Peer-to-Peer Cooperative Caching in Mobile Environments*

Chi-Yin Chow Hong Va Leong Alvin Chan
Department of Computing, Hong Kong Polytechnic University, Hong Kong
E-mail: {*cscychow, cshleong, cstschan*}*@comp.polyu.edu.hk*

Abstract

Caching is a key technique for improving the data retrieval performance of mobile clients in mobile environments. The emergence of robust and reliable peer-to-peer (P2P) technologies now brings to reality what we call "cooperative caching" in which mobile clients can access data items from the cache in their neighboring peers. This paper discusses cooperative caching in mobile environments and proposes a ***CO****operative* ***CA****ching scheme for mobile systems, called COCA. In COCA, we identify two types of mobile clients: low activity and high activity. They are referred to as Low Activity Mobile client/host (LAM) and High Activity Mobile client/host (HAM) respectively. Both LAM and HAM can share their cache. The server replicates appropriate data items to LAMs so that HAMs can take advantages of the LAM replicas. The performance of pure COCA and COCA with the data replication scheme is evaluated through a number of simulated experiments which show that COCA significantly reduces both the server workload in terms of number of requests and the access miss ratio when the MHs reside outside of the service area. The COCA with the data replication scheme can improve the overall system performance in other aspects as well.*

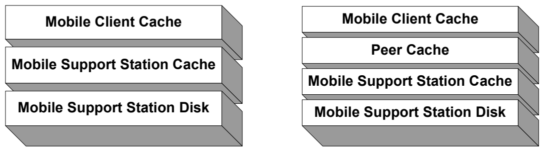
1. Introduction

With the recent widespread deployment of new peer-to-peer (P2P) communication technologies, such as IEEE 802.11 and Bluetooth, there is a new alternative for information sharing among clients over the standard client/server model. Coupled with the fact that the computation power and storage capacity of mobile devices have been improving at a fast pace, mobile clients can now communicate among themselves to share information rather than having to rely on their connection to the server for each request. This new information sharing alternative is known as *mobile cooperative caching*.

In the past, cooperative caching schemes were exclusively studied in wired networks [4, 5, 9, 11]. Recently, cooperative caching schemes in mobile environments have been drawing increasing attention. In [6, 7, 10], cooperative caching schemes were proposed in mobile ad hoc networks. Our work is different from previous works in that we propose a cooperative caching scheme in conventional mobile systems, in the presence of *mobile support stations* (MSSs) and is based on single hop communication within their servicing cells. We also distinguish our work in investigating the performance of the cooperative caching scheme in a highly dynamic environment where *mobile hosts* (MHs) or *mobile clients* roam freely, and may experience disconnection from the network, fall into idle mode and move away from the service area unconsciously [1, 2, 8].

In this paper, a **CO**operative **CA**ching (COCA) scheme is proposed for MHs in mobile environments. In conventional mobile systems, the storage hierarchy consists of three layers: Mobile Client Cache, MSS Cache and MSS Disk, as depicted in Figure 1(a). MHs send requests to the MSS, if they cannot find the required data items in their cache (called a *cache miss*). The MSS grabs the required data items from its disk if the data items are not cached in the MSS cache. In COCA, we insert a new logical layer between the Mobile Client Cache layer and the MSS Cache layer. This layer is called the *Peer Cache* layer, as depicted in Figure 1(b). In the context of cooperative caching, when an MH suffers from a cache miss (called a *local cache miss*), the MH will look up the required data item from its neighboring peers' cache before enlisting the MSS for help. Only when the MH cannot find it in the peers' cache (called a *global cache miss*) will it request the item from the MSS as in the conventional system. However, if the MH does not reside in the service area, it encounters an *access miss*.

In COCA, each MH and its neighboring peers, i.e., the MHs residing in the transmission range of the MH, work to-


Mobile Client Cache
Mobile Support Station Cache
Mobile Support Station Disk
Mobile Client Cache
Peer Cache
Mobile Support Station Cache
Mobile Support Station Disk

(a) Conventional (b) COCA

Figure 1. Storage hierarchy of mobile systems.

*Research supported in part by the Research Grant Council and the Hong Kong Polytechnic University under grant numbers PolyU 5084/01E and H-ZJ86.

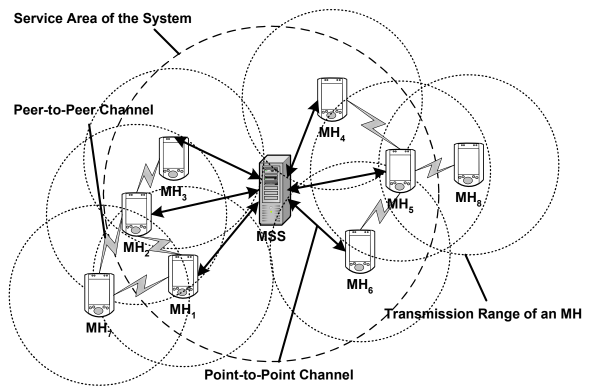


Figure 2. System architecture of COCA.

gether as a *dynamic group* to share their cached data items cooperatively via the P2P channels, as illustrated in Figure 2. For instance, when *MH*$_2$ encounters a local cache miss, it attempts to obtain the required data items from its members in its dynamic group: *MH*$_1$, *MH*$_3$ and *MH*$_7$, before sending the request to the MSS. In conventional mobile systems, *MH*$_7$ is essentially disconnected from the network, since it resides outside of the service area. Therefore, it encounters an *access miss* when the required data item cannot be found in its local cache with traditional mobile data management schemes. Fortunately, in COCA, *MH*$_7$ can try to get help from its neighboring peers: *MH*$_1$ and *MH*$_2$, before constituting an access miss. If its peers can turn in the required data item, it constitutes a *global cache hit*.

COCA is appropriate for an environment in which a group of MHs possesses a common access interest. For example, in an exhibition, the MHs residing near the same booth or an area are probably interested in the information related to the booth. Likewise, in a shopping mall, the MHs residing in the same shop would likely access the information related to the shop. When the MHs share a common access interest, there is a higher probability for them to retrieve the required data items from their neighboring peers.

The rest of this paper is organized as follows. Section 2 describes the COCA model. Section 3 describes the characterization of the client's activities in the COCA system. Section 4 presents the data replication scheme in COCA. Section 5 defines the simulation model and studies the performance of COCA and the data replication scheme through a number of simulated experiments. Finally, Section 6 offers brief concluding remarks.

2. COCA Model

In COCA, we assume that each MH has its own cache space with the same capacity ($ClientCacheSize$). An MH becomes disconnected from the network when it enters *sleep* mode by turning off the device. When an MH falls into *idle* mode, the device is still powered on, but it does not generate any request. Also, an MH residing outside the service area cannot communicate with the MSS. For simplicity, we further assume that there is no update of data items. This assumption will be relaxed in our future work. In addition, we assume that a P2P communication channel is available for the MHs to communicate with their neighboring peers. Each MH's request can be satisfied by either connecting to the MSS or its neighboring peers. Before describing the COCA request handling mechanism, we first discuss the four possible outcomes of each client request.

Local Cache Hit (LCH). The local cache space caches the data items that were obtained by the MH. If the required data item is found in the MH's local cache, it constitutes a local cache hit.

Global Cache Hit (GCH). When the required data item is not cached, the MH attempts to retrieve it from its neighboring peers. If some of the peers can turn in the data item, that constitutes a global cache hit.

Cache Miss. If the MH residing in the service area fails to achieve a local cache hit or a global cache hit (i.e., a cache miss), it can make a connection to the MSS to obtain the required data item.

Access Miss. When the MH which does not reside in the service area fails to achieve a local cache hit or a global cache hit, it suffers from an access miss.

For each request, the MH first looks for the required data item at the local cache. If it cannot find it, it broadcasts the request to its neighboring peers. The peers who cache the data item return the data item to the MH. If no neighboring peer caches it, the MH has to retrieve it from the MSS. However, when the MH is residing outside of the service area and encounters a global cache miss, it suffers from an access miss. The upper bound on the time that an MH spends on the Peer Cache layer is a timeout period when there is no peer returning the required data item.

Consider an MH M, with a set of neighboring peers $\mathcal{P}(M)$, which M can directly communicate with in P2P mode. Let A_M be the set of accessed data items, C_M be the set of data items cached and P_M be the set of data items cached by its neighboring peers $\mathcal{P}(M)$. Thus, $P_M = \cup_{m \in \mathcal{P}(M)} C_m$. The LCH ratio $Hit_L(M)$ is $\frac{|A_M \cap C_M|}{|A_M|}$ and GCH ratio $Hit_G(M)$ is $\frac{|(A_M \cap P_M) - (C_M \cap P_M)|}{|A_M|}$. The overall cache hit rate $Hit(M) = Hit_L(M) + Hit_G(M) = \frac{|A_M \cap (C_M \cup P_M)|}{|A_M|}$. COCA can achieve better system performance, as $Hit_G(M)$ is normally larger than zero, i.e., $(A_M \cap P_M) - (C_M \cap P_M) \neq \emptyset$. To obtain higher Hit_G, M and its neighboring peers $\mathcal{P}(M)$ are expected to own similar access set $(A \cap P)$ and should try to cache more *distinct* items in the set $(C \cup P)$.

COCA brings several benefits to the MH and the MSS. In some cases, the telecommunication service providers charge the MHs at a unit cost $Cost$ per access when they access remote data items in the mobile system. The cost can be based on the number of sessions, connection time or message volume. The MHs have to pay for the cost of $|A| \times (1 - Hit_L) \times Cost$. With COCA, the MHs can save a cost of $|A| \times Hit_G \times Cost$. Likewise, the access miss ratio can be reduced by the amount of Hit_G. Furthermore, the server workload can be reduced from $N \times |A| \times (1 - Hit_L)$ to $N \times |A| \times (1 - Hit_L - Hit_G)$, where N is the number of MHs in the system.

3. Client Activity

In COCA, we assume that each MH can be in one of two states: low activity (LA) and high activity (HA). An MH in LA state is called a Low Activity Mobile host (LAM) and called a High Activity Mobile host (HAM) in HA state. LAMs rarely generate accesses to the system; however, they do not turn off their devices, since it is common that these MHs may use their devices for other purposes, such as listening to music, reading electronic books, playing games and so on. Each MH detects its own state by monitoring the *cache access rate* (CAR). CAR is re-calculated every period of time ϕ and is defined as:

$$CAR(t_c, t_l) = \frac{N_a}{t_c - t_l} \quad (1)$$

where $CAR(t_c, t_l)$ is CAR in the time interval $[t_c, t_l)$, N_a is the number of cache accesses during ϕ, t_c is the current time and t_l is the time when CAR was last calculated.

If an MH has a high *sharing ability* (SA), it is considered as a LAM. The required SA value of an MH at a particular point of time is inversely proportional to CAR, i.e., the higher the CAR value, the lower the SA value. Thus, SA can be defined to be $1/(CAR + 1)$, where the offset of one in the denominator ensures that the SA value is always between zero and one. Initially, SA is set to zero. To learn about the gradual change in SA over a period of time by utilizing a summary of the history, we define SA in form of a weighted sum between the new and old estimates as:

$$SA_t = \omega \times \frac{1}{CAR + 1} + (1 - \omega) \times SA_{t-1} \quad (2)$$

where ω is a parameter to weight the importance of the most recent SA value at time t with that of the previous SA value at time $t - 1$.

The state of an MH is defined by a threshold μ. If the SA value is higher than or equal to μ, the MH is considered as a LAM. Likewise, if the SA value is lower than μ, the MH is classified as a HAM. The threshold μ must be set carefully. In the case that the MSS keeps on replicating data items to an MH whose state has changed from LAM to HAM, the system performance of the MH will be degraded. Thus, a threshold value that is able to detect the client activity changes sensibly is needed.

To use μ to detect an MH changing from LA to HA state within a period T, $T = i\phi$, where $i = 1, 2, 3 \ldots$ and $T > 0$, μ can be set by Equation 3:

$$\mu = SA_{t-T}(1 - \omega)^i + \frac{1}{\lambda + 1}(1 - (1 - \omega)^i) \quad (3)$$

where SA_{t-T} is the SA value before time T, $i = \frac{T}{\phi}$ and λ is the CAR obtained by an MH during the time interval T.

For instance, we would like to detect the state change from LAM to HAM within two cycles, assuming a cycle length of $\phi = 10$, the detection period is $2\phi = 20$. Further assume that we use a value of $\omega = 0.25$. An MH with an SA value of 1 at time $t - 20$ can observe 5 data accesses on average during each CAR cycle, $\lambda = \frac{5}{\phi} = 0.5$ and the required value of μ will be 0.85.

4. Data Replication Scheme

To utilize the cache space of the LAMs, the MSS replicates appropriate data items to them so as to allow HAMs to access the replicas. The selection of data items sent to the LAMs is based on their access probabilities. The MSS records the access probability p of each data item. For data item i, p_i is initially set to zero and t_l is set to the current time, and then p_i is updated as it is requested by an MH based on the following equation:

$$p_i^{new} = \omega \times \frac{1}{t_c - t_l} + (1 - \omega) \times p_i^{old} \quad (4)$$

where t_c is the current time, t_l is the last access time and ω is a parameter to weight the importance of the most recent access. Then, t_l is set to the current time.

Since the access probability collected by the server is past information, it can only be used to predict the trend of the access pattern. For example, the 100 data items with the highest access probability at time t_c are likely cached by the MHs, which possibly cache these data items after they are obtained from the MSS. Thus, replication of these 100 data items to LAMs may not help HAMs when the latter encounter a local cache miss. We find that the skewness of the access probability to the replicated data items can be useful to the improvement of the overall system performance.

Each LAM can keep the most α recently accessed data items in its local cache, where α is a parameter set by the MH. When the state of an MH residing in the service area changes from HA to LA, the MH sends a *replication* request to the MSS along with the information about what data items are cached in its local cache. After the MSS receives the request, it considers the list of data items maintained in a descending ordering of p. The MSS skips the δ data items with the highest access probability and considers data items from the $\delta + 1^{st}$ position to the $\delta + ClientCacheSize^{th}$ position. Then, the MSS will send the items to the MH, omitting those already cached at the MH to reduce system traffic. For each replication period η, if the MH remains to be LAM, it sends another replication request to the MSS. The MSS then processes the request again as above.

5. Simulation Model

In this section, we present the simulation model used to evaluate the performance of COCA and the data replication scheme. The simulation model is implemented in C++ using CSIM. It consists of a client model and a server model. The simulated mobile environment is composed of a single MSS and $NumClient$ MHs. There are $NumLAM$ LAMs and $NumClient{-}NumLAM$ HAMs. The MHs can move freely in a 500 m $\times$ 500 m space ($Area$). The service area ($ServiceArea$) is represented by a 450 m $\times$ 450 m space centrally wrapped by $Area$. In the simulation, the database in the MSS contains $NumData$ equal-sized (1 KByte) data items. In the simulation model, there are wireless communication channel with a bandwidth 10 Mbit/s and P2P channel with a bandwidth 1 Mbit/s, through which an MH retrieves the data items from MSS and communicates with its neighboring peers. The MHs' devices adopt the same P2P communication interface, with a transmission range of 50 meters.

Client Model. The time interval between two consecutive requests generated from the MHs follows an exponential distribution with a mean of one second. Each MH generates accesses to the data items following a Zipf distribution with a skewness parameter θ. If θ is set to zero, MHs uniformly access the data items. As θ is increasing, the access to the data items becomes more skewed. After finishing a request, an MH has a probability P_{Sleep} (P_{Idle}) to fall into the sleep (idle) mode with a period following an exponential distribution with a mean of $SleepTime$ ($IdleTime$). For the LAM and HAM, they have different P_{Idle} and $IdleTime$.

In our simulation, a separate autonomous process controls the movement pattern for each MH. The MHs move according to the "random waypoint" model [3]. At the beginning, the MHs are randomly distributed in $Area$. Each MH randomly chooses its own destination in $Area$ with a randomly determined speed s from a uniform distribution U (v_{min}, v_{max}). It then travels with the constant speed s. When it reaches the destination, it comes to a standstill for a constant time ($PauseTime$) to determine its next destination. It then moves towards its new destination with another random speed $s' \sim U$ (v_{min}, v_{max}). All MHs repeat this movement behavior during the simulation. When an MH moves away from the $ServiceArea$, it becomes disconnected from the MSS.

Server Model. There is a single MSS in the system. The MSS receives and processes requests from the MHs with a first-come-first-serve policy. An infinite queue is used to buffer the requests from the MHs when the MSS is busy. The MSS has a cache capacity of $ServerCacheSize$, and it adopts the LRU cache replacement policy. Each cache miss in the MSS incurs disk I/O access time (10 ms) for retrieving the required data item from the disk.

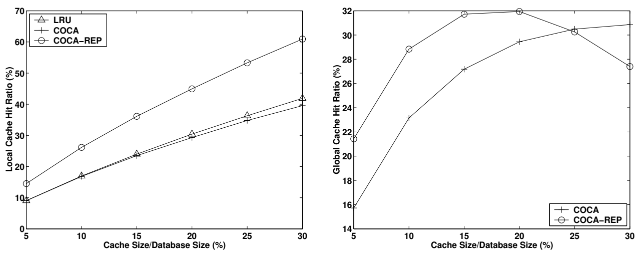
Table 1 summarizes the default parameters used in the simulated experiments. The first part of the table gives the default parameters used in the system and server models, and the second part specifies the default parameters for the client model.

Table 1. Simulation default parameters.

Parameters	Values
$NumClient$	100
$NumLAM$	20 % of $NumClient$
$NumData$	1000 items
$ServerCacheSize$	50 % of *NumData*
δ	$ClientCacheSize$
ω	0.25
ϕ	10 s
η	10 s
μ	0.85
$ClientCacheSize$	10 % of *NumData*
$Speed$ ($v_{min} \sim v_{max}$)	1 $\sim$ 5 m/s
θ	0.5
P_{Sleep}	0.1
P_{Idle}	LA: 0.95, HA: 0.05
$SleepTime$	10.0 s
$IdleTime$	LA: 100 s, HA: 10 s
$PauseTime$	1.0 s

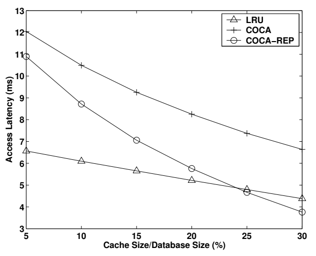
5.1. Simulation Experiments

In the simulation, we compare COCA schemes with a conventional caching approach that does not involve cooperation among peers. This serves as a base case for comparison. LRU replacement strategy is applied in the base case (LRU) and in pure COCA (COCA). Likewise, we evaluate the performance of COCA with the data replication scheme (COCA-REP) by comparing it with COCA. All simulation results are recorded after the system reaches a stable state. Each MH generates 20,000 queries where 2,000 are warm-up queries in order to avoid a transient effect. We conduct the experiment by varying the parameters: cache size, transmission range and percentage of LAMs. In our simulation, the performance metrics include access latency, LCH and GCH ratios, access miss ratio and server request ratio that indicates the server workload, and the amount of messages transmitted in the system. The last metric includes messages transmitted via both the wireless channel connecting the MSS and the P2P channels. In order to be fair to other caching strategies, each data item replicated to the LAMs by MSS is counted towards one message. In reality, the

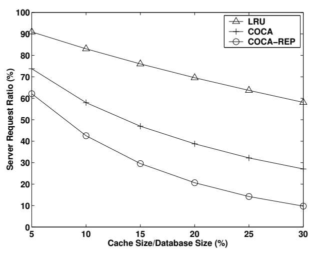


(a) LCH Ratio (b) GCH Ratio

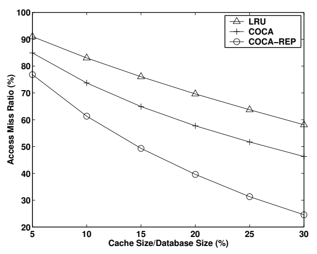
Figure 3. Breakdown of cache hit ratios.



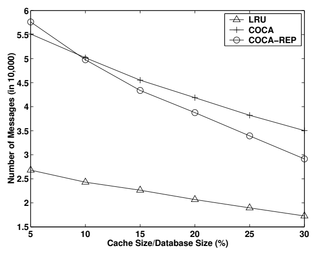
(a) Access Latency



(b) Server Request Ratio

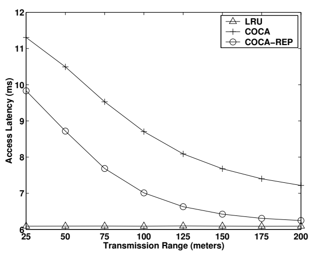


(c) Access Miss Ratio

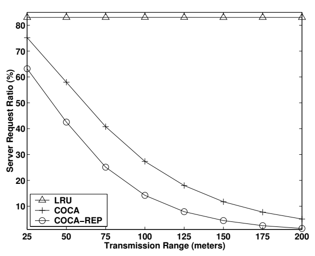


(d) Number of Messages

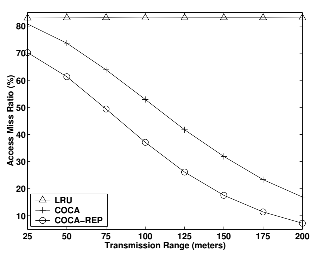
Figure 4. Performance studies on various cache size.



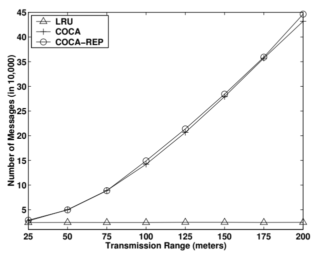
(a) Access Latency



(b) Server Request Ratio



(c) Access Miss Ratio



(d) Number of Messages

Figure 5. Performance studies on various transmission ranges.

data items are delivered in bulk and the actual cost is lower.

5.2. Effects of Cache Size

Our first experiment studied the effect of cache size on the system performance by varying the ratio of cache size and database size from 5 percent to 30 percent.

Figure 3 shows that the LCH and GCH ratios increase with the increasing cache size. COCA-REP achieves the highest LCH ratio because the data items replicated to LAMs enable them to contain more valuable data items accessible when they become HAMs and also to other peer HAMs. From Figure 3(b), the GCH ratio of COCA-REP peaks at a cache size of 20 percent and then drops. This is because when more required data items can be found in the local cache, as depicted in Figure 3, the need for accessing the global cache is alleviated, and the contribution by LAMs reduces.

Figure 4 illustrates the system performance improvement as the cache gets larger. When the cache size is larger than 25 percent of the database, COCA-REP performs the best in terms of access latency. As the global cache is large enough, the MHs can access most required data items from their local cache and their neighboring peers' cache, so it eases the access delay caused by cache misses to the server. COCA-REP is the best performer in both the server request and access miss ratios, as shown in Figures 4(b) and 4(c) respectively, because COCA-REP achieves the highest cache hit ratio.

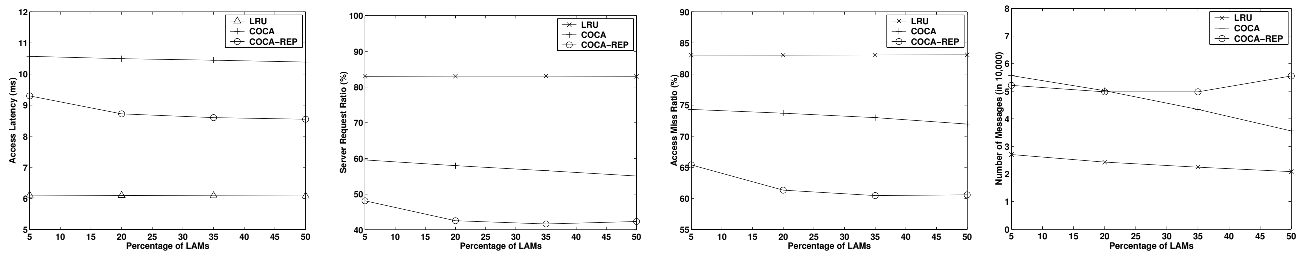
The amount of messages transmitted in COCA schemes is about double that of LRU, as shown in Figure 4(d). The amount of messages of LRU decreases with cache size, as more data items can be found in the local cache. Likewise, the amount of messages of COCA schemes decreases with cache size, as more data items can be accessed in the local cache and global cache. Further, the message count for COCA-REP is lower than COCA after the cache size reaches 10 percent of the database size. This is because for the LAMs, they have higher LCH ratio in COCA-REP than those in COCA and LRU, so the LAMs rely decreasingly on their peers' cache.

5.3. Effects of Transmission Range

In this series of simulated experiments, we examined the influence to system performance by varying the transmission range, from 25 to 200 meters.

Figure 5 indicates that the performance of COCA schemes improves as the transmission range increases. This is because the more neighboring peers' cache an MH can access, the better performance can be achieved. COCA-REP also performs better than COCA in access latency, server request ratio and access miss ratio, as shown in Figures 5(a), 5(b) and 5(c), because the data replication scheme improves data availability in the system.

The amount of messages transmitted in the system significantly increases as the transmission range gets larger. This is due to the fact that as an MH can communicate with more neighboring peers, it needs to send more messages to its peers when it encounters a local cache miss. To alleviate this problem, the MHs should maintain hints on the set of data items possibly cached by their neighboring peers. The overheads of hints exchange can be reduced by means of lazy updates on the hints.



(a) Access Latency (b) Server Request Ratio (c) Access Miss Ratio (d) Number of Messages

Figure 6. Performance studies on various percentage of LAMs.

5.4. Effects of Percentage of LAMs

Finally, we studied the effect of percentage of LAMs on the system performance by varying the percentage of LAMs from 5 percent to 50 percent.

COCA-REP performs better than COCA in access latency, server request ratio and access miss ratio, as depicted in Figures 6(a), 6(b) and 6(c). However, the performance of COCA-REP is stabilized after the percentage of LAMs in the system reaches 20 percent. This is because the MSS adopts the same replication mechanism and δ parameter to replicate data items to LAMs. The degree of overlapping among the local cache of the receiving LAMs increases, and the effectiveness of cooperative caching diminishes. To further improve the performance in COCA-REP, different replication mechanisms or δ parameter should be adopted to a group of LAMs which reside closely to one another.

Figure 6(d) shows that the amount of messages of COCA and LRU decreases as there are more LAMs in the system because there is a higher chance for the MHs to fall into the idle mode, with longer idle period. These LAMs generate fewer requests. In a COCA system, the more the LAMs are, the lower is the traffic generated. However, the amount of messages of COCA-REP increases with the percentage of LAMs in the system, due to the overheads of the data replication to LAMs from the MSS. To alleviate this problem, the MSS can select some representatives of a group of LAMs which reside closely to one another to disseminate the data items. This arrangement can reduce the amount of messages required in the data replication scheme and the degree of overlapping of data items in the cache of a group of closely located LAMs.

6. Conclusion

In this paper, we have proposed a cooperative caching strategy, called COCA, and a data replication scheme adapted to COCA for mobile environments. In COCA, the MHs share their cache with each other to reduce both the number of server requests and access misses. In order to utilize the cache space of the MHs with low activity, the data replication scheme is applied to COCA. The performance of pure COCA and COCA with the data replication scheme is evaluated through a series of simulated experiments showing that they reduce the number of server requests and access miss ratio; however, in general, they incur longer access latency when the MHs encounter a global cache miss. The results also show that COCA with the data replication scheme can effectively reduce the access latency and further reduce the number of server requests and access misses.

References

[1] D. Barbara. Mobile computing and databases - a survey. *IEEE TKDE*, 11(1):108–117, January 1999.

[2] D. Barbara and T. Imielinski. Sleepers and workaholics: Caching strategies for mobile environments. In *Proc. of the ACM SIGMOD*, pages 1–12, May 1994.

[3] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. of the 4th MobiCom*, pages 85–97, October 1998.

[4] M. Dahlin, R. Y. Wang, T. E. Anderson, and D. A. Patterson. Cooperative caching: Using remote client memory to improve file system performance. In *Proc. of the 1st USENIX OSDI*, pages 267–280, November 1994.

[5] M. J. Feeley, W. E. Morgan, F. H. Pighin, A. R. Karlin, H. M. Levy, and C. A. Thekkath. Implementing global memory management in a workstation cluster. In *Proc. of the 15th ACM SOSP*, pages 201–212, December 1995.

[6] T. Hara. Effective replica allocation in ad hoc networks for improving data accessibility. In *Proc. of the 20th INFOCOM*, pages 1568–1576, April 2001.

[7] T. Hara. Cooperative caching by mobile clients in push-based information systems. In *Proc. of the 11th CIKM*, pages 186–193, November 2002.

[8] T. Imielinski and B. R. Badrinath. Mobile wireless computing: Challenges in data management. *Communications of the ACM*, 37(10):18–28, October 1994.

[9] L. Ramaswamy and L. Liu. A new document placement scheme for cooperative caching on the internet. In *Proc. of the 22nd ICDCS*, pages 95–103, July 2002.

[10] F. Sailhan and V. Issarny. Cooperative caching in ad hoc networks. In *Proc. of the 4th MDM*, pages 13–28, January 2003.

[11] P. Sarkar and J. H. Hartman. Hint-based cooperative caching. *ACM TOCS*, 18(4):387–419, November 2000.