Distributed Group-based Cooperative Caching in a Mobile Broadcast Environment[*]

Chi-Yin Chow
Department of Computing
The Hong Kong Polytechnic University
Hong Kong
cscychow@comp.polyu.edu.hk

Hong Va Leong
Department of Computing
The Hong Kong Polytechnic University
Hong Kong
cshleong@comp.polyu.edu.hk

Alvin T. S. Chan
Department of Computing
The Hong Kong Polytechnic University
Hong Kong
cstschan@comp.polyu.edu.hk

ABSTRACT

Caching is a key technique for improving data retrieval performance of mobile clients. The emergence of state-of-the-art peer-to-peer communication technologies now brings to reality what we call "cooperative caching" in which mobile clients not only can retrieve data items from mobile support stations, but also from the cache in their peers, thereby inducing a new dimension for mobile data caching. In this paper, we propose a distributed group-based cooperative caching scheme, in which we define the concept of a tightly-coupled group (TCG) by capturing the data affinity of individual peers and their mobility patterns, in a mobile broadcast environment. A distributed stable peer discovery protocol is proposed for discovering all TCGs dynamically. In addition, a cache signature scheme is adopted to provide hints for the mobile clients to determine whether their required data items are cached by their neighboring peers, and to perform cooperative cache replacement to increase overall data availability. Simulation studies are conducted to evaluate the effectiveness of our distributed group-based cooperative caching scheme.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*distributed systems*; H.2.4 [**Database Management**]: Systems—*distributed databases*

General Terms

Algorithms, Design, Performance

Keywords

Mobile computing, mobile data broadcast, peer-to-peer computing, peer groups, cooperative caching

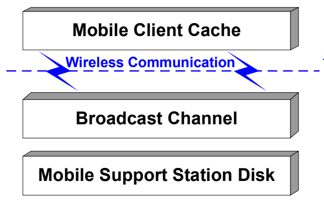
[*]This research is supported in part by the Research Grant Council and the Hong Kong Polytechnic University under grant numbers PolyU 5084/01E and H-ZJ86.

1. INTRODUCTION

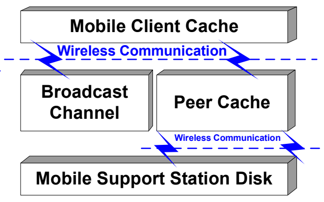
A mobile system is formed with a wireless network connecting mobile hosts (MHs) and mobile support stations (MSSs), in which MHs are clients equipped with portable devices and MSSs are stationary servers providing information access for the MHs residing in their predefined service areas. This kind of network architecture, with the presence of MSSs, is known as an *infrastructure network*. It is most commonly deployed in real life. In the infrastructure network, the MHs can retrieve data items from the MSSs, either by requesting them over shared point-to-point channels, i.e., pull-based data dissemination model, or catching them from scalable broadcast channels, i.e., push-based data dissemination model, or through the utilization of both types of channels, i.e., hybrid data dissemination model. In this work, we consider a push-based data dissemination model, in which if an MH cannot find the required data item in its cache, i.e., a cache miss, it obtains the data item from the broadcast channel.

The push-based mobile data dissemination model is also known as a data broadcast model, in which the MSS periodically broadcasts data items to the MHs via a broadcast channel. When the MHs encounter cache misses, they have to listen to the broadcast channel and "catch" the required data items. The strength of the mobile broadcast model is its scalability to an immense number of MHs. However, its major weakness lies in the sequential data item broadcast; the access latency gets longer with increasing number of data items being broadcast. The MHs also have to consume substantial power to listen to the broadcast channel until the required data items appear in the channel. Many researchers adopt different techniques to overcome the downside of mobile broadcast model. Hybrid data dissemination models [15, 22] are proposed to make use of both pull- and push-based models at the same time; indexing techniques are proposed to shorten the tune-in time and power consumption [2, 16, 20]; and semantic-based techniques [19] are adopted to improve access latency and tuning time. In this paper, we propose a distributed group-based cooperative caching scheme to improve access latency and reduce power consumption.

COCA [6] is a mobile **CO**operative **CA**ching framework that combines the peer-to-peer communication technology (known as P2P in this paper) with a conventional mobile system. In a traditional push-based mobile system, the storage hierarchy commonly consists of three layers: Mo-



(a) Conventional



(b) COCA

Figure 1: Storage hierarchy of push-based mobile systems.

bile Client Cache, Broadcast Channel and MSS Disk, as shown in Figure 1(a). The MSS grabs the data items from the disk or database, and allocates them to the broadcast channel. If an MH encounters a local cache miss, it tunes in to the broadcast channel, and catches the required data item when the data item appears in the broadcast channel. In COCA, we insert a logical layer, called Peer Cache layer, as a supplementary component of the Broadcast Channel layer, as shown in Figure 1(b). When an MH suffers from a local cache miss, the MH tunes in to the broadcast channel; meanwhile, it searches for the required data item in the Peer Cache layer. There is a global cache hit, if some peers turn in the required data item to the MH before either the data item appears on the broadcast channel or the timeout period elapses. In case of no peer returning the required data item, i.e., a global cache miss, the MH has to wait for the data item to appear in the broadcast channel, as in the conventional case.

This work extends COCA with a distributed group-based cooperative caching scheme, namely, DGCoca, to reduce access latency and power consumption in a mobile broadcast environment. DGCoca groups the MHs into tightly-coupled groups (TCGs) based on their mobility and data access patterns, and allows them to manage their cached data items *cooperatively* with other members in the same TCG to improve data availability. In addition, a cache signature scheme is adopted to provide hints for the MHs to make local decision on whether to search the cache of their peers and to provide information for the MHs to perform cooperative cache replacement in the TCG. The performance of DGCoca is evaluated through a number of simulated experiments. The result shows that DGCoca significantly improves access latency and power consumption compared with the conventional broadcast and standard COCA schemes.

The rest of this paper is organized as follows. Section 2 surveys related works of cooperative caching in mobile environments. The COCA framework is presented in Section 3. Section 4 describes the DGCoca, cache signature scheme and two cooperative cache management protocols. In Section 5, the simulation model of the standard COCA and DGCoca is defined. Their performance is evaluated in Section 6. Finally, Section 7 offers brief concluding remarks, with an outline of our future work.

2. RELATED WORK

Recently, cooperative caching schemes in mobile environments have been drawing increasing attention. In [12, 13, 18, 23, 24, 26], several cooperative caching schemes were proposed for mobile environments. Research in [24] proposed an intuitive cooperative caching architecture for MANETs. If an MH can directly connect to an infrastructure support with a single hop, it would obtain the required data items from the infrastructure support; otherwise, the MH has to enlist its peers for help to turn in the required data items. If no peer caches the data items, the peers route the request to the nearest infrastructure support. A similar cooperative caching architecture is proposed in [18] that is designed to support continuous media access in MANETs. In [18], the MHs retrieve the multimedia objects from the nearest source that can be the cache of their peers or the original servers. In [23], cooperative caching scheme is used as a complement to the infrastructure support with power conservation. When an MH fails to access the required information from the server, it would try to access the information from its peers. The power conservation scheme adjusts the MHs' degree of activity or participation in the cooperative caching scheme based on their battery level.

In [12] and [13], replication techniques are adopted in cooperative caching schemes to improve the data availability and alleviate the network partitioning problem. One of our previous work [6] also adopts replication techniques to utilize the cache space of low-activity MHs in the system. The cooperative caching scheme in [26] is more complicated than previous works. Each MH not only caches popular data items or paths on behalf of its peers, but also extracts useful information from the messages passing-by. This information can be used for routing future requests to the nearest data source. The techniques of searching for desired data items in MANETs are extensively studied in [9].

In [5], we propose a centralized group-based COCA scheme that the MSS uses an incremental clustering algorithm to discover TCGs based on the distance and data access similarity between MHs. Since it depends on precise location information of each MH, it is not always feasible in a mobile environment. In contrast, our newly proposed DGCoca *does not depend* on any positioning system or device, e.g., the global positioning system (GPS) or sensor-based local positioning systems, to capture the mobility pattern of itself or those of other MHs.

This work is different from previous works in that we propose a distributed algorithm to discover TCGs dynamically, and employ the cache signature scheme to provide hints for the MHs to determine whether to bypass the search in their peers' cache, and to provide local information for the tightly-coupled MHs to perform cooperative cache replacement in a distributed manner.

3. COCA

In this work, we assume that each MH, m_i, has a unique identifier, i.e., $\mathcal{M} = \{m_1, m_2, ..., m_{NumClient}\}$, where *NumClient* is the total number of MHs in the system. Each MH is equipped with two wireless network interface cards (WNICs); one of them is dedicated to communicate with MSS, while the other one is devoted to the communication with other peers. For simplicity, we further assume that there is no update of data items.

COCA is based on the system architecture depicted in Figure 2. If an MH, m_2, finds the required data item in its local cache, it constitutes a *local cache hit*; otherwise, it is a *local cache miss.* When m_2 encounters a local cache miss, it tunes in to the broadcast channel, and sends a request to its neighboring peers, m_1 and m_3. If one of them can turn

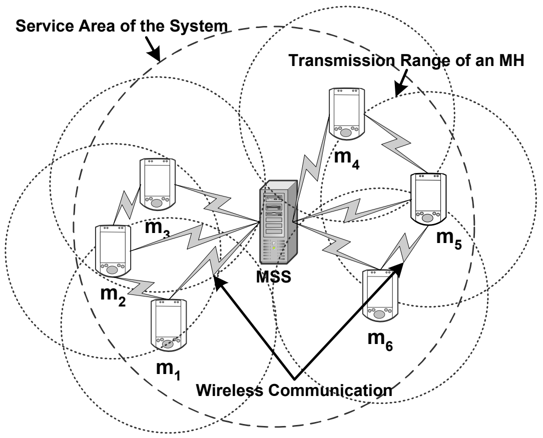


Figure 2: The COCA system architecture.

in the required data item before it appears in the broadcast channel, a *global cache hit* is resulted; otherwise, it is a *global cache miss*, and m_2 has to wait for the data item to appear in the broadcast channel. Based on the COCA system architecture, the COCA network model and communication protocol are described next.

3.1 Network Model

In COCA, a neighbor discovery protocol (NDP) [11, 21] is available in the service of the underlying network. NDP is a simple protocol in which the neighbor connectivity is maintained through a periodic beacon of "hello" message with useful information, e.g., an identifier of the sender. As the beacon period ($BeaconPeriod$) is short enough, NDP can precisely detect a new link to a newly admitted MH and a disconnected link to a leaving MH. In P2P model, there are two P2P communication paradigms: point-to-point and broadcast. For the former one, there is only one destination MH for the message being sent from the source MH, while in the latter, all MHs residing in the transmission range of the source MH receive the broadcast message.

3.2 Communication Protocol

For each request, an MH first finds the required data item in its local cache. If it encounters a local cache miss, it tunes in to the broadcast channel; meanwhile, it broadcasts a *request* message to its neighboring peers via P2P broadcast communication. A peer caching the required data item sends a *reply* message to the requesting MH via P2P point-to-point communication. The MH selects the first peer from those sending *reply* messages as a target peer. It next sends a *retrieve* message to the target peer via P2P point-to-point communication. Finally, the target peer turns in the required data item via P2P point-to-point communication. If the data item appears in the broadcast channel *before* any reply from peers, the MH catches it from the channel and does not send out any *retrieve* message. In case of no peer being found caching the required data item for a period of time, i.e., a timeout period, the MH needs to wait for the data item to appear in the broadcast channel.

The timeout period is set to a default value that is defined as the round-trip time of P2P communication scaled up by a congestion factor, φ, i.e., $\frac{size\ of\ request + size\ of\ reply}{BW_{\mathsf{P2P}}} \times \varphi$, where BW_{P2P} is the bandwidth of the P2P communication channel. For each search in the peers' cache, an MH records the time duration, τ, spent, i.e., the time when the MH broadcasts a *request* message to the time when a *reply* message is received. Then, the timeout period is set to the average time duration $\overline{\tau}$ plus another system parameter φ' times the standard deviation, σ_τ, of τ, i.e., timeout period $= \overline{\tau} + \varphi'\sigma_\tau$.

4. DGCOCA

In DGCoca, a *stable neighbor discovery algorithm* (SND) that is extended from NDP is proposed for an MH to discover the common mobility pattern between itself and its neighboring peers. Signature techniques are adopted to capture the data access histories for the MH to determine the data access similarity between itself and others.

4.1 Stable Neighbor Discovery Algorithm

In SND, there are four kinds of relationships between any two MHs: *unknown*, *stranger*, *friend* and *member* (in an increasing order of the closeness of relationship between two MHs). SND is a memorization-based algorithm. When an MH has consecutively received "hello" beacon from a peer for a period of time, π, the relationship between them becomes closer. On the other hand, if the MH has not obtained any "hello" beacon from the peer for the same period of time, π, their relationship will be degraded. Initially, the relationship between any pair of MHs is *unknown*.

For two MHs, m_i and m_j, with an *unknown* relationship, after m_i receives a "hello" beacon from m_j, m_j becomes a *stranger* to m_i. When m_i has consecutively received beacons from m_j for a period of time π, m_j becomes a *friend* of m_i. Then, if m_i obtains "hello" beacons from m_j consecutively for another period of time π, m_i treats m_j as a *member* in its TCG. Nevertheless, if m_i cannot receive any beacon from m_j beyond a period of π, their relationship is downgraded to *friend*. After another period of π, if m_i still cannot receive any beacon from m_j, the memory for m_j is faded away, and m_j becomes a *stranger* to m_i. m_i will next remove the relationship information about m_j, and m_j reverts to be *unknown* to m_i. Figure 3 illustrates a state transition diagram of MH relationships in SND.

SND consists of two components (Algorithms 1 and 2). Algorithm 1 is a continuous algorithm that is periodically executed to learn about any changes in the relationship between other MHs. Algorithm 2 is executed when the MH receives a "hello" beacon through NDP.

4.2 Cache Signature Scheme

There are four types of signatures, *data signature*, *cache signature*, *peer signature* and *search signature*. Each signature is stored as a bloom filter [3] with m bits, i.e., the size of a bloom filter is m. A data signature of a data item is

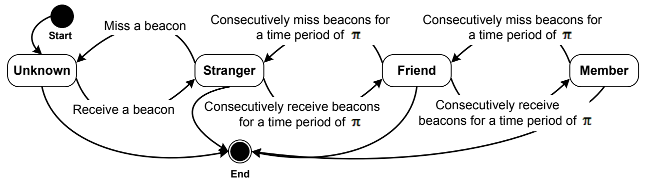

Start
Miss a beacon
Consecutively miss beacons for a time period of π
Consecutively miss beacons for a time period of π
Unknown
Stranger
Friend
Member
Receive a beacon
Consecutively receive beacons for a time period of π
Consecutively receive beacons for a time period of π
End

Figure 3: A state transition diagram of MH relationships.

Algorithm 1 Stable Neighbor Discovery Algorithm at m_i

```
1: DiscoverStableNeighbors(Relation $\mathcal{R}_i$,
       CacheSignature $\mathcal{S}_i$)
   // $rel(i, j)$ maintains the current relationship between
       $m_i$ and $m_j$
   // $\mathcal{R}_i$ contains only relationships with $m_i$,
       i.e., $\forall k \neq i, rel(k, j) \notin \mathcal{R}_i$
   // furthermore, $rel(i, i) \notin \mathcal{R}_i$
2: for all $rel(i, j) \in \mathcal{R}_i$ do
3:   // now() returns the current time
4:   if now()$-last\_beacon\_ts_j > BeaconPeriod$ then
5:     if $rel(i, j) =$ stranger then
6:       $rel(i, j) \leftarrow$ unknown;
7:       $\mathcal{R}_i \leftarrow \mathcal{R}_i - \{rel(i, j)\}$;
8:       next $m_j$;
9:     end if
10:    if not $disappeared_j$ then
11:      $last\_disappeared\_ts_j \leftarrow$ now();
12:      $disappeared_j \leftarrow$ true;
13:    end if
14:  end if
15:  if $disappeared_j$ then
16:    if now()$-last\_disappeared\_ts_j \geq \pi$ then
17:      if $rel(i, j) =$ member then
18:        $rel(i, j) \leftarrow$ friend;
19:      else if $rel(i, j) =$ friend then
20:        $rel(i, j) \leftarrow$ stranger;
21:        // remove $m_j$'s cache signature
22:        if $S_j \in \mathcal{S}_i$ then
23:          $\mathcal{S}_i \leftarrow \mathcal{S}_i - \{S_j\}$;
24:        end if
25:      end if
26:      $last\_disappeared\_ts_j \leftarrow$ now();
27:    end if
28:  else
29:    if now()$-first\_consecutive\_beacon\_ts_j \geq \pi$ then
30:      if $rel(i, j) =$ stranger then
31:        $rel(i, j) \leftarrow$ friend;
32:        $first\_consecutive\_beacon\_ts_j \leftarrow$ now();
33:      else if $rel(i, j) =$ friend then
34:        if sim($m_i$, $m_j$) $\geq \delta$ then
35:          // sim() will be defined in Section 4.2.4
36:          $rel(i, j) \leftarrow$ member;
37:          // collect $m_j$'s cache signature
38:          if $S_j \notin \mathcal{S}_i$ then
39:            $\mathcal{S}_i \leftarrow \mathcal{S}_i \cup \{S_j\}$;
40:          end if
41:        end if
42:      end if
43:    end if
44:  end if
45: end for
```

Algorithm 2 Stable Neighbor Discovery Algorithm at m_i

```
1: OnReceiveBeacon(Relation $\mathcal{R}_i$, MH $m_j$)
   // $m_i$ receives a "hello" beacon message from $m_j$
2: $last\_beacon\_ts_j \leftarrow$ now();
3: if $rel(i, j) \notin \mathcal{R}_i$ then
4:   $rel(i, j) \leftarrow$ stranger;
5:   $\mathcal{R}_i \leftarrow \mathcal{R}_i \cup \{rel(i, j)\}$;
6:   $first\_consecutive\_beacon\_ts_j \leftarrow$ now();
7: else if $disappeared_j$ then
8:   $disappeared_j \leftarrow$ false;
9:   $first\_consecutive\_beacon\_ts_j \leftarrow$ now();
10: end if
```

a bit string that is produced by hashing its unique identifier, keywords, attribute values, URL, the document content or other identifiers with single or multiple independent hash functions (we use k hash functions). A cache signature summarizes the cache content by superimposing all data signatures of the data items in a cache. For a peer signature, it is produced by superimposing the cache signatures from the peers to provide hints for the MH to determine whether to search the peers' cache. The generation of a data signature and search signature is the same. A data signature is for a cached data item, while a search signature is for a required data item that cannot be found in the local cache. The search signature is used for comparing with the peer signature by a bitwise AND operation, $\otimes$. If the result is the same as the search signature, it indicates that some of the MH's peers are likely caching the required data item, so the MH will search the peers' cache. Otherwise, the MH will not search the peers' cache and will instead obtain the data item from the broadcast channel directly.

To reduce transmission overheads, i.e., power consumption and transmission time, the cache signature is compressed before transmitting to other peers. A variable-length-to-fixed-length (VLFL) run-length encoding [17] 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 (L) which are terminated by two cases:

1. There are R consecutive "zeros", where $R = 2^n - 1$ for any positive integer n.
2. There are z consecutive "zeros" followed by a single "one", where $0 \leq z < R$.

Second, a fixed-length codeword, which indicates the size of the run-length, is assigned to each run-length.

The generation mechanism of a cache signature is similar to that of a summary cache [7]. The identifier of a data item is first hashed by a 128-bit MD5 hash function. The 128-bit hash value is then divided into four 32-bit words. A modulus function is next applied on each 32-bit word to yield a hash value bounded by m (the bloom filter size), i.e., between zero and $m-1$. The bits at the position of the four hash values are set to one, in case of $k = 4$. This process is repeated for each data item stored in the local cache. Then the cache signature is compressed by the VLFL run-length encoding. If $k < 4$, only the bits at the position of the first k hash values are set to one. If $k > 4$, other additional hash functions, such as cyclic redundancy check (CRC32) [10], could be used.

4.2.1 *Cache Signatures with Proactive Generation*

Since the cache content is changed frequently, i.e., every cache insertion or eviction changes the cache signature, the MH has to re-generate the cache signature before sending it to other peers. To reduce the overhead of re-generating cache signatures, we extend the counter vector [7] with *dynamic counter size* to maintain the information of a cache signature. An MH maintains a vector of m counters, each counter with ξ bits. Initially, $\xi = 1$, and all counters are set to zero. When a data item is inserted into the local cache, a data signature is generated for the data item. If k independent hash functions are used to construct the bloom filter, k bits are set in the data signature. The counters at the position of the bits that are set in the data signature are incremented. Likewise, when a data item is removed from the cache, a data signature is also generated for the data item. The corresponding k counters are decremented. By using this proactive signature generation approach, the MH can construct a new cache signature by simply setting the bits corresponding to the position of the counters with non-zero values. When an MH increments a counter to a value of 2^{ξ}, the size of all counters will be increased by one to deal with counter overflow. Likewise, in case that all counter values fall below $2^{\xi-1}$, the counter size ξ will be decremented to conserve cache space.

4.2.2 Peer Signatures Structure

Similar to the proactive generation approach, each MH maintains another dynamic counter vector structure for storing its peer signatures, in a parallel manner to the cache signatures. The counter size ξ is also dynamically adjusted to avoid overflowing.

4.2.3 Cache Signature Exchange Protocol

When an MH has detected a new member in its TCG, i.e., an event of the relationship of the MH to a peer changing from *friend* to *member*, it sends a signature request message to the new member. Then, the new member turns in its full cache signature to the MH. After that, the member embeds the signature update information to the request message that is broadcast to all its neighboring peers. Thus, members in its TCG can utilize such information to update their peer-signature counter vectors. An MH that has not generated any request to its neighboring peers for a period of time μ is required to broadcast the changes in its cache signature to its peers, unless there is no change in the cache signature. The signature update information is maintained in two lists, one list storing the bit position that has been reset, and another list the bit position that has been set since the last time the MH broadcasts a request message. If a bit position exists in both lists, the change is annihilated.

If the MH detects that the relationship of itself to its peer changes from *member* to *stranger*, the MH resets the peer-signature counter vector, and sends a request to all of its remaining members to turn in their full cache signatures via P2P point-to-point communication. The MH uses the received cache signatures to reconstruct the counter vector.

4.2.4 Signature for Data Access Pattern

In DGCoca, an *access history signature* is used to store the access history of an MH. When the MH accesses a data item, it generates a data signature for it. Then, the access history signature is updated by performing a bitwise OR operation, $\oplus$, on the data signature with the last access history signature, i.e., (access history signature)^{new} = (data signature) $\oplus$ (access history signature)^{old}. The data access pattern introduces an additional condition to the SND algorithm. When an MH, m_i, is going to upgrade a relationship with a peer, m_j, from *friend* to *member*, it requests m_j to turn in its access history signature. m_i then calculates a score on the data access similarity as the Jaccard Coefficient, i.e., $\mathsf{sim}(A_i, A_j) = \frac{A_i \cap A_j}{A_i \cup A_j}$ (note that $0 \leq \mathsf{sim}(A_i, A_j) \leq 1$), where A_i and A_j are the access history signatures of m_i and m_j respectively. If $\mathsf{sim}(A_i, A_j)$ is equal to or larger than a similarity threshold δ, the peer becomes the MH's *member*; otherwise, the peer only remains to be its *friend*, because they do not share sufficient common interest. In DGCoca, a small value is set for δ because we would like to shorten the evaluation period and obtain a higher global cache hit ratio, screening out only those peers whose access patterns are highly different.

To reduce communication overheads, a similarity score will be maintained for a period of time TTL. The calculated scores will be reused before they expire, and all expired scores are removed.

4.3 Cooperative Cache Management Protocols

Two cooperative cache management protocols: *cooperative cache admission control* and *cooperative cache replacement*, are proposed for the TCG MHs to work together to manage their cache space as a *global cache* or *aggregate cache*.

4.3.1 Cooperative Cache Admission Control

When an MH encounters a local cache miss, it sends a request to its peers. If some peers turn in the required data item to the MH and the local cache is not full, the MH caches the data 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 data item when it is supplied by a peer in the same TCG, on the belief that that data 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 data item in its local cache by removing the least valuable cached item, since the providing peer may move far away in future. After a peer sends the required data item to the requesting MH and if they belong to the same TCG, the peer updates the last access timestamp of the data item, so that the item can have a longer *time-to-live* in the global cache. If there are more than one TCG member caching the required data item, the MH randomly selects one of them to update the last access timestamp of the data item.

4.3.2 Cooperative Cache Replacement

This protocol allows the MH to replace the least valuable data item with respect to the MH itself and other members in the same TCG. This scheme satisfies three useful properties. First, the most valuable data items are always retained in the local cache. Second, in a local cache, a data item which has not been accessed for a long period will be replaced eventually. Third, in a global cache, a data item which “spawns” replica is first replaced in order to increase the effective cache size.

The LRU (least recently used) cache replacement policy is used. To retain valuable data items in the local cache, only a number of $NumReplaceCandidate$ least valuable data items are selected as the candidates for replacement. Among the candidates, the least valuable data item is first chosen, and the MH generates a data signature for that data item. The data signature is then compared with the peer signature by a bitwise AND operation, $\otimes$. If the result is the same as the data signature, the data item is likely a replica in the global cache, so it is removed from the cache. Otherwise, the second least valuable data item is examined, and so on.

If no candidate is probably replicated in the TCG, the least valuable data item is removed from the cache.

There could be a drawback with the above arrangement: a data item without any replica could always be retained in the local cache, even though it will not be accessed again. If most data items cached in the local cache of an MH belong to such type of data item, the MH's cache will be populated with useless data items. As a result, the cache hit ratio will be degraded. To solve this problem, a *time-to-live* counter is associated with each candidate. The counter value is initially set to *ReplaceDelay*. When the least valuable data item is not replaced because it does not have any replica, its counter is decremented. If the counter of the least valuable data item is equal to zero, the cooperative cache replacement protocol is skipped, and the MH simply drops that data item from the cache. The counter is reset to *ReplaceDelay*, when the data item is accessed by the MH itself or other members in its TCG.

4.4 Handling Mobile Client Disconnection

To deal with the possibility of mobile client disconnection, an MH will collect and store the relationships between its peers when it is disconnected from the network voluntarily. Upon its reconnection to the network, it restores the saved relationships, and then broadcasts a request containing its peers' identities and their corresponding relationships. The peers receiving the request also restore the relationship, which is gradually downgraded to *unknown* after prolonged absence of beacons. However, other disconnected peers departing the network after the MH reconnecting cannot receive this request and may lose the relationship; they have to rebuild their relationship from scratch upon their reconnection.

In DGCoca, the MHs check for the validity of their cached cache signatures with timestamping mechanism after they reconnect to the network. When an MH, m_i, broadcasts a request with signature update information to its peers, it records the time, *last_signature_update_ts*. In addition, when m_i disconnects from the network, it records the time, *disconnection_ts*. After m_i reconnects to the network, *disconnection_ts* will be included in the broadcast request. The group members of m_i reply to m_i with a message indicating whether its peer signature is valid ($last\_signature\_update\_ts < disconnection\_ts$) or obsolete ($last\_signature\_update\_ts \geq disconnection\_ts$). If the MH stores any stale peer signatures, it resets the peer-signature counter vector and broadcasts a request to its members. The peer receiving the request returns its full cache signature to the MH. The MH then reconstructs the counter vector based on the collected cache signatures.

5. SIMULATION MODEL

The simulation model is implemented in C++ using CSIM 19 [25]. The simulated mobile environment is composed of an MSS and *NumClient* MHs. The MHs move in a 1000 m × 1000 m (*Area*) space, which constitutes the service area of the MSS. The total bandwidth of the broadcast channel is 10 Mbps. There is also a wireless communication channel for P2P communication, with a bandwidth of 2 Mbps and a transmission range of *TranRange*. When an MH sends a message to another peer, it has to wait if its required channel is busy.

5.1 Power Consumption Model

All MHs are assumed to be equipped with the same type of WNICs for P2P communication with an omnidirectional antenna. The communication power consumption measurement is based on [8], which uses 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, the destination MH, D_R, and both the source and destination MHs, as shown in Figure 4(a). The P2P point-to-point power consumption is measured by Equation 1.

$$P_{p2p} = \begin{cases} (v_{send} \times |msg|) + f_{send}, & \text{for } m = S \\ (v_{recv} \times |msg|) + f_{recv}, & \text{for } m = D \\ (v_{sd\_disc} \times |msg|) + f_{sd\_disc}, & \text{for } m \in S_R \wedge m \in D_R \\ (v_{s\_disc} \times |msg|) + f_{s\_disc}, & \text{for } m \in S_R \wedge m \notin D_R \\ (v_{d\_disc} \times |msg|) + f_{d\_disc}, & \text{for } m \notin S_R \wedge m \in D_R \end{cases} \tag{1}$$

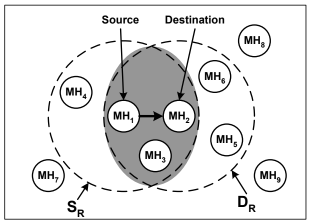
where f is the fixed setup cost for message transmission, and v is the variable power consumption based on the size of a message msg in byte ($|msg|$).

Power consumption of source MH, S, and other MHs residing in S_R of a broadcast communication, as depicted in Figure 4(b), can be measured by Equation 2.

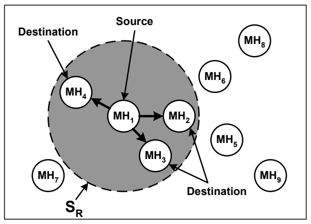
$$P_{bc} = \begin{cases} (v_{bsend} \times |msg|) + f_{bsend}, & \text{for } m = S \\ (v_{brecv} \times |msg|) + f_{brecv}, & \text{for } m \in S_R \end{cases} \tag{2}$$

Although the power consumption model in [8] is based on an ad hoc network mode, it is also used to approximate the power consumption in the communication between the MH and the MSS. When an MH communicates with the MSS, the surrounding MHs are not affected by the transmission so that no power is consumed by their WNICs. Tables 1 and 2 show the parameter settings for MHs playing different roles, as used in power consumption measurement for P2P point-to-point and broadcast communication respectively [8].

To reduce power consumption in the mobile broadcast environment, the MSS periodically broadcasts the index of the data items in the broadcast cycle to the MHs, so that the MHs are able to determine when the required data items will appear in the broadcast channel. The index contains the identifier of each data item being broadcast and its broadcast latency, which is defined by subtracting the current broadcast slot from the slot containing the data item. The index is broadcast to the MHs every $IndexInterval$. When the MHs encounter a local cache miss, they have to listen



(a) P2P point-to-point communication



(b) P2P broadcast communication

Figure 4: Power consumption model.

Table 1: Parameter settings for power consumption model in P2P point-to-point communication.

Conditions	$\mu W \cdot s/byte$	$\mu W \cdot s$
$m = S$	$v_{send} = 1.9$	$f_{send} = 454$
$m = D$	$v_{recv} = 0.5$	$f_{recv} = 356$
$m \in S_R \wedge m \in D_R$	$v_{sd\_disc} = 0$	$f_{sd\_disc} = 70$
$m \in S_R \wedge m \notin D_R$	$v_{s\_disc} = 0$	$f_{s\_disc} = 24$
$m \notin S_R \wedge m \in D_R$	$v_{d\_disc} = 0$	$f_{d\_disc} = 56$

Table 2: Parameter settings for power consumption model in P2P broadcast communication.

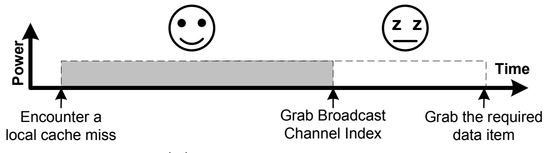
Conditions	$\mu W \cdot s/byte$	$\mu W \cdot s$
$m = S$	$v_{bsend} = 1.9$	$f_{bsend} = 266$
$m \in S_R$	$v_{brecv} = 0.5$	$f_{brecv} = 56$

to the broadcast channel until they obtain the required data items or the index. In the latter case, the MHs can determine when the required data items appear in the broadcast channel by looking up the index. The MHs can keep themselves in a doze mode for a certain period. Then, they wake up and catch the required data items, as shown in Figure 5(a). In COCA, there is a chance for the MHs to save power, when some of their peers can turn in the required data items before the data items or the index appear in the broadcast channel, as depicted in Figure 5(b). Figure 5(c) shows the situation when the MHs encounter global cache misses.

5.2 Client Model

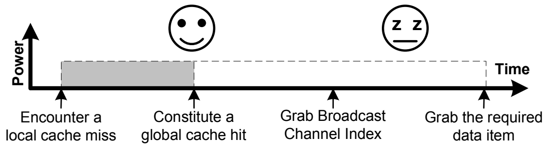
The MHs are divided into several motion groups, and each group consists of *GroupSize* MHs. The mobility of the MHs is based on the "reference point group mobility" model [14] which is a random mobility model for a group of MHs and for an individual MH within its group. Each group has a logical center which represents a motion reference point for the MHs belonging to the group. The movement of each motion group is based on the "random waypoint" mobility model [4], with a uniformly determined moving speed from v_{min} to v_{max}, and one-second pause time.

The MHs generate accesses to the data items following a Zipf [27] distribution with a skewness parameter θ. If θ is set to zero, MHs access the data items uniformly. By increasing θ, we are able to model a more skewed access pattern to the data items. The time interval between two consecutive accesses generated by an MH follows an exponential distribution with a mean of one second. The MHs in the same motion group share a certain percentage of common range of hot spot (*CommonHotSpot*) in their individual access range *AccessRange*, and the remaining access ranges are randomly selected for each MH. The access range of each motion group is randomly selected. After an MH finishes with a request, there is a probability that it will become disconnected from the network for a certain period. The MHs cannot communicate with disconnected peers. This disconnection is modeled by client disconnection probability. By default, this disconnection probability is zero. The disconnection period is 10 seconds. To produce a fair comparison, signature storage also consumes client cache space in our model.



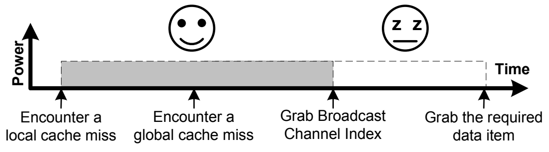
Power
Time
Encounter a local cache miss
Grab Broadcast Channel Index
Grab the required data item

(a) Conventional model



Power
Time
Encounter a local cache miss
Constitute a global cache hit
Grab Broadcast Channel Index
Grab the required data item

(b) COCA with a global cache hit



Power
Time
Encounter a local cache miss
Encounter a global cache miss
Grab Broadcast Channel Index
Grab the required data item

(c) COCA with a global cache miss

Figure 5: Power consumption model in a broadcast environment.

Table 3: Default parameter settings.

Parameters	Values
NumClient	100
GroupSize	10
DataSize	8 kbytes
AccessRange	1000 data items
CommonHotSpot	100%
CacheSize	100 data items
Speed ($v_{min} \sim v_{max}$)	$1 \sim 5$ m/s
NumReplaceCandidate	20 data items
ReplaceDelay	2
IndexInterval	300 data items
TranRange	100 m
BeaconPeriod	1 s
θ	0.5
R	511
m, k	40000, 2
μ	10 s
φ, φ'	10, 3
π	10 s
δ	0.05
TTL	30 s

5.3 Server Model

There is a single MSS in the simulated environment. A database connected to the MSS contains 10000 equal-sized data items of size *DataSize*. The MSS adopts a broadcast disk scheme [1] with three disks; their relative frequencies are 4:2:1 and their sizes are 1000, 3000 and 6000 respectively. Table 3 summarizes the default parameter settings used in the simulated experiments.

6. SIMULATION RESULTS

In the simulated experiments, we compare the performance of DGCoca (denoted as DGCoca) with a conventional caching scheme that does not involve any cooperation among MHs (denoted as non-COCA or NC) and with standard

COCA (denoted as COCA) along different dimensions. The non-COCA scheme uses LRU cache replacement policy. All simulation results are recorded only after the system reaches a stable state, in which all client caches are fully occupied, in order to avoid a transient effect. A simulated experiment ends when each MH generates about 10000 requests. The performance metrics include the access latency and power consumption on communication. The access latency is defined as the sum of transmission time, and the time spent on waiting for required communication channels, if they are busy.

6.1 Effect of Cache Size

Our first experiment studies the effect of cache size on system performance by varying the cache size from 50 to 250 data items.

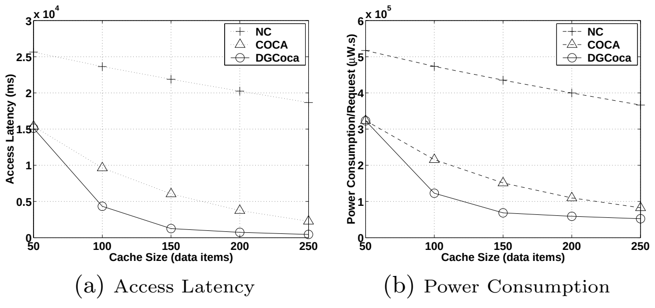


Figure 6: Effect of cache size.

Figure 6 shows that all schemes exhibit better performance with increasing cache size because the MHs experience a higher local cache hit ratio as the cache size gets larger. COCA and DGCoca outperform NC. For the MHs adopting COCA schemes, other than achieving a higher local cache hit ratio with increasing cache size, they also enjoy a higher global cache hit ratio, because the chance that some peers can return the required data items increases with a larger cache size (cache hit ratio figures omitted due to space limitation). When the MHs record a higher global cache hit ratio, they achieve better performance, due to the fact that a broadcast channel access incurs longer access latency and higher power consumption than a global cache access. As DGCoca can improve the effective cache size in TCGs, it thus improves the global cache hit ratio, leading to better performance than COCA.

6.2 Effect of Data Access Pattern

The effect of the percentage of common hot spot of each motion group on system performance is studied by varying the percentage from 0 to 100 percent.

Figure 7 shows that the performance of COCA schemes improves with increasing percentage of common hot spot. This is because there is a higher chance for the MHs to obtain the required data items from their peers when they share a higher similarity in hot spot. DGCoca outperforms NC. It also performs better than COCA after the common hot spot is over 20 percent because it effectively improves the data availability in TCGs.

We next study the effect of skewness in access pattern by varying the skewness parameter value from zero to one.

Figure 8 shows that system performance of all schemes

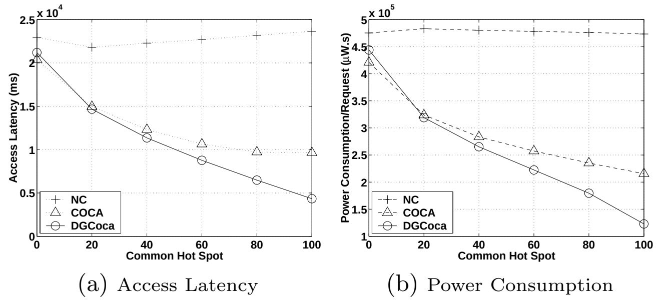


Figure 7: Effect of percentage of common hot spot.

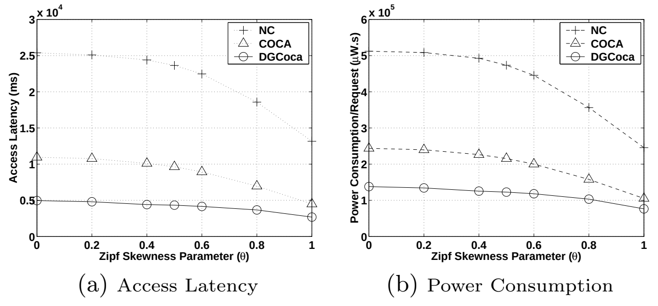


Figure 8: Effect of skewness in access pattern.

improves with increasing skewness in access pattern. When the skewness is zero, the MHs access the data items uniformly. Thus, there is a higher chance for them to encounter local cache misses. When the data access pattern becomes more skewed, the MHs witness a higher local cache hit ratio, leading to improvement of system performance. For the MHs adopting COCA schemes, when the skewness parameter value increases, they not only enjoy a higher local cache hit ratio, but also achieve a higher global cache hit ratio, as there is a higher chance for them to obtain the required data items from the peers of their own motion group. In this experiment, DGCoca always yields the best performance.

6.3 Effect of Data Item Size

The effect of data item size on system performance is studied by varying the item size from 1 to 8 kbytes.

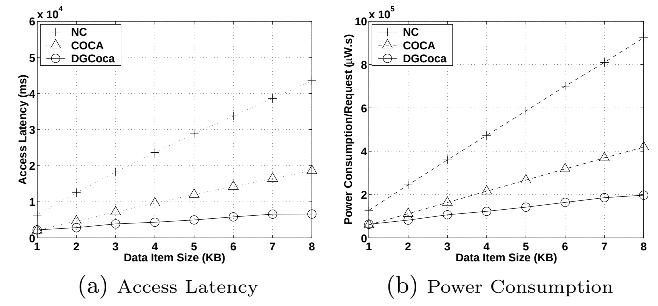
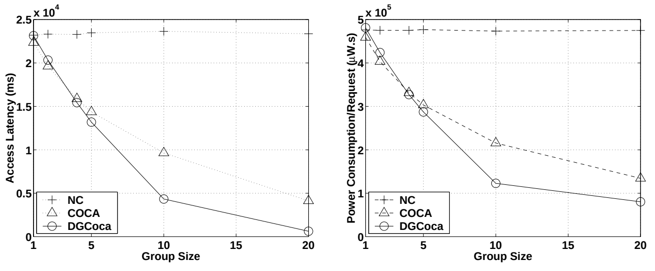


Figure 9: Effect of data item size.

As shown in Figure 9, the access latency and power consumption of all schemes increase as the data item size gets larger. When the data item size increases, the MHs have to wait longer to catch the required data items from the broadcast channel, because the broadcast cycle gets longer, as depicted in Figure 9(a). They also need to spend more power on waiting for the required data items or the broadcast channel index, and obtain the required items from the broadcast channel. For the MHs adopting COCA schemes, they consume more power not only to catch their required data items from the broadcast channel, but also to turn in or receive data items to or from other peers. As a result, the power consumption gets higher with increasing data item size, as shown in Figure 9(b). The result also shows that DGCoca can still effectively improve system performance with large data item size. This is because the storage overheads of cache signatures and access history signatures become insignificant, as the data item size gets larger.

6.4 Effect of Group Size

In this series of simulated experiments, we examine the influence on system performance with various motion group size (1, 2, 4, 5, 10 and 20).



(a) Access Latency (b) Power Consumption

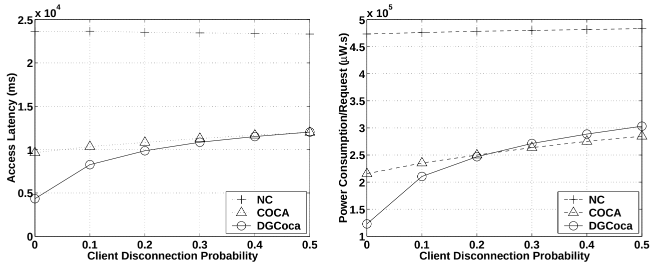
Figure 10: Effect of group size.

From Figure 10, it can be observed that the performance of COCA schemes improves with increasing group size. The MHs adopting COCA schemes can achieve a higher global cache hit ratio because they could enlist more peers for help with larger group size when they encounter local cache misses. When the MHs record a higher global cache hit ratio, they get better performance, due to the fact that a global cache access incurs shorter access latency and lower power consumption than a broadcast channel access. Furthermore, DGCoca performs better than COCA when the group size is beyond four.

6.5 Effect of Client Disconnection Probability

We finally study the effect of client disconnection probability on system performance by varying the disconnection probability from 0 to 0.5.

As depicted in Figure 11, NC is not affected by varying the disconnection probability, as there is no cooperation among peers. However, the performance of COCA schemes is more sensitive to the disconnection probability, degrading when the latter increases, due to fewer peers available in helping the requesting MHs. Thus, the MHs suffer from a higher global cache miss. When the disconnection probability is high, the MHs adopting DGCoca have to spend more power than COCA, in order to rebuild the relationships between themselves and their members, and to reconstruct the counter vector of storing peer signatures, as exhibited in



(a) Access Latency (b) Power Consumption

Figure 11: Effect of client disconnection probability.

Figure 11(b).

7. CONCLUSION

In this paper, we have described a mobile cooperative caching scheme, called COCA, in a mobile broadcast environment, and proposed a distributed group-based extension, DGCoca, with a cache signature scheme. Groups of MHs sharing common mobility and data access patterns are dynamically formed, each known as a tightly-coupled group (TCG). In a TCG, the MHs manage their cached data items cooperatively as a *global cache* or *aggregate cache*, in order to improve overall data availability. In DGCoca, a stable neighbor discovery algorithm is proposed to allow the MHs to form their own TCGs in a distributed manner. The cache signature scheme is adopted to provide hints for the MHs to determine whether the required data items are cached by their neighboring peers, and information for the MHs to perform cooperative cache replacement in their own TCGs. The performance of the DGCoca scheme is evaluated through a number of simulated experiments. The result indicates that DGCoca outperforms the standard COCA and conventional caching schemes in terms of access latency and power consumption.

In the future, we would like to explore on the semantics of data items cached by the MH groups to further improve the efficiency of caching and to reduce the overhead of maintaining and identifying the cache items. Furthermore, semantic caching is highly beneficial in a mobile environment [19], by enabling queries to be self-answerable, even with limited support from peers and in the absence of server connectivity, through the association of descriptive semantics to cached data.

8. REFERENCES

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. Broadcast disks: Data management for asymmetric communication environments. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 199–210, May 1995.

[2] S. Acharya, M. Franklin, and S. Zdonik. Balancing push and pull for data broadcast. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 183–194, May 1997.

[3] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM (CACM)*, 13(7):422–426, July 1970.

[4] 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 *Proceedings of the 4th Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 85–97, October 1998.

[5] C.-Y. Chow, H. V. Leong, and A. T. S. Chan. Group-based cooperative cache management for mobile clients in a mobile environment. In *Proceedings of the 33rd International Conference on Parallel Processing (ICPP)*, pages 83–90, August 2004.

[6] C.-Y. Chow, H. V. Leong, and A. T. S. Chan. Utilizing the cache space of low-activity clients in a mobile cooperative caching environment. *International Journal of Wireless and Mobile Computing (IJWMC)*, to appear.

[7] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary cache: A scalable wide-area web cache sharing protocol. *IEEE/ACM Transactions on Networking (TON)*, 8(3):281–293, June 2000.

[8] L. M. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1548–1557, April 2001.

[9] R. Friedman, M. Gradinariu, and G. Simon. Locating cache proxies in MANETs. In *Proceedings of the 5th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 175–186, May 2004.

[10] Z. Genova and K. Christensen. Using signatures to improve URL routing. In *Proceedings of the 21st IEEE International Performance Computing and Communications Conference (IPCCC)*, pages 45–52, April 2002.

[11] Z. J. Haas and M. R. Pearlman. The performance of query control schemes for the zone routing protocol. *IEEE/ACM Transactions on Networking (TON)*, 9(4):427–438, 2001.

[12] T. Hara. Effective replica allocation in ad hoc networks for improving data accessibility. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1568–1576, April 2001.

[13] T. Hara. Cooperative caching by mobile clients in push-based information systems. In *Proceedings of the 11th International Conference on Information and Knowledge Management (CIKM)*, pages 186–193, November 2002.

[14] X. Hong, M. Gerla, G. Pei, and C.-C. Chiang. A group mobility model for ad hoc wireless networks. In *Proceedings of the 2nd International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pages 53–60, August 1999.

[15] Q. Hu, D. L. Lee, and W.-C. Lee. Performance evaluation of a wireless hierarchical data dissemination system. In *Proceedings of the 5th International Conference Mobile Computing and Networking (MobiCom)*, pages 163–173, August 1999.

[16] Q. Hu, W.-C. Lee, and D. L. Lee. A hybrid index technique for power efficient data broadcast. *Distributed and Parallel Databases*, 9(2):151–177, March 2001.

[17] H. Kobayashi and L. R. Bahl. Image data compression by predictive coding I: Prediction algorithms. *IBM Journal of Research and Development*, 18(2):172–179, March 1974.

[18] W. H. O. Lau, M. Kumar, and S. Venkatesh. A cooperative cache architecture in support of caching multimedia objects in MANETs. In *Proceedings of the 5th ACM International Workshop on Wireless Mobile Multimedia (WoWMoM), in conjunction with the 8th International Conference on Mobile Computing and Networking (MobiCom)*, pages 56–63, September 2002.

[19] K. C. K. Lee, H. V. Leong, and A. Si. Semantic data broadcast for a mobile environment based on dynamic and adaptive chunking. *IEEE Transactions on Computers (TOC)*, 51(10):1253–1268, October 2002.

[20] W.-C. Lee and D. L. Lee. Signature caching techniques for information filtering in mobile environments. *Wireless Networks (WINET)*, 5(1):57–67, January 1999.

[21] 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 *Proceedings of the 20th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 264–273, August 2001.

[22] E. Mok, H. V. Leong, and A. Si. Transaction processing in an asymmetric mobile environment. In *Proceedings of the 1st International Conference on Mobile Data Access (MDA)*, pages 71–81, December 1999.

[23] M. Papadopouli and H. Schulzrinne. Effects of power conservation, wireless coverage and cooperation on data dissemination among mobile devices. In *Proceedings of the 2nd ACM Interational Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 117–127, October 2001.

[24] F. Sailhan and V. Issarny. Cooperative caching in ad hoc networks. In *Proceedings of the 4th International Conference on Mobile Data Management (MDM)*, pages 13–28, January 2003.

[25] H. Schwetman. *User's Guide CSIM19 Simulation Engine (C++ Version)*. Mesquite Software Inc., 1998.

[26] L. Yin and G. Cao. Supporting cooperative caching in ad hoc networks. In *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, March 2004.

[27] G. K. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, 1949.