Media Caching Support for Mobile Transit Clients

Hazem Gomaa Geoffrey Messier Robert Davies
Department of Electrical and Computer Engineering
University of Calgary
Calgary, AB, Canada
Email: {hagomaa,gmessier,davies}@ucalgary.ca

Carey Williamson
Department of Computer Science
University of Calgary
Calgary, AB, Canada
Email: carey@cpsc.ucalgary.ca

***Abstract*—In this paper, we consider the design of caching infrastructure to enhance the client-perceived performance of mobile wireless clients retrieving multimedia objects from the Internet. We consider three primary issues: location of the cache, size of the cache, and management policy for the cache. We consider both infrastructure-oriented caching at the Access Point (AP), as well as peer-assisted caching at the mobile clients. Simulation is used as the methodology for evaluation and comparison of caching strategies. The simulation results show that AP caching is generally more effective than client-side caching, that adequate performance is achievable with a mix of rather modest AP and client-side caches, and that Least Frequently Used (LFU) is the most effective cache replacement policy. Additional simulation experiments show that our results are robust across different request generation rates and client turnover rates.**

***Index Terms*—Multimedia, Caching, Wireless Networks, Mobility, Simulation**

I. Introduction

Consumer demand for multimedia services (i.e., audio and video) using Internet-enabled mobile devices such as PDAs, smart phones, and wireless laptops, is increasing rapidly. However, there are many challenges that occur in supporting high-quality media streaming for mobile clients. Streaming videos from movies, news, and sports Web sites require a sustained bandwidth of approximately 1 Mbps from the media server to clients [1], [2]. The limited and unpredictable throughput of Internet links, particularly in wireless access networks, often degrades video quality.

Caching of streaming media objects near clients is a commonly used approach to overcome these challenges [3]. In a wireless network, caching on the edge of the network typically refers to caching at the Access Point (AP), which provides Internet access for the wireless clients.

The purpose of this paper is to explore novel caching approaches for wireless mobile clients who are retrieving and viewing multimedia objects via the Internet. We consider both infrastructure-oriented caching at the Access Point (AP), as well as peer-assisted caching at the mobile client devices themselves. The latter caching philosophy is very much in line with the recent trend in cooperative communications, where mobile nodes participate actively in the network operation [4]. However, current cooperative communication research has explored the use of mobile nodes primarily for packet forwarding at the physical, data link, and network layers. There has been little work on using mobiles as application-layer caching elements for other network clients.

The paradigm considered in our work is that each mobile device dedicates a portion of its memory or disk resources to serve as media object cache space that can be accessed by other devices. That is, the caching resources of the WLAN are implemented by hosting caching proxy servers on the AP as well as on the mobile devices. This mixes two types of proxy server models. The caching proxy on the AP corresponds to the model where the proxy server runs on a dedicated machine with a large storage capacity [3], [5], [6]. The mobile proxies correspond to the model where the caching proxy runs directly on the LAN clients, with a central entity, perhaps hosted on the AP, coordinating the system operation [7].

This paper also evaluates cache management strategies in the case where mobile clients constitute part of the aggregate logical cache. Cache replacement policies are usually evaluated assuming a well-connected fixed network, where the cached object is always accessible by clients on the same LAN. In WLANs, on the other hand, clients may connect and disconnect as they roam into and out of the range of the AP. For example, mobile clients waiting at a train station or in an airport may use wireless devices to connect to an AP, which is installed to provide clients with wireless Internet access. However, the connections between the clients and the AP are severed when clients leave. This WLAN is a dynamic network because the set of connected clients varies with time, as does the accessibility to the set of cached content.

We use simulation as the performance evaluation methodology to evaluate and compare different caching strategies. The simulation results show that AP caching is generally more effective than client-side caching, and Least Frequently Used (LFU) is the most effective cache replacement policy. Fortunately, adequate caching performance is achievable with a mix of rather modest-sized AP and client caches. Furthermore, additional simulation experiments for sensitivity analysis show that our results are quite robust across a range of client turnover rates and request generation rates.

The remainder of the paper is organized as follows. First, related work is summarized in Section II. In Section III, the proposed caching system is introduced. Section IV describes the experimental methodology for our simulation work, while Section V presents the simulation results. Finally, Section VI concludes the paper and discusses future work.

II. Related Work

While the notion of a cooperative distributed mobile cache is new, this work will make use of previous research into caching replacement algorithms, evaluation methodologies, evaluation metrics, and different techniques for simulating cache performance.

As in all caches, the scheme in this paper requires a replacement algorithm to decide which media objects are cached, and which media objects are replaced. This work focuses on stream-aware replacement algorithms that take into account the characteristics of the streaming media object such as popularity, size, and bit rate. The algorithms considered include First-In-First-Out (FIFO), Least Recently Used (LRU), Least Frequently Used (LFU), and SIZE. FIFO removes objects in the same order that they initially arrived. LRU discards the least recently requested object [5], [8]. LFU removes unpopular (i.e., least frequently requested) objects [3], [5], [6]. SIZE always removes the largest object [9].

Other caching replacement algorithms include network-aware algorithms that take into account the network conditions such as available bandwidth and number of hops [10] and hybrid versions of stream-aware and network-aware algorithms [1], [11]. While network-aware and hybrid algorithms could be used with the caching architecture presented in this paper, they are not considered in this paper.

As with many other caching studies [3], [6], [7], [12]–[18] the performance of this scheme is evaluated using simulation. The streaming media traffic used to evaluate the cache performance is generated using a synthetic workload based on GISMO (Generator for Internet Streaming Media Objects) [13].

III. Proposed Caching System

The proposed caching system reflects caching activity in a WiFi hotspot contained in a heterogeneous wireless network. In this network, clients are able to access meida content over a cellular network covering the full network area and then take advantage of WiFi hotspots in certain high traffic locations.

The architecture of the system considered in this paper is shown in Figure 1. The system consists of a media server, mobile clients, a fixed wireless AP, and multiple storage proxy caches that reside on the clients and AP. The flow of media content (whether cached or not) always occurs via the AP, either between a client and the server, the AP and the client, or between two clients via the AP. No direct peer-to-peer data transfers are permitted.

Our WLAN model considers mobile transit clients, such as airport travelers or train passengers, who move at random, but perhaps en masse, between points of wireless connectivity. The simulation assumes that the *turnover rate* defines the rate at which new clients enter the WLAN, as well as the rate at which former clients leave. Thus there is always a fixed number N of concurrent clients, but their identities change with time. Clients first entering the WLAN are assumed to have empty caches, and the clients that leave the WLAN are selected at random. Once a client leaves the WLAN, the information in that client's cache is effectively lost. A client does not re-enter the WLAN after leaving. The data cached in the AP, however, is always available to the clients in the network, unless evicted by the cache replacement algorithm.

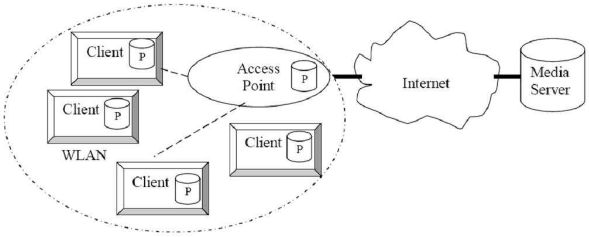


Fig. 1. Network model for wireless media caching

Media objects that are not present in the client or Access Point caches flow from the media server to the mobile client via the AP (see Figure 1). The caching algorithm fills the AP cache first, since it is a central resource shared by all clients. If a client requests a media object and the AP cache has sufficient capacity remaining, then the object is cached at the AP, regardless of the state of the client cache. If the AP cache is full, then the client will voluntarily cache the media object that it just requested, in anticipation that it may be useful for another WLAN client in the near future.

The aggregate WLAN cache never contains more than one copy of any given media object at a time. If a requested object already resides in the proxy cache of another client served by the AP, then a copy of this object is provided to the requesting client (but not cached there). Also, media objects are cached in their entirety; no partial object caching (such as prefix caching) is allowed.

There are only two possible locations in which a media object can be cached: the AP, and the requesting client. If those two locations are both full, then a cache replacement policy is used to make room for the new object. From the perspective of the replacement algorithm, the combination of the proxy of the client requesting the media object and the AP form a Logical Proxy (LP), as shown in Figure 2. The replacement algorithm operates on the LP as if it was a single unified cache. We consider recency-based, frequency-based, and size-based cache replacement policies, namely FIFO, LRU, LFU, and SIZE.

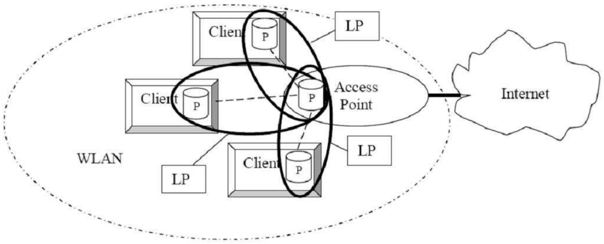


Fig. 2. WLAN example with 4 physical (P) and 3 logical proxies (LP)

In the proposed caching architecture, the mobile client sends a request for a media object to the Access Point. If the AP cache has that object, then the object is sent directly to the client. If not, the AP queries the other clients in the WLAN. If one of these clients has the object, it sends the object to the AP, which then relays it to the client making the request. Note that this object is not added to the cache of the requesting client since it is already present elsewhere in the WLAN cache. Finally, if the object is not present anywhere in the WLAN cache, then a request for the object is sent to the media server. The object is then added to the LP consisting of the AP and the client that requested the object.

IV. Evaluation Methodology

This section describes the simulation methodology used in the evaluation of the proposed system. First, the generation of synthetic workload for the network is described. Next, the replacement algorithms and the performance metrics used in the evaluation are discussed.

The proposed caching system was evaluated using a discrete-event simulator developed using C++. The primary inputs to the simulator are the traffic workload, the network model, and the client behavioural model.

Table I lists the characteristics of the generated multimedia workload. The GISMO toolset [13] was modified to generate a synthetic workload that models the demand for media objects during a media session. We consider 1000 multimedia objects, averaging 122 KB in size. The popularities of the objects vary according to a Zipf-like distribution. In particular, the most popular item is requested several hundred times, while the least popular items are requested only once.

TABLE I
Multimedia Workload Characteristics

Number of Media Objects	1000
Object Popularity	Zipf-like ($\theta = 0.73$)
Object Size Distribution	Lognormal ($\mu = 7$, $\sigma = 0.5$)
Smallest Object Size	17 KB
Median Object Size	110 KB
Mean Object Size	122 KB
Largest Object Size	485 KB
Cumulative Object Size	125 MB

Table II lists the characteristics of the network and client models. We assume that the Internet link to the media server is the bottleneck (1 Mbps), and that WLAN throughput is 10 Mbps. We consider a batch arrival process for network clients, with a new batch of B clients arriving every hour, and B randomly chosen clients departing every hour. We also assume that a randomly chosen subset of clients remain from one hour to the next. We call this set of clients *residual clients*, and denote them using R. Each of the $N = B + R$ concurrent clients in the WLAN is equally likely to start a media session.

The total number of clients simulated depends on the client mobility model (i.e., how quickly the clients enter and leave the WLAN), and the total number of active sessions depends on the request rate. To illustrate this, let Q denote the number of new media queries from a client per hour, and let B denote the number of clients joining and leaving the WLAN per hour (i.e., the turnover rate). If the duration of the simulation is T hours, then the total number of individual clients observed is $U = BT + R$ and the total number of simulated sessions will be $X = NQT$.

TABLE II
Network and Client Model Parameters

Parameter	Description	Value
$C_{Internet}$	Wired Internet Link Capacity	1 Mbps
C_{WLAN}	WLAN Link Capacity	10 Mbps
S	Aggregate WLAN Cache Size	0 - 100%
T	Time Duration	6 hours
N	Total Concurrent Clients	120
B	Client Batch Size	100
R	Residual Clients	20
Q	Request Arrival Rate	8 per hour
X	Total Sessions	5,760

Note that the clients arrive according to a batch arrival process, while media object requests are generated according to a Poisson arrival process. As a result, the total number of clients accommodated in the simulation and the total number of media requests is a function of client mobility and request rate. This is in contrast to fixed networking caching studies where the number of clients is fixed.

Table III summarizes the factors and levels considered in our simulation experiments. We use a one-factor-at-a-time experimental design, and consider three primary issues: location of the cache, size of the cache, and management policy for the cache.

TABLE III
Factors and Levels for Simulation Experiments

Factor	Description	Levels
S_{Agg}	Aggregate Cache Size	0 - 100% (0-125 MB)
S_{AP}	AP Cache Size	10% (12.5 MB)
S_{Client}	Client Cache Size	0 - 2% (0-2.5 MB)
P	Cache Replacement Policy	FIFO, LRU, LFU, SIZE
Q	Request Arrival Rate (per hour)	8, 40
X	Total Sessions	5,760–28,800
B	Client Batch Size	50–150

As mentioned previously, the cache replacement algorithms considered are FIFO, LRU, LFU, and SIZE. To help evaluate the effect of varying cache size, several special cases are considered: Infinite AP Cache (IAPC), Infinite Client Cache (ICC), and Infinite Logical Cache (ILC). IAPC assumes that all objects can be cached at the AP, ICC assumes that all objects can be cached at one client, and ILC assumes that half of the media objects are cached on the client and the other half at the AP. Due to the infinite capacity assumption, the cache replacement algorithm is never triggered for the IAPC, ICC, and ILC cases.

The performance of the FIFO, LRU, LFU and SIZE replacement algorithms is evaluated assuming a cache of finite size. The total aggregate cache size for the entire WLAN, S, is divided such that the Access Point cache size is $S/2$, and the cache size for each client is $S/2U$.

Caching performance is evaluated using hit ratio, byte hit ratio, and service delay. Hit ratio is the total number of requests served by the caches divided by the total number of requests made by all clients during the simulation. Byte hit ratio is the total number of bytes served by the caches divided by the total bytes requested during the simulation. Normalized service delay is the time required for requesting and retrieving objects during the simulation divided by the time for requesting and retrieving objects in the absence of a cache (i.e., $S = 0$).

V. Simulation Results

Four experiments are done to compare the performance of the replacement algorithms under different network conditions.

A. Aggregate Cache Size and Location

In the first experiment, the performance of the replacement algorithms is evaluated as a function of the total aggregate cache size S available in the WLAN. The value of the aggregate cache size is varied from 0% to 100% of the total object sizes in steps of 10%.

Figure 3 shows the results from the first simulation experiment regarding cache size and cache location. The leftmost graph is for the request hit ratio, the middle graph is for the byte hit ratio, and the rightmost graph is for the normalized service delay. The same format is used for reporting all of the simulation results.

Figure 3(a) shows the request hit ratio for the cache on the vertical axis, as a function of increasing cache size on the horizontal axis. The three horizontal lines show the baseline results for the infinite cache policies. The results show that hosting an infinite cache solely at the AP is the best choice, with a hit ratio of 84%, while distributing the cache only among the clients is a poor choice, with a hit ratio of 55% (because of the client mobility every hour). Splitting the cache space equally between the AP and the clients yields an intermediate hit ratio result of 75%. These results provide a reference point for assessing the performance of practical (finite) cache sizes.

The four remaining lines in Figure 3(a) show the results for the different cache replacement policies considered. LFU provides the best hit ratio, followed by LRU and FIFO, and SIZE, which provides the lowest hit ratio at small cache sizes, but improves the most as cache size is increased. The advantages of LFU are attributable to the strong skew in object popularity from the Zipf-like distribution in the workload model; by retaining popular content in the cache, many subsequent hits can occur. The advantage of LFU is most evident at small cache sizes, and diminishes as cache size is increased, since replacement decisions occur less often. With adequate cache space, all four policies can achieve performance comparable to the 50-50 split Infinite Logical Cache (ILC). Note that the hit ratio results for the four policies do not converge to an identical value, even at 100% aggregate cache size. The reason is that the cache space is partitioned, with only 50% at the shared AP. Thus cache replacement decisions can still occur, resulting in performance differences between these policies.

Similar observations apply for the byte hit ratio results in Figure 3(b). LFU provides the highest byte hit ratio results observed, while SIZE provides the lowest, since its cache hits tend to be for the small(er) objects that are retained in its cache.

Figure 3(c) expresses the system performance in terms of normalized service delay. When the cache size is zero, the average delay for clients is exactly the same as the no cache configuration. With a very large cache, the average delay is reduced by a factor of 2 or more compared to the no cache scenario. LFU consistently provides the best performance among the four cache replacement policies considered.

Interestingly, a large but finite cache using any of the replacement algorithms performs better than an Infinite Client Cache (ICC). The first reason is due to client mobility. The ICC can store lots of information on the mobile clients, but those clients randomly leave the WLAN over the course of the simulation. The result is a loss of the cached information resulting in a subsequent request and retrieval of a media object from the media server over the bottleneck link. A second reason is due to the assumption that direct peer-to-peer links do not exist between clients. Therefore, it takes longer to access cached objects on the mobile clients since all cached client data must be relayed via the AP. Hosting a cache directly at the AP is clearly more efficient.

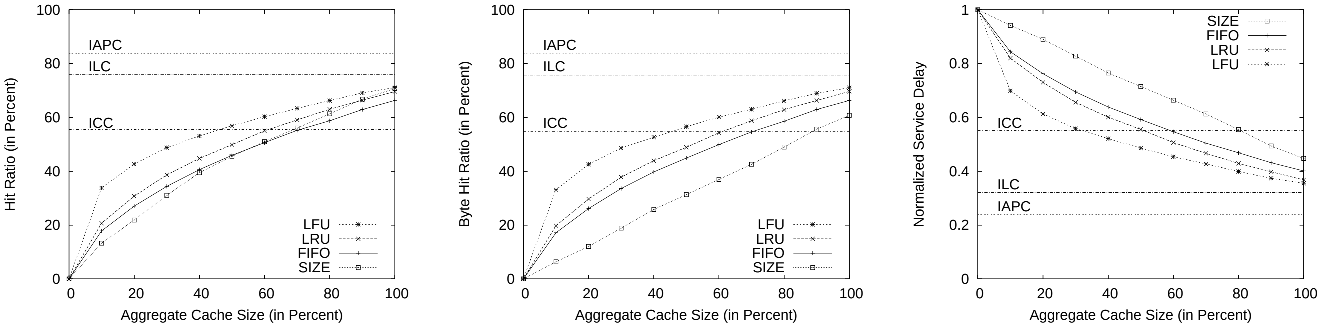
B. Client Cache Size

Figure 4 shows the results from the second simulation experiment that varies the client cache size. In these simulations, the size of the AP cache is fixed at 10% (approximately 12.5 MB) of the aggregate object sizes. The results show how the caching performance improves as the per-client cache size (on the horizontal axis) is increased.

Figure 4(a) shows that only modest client caches up to 2 MB are needed. With no client cache at all, the average hit ratio at the AP cache is about 30-40%. With a small client cache (e.g., 1-2 MB) the cache hit ratio improves to about 60%. However, there is a diminishing returns effect, in that additional client cache space offers little or no additional improvement. The asymptotic hit ratio achieved is only slightly better than that achieved with an infinite client cache, with the improvement attributable to the AP portion of the logical cache. The performance differences between different cache replacement algorithms are minor, but LFU still outperforms LRU, FIFO, and SIZE. Similar observations apply for byte hit ratio in Figure 4(b), and for normalized service delay in Figure 4(c).

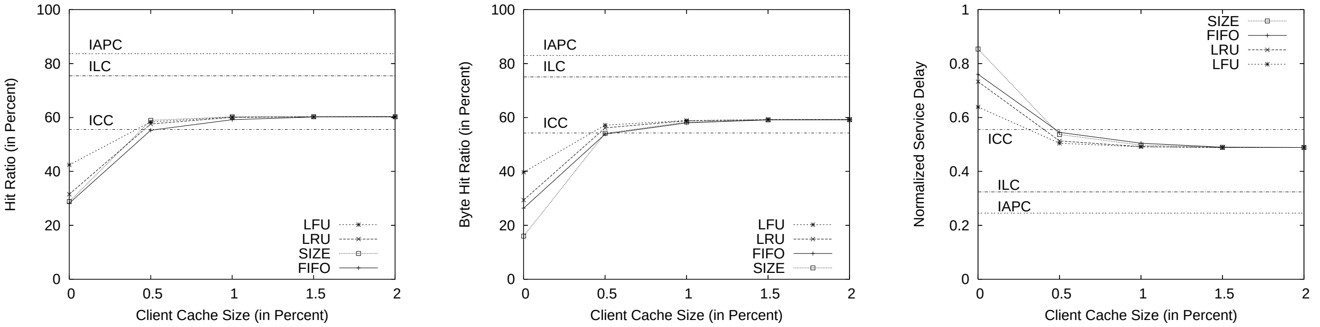
C. Client Request Rate

Figure 5 shows the results from the third simulation experiment that increases the client request rate from $Q = 8$ to $Q = 40$. Note that network congestion is not modelled in this



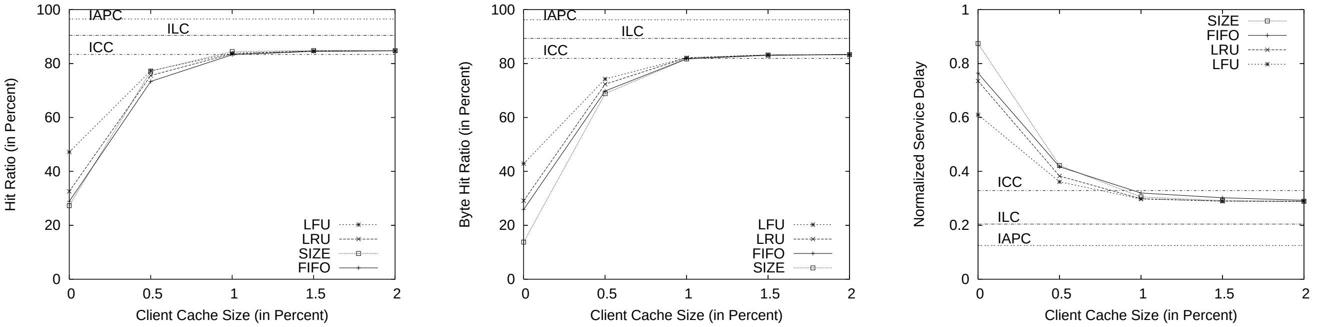
(a) Request Hit Ratio (b) Byte Hit Ratio (c) Service Delay

Fig. 3. Simulation Results for Experiment 1: Effect of Aggregate Cache Size ($N = 120$, $Q = 8$, $X = 5,760$)



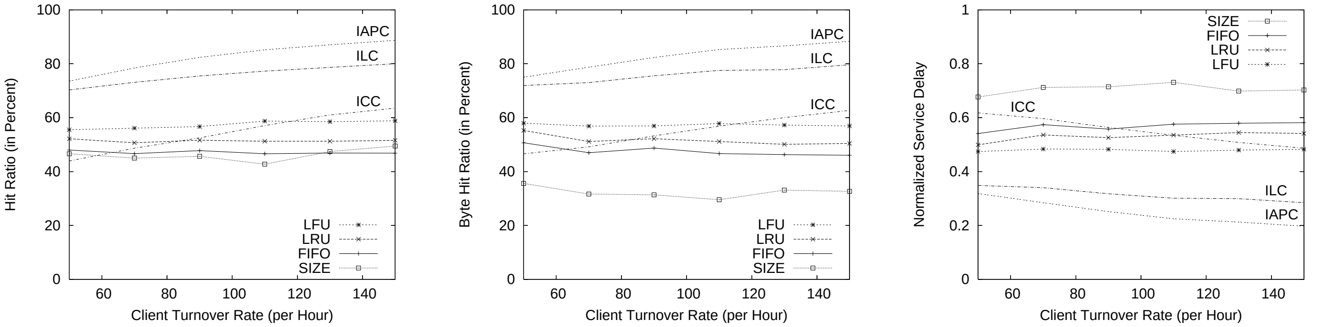
(a) Request Hit Ratio (b) Byte Hit Ratio (c) Service Delay

Fig. 4. Simulation Results for Experiment 2: Effect of Client Cache Capacity ($N = 120$, $Q = 8$, $X = 5,760$)



(a) Request Hit Ratio (b) Byte Hit Ratio (c) Service Delay

Fig. 5. Simulation Results for Experiment 3: Effect of Client Cache Capacity ($N = 120$, $Q = 40$, $X = 28,800$)



(a) Request Hit Ratio (b) Byte Hit Ratio (c) Service Delay

Fig. 6. Simulation Results for Experiment 4: Effect of Client Turnover Rate ($Q = 8$)

study. That means that each client always experiences 1 Mbps Internet link throughput even for the higher request rate of $Q = 40$. The performance results are qualitatively consistent with those observed in Figure 4. The primary difference in Figure 5(a) is the higher hit ratios achieved. With a higher request rate from the same set of clients to the same set of media objects, there are more repeated requests to the popular content, and more opportunities for caching. The asymptotic hit ratio achieved is approximately 82%. LFU continues to maintain its advantage over the other three cache replacement policies considered. The performance results are consistent for byte hit ratio and service delay, as presented in Figure 5(b) and Figure 5(c), respectively.

D. Client Turnover Rate

Figure 6 shows the results from the fourth and final simulation experiment, which studies the effect of client turnover rate. Simulation results are plotted as a function of the client batch size B, ranging from 50 to 150 clients per hour on the horizontal axis. As the turnover rate increases, one expects to see fewer opportunites for caching, since new clients are arriving into the system with empty caches, and old clients are leaving the system with their cached content.

The results in Figure 6, however, show that the system is quite robust in the face of increased client turnover. That is, the hit ratio results and the service delay results are fairly consistent for the range of turnover rates considered. The most pronounced impact is for the infinite client cache configuration, in which the hit ratio actually *improves* with increasing turnover. The explanation for this effect is the growth in the number of clients. With more clients present, the number of requests increases, and more overlaps in their media object requests produce cache hits.

VI. Conclusions

In this paper, we proposed a novel wireless media caching system for mobile transit clients. The system uses storage proxies, which are hosted by mobile clients and APs, to store and share media objects on an on-demand basis among WLAN clients. The performance of cache management strategies was evaluated in this dynamic wireless network context, where clients may connect and disconnect from the WLAN at random.

Four simulation experiments were conducted to compare the performance of the replacement algorithms in different network and workload conditions. These experiments study the location, size, and management of the cache space. In dynamic networks, the increase in aggregate cache size, which consists of clients and AP caches, leads to improved caching performance. The increase in client cache size provides a small additional improvement, but this effect is minimal if the AP cache size is adequately provisioned. LFU is consistently the best cache replacement policy in the network and workload scenarios that we considered.

More work remains to be done to examine how the cache capacity distribution affects caching performance. Also, more caching algorithms will be implemented to find the best caching performance in the dynamic network context. Finally, the proposed system will be evaluated in terms of energy consumption tradeoffs. That is, does power consumption in the mobile device increase due to cache sharing, or decrease because of the reduced time for object retrievals.

Acknowledgments

The authors would like to thank the WiMob 2009 reviewers for their positive feedback on an earlier version of this paper. Financial support for this research was provided by TRLabs (Telecommunications Research Laboratories, Alberta), and NSERC (Natural Sciences and Engineering Research Council).

References

[1] S. Acharya and B. Smith, “An Experiment to Characterize Videos Stored on the Web”, *Proceedings of SPIE Multimedia Computing and Networking* (MMCN), 1998.

[2] S. Acharya, B. Smith, and P. Parns, “Characterizing User Access to Videos on the World Wide Web”, *Proceedings of SPIE Multimedia Computing and Networking* (MMCN), January 2000.

[3] S. Jin, A. Bestavros, and A. Iyengar, “Accelerating Internet Streaming Media Delivery Using Network-Aware Partial Caching”, *Proceedings of IEEE International Conference on Distributed Computing Systems* (ICDCS), Vienna, Austria, pp. 153-160, July 2002.

[4] S. Jin and A. Bestavros, “Cache-and-Relay Streaming Media Delivery for Asynchronous Clients”, *Proceedings of the 4th International Workshop on Networked Group Communication*, Boston, MA, October 2002.

[5] P. Cao and S. Irani, “Cost Aware WWW Proxy Caching Algorithms”, *Proceedings of USITS*, December 1997.

[6] Y. Wang, Z. Zhang, D. Du, and D. Su, “A Network-Conscious Approach to End-to-End Video Delivery over Wide Area Networks using Proxy Servers”, *Proceedings of IEEE INFOCOM*, 1998.

[7] S. Acharya and B. Smith, “MiddleMan: A Video Caching Proxy Server”, *Proceedings of ACM NOSSDAV*, June 2000.

[8] E. O’Neil, P. O’Neil, and G. Weikum, “The LRU-k Page Replacement Algorithm for Database Disk Buffering”, *Proceedings of International Conference on Management of Data*, May 1993.

[9] S. Williams, M. Abrams, C. Standbridge, G. Abdulla, and E. Fox, “Removal Policies in Network Caches for World-Wide Web Documents”, *Proceedings of the ACM SIGCOMM*, August 1996.

[10] R. Wooster and M. Abrams, “Proxy Caching the Estimates Page Load Delays”, *Proceedings of the 6th International World Wide Web Conference*, April 1997.

[11] M. Abrams, C. Standbridge, G. Abdulla, S. Williams, and E. Fox, “Caching Proxies: Limitations and Potentials”, *Proceedings of the 4th International World Wide Web Conference*, Boston, MA, December 1995.

[12] A. Bestavros and S. Jin, “OSMOSIS: Scalable Delivery of Real-Time Streaming Media in Ad-Hoc Overlay Networks”, *Proceedings of IEEE ICDCS Workshop on Data Distribution in Real-Time Systems*, Providence, RI, pp. 184-195, May 2004.

[13] S. Jin and A. Bestavros, “GISMO: Generator of Streaming Media Objects and Workloads”, *ACM Performance Evaluation Review*, Vol. 29, No. 3, 2001.

[14] R. Rejaie, H. Yu, M. Handley, and D. Estrin, “Multimedia Proxy Caching Mechanism for Quality Adaptive Streaming Applications in the Internet”, *Proceedings of IEEE INFOCOM*, March 2000.

[15] S. Sen, J. Rexford, and D. Towsley, ”Proxy prefix caching for multimedia streams”, *Proceedings of INFOCOM*, April 1999.

[16] S. Sheu, K. Hua, and W. Tavanapong. ”Chaining: A generalized batching technique for video on demand”, *Proceedings of IEEE ICMCS*, 1997.

[17] A. Dan, ”Buffering and Caching in Large Scale Multimedia Servers”, *Proceedings of IEEE COMPCON*, March 1995.

[18] A. Dan and D. Sitaram, ”Buffer Management Policy for an On-Demand Video Server”, *IBM Research Report RC 19347, T. J. Watson Research Center, Yorktown Heights, New York* ,January 1993.