (MacKie-Mason and Varian). The use of markets in the building of computational economies is another example of advantage that gained by decentralized resource allocation in a distributed environment (Kurose and Simha; Wellman). Waldspurger et al. focuses on distributed agents buying processor time (Waldspurger et al.). In their system the agents have a clear idea of value of processor speed but in this paper an agent's value of cache space depends on others; how many times the space is accessed by end users. Other research has been done combining economic principles and caching (Karaul, Korilis and Orda). It focuses on distributing requests to replicated servers based on market mechanisms. A recent dissertation describes the larger situation of market driven caches and networks (see chapter 4) but does not look at a singular cache market in detail (Chuang). A recent paper focuses on weighting LRU by a cost metric (Cao and Irani). However, the cost is based on congestion and hardware prices not user valuation or the welfare of the system. In addition, by using LRU the paper makes assumptions about the pattern of future requests.

Although an important contribution is to demonstrate that user-directed replacement policies could greatly increase caching value, we also have something to say about how to implement user-directed replacement policies. One approach is to offer a *stochastic QoS* service. That is, users could provide valuation and forecasting information, and the cache could promise in exchange a service policy that increases or decreases the probability that an object will be found in the cache according to the communicated information. A second approach is to offer a *deterministic QoS* service. In this approach, users communicate with the cache and receive deterministic guarantees on whether specific objects will be in the cache.

A stochastic QoS Web caching service for servers was recently described (Kelly et al.). This service, called server-weighted LFU (swLFU), responds to different levels of value for cache hits by weighting their frequency counts accordingly. Those with a higher value per hit receive a higher byte hit rate on average. Further, the algorithm provides a higher average social welfare than LRU or LFU to the aggregate of users.

In this paper we develop and analyze a deterministic QoS Web caching service. This information has value to the users and they need proper incentives to reveal that information. Second, the cache system in this paper allows for more responsive shifts in the content of the cache. This feature could be important if the preferences of the users shift or network resources fail. Third, swLFU makes assumptions about the pattern of object usage. After accounting for the valuation weights swLFU inherently biases toward the most frequently used objects. Under our market mechanism the prediction decision is decentralized down to the users who have a clearer idea of the specific pattern of usage for each object. In this way a more accurate estimation of the periodicity of an object's request stream can be determined. Finally, Kelly et al. focused on the performance of a user-directed replacement policy without designing a service policy to solve the incentives problem. That is, it is rational for participants to misrepresent their valuations in that example and misrepresentations would reduce the performance of the cache. In this paper, we propose a market-based service policy to provide the necessary incentives to make the replacement policy work.

2. Server Cache Market

The focus of this paper is on shared caches, caches that serve many end users. Figure 1 shows a general view of the Internet cache hierarchy. The caches at level 2 (L2) are generally proxy caches serving corporations or universities. The caches at level 3 (L3) are embedded in wide-area networks across the Internet and can often serve many requests not met by the L2 caches or serve requests directly from end users. The data used in this study came from L3 caches but the mechanism is applicable to L2 caches as well

Like Kelly et al. we will be looking exclusively at Web servers as the system users. A preliminary look at client participation and the difficulties with simulating their decision can be found in Section 4. Using trace-driven simulations we implement and evaluate a server cache market. A trace is a record of all requests for Internet objects that come to a cache over a length of time. Results from this simulation are compared to Kelly's swLFU policy as well as plain LFU (using time since last access as the secondary sort key) and LRU. We use the same three request streams collected by the National Laboratory for Applied Network Research (NLANR) L3 caches at Palo Alto (PA), Silicon Valley (SV), and the University of Illinois at Urbana-Champaign (UC) during the period 15 August-28 August 1998 as Kelly et al. in order to directly compare the results.[2] In addition the same filtering mechanism is used as in Kelly et al. to rid the streams of unsuccessful requests and dynamic content (Kelly et al. page 5).

[2] NLANR anonymized access logs. ftp://ftp.ircache.net/Traces/.

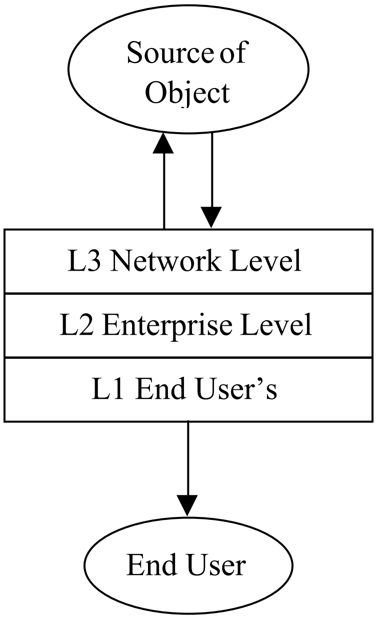

Figure 1: Internet Cache Hierarchy

	PA site	**SV site**	**UC site**
# servers	114,381	124,698	105,710
# URLs	3,412,105	3,744,274	2,884,598
# requests	7,011,622	7,897,659	5,568,112
Bytes req'd	131,665,275,644	161,620,444,331	127,346,723,989
Value req'd	264,025,996,908,451	319,466,493,484,667	264,106,487,028,562
Infinite cache			
size (bytes)	60,037,623,775	66,976,225,688	51,825,514,504
hit rate (%)	51.3364	52.5901	48.1943
byte hit rate	54.4013	58.5596	59.3036
value hit rate	48.5422	57.4670	56.5745

Table1: (from Kelly et al.) Summary statistics on our three request streams after filtering. Given our assignment of weights to URLs, "value requested" (see section 2.1) refers to the total value servers would receive if *every* request were served from the cache*, including first requests for URLs*; infinite cache value hit rate refers to the fraction of value requested that would be delivered by a cache large enough to store all requested URLs.

As in Kelly et al., we simulated cache sizes of 1, 4, 16, 64, 256, and 1024 megabytes for the three request streams listed above for the different caching mechanisms. In order to measure aggregate welfare for a given cache size we report plots of the value hit rate (VHR) as defined in the Kelly study. VHR is the value received by servers divided by the total possible value the server would receive if every request were served from the cache (see section 2.1 for valuation description). This normalized statistic provides a similar intuition as the common byte hit rate (BHR) statistic, the total number of bytes hit in the cache divided by the amount requested. In fact, Kelly points out that the two are exactly equal when all weights are equal. Summary statistics from the three caches are listed in Table 1.

2.1 Server Valuation

As we discussed earlier, servers associate a value with end users receiving the server objects from the cache rather than directly from the servers. Each server has a value per cache hit, V^i, for each object of given size, S^i, in bytes. In order to judge the total value of having object i, in the cache the server must make some judgement of the number of hits, H^i, the object will receive for a given length of time. The total value of the object being in the cache is: $T^i = V^i H^i$. In this paper we assume a server values hits on each of its web objects in the cache equally. So, $V^1 = V^2 = . . . = V^n$. In actuality each value for an object should be different, depending on their nature and their value to end users. This assumption simplified the experimentation. Undoubtedly it reduces the heterogeneity of server valuation and therefore bids. However, the large number of servers being simulated ensures a wide diversity.

Kelly et al. randomly assigns weights (value per hit, V) in equal proportions from the set {1, 10, 100, 1000, 10000} to each server in the trace streams. We use the exact assignment of values to the exact servers, not just the same conceptual method of assigning weights. By guaranteeing that both sets of research use the same weights we ensure that any difference between the caching mechanisms' results is due to the differences in the process and not random effects of weight assignment.

2.2 Motivation

Aggregate user value of the cache cannot be fully realized without accurate predictions of request streams and a method for knowing users' true valuations. If a cache was clairvoyant it could fill itself with the highest valued objects. We model such a cache here that packs the cache in order of highest marginal value. The objects are ranked by the value per byte, $B^i= T^i / S^i$. Every twenty minutes the cache is emptied and refilled based on this sort. The twenty-minute periodicity is used to compare the perfect foresight cache to the market discussed later.[3] The perfect foresight cache represents an upper bound on how well one could do.

[3] A centralized mechanism with perfect knowledge of requests and knowledge of values could bring the system to a higher level of welfare by swapping objects in and out of the cache based on their request stream over time at a higher level of granularity than twenty minutes.

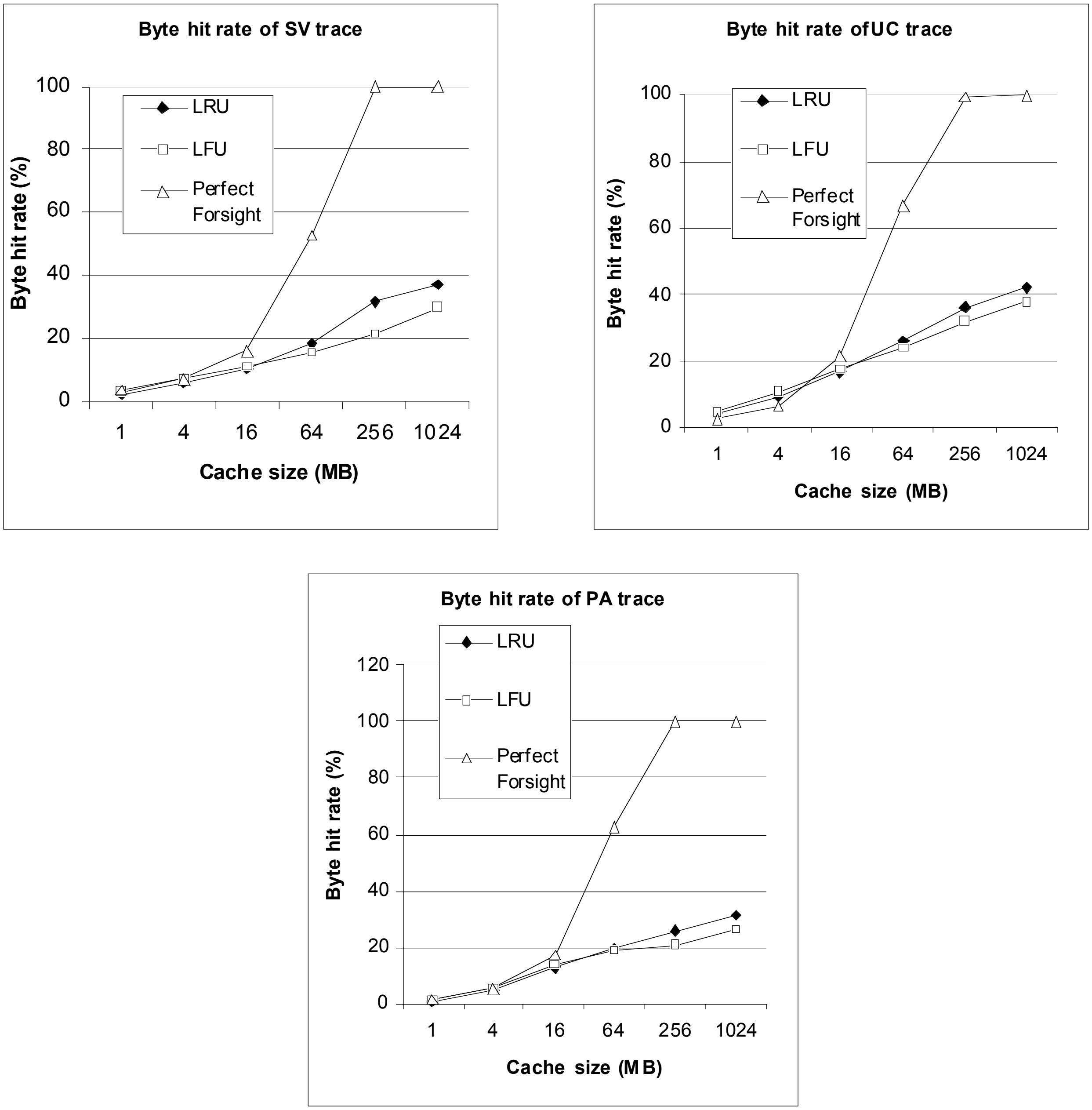

Figure 2: Byte hit rates for LFU, LRU, and Perfect Foresight.

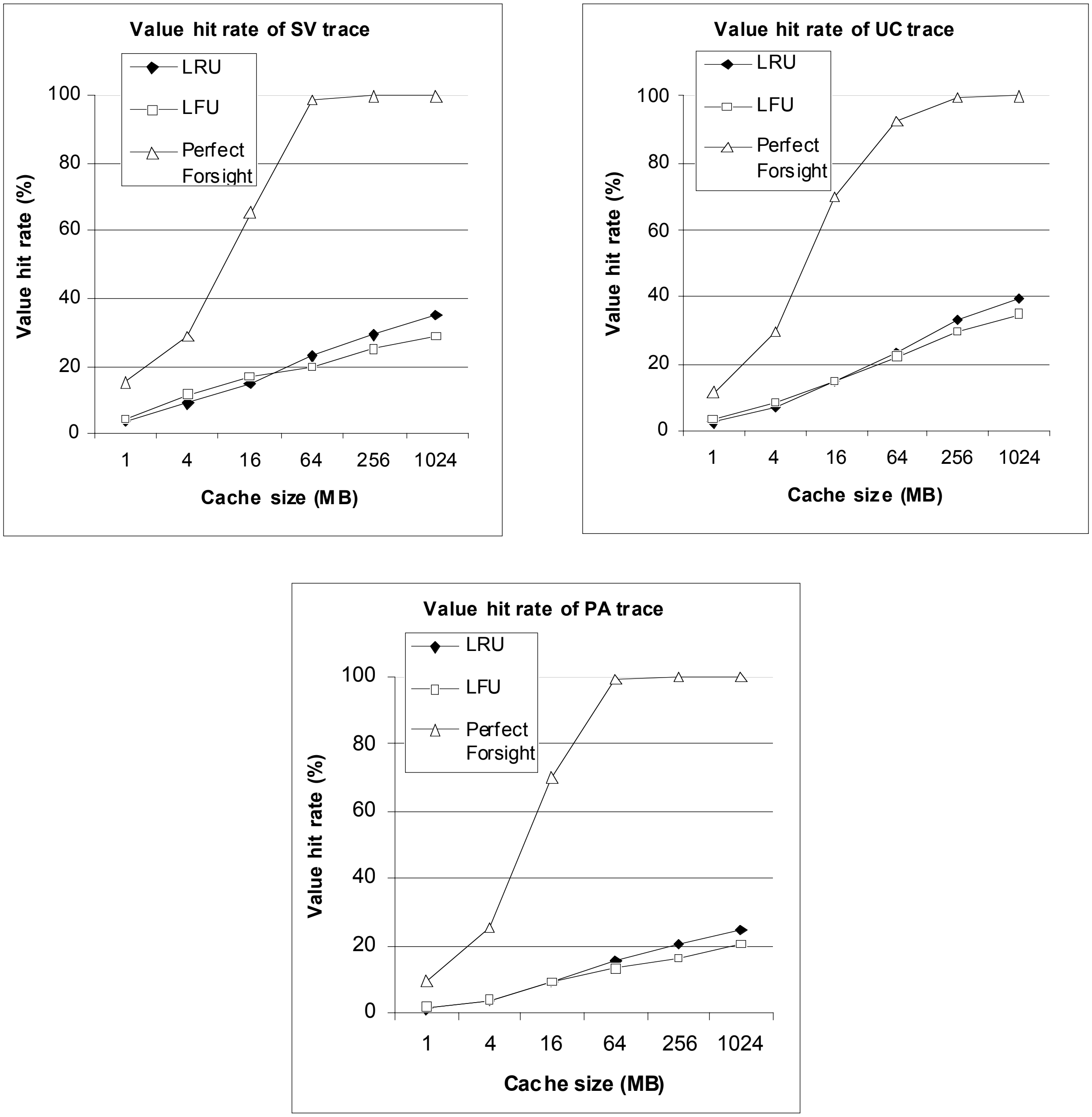

Figure 3: Value hit rates for LFU, LRU, and Perfect Foresight.

In figure 2 we see a notable gap between the percentage of byte requests served by a cache with perfect knowledge and the traditional caching mechanisms of LRU and LFU. Their predictive ability is limited by their assumptions of recency and frequency in prediction. While a larger cache would help some with BHR improvement LRU and LFU would still fail to anticipate new requests never seen before.

In figure 3 we see a larger gap when comparing the VHR of the cache mechanisms, especially at low cache sizes. Not only does LRU and LFU have problems predicting request streams but in addition, its failure to realize user valuations keeps the aggregate welfare lower. These gaps represent the possible improvement a cache can realize if accurate measurements of prediction and user value are obtained.

We implement a market for disk space, where servers buy deterministic access, as a way to gather the prediction and value information. The market works as follows: An auctioneer opens the cache space for bidding in an auction periodically throughout the day.[4] Servers bid for space at a self-determined price per block and a given number of blocks. By using an auction information is gathered directly from the servers who are expected to have the most accurate idea of their valuation and the expected number of future requests on their objects. The following is a description of the auction and determination of bids.

2.3 Auction

We divide the time axis into equal segments of 20 minutes. Servers issue bids at the beginning of every period and the auctioneer sorts the bids in descending order of value per byte where $B^i = T^i / S^i$. So the complete bid from a server for a given object is $\{B^i, S^i\}$. The cache is then filled starting at the top of the list and working its way down.[5] If the remaining space in the cache is too small for the next object on the list it is skipped and the following object is added. A bid is considered a winning bid if its object is added to the cache. We then define the clearing price – the bid value of the highest losing bid – as the value every winning bid needs to pay when the payment is due. By charging an independent clearing price, this auction is considered competitive. Pareto-efficiency can only be assured by competitive auctions (Nautz). The clearing price is assumed to be independent of each server's bid and therefore, incentive compatible. This independence is important to ensure truthful revelation of servers' valuation information, having a similar effect as a second-price auction (Mas-Colell, Whinston and Green).

Immediately following the sorting of the winners, the auctioneer starts communicating with the servers. The servers with winning bids push the winning objects into the cache. At this point the cache is ready to accept requests from web users for web objects. The auctioneer checks whether the requested object is in the cache. If it is, the object is returned. Otherwise, the request is passed to the server and the object is saved in the vacant space in the cache. Vacant space is defined to be the total cache size minus the total size of winning bid objects. When the vacant space becomes congested it is managed by LRU.

At this point certain implications of this caching market should be noticed. First, the maximum size of the vacant space will change each period when the total size of the winning bid objects changes. When the cache is congested, the total size of objects bid is greater than the cache, the vacant space will shrink to a useless size. When there is any vacant space in the cache the clearing price is zero, there is no highest losing bid because all bids were accepted. The price reflects the fact that space in the cache is no longer a scarce resource; supply exceeds demand.

[4] Although we do not explore this issue in the paper it seems reasonable that the more frequently the auctions are held the quicker the cache content could change and the more responsive the cache would be to demand shifts.

[5] Note, by filling the cache by value per byte we are filling the cache by the marginal benefit of the object. By using this metric the highest marginal benefit objects are placed in the cache which almost guarantees to maximize ex ante welfare of the cache. Almost, because filling the cache is an integer programming knapsack problem for which optimal algorithms have been developed, see Winston, Wayne L. Operations Research Applications and Algorithms. Second ed. Boston: PWS-Kent Publishing Company, 1991. The rough approach used in this paper should capture most of the welfare as long as the size of the objects is small relative to the cache size.

The objects in the cache will have a shake-up at the end of a period because the ownership of the cache changes hand. New servers can receive new space for different objects. Nothing is done if the winning bid object by a server is the same as before. Otherwise, the object is moved into the vacant space with congestion managed by LRU.

3. Experiments

The server market is implemented using the same trace data and valuations assigned to servers for the perfect foresight cache.

3.1 Server Prediction

This model of a market based web cache is evaluated with server agents that are assumed to follow a simple method of estimating future hits on web objects. These simple servers have accurate data of past request streams of generic web content for the cache in question. They assume all request patterns are the same for all objects and use an hour window of past data to predict the expected number of hits in the auction period of 20 minutes. By using a relatively unsophisticated agent in prediction ability we can judge the lower bound of the market mechanism. Others have used past trace data to predict the probability of future requests to develop cache replacement policies (Rizzo and Vicisano). A simple regression of requests for an object in the window against the number of requests for the object in the auction period was completed for each of the cache sites for August 1-14. The first two weeks of the month were used because realistically servers would use prior request streams to generate models for the future, in this case August 15-28. In example the linear model for the SV site was estimated as:[6]

Period hits = -0.302478 + 0.303812*Window hits

Very similar results were computed for the other two sites. While very simple and only accounting for the impact of recent requests in predicting the future, the model provides some interesting intuition. Data analysis showed us that close to 85% of objects are only requested once in a day. This characteristic is reconfirmed in the model. If the object only receives one window hit its predicted number of period hits is 0.001334.

3.2 Results

Figures 4 and 5 provide BHR and VHR information for LRU, LFU, swLFU, and the simple server market for various cache sizes. The most significant predictive characteristics of the server market performance seem to be the congestion of the cache and the replacement policy of the vacant space, LRU. It appears that the simple server market provides a better VHR then LFU and LRU when the cache size is relatively small. This result seems to be correlated with the cache being congested, meaning the clearing price is positive. Table 2 provides a look at the average clearing price for the different size caches of the three sites. The 256 MB and 1024 MB caches do not appear congested the vast majority of the time. Once there are no more objects to push of known average positive value the VHR performance tends toward LRU. As LRU manages a greater percentage of the cache it dominates the results.

[6] The R^2 was estimated as 0.7764. Both parameter estimates were significant at less than the 0.0001 level.

Cache size (MB)	SV Trace	UC Trace	PA Trace
1	16042.8	11464.6	5515.8900
4	4030.45	4030.45	2289.2900
16	732.972	442.811	268.7020
64	9.5062	36.1828	40.9102
256	0.046931	0.178155	0.1109
1024	0	0	0

Table 2: Simple server market clearing prices for the SV, UC, and PA traces at various cache sizes.

The BHRs for the server market remains very low for small cache sizes. A smilingly contradictory result when viewed with the larger VHR results because a higher hit rate for an object means it is producing a higher value. But in fact there is a direct trade-off between the two. As more emphasis is placed on the valuation of an object in the sorting process it will necessarily be at the expense of those objects with a high hit rate but a relatively low valuation. Again, it seems once the server market cache ceases to be congested the BHR tends toward LRU.

In Figure 6 we can see how the byte hit rate for URLs changes as a function of the weights that were assigned for the 64 MB cache. The SV trace presents the most convincing case that the lower bound server market gives differential QoS in relation to the amount paid by the servers. The anomalous UC data point is also reported by Kelly et al. who attribute the result to the random assignment of many highly hit URLs to the weight of 100. The PA trend is ambiguous but based on the VHR trends in Figure 5 it seems likely that differential quality of service would be seen for smaller cache sizes.

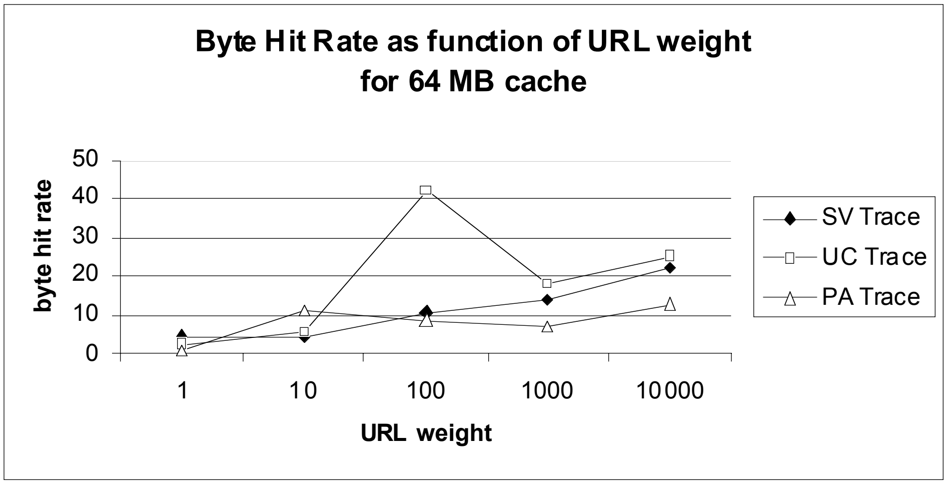

Figure 6: Byte Hit Rate for the SV, UC, and PA traces as a function of URL weight for a 64 MB cache.

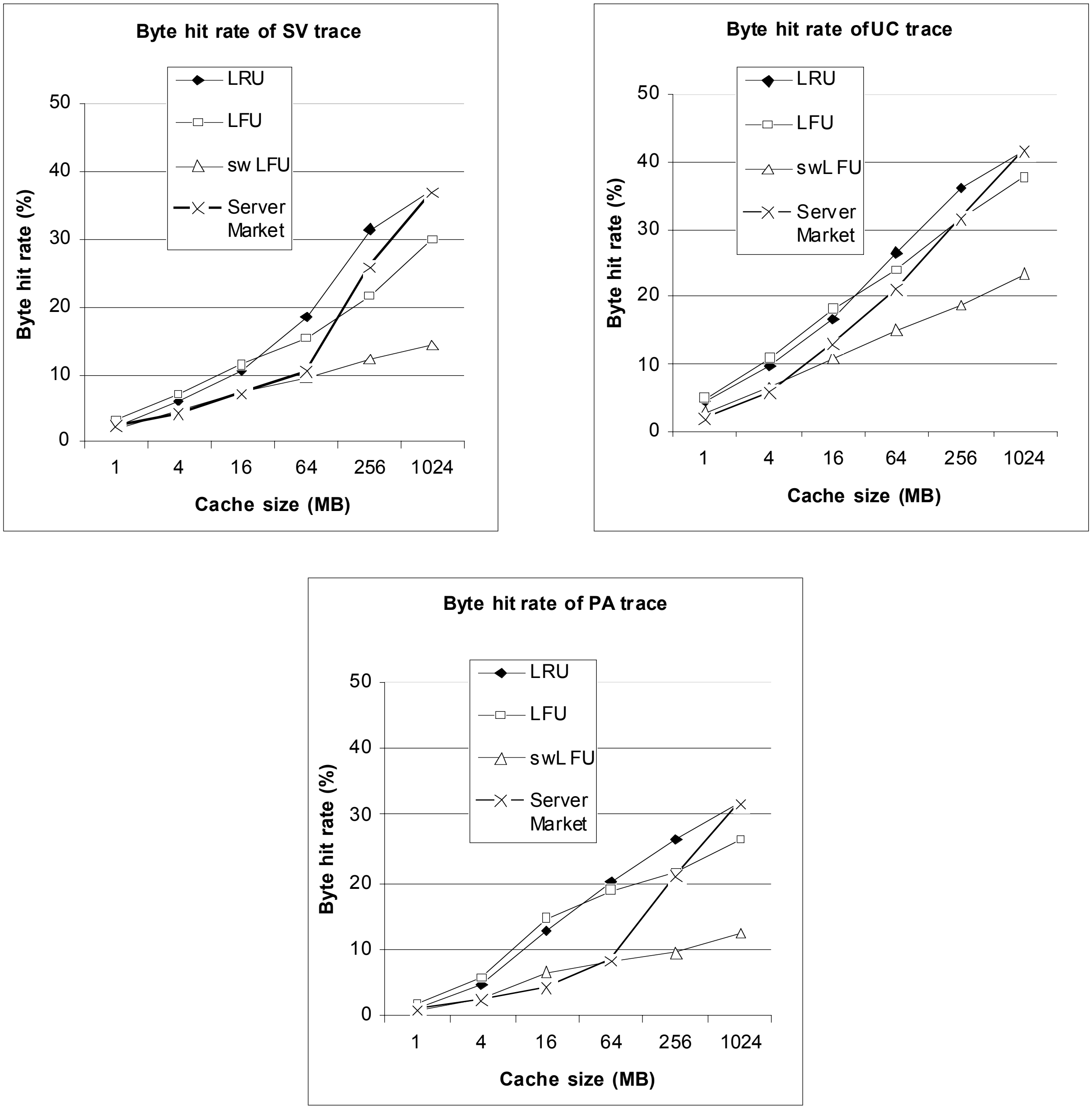

Figure 4: Byte hit rate for LFU, LRU, swLFU, and Server market.

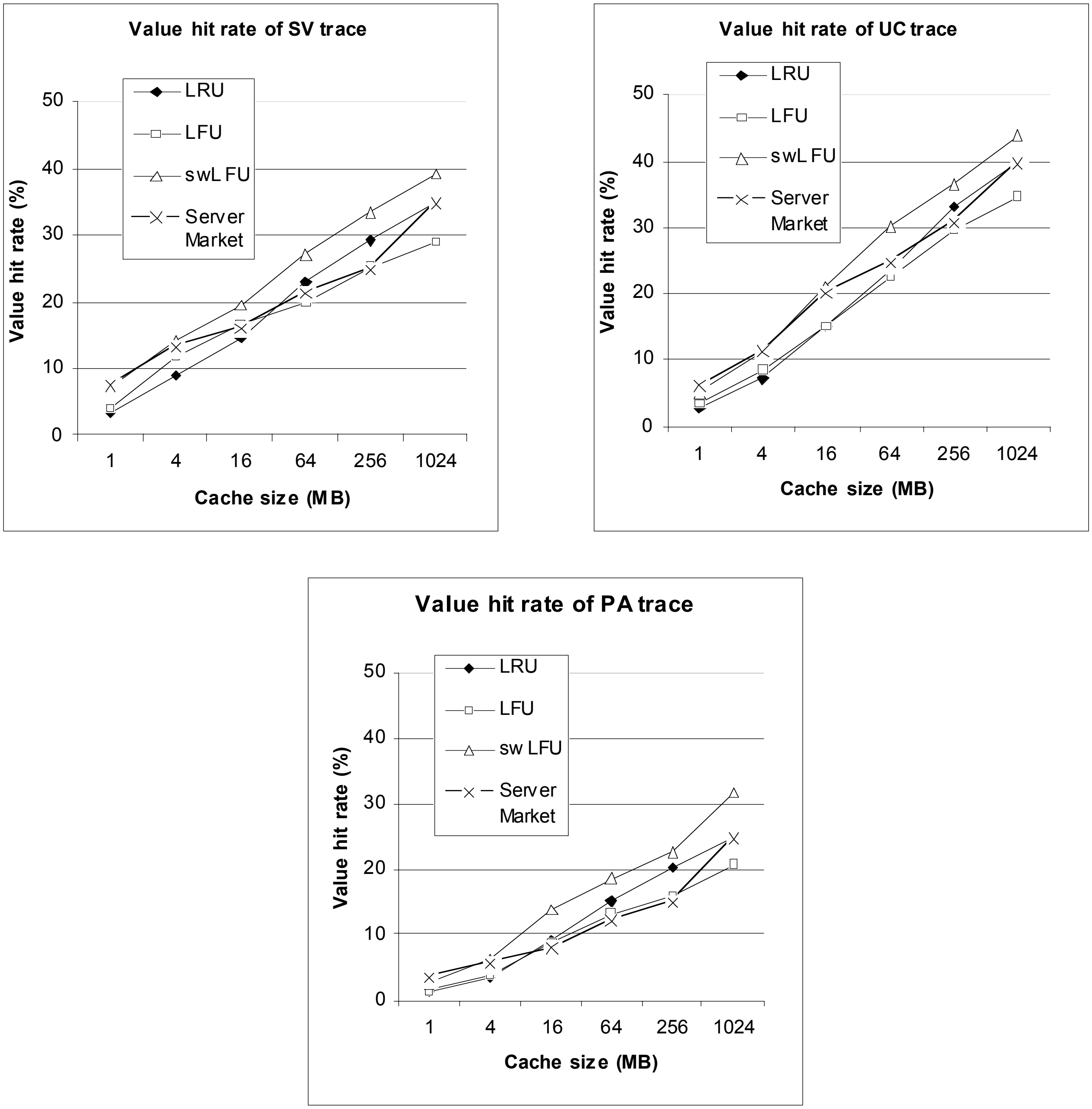

Figure 5: Value hit rate for LFU, LRU, swLFU, and Server market.

Except where caches are very small swLFU out performs the simple server market in VHR. This result is probably the case because swLFU uses more information than the simple server market in making its decisions. While the server market might make better use of the data only the last hour worth of frequency of requests is used. The difference in VHR widens as the cache becomes bigger and the simple server market has the opportunity to push less accurate material into the cache through LRU. But it is inappropriate to compare the two directly. The results for swLFU represent an upper bound under the assumption that the valuations used are reported truthfully even though there is no incentive to do so. The results for the simple server market represent a lower bound on the predictive techniques used for future hits. The possible welfare to be gained, as seen by the perfect foresight cache, leaves a lot of room for improvement for better prediction systems in the market mechanism.

4. Client Problem

In reality there is likely to be another type of agent that would be willing to purchase space in the cache that has a completely different decision problem than the server agents that have been the focus of most of the discussion. These agents are clients of the cache. Clients, in this case, refer to enterprise level institutions or ISPs who are users of the L3 cache and wish to reserve some portion of space in the cache for the requests of their end users (see figure 1).

A client's decision is more complex than a server's decision because a client's motivation to control space in the cache is directly tied to the homogeneity of content pushed by other clients. The problem is caused by the public good nature of the cache. This characteristic means any one can "consume" from the cache without other consumers losing value in their "consumption" and the cache is non-exclusive, meaning all requests are served. These aspects might motivate a particular client to cache very little if it knew other clients would cache material its users demanded. This phenomenon is known as the free rider effect. For instance, if no clients demand the same object, and the market is incentive compatible, we can be reasonably sure that their bids to cache objects will reflect their value of the objects. If many clients have users that demand the same object then their decision to bid for space for the object must account for the chance that other clients could cache these objects. In other words the decision must account for the benefit of not controlling part of the cache. In fact, one could imagine a circumstance where other clients' requests would be better at predicting a particular client's requests than the client's own past requests. The requests could be correlated with a time lag, quite possible given the trend nature of many objects. Formally the clients would try to maximize the decision:

$$\max_{Qc} \; [Hc(Qc) * V - C(Qc) + Ho(Qt - Qc) * V] * pr[win|B(Qc)] + (1 - pr[win|B(Qc)]) * Ho(Qt) * V$$

Hc(Qc) = expected number of hits from the client's cache space
Ho(Qt-Qc) = expected number of hits from the rest of the cache
V = value per hit
B(Qc) = value per disk block = Hc*V/Qc
Qc = the number of blocks bid for by the client
Qt = the total number of blocks in the cache
C(Qc) = expected cost of the blocks
pr[win|B(Qc)] = probability of winning disk blocks given clients bid of B

Notice that here we have modeled the problem with one decision variable, Qc. In fact, the client must bid two parameters in the auction, the number of blocks desired and the value per block. For simplicity value per disk block is represented as a function of the size of the cache that can be estimated from past trace data.

5. Conclusions

Caching can perform better and can support variable QoS if user valuation and prediction information can be gathered and used intelligently by the replacement algorithm. We have implemented a particular incentive-compatible market mechanism for gathering that information, and a particular replacement policy (market for write access to the cache) that uses the information. We have shown that we can improve on LRU and LFU even with very simple, naïve models of the quality of user prediction information. We have further shown that with better predictions, the potential gain in value is very high, more than twice the value obtained by LRU and LFU. It is certain that some very large servers would be able to see some of this gain. They could anticipate shifts in end user demand that the prediction algorithm explored here could not. Netscape would know to push a new version of its browser even though that particular object has no past request data (in fact Netscape does this now with proxy severs). Further work needs to be done to represent improved server prediction of end user requests. Currently LRU and LFU seem to be doing a better job at predicting than the lower bound server market. By offloading request prediction to the servers experimental results become very dependent on our ability to mimic those predictions and our assumptions of what servers would be reasonably able to predict. When cache resources are scarce this market mechanism appears to give encouraging results to those whose goal it is to maximize the value received by the users.

Future work should include estimation of the hit rate functions found in the client's decision problem and further simulations using these agents. Then simulations involving both types of agents in a more realistic market setting would be possible. Further future work will include more detailed regression models for the estimation of the number of requests in a given auction period for individual server objects. By being more specific and by exploring weekly and daily periodicity in the request streams we hope to realize more of the possible welfare demonstrated by the perfect foresight cache. In addition, the prediction models could be for specific objects and could be updated periodically instead of relying on one model for all agents in the entire simulation as was done here. These methods are not the only ways to improve prediction in Web caching replacement policies. Past research shows that decision theoretic user models can be used to improve performance (Horvitz). In the future we will also look at the balance between the auction period length and changes in the preferences of the servers. In addition, we could compare the performance of the other caching mechanisms to the server market when there are dramatic changes in server preferences (valuation of hits).

6. Acknowledgments

Terence Kelly provided significant technical help and many useful discussions throughout the research process. The authors also thank the entire Michigan Adaptive Resource Exchange (MARX) project group for continual feedback and direction. They gratefully acknowledge support from DARPA grant F30602-97-1-0228 from the Information Survivability program. The National Laboratory for Applied Network Research collected the trace data used in the simulations under National Science Foundation grants NCR-9616602 and NCR-9521745.

Jeff MacKie-Mason gratefully acknowledges the support of an IBM University Partnership grant. Sugih Jamin is supported in part by NSF CAREER Award ANI-9734145, Presidential Early Career Award for Scientists and Engineers (PECASE) 1998, and equipment support from Sun Microsystems Inc. and Compaq (Digital Equipment) Corp.

References:

Cao, Pei, and Sandy Irani. “Cost-Aware WWW Proxy Caching Algorithms.” Proceedings of the 1997 USENIX Symposium on Internet Technology and Systems. December 1997. 193-206.

Chuang, John. “Economics of Scale in Information Dissemination over the Internet.” Dissertation. Carnegie Mellon University, 1998.

Horvitz, Eric. "Continual Computation Policies for Utility-Directed Prefetching." Seventh ACM Conference on Information and Knowledge Management. Bethesda, MD: ACM Press: New York, 1998. 175-184.

Karaul, Mehmet, Yannis A. Korilis, and Ariel Orda. "A Market-Based Architecture for Management of Geographically Dispersed, replicated Web Servers." Proceedings of the First International Conference on Information and Computation Economics. 1998. 158-165.

Kelly, Terence, et al. "Biased Replacement Policies for Web Caches: Differential Quality-of-Service and Aggregate User Value." Fourth International Web Caching Workshop San Diego, California, 1999.

Kurose, James F., and Rahul Simha. “A microeconomic approach to optimal resource allocation in distributed computer systems.” IEEE Transactions on Computers 38.5 (1989): 705-717.

MacKie-Mason, Jeffrey K., and Hal R. Varian. “Pricing Contestable Network Resources.” IEEE Journal of Selected Areas in Communications 13.7 (1995).

Mas-Colell, Andreu, Michael D. Whinston, and Jerry R. Green. Microeconomic Thery. New York: Oxford University Press, 1995.

Nautz, D. “Optimal Bidding in Multi-Unit Auctions with Many Bidders.” Economic Letters 48 (1995): 301-306.

Oaks, Chris. “Can Caching Tame the Web?” Wired News August 18, 1998.

Rizzo, Luigi, and Lorenzo Vicisano. “Replacement Policies for a Proxy Cache.” Technical Report RN/98/13: University College London Department of Computer Science, 1998.

Waldspurger, C. A., et al. “Spawn: A Distributed Computational Economy.” IEEE Transactions on Software Engineering 18.2 (1992): 103-117.

Wellman, M. P. “Market-Oriented Programming: Some Early Lessons.” Market-Based Control: A paradigm for distributed resource allocation. Ed. Scott Clearwater. River Edge, NJ: World Scientific, 1996.

Williams, Stephen, et al. “Removal Policies in Network Caches for World-Wide Web Documents.” Proceedings of ACM SIGCOMM '96 1996. 293-305.

Winston, Wayne L. Operations Research Applications and Algorithms. Second ed. Boston: PWS-Kent Publishing Company, 1991.