GroCoca: Group-based Peer-to-Peer Cooperative Caching in Mobile Environment

Chi-Yin Chow, *Student Member, IEEE,* Hong Va Leong, *Member, IEEE Computer Society,* and Alvin T. S. Chan, *Member, IEEE*

***Abstract*— In a mobile cooperative caching environment, we observe the need for cooperating peers to cache useful data items together, so as to improve cache hit from peers. This could be achieved by capturing the data requirement of individual peers in conjunction with their mobility pattern, for which we realized via a *GROup-based COoperative CAching scheme* (GroCoca). In GroCoca, we define a *tightly-coupled group* (TCG) as a collection of peers that possess similar mobility pattern and display similar data affinity. A family of algorithms is proposed to discover and maintain all TCGs dynamically. Furthermore, two cooperative cache management protocols, namely, cooperative cache admission control and replacement, are designed to control data replicas and improve data accessibility in TCGs. A cache signature scheme is also adopted in GroCoca in order to provide information for the mobile clients to determine whether their TCG members are likely caching their desired data items and to perform cooperative cache replacement. Experimental results show that GroCoca outperforms the conventional caching scheme and standard *COoperative CAching scheme* (COCA) in terms of access latency and global cache hit ratio. However, GroCoca generally incurs higher power consumption.**

***Index Terms*— Mobile computing, peer-to-peer computing, mobile data management, cooperative caching, cache signatures.**

I. INTRODUCTION

With recent widespread deployment of new peer-to-peer wireless communication technologies, such as IEEE 802.11 and Bluetooth, coupled with the fast improvement in the computation processing power and storage capacity of most portable devices, a new information sharing paradigm, known as peer-to-peer (referred to as P2P) information access has rapidly taken shape. The mobile clients can communicate among themselves to share information rather than having to rely solely on the server. This paradigm of sharing cached information among the mobile clients voluntarily is called *mobile* P2P *cooperative caching*.

Conventionally, mobile systems are built based on *infrastructure-based* and *ad-hoc-based* architecture. An infrastructure-based mobile system is formed with a wireless network connecting mobile hosts (MHs) and mobile support stations (MSSs). The MHs are clients equipped with portable devices, while MSSs are stationary servers supporting information access for the MHs residing in their service areas. The MHs can retrieve their desired information from the MSSs, by requesting over shared point-to-point channels (pull-based data dissemination model), downloading from scalable broadcast channels (push-based data dissemination model), or utilizing both types of channels (hybrid data dissemination model). For ad-hoc-based mobile communication architecture (mobile ad hoc network or MANET), the MHs can share information among themselves without any help from the MSSs, being referred to as a P2P data dissemination model. In this work, we propose a novel communication architecture, in which P2P data dissemination model is used in combination with a mobile environment supporting pull-based dissemination.

In a pull-based mobile environment, MHs have to retrieve their desired data items from an MSS when they encounter local cache misses. Since a mobile environment is characterized by limited bandwidth, the communication channel between the MSS and the MHs could become a scalability bottleneck. Although the push-based and hybrid data dissemination models are scalable, the MHs adopting these models generally suffer from longer access latency and higher power consumption, as they need to tune in to the broadcast and wait for the broadcast index or their desired items to appear.

MANET is practical to a mobile system with no fixed infrastructure support, such as battlefield, rescue operations and so on [1]. However, it is not as suitable for commercial mobile applications. In MANETs, the MHs can rove freely and disconnect themselves from the network at any instant. These two characteristics lead to dynamic changes in the network topology. As a result, the MHs could suffer from poor access latency and access failure rate, when the peers holding the desired data items are far way or unreachable (disconnected).

The inherent shortcomings of infrastructure-based architecture and MANET render mobile applications adopting either architecture alone not as appropriate in most real commercial settings. In reality, poor access latency and access failure rate could cause the abortion of valuable transactions or the suspension of critical activities, reducing user satisfaction and loyalty, and potentially bringing damages to the organization involved. The drawbacks of existing mobile data dissemination models motivate us to develop a novel data accessing scheme, based on an *integrated communication structure*, for deploying mobile information access applications in reality. In [2], [3], [4], [5], we have proposed a mobile COoperative CAching scheme (COCA) that combines the conventional infrastructure-based architecture with P2P communication technology. The MHs

Manuscript received December 15, 2005. This work was presented in part at the 33rd International Conference on Parallel Processing (ICPP), Montreal, Quebec, Canada, August 2004. This manuscript contains the following new materials: cache signature scheme in Section IV.D, cooperative cache management protocols in Section IV.E, cache consistency in Section IV.F, and three new sets of experiments (effect of access pattern, data update rate, and client disconnection) in Section VI.

This research is supported in part by the Hong Kong Research Grant Council under grant number PolyU 5084/01E and The Hong Kong Polytechnic University under grant number H-ZJ86.

C.-Y. Chow is with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, email: cchow@cs.umn.edu. H.V. Leong and A.T.S. Chan are with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, email: {cshleong, cstschan}@comp.polyu.edu.hk

adopting COCA can obtain their desired data items only not from the MSS, but also from the cache of their peers.

In a mobile environment, client mobility and data access patterns are key factors in cache management. Major issues in cooperative cache management include cache replacement and admission control strategies to increase data accessibility, with respect to individual MHs and their peers. The decision of whom a cached data item should be forwarded to thus depends on both access affinity on data items and the mobility pattern.

In this paper, we propose a *GROup-based COoperative CAching scheme* (GroCoca) based on the concept of a *tightly-coupled group* (TCG), which is a group of MHs that are *geographically and operationally close*, i.e., sharing common mobility and data access pattern. It is not difficult to define whether two MHs are *geographically close*, based on their locations. Two MHs are said to be *operationally close*, if they perform similar operations and access similar set of data items. In GroCoca, two cooperative cache management protocols, *cooperative cache admission control* and *cooperative cache replacement*, are designed for the MHs to manage their cache space with respect to themselves and their TCG members.

The rest of this paper is organized as follows. Section II gives a survey on some important work related to mobile cooperative caching and group formation techniques in mobile environment. Section III presents the system architecture and communication protocol of COCA. Section IV delineates the various aspects of GroCoca. Section V describes the simulation model of GroCoca and its performance is evaluated in Section VI. Finally, Section VII concludes the paper.

II. Related Work

Recently, cooperative caching in mobile environment has been drawing increasing attention. Several cooperative caching schemes were proposed within recent years. These works can be divided into two major categories: *cooperative data dissemination* and *cooperative cache management*. Cooperative data dissemination research mainly focuses on how to search for desired data items and how to forward the data items from the source MHs or the MSSs to the requesting MHs [6], [7], [8], [9]. Cooperative cache management research focuses on how MHs cooperatively manage their cache as an aggregate cache to improve system performance along such design dimensions as *cooperative data replica allocation* [10], [11], [12], *cooperative cache invalidation* [13] and *cooperative cache admission control and replication* [14].

In the past, several distributed clustering algorithms have been proposed for mobile environment. The two simplest distributed clustering algorithms are the lowest-ID [15] and largest-connectivity (degree) [16] algorithms. To improve the stability of clusters in MANETs, several mobility-based clustering algorithms [17], [18], [10], [19], [20], [21] are proposed. A bottom-up clustering algorithm considering the mobility pattern of MHs is proposed [17]. The MHs record their own mobility pattern in a mobility profile, which is periodically exchanged with other peers. The clustering criterion is based on the relative velocity between MHs. Inter-cluster merging algorithm is then executed to combine clusters that are within a certain hop distance.

A *dynamic connectivity based group* formation scheme is proposed to group the MHs with highly stable connection together [10]. A group of MHs possesses a high connection stability, as they form a biconnected component in the network. An incremental clustering algorithm is proposed to discover group mobility pattern, to alleviate network partitioning [21]. When the system detects that some network partitions are likely to materialize, it replicates appropriate data items among mobility groups to improve data accessibility. The DRAM mobility-based clustering algorithm improves data accessibility in the system, by considering not only the current motion information of the MHs, but also their historical motions [19]. When a cluster is constructed, the data replica allocation algorithm is executed to place appropriate data items in cluster members to improve data accessibility.

GBL is another distributed mobility-based clustering algorithm for group-based location update in mobile environment [20]. The MH with the highest degree of affinity in neighborhood is assigned as the cluster leader. The degree of affinity is defined by the distance between an MH and its peers, together with the similarity in their movement vectors. Each cluster leader is responsible for updating the location of its cluster members to location databases. Other than using velocity or distance to determine the mobility pattern, signal power detection is also adopted to calculate the relative mobility metric for distributed clustering in MANETs [18].

Our work distinguishes itself from previous works in that we propose a group formation algorithm which takes into account the *geographical vicinity* property of communication groups, as well as *operational vicinity* property of computation groups. Two cooperative cache management protocols are proposed for the MHs to control data replicas and improve data accessibility within their TCGs.

III. COCA

In COCA, a neighbor discovery protocol (NDP) [22], [23] is assumed to be available. NDP is a simple protocol in which the neighbor connectivity is maintained through a periodic beacon of “hello” message with other useful information, e.g., the identifier or communication address of the sender. Each MH broadcasts a “hello” message to its neighboring peers for every beacon interval. If an MH has not received a beacon message from a known peer for some beacon cycles, it considers that there is a link failure with that peer. In addition, there are two P2P communication paradigms: *point-to-point* and *broadcast*. In P2P point-to-point communication, there is only one destination MH for the message sent from the source MH. In P2P broadcast communication, all MHs residing in the transmission range of the source MH receive the broadcast message. Before describing the system model, we first examine the four possible outcomes of a client request.

1) **Local Cache Hit (LCH).** If the required data item is found in the MH’s local cache, it constitutes a local cache hit; otherwise, it is a local cache miss.
2) **Global Cache Hit (GCH).** When the required data item is not cached, the MH attempts to retrieve it from its peers. If some peers can turn in the required item, that constitutes a global cache hit.

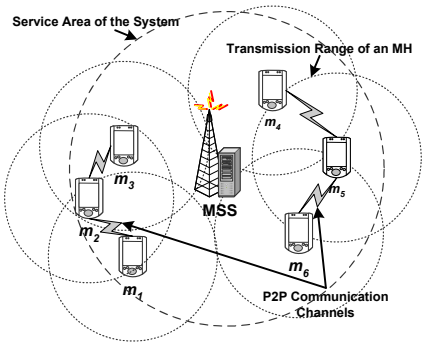


Fig. 1. System architecture of COCA.

3) **Cache Miss (or Server Request).** If the MH fails to achieve a local cache hit or a global cache hit, it encounters a cache miss, and has to make a contact with the MSS to obtain the item.
4) **Access Failure.** If the MH encounters a cache miss, and fails to access the item from the MSS, as it is residing outside of the service area or the MSS is down or overloaded, that is an access failure.

COCA is based on the system architecture in Figure 1. Each MH and its peers work together to share their cached data items cooperatively via P2P communication. For instance, when an MH, m_2, encounters a local cache miss, it broadcasts a request to its peers, m_1 and m_3. If any peer turns in the required item, a *global cache hit* is recorded; otherwise, it is a *global cache miss*, and m_2 has to enlist the MSS for help.

The communication protocol of COCA specifies that an MH should first find its desired data item in its local cache for each query. If it encounters a local cache miss, it broadcasts a request message to its peers within the distance of a prescribed maximum number of hops (*HopDist*), via P2P broadcast communication. Any peer caching the required item will send a reply message back to the requestor via P2P point-to-point communication. When the MH receives the first reply from a peer, that peer is selected as a *target peer*. The MH next sends a retrieve message to the target peer via P2P point-to-point communication. The peer receiving retrieve turns in the required item through P2P point-to-point communication. If no peer sends a reply back to the requestor upon timeout, the item needs to be requested from the MSS.

The timeout period, τ, is adaptive to network congestion. Initially, τ is set to a default value of the round-trip time of a P2P transmission between two MHs at a distance of *HopDist* scaled up by a congestion factor, φ, i.e., $\frac{(|\text{request}|+|\text{reply}|)}{BW_{\text{P2P}}} \times HopDist \times \varphi$, where BW_{P2P} is the bandwidth of the P2P communication channel and $|\text{request}|$ and $|\text{reply}|$ are the size of a request and reply message respectively. For each search in the peers' cache, an MH records the time duration, τ', from the moment when the MH broadcasts a request to the moment when a reply is received. Then, τ is set to the average time duration, $\overline{\tau'}$, plus another system parameter, φ', times the standard deviation of τ', $\sigma_{\tau'}$, i.e., $\tau = \overline{\tau'} + \varphi'\sigma_{\tau'}$. Both $\overline{\tau'}$ and $\sigma_{\tau'}$ can be calculated incrementally [24].

IV. GroCoca

GroCoca defines and makes use of the concept of a *tightly-coupled group* (TCG), which is defined as a group of MHs sharing common mobility pattern and data affinity. Two cooperative cache management protocols, namely, *cooperative cache admission control* and *cooperative cache replacement* protocols, are proposed for an MH to manage its own cache space with respect not only to itself, but also to its TCG members. In GroCoca, an exponentially weighted moving average (EWMA) measure [25] is used to discover common mobility pattern, while the similarity in access pattern is captured via a vector space model [26].

A. Similarity Measurement in Mobility Patterns

The mobility pattern is modeled by the weighted average distance between any two MHs. The MHs need not explicitly update their locations, but they piggyback the location information on the request sent to the MSS when they are requesting data items. The location information represented by a coordinate (x, y) is returned from a global positioning system (GPS) or indoor sensor-based positioning system, like MIT BAT or Cricket. For two MHs, m_i and m_j, the distance between them is calculated as their Euclidean distance, $|m_i m_j|$. EWMA is used to forecast the future distance of each pair of MHs based on their historical mobility patterns. The weighted average distance between two MHs, m_i and m_j, is denoted as $||m_i m_j||$. After the MSS receives the location information of both m_i and m_j for the first time, $||m_i m_j||$ is set to $|m_i m_j|$. Then $||m_i m_j||$ is updated when either m_i or m_j sends its new location to the MSS:

$$\|m_i m_j\|^{new} = \omega \times |m_i m_j| + (1-\omega) \times \|m_i m_j\|^{old}, \qquad (1)$$

where ω ($0 \leq \omega \leq 1$) is a parameter to weight the importance of the most recent distance. A pair of MHs are considered to possess common mobility pattern, if their weighted average distance is less than or equal to a prescribed threshold, Δ. The weighted average distance of each pair of MHs is stored in a two-dimensional *weighted average distance matrix* (WADM).

B. Similarity Measurement in Data Access Patterns

GroCoca considers also the similarity of MH accesses to the data items. In the MSS, each data item is associated with an identifier. The MSS maintains a counter vector of length *NData* for each MH to store its data access pattern, where *NData* is the number of data items at the server.

When an MH accesses a data item, the MSS increments the corresponding counter for the MH. The similarity score of the access pattern of two MHs, m_i and m_j, is calculated by:

$$\text{sim}(m_i, m_j) = \frac{\sum_{d=1}^{NData} A_i(d) \times A_j(d)}{\sqrt{\sum_{d=1}^{NData} A_i(d)^2} \times \sqrt{\sum_{d=1}^{NData} A_j(d)^2}}, \qquad (2)$$

where $A_i(d)$ is the frequency that m_i accessed a data item, d, and $0 \leq \text{sim}(m_i, m_j) \leq 1$. If the measured similarity score of two MHs is larger than or equal to a prescribed threshold, δ, they are considered to possess a similar access pattern. The similarity score of each pair of MHs is stored in a two-dimensional *access similarity matrix* (ASM).

Since the uplink channel is scarce, a passive approach is adopted for collecting the data access pattern. An MH does not send any data access information actively to the MSS, but

the MSS learns this access pattern from the received requests along with location information sent by the MH. The collected information is just a sample of the mobility and access pattern. To improve accuracy, the thresholds for weighted average distance and data access similarity should be made lower. When an MH has not sent any request to the MSS for a period of time, τ_P, the MH explicitly reports its location and a portion, ρ_P, of the data access history, i.e., items retrieved from other peers since last explicit update or request.

C. Tightly-Coupled Group Discovery Algorithm

The Tightly-Coupled Group (TCG) discovery algorithm is executed in the MSS. When the MSS receives a request from an MH, it extracts the location information from the request and then calculates its weighted average distance to its every peer (Algorithm 1). Meanwhile, the MSS updates the score on data access similarity between the MH and each peer (Algorithm 2). A peer, m_j, is considered to be a TCG member of an MH, m_i, if two conditions, $||m_i m_j|| \leq \Delta$ and $\mathsf{sim}(m_i, m_j) \geq \delta$, are satisfied (Algorithm 3). Since $||m_i m_j|| = ||m_j m_i||$ and $\mathsf{sim}(m_i, m_j) = \mathsf{sim}(m_j, m_i)$, the TCG relation is symmetric. Thus, when the MSS adds m_i into m_j's TCG, it also adds m_j into m_i's TCG.

The MSS postpones the announcement of changes in TCG membership to the affected MHs until they send explicit updates or requests to the MSS. In the former case, the MSS only sends the changes in the TCG membership to the MHs. In the latter case, the MSS processes the request and sends the required data items along with the changes in the TCG membership to the requesting MHs. In effect, we are performing an asynchronous group view change [27] without enforcing stringent consistency among group members.

Algorithm 1 TCG Discovery Algorithm: location update

```
1: procedure LocationUpdate(M, WADM, ASM, TCG_i, Δ, δ, ω)
   // M: the set of MHs in the system
   // WADM and ASM: weighted average distance matrix and access similarity matrix
   // TCG_i: identifiers of MHs in the TCG of m_i
   // Δ, δ: prescribed weighted average distance and data access similarity thresholds
   // ω: weight parameter for weighted average distance between two MHs
2: for all m_j ∈ M do
3:     if m_i ≠ m_j then
4:         ||m_i m_j||^new ← ω × |m_i m_j| + (1 − ω) × ||m_i m_j||^old;
5:         CheckTCGMembership(m_i, m_j, WADM, ASM, TCG_i, Δ, δ);
6:     end if
7: end for
```

Algorithm 2 TCG Discovery Algorithm: data access pattern

```
1: procedure ReceiveRequest(M, A, WADM, ASM, TCG_i, Δ, δ)
   // A: the set of vector A for each MH
2: for all m_j ∈ M do
3:     if m_i ≠ m_j then
4:         sim(m_i, m_j) ← (Σ_{d=1}^{NData} A_i(d) × A_j(d)) / (√(Σ_{d=1}^{NData} A_i(d)^2) × √(Σ_{d=1}^{NData} A_j(d)^2));
5:         CheckTCGMembership(m_i, m_j, WADM, ASM, TCG_i, Δ, δ);
6:     end if
7: end for
```

D. Cache Signature Scheme

Our cache signature is based on the bloom filter and is compressed based on a variable-length-to-fixed-length (VLFL) run-length encoding scheme. After a brief review of the techniques adopted, we describe the use of cache signatures to filter unnecessary search and the signature exchange protocol and extend the protocol to cater for client disconnection.

Algorithm 3 TCG Discovery Algorithm: membership checking

```
1: procedure CheckTCGMembership(m_i, m_j, WADM, ASM, TCG_i, Δ, δ)
2: if ||m_i m_j|| ≤ Δ and sim(m_i, m_j) ≥ δ then
3:     if m_j ∉ TCG_i then
4:         TCG_i ← TCG_i ∪ {m_j}; TCG_j ← TCG_j ∪ {m_i};
5:     end if
6: else
7:     if m_j ∈ TCG_i then
8:         TCG_i ← TCG_i − {m_j}; TCG_j ← TCG_j − {m_i};
9:     end if
10: end if
```

1) Bloom Filter Data Structure: A bloom filter is used to represent a set $S = \{s_1, s_2, \ldots s_n\}$ of n data items. A vector with σ bits, $V = \{v_i | i \in [0, \sigma - 1]\}$, is initially set to zero. There are k independent hash functions, $\{h_i | i \in [1, k]\}$, each yielding a hash value between 0 and $\sigma - 1$. To construct a bloom filter, an element $s \in S$ is hashed by the k hash functions to produce k hash values, $h_i(s)$. Then, the corresponding bits in V at the position of the hash values are set to one, i.e., $\forall j \in [1, k], v_{h_j(s)} = 1$. This procedure is repeated for each element in S.

To check whether s' is in V, s' is first hashed by the k hash functions. If all bits at positions indicated by the hash functions are set in V, verified with a bitwise and operation, i.e., $\forall j \in [1, k], v_{h_j(s')} = 1$, s' is probably in V. Otherwise, $\exists j \in [1, k], v_{h_j(s')} = 0$ and s' is definitely not in V.

Since each bit in V can be randomly set multiple times by different elements, a bloom filter could result in a *false positive*; this occurs when the existence of an element is indicated, but the element is actually not stored in it. For mathematical simplicity, it is commonly assumed that all bits are independently set to zero or one [28]. After hashing all elements in S into V, the probability that a particular bit still remains zero is $\left(1 - \frac{1}{\sigma}\right)^{nk}$. The probability of the occurrence of a false positive is $\left[1 - (1 - \frac{1}{\sigma})^{nk}\right]^k$. The optimal number of hash functions, k, to minimize false positive probability is $k = (\ln 2)\left(\frac{\sigma}{n}\right)$ [28].

2) Compressed Cache Signatures: When an MH merely caches a small portion of data items, there are many "zeros" in its cache signature. The transmission overhead can be reduced, if the bloom filter is compressed before being transmitted to other peers. A VLFL run-length encoding is used to compress the cache signature because it is simple and yet practical for most mobile devices. The VLFL run-length encoding consists of two steps. First, the sequence of bits is decomposed into run-lengths terminated by two cases: there are R consecutive "zeros", where $R = 2^l - 1$ for a positive integer l, and there are L consecutive "zeros" followed by a single "one", where $0 \leq L < R$. Second, a fixed-length codeword, which indicates the length of the run-length, is assigned to each run-length. Assuming that all "zeros" are uniformly distributed, the probability of the VLFL encoding process encountering a "zero" in a signature is $\phi = \left(1 - \frac{1}{\sigma}\right)^{nk}$. Thus, the probability of different run-lengths, $0 \leq L \leq R$ would be $P(L) =$

$\phi^L(1-\phi)$ when $0 \leq L < R$ and $P(L) = \phi^R$ when $L = R$. Thus, the expected length of an intermediate symbol, that is either a run-length with R consecutive "zeros" or a run-length with L consecutive "zeros" and a terminator "one", is $\eta = \sum_{i=0}^{R-1}(i+1)P(i)+RP(R) = \frac{1-\phi^R}{1-\phi}$. The expected length of a compressed cache signature, σ', is $\sigma' = \frac{\sigma}{\eta}\log_2(R+1)$. Algorithm 4 is used to find an optimal value of R. We take advantage of the VLFL run-length encoding when $\log_2(R+1) < \eta$. An MH makes a local decision on whether to compress its cache signature before transmitting it to other peers based on three factors: its cache size (ε) in terms of number of data items that can be stored in the cache, the bloom filter size (σ) and the number of independent hash functions (k). A cache signature should be compressed, if $\log_2(\widehat{R}+1) < \frac{1-\phi^{\widehat{R}}}{1-\phi}$, where $\widehat{R} = \mathsf{FindOptimalR}(\varepsilon, \sigma, k)$; otherwise, the MH simply sends its cache signature without compression.

Algorithm 4 Finding an optimal value of *R*.

```
1:  procedure FindOptimalR (ε, σ, k)
2:  φ ← (1 − (1/σ)^(εk)); min_σ' ← +∞; R ← 1; R̂ ← 1; i ← 1;
3:  while i ≤ (1−φ^R)/(1−φ) do
4:      tmp_σ' ← σi(1−φ)/(1−φ^R);
5:      if tmp_σ' < min_σ' then
6:          min_σ' ← tmp_σ'; R̂ ← R;
7:      else
8:          break;
9:      end if
10:     i ← i + 1; R ← 2^i − 1;
11: end while
12: return R̂;
```

3) Filtering Mechanism: In the cache signature scheme, there are four types of signatures, *data signature*, *cache signature*, *peer signature* and *search signature*. A data signature for a data item is a bloom filter produced by hashing its unique identifier, such as integral identifier, keywords, attribute values, URL or content with k hash functions. A cache signature is a bloom filter that summarizes a cache content by superimposing all data signatures of the cached data items. An MH produces a peer signature by superimposing all cache signatures of its neighboring peers. When an MH encounters a local cache miss, it generates a search signature for its desired item by setting the bit at the appropriate positions for the item. A bitwise and operation is applied on the search signature and peer signature. If the result is the same as the search signature, indicating that some peers are likely caching the item, the MH searches the peers' cache for it; otherwise, it bypasses the peers' cache, and obtains the item from the MSS directly.

An MH has to regenerate the cache signature after a cache insertion or eviction. To reduce processing overhead, a proactive approach is used to generate cache signatures. Each MH maintains a counter vector with σ counters, each counter with π_c bits. When a data item is inserted into (evicted from) the local cache, a data signature is generated for the item, and counters at the position of the bits set in the data signature are incremented (decremented). The MH can construct a new cache signature by simply setting the bits at the position of counters with non-zero values in a bloom filter. The value of a counter is bounded by its size, π_c. Increment operations will not be executed on counters with value $2^{\pi_c} - 1$. Likewise, if the value of a counter becomes zero, decrement operation would be discarded, and the MH would reset and reconstruct the counter vector to avoid false negative.

4) Cache Signature Exchange Protocol: In GroCoca, the MHs adopt a counter vector structure with dynamic counter size to store cache signatures from their peers. They only exchange their signatures with those peers belonging to their own TCGs to enhance the stability of the counter vector. Each MH maintains a vector of σ peer counters, where σ is the size of bloom filter, each counter with π_p bits. If an MH does not have any TCG member, π_p is zero. Once the MH discovers a newly joined member in its TCG, it creates a one-bit counter vector and sends a SigRequest message to it. The peer receiving SigRequest returns its full cache signature to the requesting MH. Then, the MH updates the counter vector by incrementing the counters at the position of the bits that are set in the received cache signature. When the MH detects a new TCG member, it sends SigRequest to the member, and the member turns in its cache signature. After the MH receives the cache signature, it updates the counter vector. When the MH is going to increase a counter to 2^{π_p}, the counter size, π_p, will expand. Likewise, in case that all counter values fall below 2^{π_p-1}, the counter size will contract.

After the MH synchronizes the TCG membership information with the MSS, when it detects a TCG member departing, it resets the counter vector, and then recollects all cache signatures from its remaining TCG members. The MH broadcasts SigRequest to its neighborhood with the membership information. A peer receiving the request checks whether itself is a member. If it is, it returns its cache signature; otherwise, it simply drops the request. In case that the network is extremely dynamic, the MH does not recollect all cache signatures from its remaining TCG members. Instead, it recollects the cache signatures only after a certain number of members have departed the TCG. However, the extra delay on the recollection could lead to higher false positives.

To reduce communication overhead on updating cache signature, the MHs embed the signature update information to request message broadcast to all their neighboring peers. When an MH receives request from a peer, the MH checks whether the peer belongs to its own TCG. If so, the MH extracts the signature update information, and updates the counter vector accordingly. The signature update information is maintained in two lists: *insertion list* and *eviction list*. The insertion list stores the bit position set, and the eviction list stores the bit position reset, since the last time the MH broadcasts request along with signature update information. If a bit position exists in both lists, the change is annihilated.

5) Client Disconnection Handling Protocol: Since client disconnection is one of the most common phenomena in mobile environment, GroCoca is extended to handle client disconnection. After a disconnected MH reconnects to the network, it synchronizes its TCG membership information with the MSS. The MH resets the counter vector and sends SigRequest along with its TCG membership information to its peers. The peers finding their identifiers in the membership information send their full cache signatures back to the MH. Then, the MH reconstructs the counter vector based on the re-

ceived cache signatures. However, some members may disconnect from the network, when the MH is recollecting their cache signatures. Each MH thus maintains a list, *OutstandSigList*, to record members not yet turned in their cache signatures. After the MH obtains the up-to-date membership information from the MSS, it resets *OutstandSigList* and inserts all members to the list. When an MH detects a peer in *OutstandSigList* reconnecting to network, it sends SigRequest to the peer. When the MH receives the cache signature from the peer, it updates the counter vector and removes the peer from the list.

E. Cooperative Cache Management Protocols

In GroCoca, two cooperative cache management protocols: cooperative cache admission control and cooperative cache replacement, are designed for an MH to manage its own cache space with respect to itself and its TCG members.

The cooperative cache admission control protocol provides a means for the MHs to control data replicas in their TCGs. When an MH encounters a local cache miss, it sends a request to its peers. If some peers turn in the required item and the local cache is not full, the MH caches the item, no matter it is returned by a peer in the same TCG or not. However, if the local cache is full, the MH does not cache the item when it is supplied by a peer in the same TCG, on the belief that the item can be readily available from the peer if needed. If the cache is full but the data item comes from a peer outside of its TCG, the MH would rather cache the item by removing the least valuable data item, since the providing peer may move far away in future. After a peer sends the required item to the requesting MH, if they are belonging to the same TCG, the peer updates the last access timestamp of the item, so that the item can be retained longer in the global cache. If there are more than one member caching the same item, the MH selects the member caching the item with the longest time-to-live (*TTL*) to update the last access timestamp of the item.

The cooperative cache replacement protocol allows the MH to collaborate with its TCG members to replace the least valuable item with respect to the MH and other TCG members. It satisfies the three desirable properties for caching [4]. First, the most valuable items are always retained in the local cache. Second, in a local cache, an item which has not been accessed for a long period, is replaced eventually. Third, in a global cache, an item which "spawns" replica is first replaced to increase the effective cache size.

To retain the most valuable data items in the local cache, only a number of $ReplaceCandidate$ least valuable items are selected as the replacement candidates. Among the candidates, the least valuable item is first chosen, and the MH generates a data signature for it. The data signature is then compared with the peer signature by a bitwise and operation. If the result is the same as the data signature, the item is likely a replica in the global cache, so it is replaced. Otherwise, the second least valuable item is considered. If no candidate is probably replicated in the TCG, the least valuable item is replaced.

There could be a wasting issue with the arrangement above: a data item without any replica could always be retained in the local cache, even though it will not be accessed again. If most items cached belong to such type of item, the MH's cache will be populated with useless items, degrading the LCH ratio. To solve this problem, a *SingletTTL* counter is associated with each candidate. *SingletTTL* is initially set to $ReplaceDelay$. When the least valuable item is not replaced because it does not have any replica, its counter is decreased by one. Cache replacement mechanism is skipped if the counter value of the least valuable item becomes zero, and the MH simply drops that item from the cache. The counter is reset to $ReplaceDelay$, when the item is accessed by the MH or its TCG members.

F. Cache Consistency

Since maintaining strong cache consistency in mobile environment is very costly, an on-demand lazy cache consistency strategy is more suitable than a strict one [25]. In COCA, a timestamp-based on-demand cache consistency strategy is adopted. When an MH retrieves a data item from the MSS, the MSS assigns a *TTL* to the item, and then the MH records the retrieve time, t_r, for the item. When the MH accesses the item in response to a request, it checks the validity of the item by determining whether *TTL* has expired. The MH considers the cached item to be valid, if its *TTL* has not expired. Otherwise, it validates the item by consulting the MSS. If the item has been updated in the MSS, i.e., $t_r < t_l$, where t_l is its last updated timestamp, the MSS returns the up-to-date copy of the item; otherwise, the MSS just approves its validity, and then the MH accesses the cached copy. The MSS records the EWMA update interval for each item. Consider a data item, x, with a last updated timestamp, t_l^x, the data update interval, u_x, is re-calculated when x is updated at t_c, as $u_x^{new} = \alpha(t_c - t_l^x) + (1-\alpha)u_x^{old}$, where α is a parameter to weight the importance of the most recent data update. After that, t_l^x is set to t_c. To deal with items without any update for a long period, the EWMA update rates of all items are examined periodically. If an item, x, has not been updated for a time period longer than u_x, u_x^{new} is updated to $\alpha(t_c - t_l^x) + (1-\alpha)u_x^{old}$. When an MH accesses x from the MSS, the MSS assigns a new *TTL* to x as $\max(u_x - (t_c - t_l^x), 0)$. When an MH encounters a local cache miss, it searches the peers' cache for its desired item. The peer only turns in the required item to the MH, if the *TTL* of the item has not expired. If no peer caches a valid copy of the item, the MH has to enlist the MSS for help.

V. Simulation Model

The simulation model for GroCoca is implemented in C++ using CSIM. The simulated mobile environment is composed of an MSS and *NClient* MHs in a space of 1000 $\times$ 1000 meters. The total bandwidth of wireless communication between the MSS and the MHs is BW_{server}. There is a half-duplex wireless channel for an MH to communicate with its peers with a total bandwidth of BW_{P2P} and a transmission range of *TranRange*.

A. Power Consumption Model

Each MH is equipped with two wireless network interface cards (NICs), in which one is dedicated to communicate with

Condition	P_{p2p} power	v	f
$m = S$	$(v_{send} \times b) + f_{send}$	1.9	454
$m = D$	$(v_{recv} \times b) + f_{recv}$	0.5	356
$m \in S_R \wedge m \in D_R$	$(v_{sd\_disc} \times b) + f_{sd\_disc}$	0	70
$m \in S_R \wedge m \notin D_R$	$(v_{s\_disc} \times b) + f_{s\_disc}$	0	24
$m \notin S_R \wedge m \in D_R$	$(v_{d\_disc} \times b) + f_{d\_disc}$	0	56

Condition	P_{bc} Power	v	f
$m = S$	$(v_{bsend} \times b) + f_{bsend}$	1.9	266
$m \in S_R$	$(v_{brecv} \times b) + f_{brecv}$	0.5	56

TABLE I

POWER CONSUMPTION MEASUREMENT IN P2P COMMUNICATION.

MSS, while the other is devoted to communicate with other peers. For P2P communication, all MHs are assumed to be equipped with the same type of wireless NICs with an omnidirectional antenna so that all MHs within the transmission range of a transmitting MH can receive its transmission. Furthermore, the wireless NIC of the non-destination MH is operated in an idle mode during the transmission. The power consumption measurement model uses a set of linear formulas to measure the power consumption of the source MH, S, the destination MH, D, and other *remaining* MHs residing in the transmission range of the source MH, S_R, and the destination MH, D_R [29], The power consumption of P2P point-to-point and broadcast are measured as in Table I, where f ($\mu W \cdot s$) is the fixed setup cost for a transmission, and v ($\mu W \cdot s/byte$) is the variable power consumption on message size in bytes (b).

B. Client Model

The MHs are divided into several motion groups, each with $GroupSize$ MHs. The mobility of the MHs is based on the *reference point group mobility model* [30], a random mobility model for a group of MHs. The movement of each group is based on the *random waypoint mobility model* [31], with a uniformly distributed moving speed from v_{min} to v_{max}, and one-second pause time. The MHs of the same motion group share a common access range on data items, generating accesses following a Zipf distribution with a skewness parameter θ. The interarrival time between data accesses from an MH follows an exponential distribution with a mean of one second. An MH disconnects from the network after completing a request with a probability, P_{disc}, for a period of time, *DiscTime*, uniformly distributed from d_{min} to d_{max}.

C. Server Model

There are *NData* equal-sized (*DataSize*) data items. The MSS receives and processes the requests sent by the MHs with a first-come-first-serve policy. An infinite queue is used to buffer the outstanding requests from the MHs when the MSS is busy. All data items are randomly updated in the MSS with an update rate, *DataUpdateRate* items per second. Table II shows the default setting and varying range of all simulation parameters in the simulated experiments.

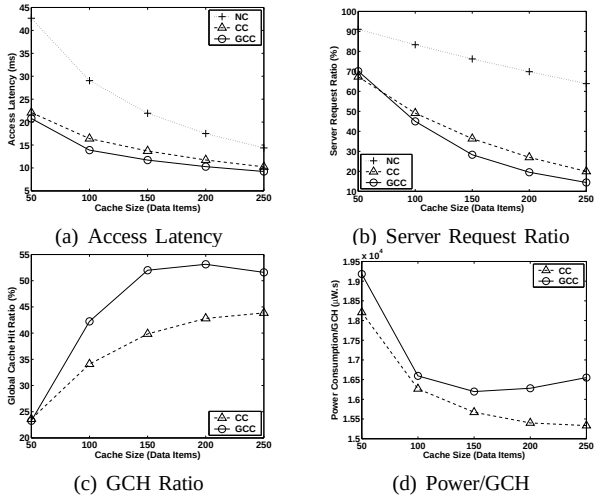
VI. SIMULATION RESULTS

The performance of GroCoca (denoted as GCC) is compared with a conventional caching scheme that does not involve any cooperation among MHs (denoted as NC) and standard COCA (denoted as CC). All schemes adopt least recently used (LRU) cache replacement policy. We start the recording of simulation results after the system reaches a stable state, in which all client caches are full, in order to avoid a transient effect. A simulated experiment terminates after each MH generates over 2000 requests beyond the warmup period.

Parameter	Default Value	Range
NClient	100	$50 - 400$
NData	10,000	-
BW_{server}	Downlink 2,400 Kbits/s	-
	Uplink 153.6 Kbits/s	-
BW_{P2P}	2,000 Kbits/s	-
$TranRange$	100 m	-
$DataSize$	3 KBytes	-
$DataUpdateRate$	0 (no data update)	$0 - 500$
$CacheSize$	100 data items	$50 - 250$
$AccessRange$	1000 data items	$500 - 10,000$
σ, k	40,000, 2	-
θ	0.5	$0 - 1$
$Speed$ ($v_{min} \sim v_{max}$)	$1 \sim 5$ m/s	$1 - 30$
P_{disc}	0	$0 - 0.3$
$DiscTime$ ($d_{min} \sim d_{max}$)	$1 \sim 5$ s	-
$ReplaceCandidate$	20 data items	-
$ReplaceDelay$	2	-
ω, α	0.25, 0.25	-
φ, φ'	10, 3	-
τ_P, ρ_P	10 s, 0.5	-

TABLE II

SIMULATION PARAMETERS.



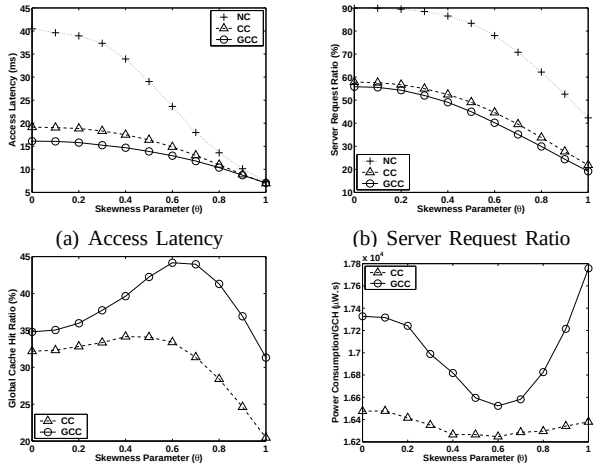
(a) Access Latency (b) Server Request Ratio

(c) GCH Ratio (d) Power/GCH

Fig. 2. Effect of cache size on system performance.

A. Effect of Cache Size

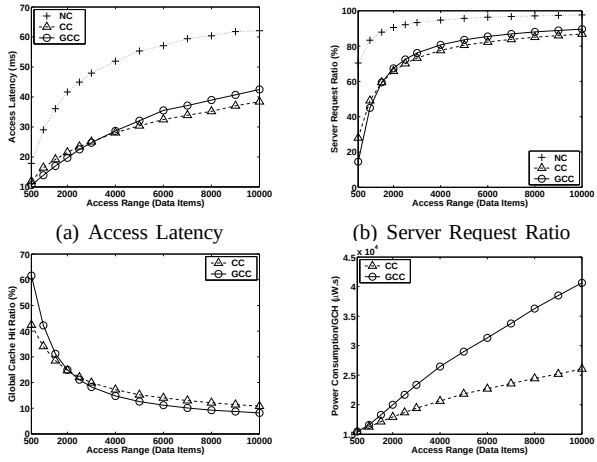
Our first simulated experiment studies the effect of cache size on system performance by increasing the cache size from 50 to 250 data items. Figures 2(a) and 2(b) show that the access latency and server request ratio improve, as the cache gets larger. The MHs can cache more items with increasing cache size, so it improves the LCH ratio. For the MHs adopting COCA schemes, it not only achieves a higher LCH ratio, but it also enjoys a higher GCH ratio, as depicted in Figure 2(c). This is because there is a higher chance for the MHs to obtain their desired items from their peers, with larger cache. Since GCC further improves the GCH ratio in TCGs, the MHs with GCC records the highest GCH ratio. The access latency and server request ratio of GCC are thus better than NC and CC. Although the MHs adopting GCC enjoy a higher GCH ratio, they have to consume more power on cache signature scheme. However, we still observe a lower power consumption per GCH due to higher GCH. With a much larger cache, the larger



(a) Access Latency (b) Server Request Ratio

(c) GCH Ratio (d) Power/GCH

Fig. 3. Effect of skewness in access pattern on system performance.



(a) Access Latency (b) Server Request Ratio

(c) GCH Ratio (d) Power/GCH

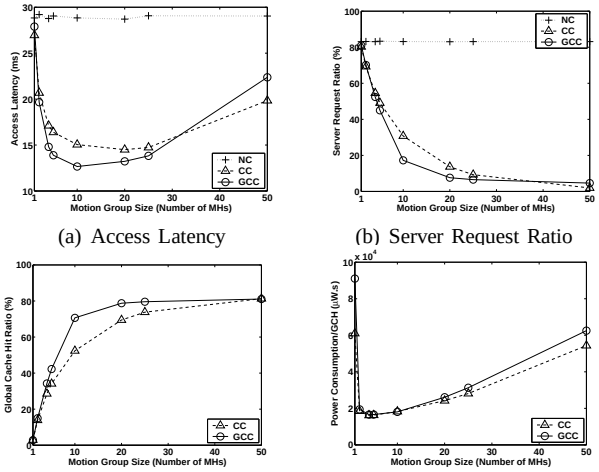
Fig. 4. Effect of access range in access pattern on system performance.

increase in LCH leads to a slightly diminishing GCH, hence a higher power consumption per GCH.

B. Effect of Access Pattern

In the second experiment, we study the effect of access pattern on system performance by varying the Zipfian skewness parameter value, θ, from zero to one and the data access range from 500 to 10,000 data items. Figures 3(a) and 3(b) reveal that the access latency and server request ratio improve with increasing θ for all schemes. When $\theta = 0$, the MHs access the data items uniformly. On the other hand, their access patterns become more skewed, with larger θ. The MHs with a more skewed access pattern are likely to find their desired items in the local cache, so the LCH ratio improves, leading to an improvement in access latency and server request ratio.

The GCH ratio of CC and GCC initially improves, but it drops with further increasing θ, as illustrated in Figure 3(c). When θ increases, the range of hot data items becomes smaller, so there is a higher probability for an MH to retrieve its desired items from its TCG members. However, as θ further increases, the MH enjoys a much higher LCH ratio that eases the demand for the global cache, so the GCH ratio drops. When the GCH ratio drops, the power consumption of the cache signature



(a) Access Latency (b) Server Request Ratio

(c) GCH Ratio (d) Power/GCH

Fig. 5. Effect of motion group size on system performance.

scheme is amortized by fewer number of GCHs. Thus, the power consumption per GCH gets higher due to reduced GCH ratio at high access skewness, as illustrated in Figure 3(d).

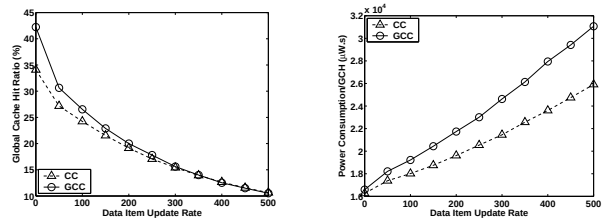
The performance of all schemes gets worse, when the access range increases, as exhibited in Figure 4. The larger access range, the more distinct data items the MHs are likely to access, so they suffer from lower LCH and GCH ratios. These lower LCH and GCH ratios lead to system performance degradation, in terms of access latency, server request ratio and power consumption per GCH. The result also reveals that CC and GCC perform better than NC, but CC is more effective than GCC, when the access range gets larger.

C. Effect of Motion Group Size

To evaluate the effect of motion group size on system performance, we increase $GroupSize$ from 1 to 50. When the motion group size is equal to one, the mobility model is equivalent to an individual *random waypoint mobility model*. The MHs adopting CC and GCC score the worst GCH ratio when the group size is equal to one, as shown in Figure 5(c), that constitutes the worst case of CC and GCC in terms of access latency, server request ratio and power consumption per GCH, as depicted in Figures 5(a), 5(b) and 5(d) respectively. The server request and GCH ratios of CC and GCC improve with increasing motion group size because there is a higher chance for the MHs to obtain their desired items from their peers, when there are more neighboring peers with similar data affinity from which help can be sought. The increasing motion group size brings two effects on system performance. First, the larger motion group size leads to higher power consumption per GCH because the MHs have to handle more global cache queries and discard more unintended messages, as illustrated in Figure 5(d). Second, it also induces higher network traffic around the vicinity of a motion group, which in turn increases the latency of global cache accesses, as shown in Figure 5(a).

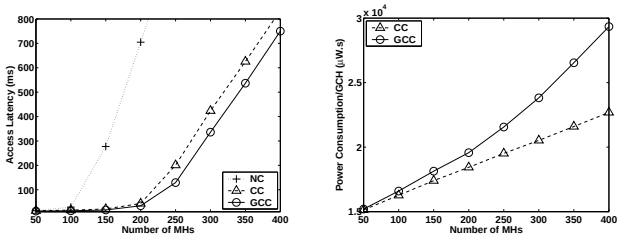
D. Effect of Data Item Update Rate

We next study the effect of data item update rate by increasing the rate from 0 to 100 items per second. The



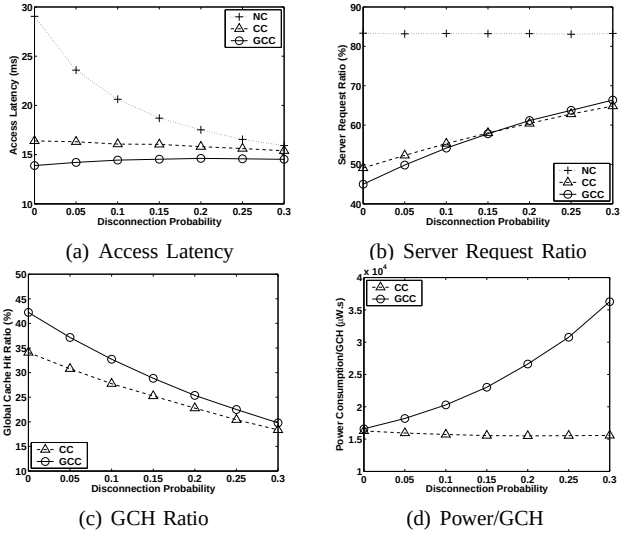
(a) GCH Ratio (b) Power/GCH

Fig. 6. Effect of data update rate on system performance.



(a) Access Latency (b) Power/GCH

Fig. 7. Effect of number of MHs on system performance.



(a) Access Latency (b) Server Request Ratio

(c) GCH Ratio (d) Power/GCH

Fig. 8. Effect of disconnection probability on system performance.

performance of all schemes gets worse with increasing data item update rate, as shown in Figure 6. This is because there is a higher chance for the MHs to encounter local cache or global cache misses, when the update rate increases. The lower LCH and GCH ratios lead to system performance degradation in access latency and server request ratio. When the GCH ratio drops, the power consumption on searching the peers' cache of CC and GCC and cache signature scheme of GCC is amortized over fewer GCHs, so power consumption per GCH rises with increasing update rate.

E. Effect of Number of Mobile Hosts

The system scalability is evaluated by increasing the number of MHs from 50 to 400. Figure 7(a) illustrates that the access latency of NC increases sharply, when there are more than 100 MHs; it also reveals that CC and GCC can effectively improve system scalability. Since the access range of each motion group is randomly assigned, the increasing number of MHs does not bring in very impressive improvement in the GCH ratio as anticipated. When two motion groups with no or little common access range come close together, they may not be able to take advantage of the cached data items from one another; they actually degrade the system performance in terms of power consumption per GCH, as depicted in Figure 7(b). This is because they have to consume more power not only to receive more broadcast requests from the peers of another motion group, but also to discard unintended messages.

F. Effect of Client Disconnection

Finally, we study the effect of client disconnection by varying the disconnection probability, P_{disc}, from 0 to 0.3, as shown in Figure 8. The access latency of NC decreases with increasing P_{disc}, as depicted in Figure 8(a). This is because the congestion of the downlink channel relieves, when there are more disconnected MHs. However, the more disconnected MHs, the lower GCH ratio the MHs adopting CC and GCC experience, as shown in Figure 8(c). It is due to the fact that there are fewer peers that an MH can enlist for help, when P_{disc} gets larger. When P_{disc} is 0.15, the MHs adopting GCC suffer from a higher server request ratio than those adopting CC, as shown in Figure 8(b).

The MHs adopting CC consume less power per GCH with increasing P_{disc}, as shown in Figures 8(d). The power consumption on receiving broadcast requests and discarding unintended messages reduces, when there are more disconnected MHs in the system. However, the MHs with GCC suffer from higher power consumption per GCH, as P_{disc} increases. This is because the MHs have to execute client disconnection handling protocol that leads to higher power consumption on receiving (transmitting) cache signatures from (to) their TCG members, after they reconnect to the network. Therefore, when the MHs disconnect from the network more frequently, they not only suffer from a lower GCH ratio, but they also have to consume more power per GCH.

VII. CONCLUSION

In this paper, we propose a *GROup-based COoperative CAching scheme*, namely, GroCoca, for MHs in mobile environment. In GroCoca, a collection of MHs that possess similar mobility pattern and data affinity form a group, called *tightly-coupled group* (TCG). A TCG discovery algorithm is proposed to discover all TCGs dynamically in system. The cache signature scheme is adopted not only to provide hints for the MHs to make local decision on whether to search the peers' cache for their desired data items, but it also provides information for the MHs to perform cooperative cache replacement in their TCGs. We then compress the cache signature to reduce the power consumption on transmitting cache signatures between MHs. Within a TCG, two cooperative cache management protocols are designed for the MHs to work together to manage their cache space as an aggregate cache. These two *cooperative cache admission control* and *cooperative cache replacement* protocols are adopted to control data replicas and improve data accessibility in a TCG respectively. Experimental results depict that GroCoca further improves system performance in comparison to standard cooperative

caching scheme, which already yields an improvement over conventional caching scheme. However, the MHs adopting GroCoca generally suffer from higher power consumption.

REFERENCES

[1] L. D. Fife and L. Gruenwald, "Research issues for data communication in mobile ad-hoc network database systems," *ACM SIGMOD Record*, vol. 32, no. 2, pp. 42–47, June 2003.

[2] C.-Y. Chow, H. V. Leong, and A. T. S. Chan, "Cache signatures for peer-to-peer cooperative caching in mobile environments," in *Proc. of International Conference on Advanced Information Networking and Applications*, Mar. 2004, pp. 96–101.

[3] ——, "Group-based cooperative cache management for mobile clients in a mobile environment," in *Proc. of ICPP*, Aug. 2004, pp. 83–90.

[4] ——, "Peer-to-peer cooperative caching in a hybrid data delivery environment," in *Proc. of International Symposium on Parallel Architectures, Algorithms, and Networks*, May 2004, pp. 79–84.

[5] ——, "Utilizing the cache space of low-activity clients in a mobile cooperative caching environment," *International Journal of Wireless and Mobile Computing*, to appear.

[6] W. H. O. Lau, M. Kumar, and S. Venkatesh, "A cooperative cache architecture in support of caching multimedia objects in MANETs," in *Proc. of MobiCom Workshop on Wireless Mobile Multimedia*, Sep. 2002, pp. 56–63.

[7] M. Papadopouli and H. Schulzrinne, "Effects of power conservation, wireless coverage and cooperation on data dissemination among mobile devices," in *Proc. of MobiHoc*, Oct. 2001, pp. 117–127.

[8] F. Sailhan and V. Issarny, "Cooperative caching in ad hoc networks," in *Proc. of International Conference on Mobile Data Management*, Jan. 2003, pp. 13–28.

[9] H. Shen, S. K. Das, M. Kumar, and Z. Wang, "Cooperative caching with optimal radius in hybrid wireless network," in *Proc. of International IFIP-TC6 Networking Conference*, May 2004, pp. 841–853.

[10] T. Hara, "Effective replica allocation in ad hoc networks for improving data accessibility," in *Proc. of INFOCOM*, Apr. 2001, pp. 1568–1576.

[11] ——, "Cooperative caching by mobile clients in push-based information systems," in *Proc. of CIKM*, Nov. 2002, pp. 186–193.

[12] ——, "Replica allocation in ad hoc networks with periodic data update," in *Proc. of Interntional Conference on Mobile Data Management*, Jan. 2002, pp. 79–86.

[13] H. Hayashi, T. Hara, and S. Nishio, "Cache invalidation for updated data in ad hoc networks," in *Proc. of International Conference on Cooperative Information Systems*, Nov. 2003, pp. 516–535.

[14] S. Lim, W.-C. Lee, G. Cao, and C. R. Das, "A novel caching scheme for internet based mobile ad hoc networks," in *Proc. of ICCCN*, Oct. 2003, pp. 38–43.

[15] A. Ephremides, J. Wieselthier, and D. J. Baker, "A design concept for reliable mobile radio networks with frequency hopping signaling," *Proceedings of IEEE*, vol. 75, no. 1, pp. 56–73, Jan. 1987.

[16] A. K. Parekh, "Selecting routers in ad-hoc wireless network," in *Proc. of International Telecommunications Symposium*, Aug. 1994, pp. 420–424.

[17] B. An and S. Papavassiliou, "A mobility-based clustering approach to support mobility management and multicast routing in mobile ad-hoc wireless networks," *International Journal of Network Management*, vol. 11, no. 6, pp. 387–395, Nov. 2001.

[18] P. Basu, N. Khan, and T. D. Little, "A mobility based metric for clustering in mobile ad hoc networks," in *Proc. of ICDCS Workshop on Wireless Networks and Mobile Computing*, Apr. 2001, pp. 413–418.

[19] J.-L. Huang, M.-S. Chen, and W.-C. Peng, "Exploring group mobility for replica data allocation in a mobile environment," in *Proc. of CIKM*, Nov. 2003, pp. 161–168.

[20] G. H. K. Lam, H. V. Leong, and S. C. F. Chan, "GBL: Group-based location updating in mobile environment," in *Proc. of DASFAA*, Mar. 2004, pp. 762–774.

[21] K. H. Wang and B. Li, "Efficient and guaranteed service coverage in partitionable mobile ad-hoc networks," in *Proc. of INFOCOM*, June 2002, pp. 1089–1098.

[22] Z. J. Haas and M. R. Pearlman, "The performance of query control schemes for the zone routing protocol," *IEEE/ACM Trans. on Networking*, vol. 9, no. 4, pp. 427–438, Aug. 2001.

[23] L. Li, J. Y. Halpern, P. Bahl, Y.-M. Wang, and R. Wattenhofer, "Analysis of a cone-based distributed topology control algorithm for wireless multi-hop networks," in *Proc. of ACM PODC*, Aug. 2001, pp. 264–273.

[24] D. E. Knuth, *The Art of Computer Programming, Volume 2 (3rd ed.): Seminumerical Algorithms*. Addison-Wesley, 1997.

[25] B. Y. Chan, A. Si, and H. V. Leong, "A framework for cache management for mobile databases: Design and evaluation," *Journal of Distributed and Parallel Databases*, vol. 10, no. 1, pp. 23–57, July 2001.

[26] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrievel*. Addison-Wesley, 1999.

[27] K. Birman, A. Schiper, and P. Stephenson, "Lightweight causal and atomic group multicast," *ACM Trans. on Computer Systems*, vol. 9, no. 3, pp. 272–314, Aug. 1991.

[28] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: A scalable wide-area web cache sharing protocol," *IEEE/ACM Trans. on Networking*, vol. 8, no. 3, pp. 281–293, June 2000.

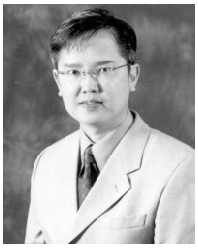
[29] L. M. Feeney and M. Nilsson, "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment," in *Proc. of INFOCOM*, Apr. 2001, pp. 1548–1557.

[30] X. Hong, M. Gerla, G. Pei, and C.-C. Chiang, "A group mobility model for ad hoc wireless networks," in *Proc. of International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, Aug. 1999, pp. 53–60.

[31] 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 MobiCom*, Oct. 1998, pp. 85–97.

Chi-Yin Chow received his B.A. (Hons.) and M.Phil. degrees in computer science from the Hong Kong Polytechnic University in 2002 and 2005, respectively. He is currently a Ph.D. student in the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN. His research interests include location privacy, query processing in mobile computing, cooperative caching and mobile data management. He is a student member of the ACM, ACM SIGMOD and IEEE.

Hong Va Leong received his Ph.D. from the University of California at Santa Barbara, and is currently an associate professor at the Hong Kong Polytechnic University. He is the program co-chairs of a number of conferences and has also served on the organizing committees and program committees of numerous conferences. He is a reviewer for IEEE Transactions on Parallel and Distributed Systems, on Knowledge and Data Engineering, on Computers, on Mobile Computing, ACM Transactions on Computer Systems, Information Systems, Theoretical Computer Science, and other journals. He has published over a hundred refereed research papers. His research interests are in mobile computing, internet computing, distributed systems, distributed databases, and digital libraries. He is a member of the IEEE Computer Society and Communications Society and the ACM.



Alvin T. S. Chan is currently an associate professor at the Hong Kong Polytechnic University. He graduated from the University of New South Wales with a Ph.D. degree in 1995 and was subsequently employed as a Research Scientist by the CSIRO, Australia. From 1997 to 1998, he was employed by the Center for Wireless Communications, National University of Singapore as a Program Manager. Dr. Chan is one of the founding members of a university spin-off company, Information Access Technology Limited. He is an active consultant and has been providing consultancy services to both local and overseas companies. His research interests include mobile computing, context-aware computing and smart card applications. He is a member of the IEEE and ACM.