Group-based Cooperative Cache Management for Mobile Clients in a Mobile Environment*

Chi-Yin Chow Hong Va Leong Alvin T. S. Chan

Department of Computing, Hong Kong Polytechnic University, Hong Kong
{cscychow, cshleong, cstschan}@comp.polyu.edu.hk

Abstract

Caching is a key technique for improving data retrieval performance of mobile clients. The emergence of robust and reliable peer-to-peer (P2P) communication technologies now brings to reality what we call "cooperating caching" in which mobile clients not only can retrieve data items from mobile support stations, but also can access them from the cache in their neighboring peers, thereby inducing a new dimension for mobile data caching. This paper extends a COoperative CAching *scheme, called COCA, in a pull-based mobile environment. Built upon the COCA framework, we propose a* GROup-based COoperative CAching *scheme, called GroCoca, in which we define a* tightly-coupled group (TCG) *as a set of peers that possess similar movement pattern and exhibit similar data affinity. In GroCoca, a centralized incremental clustering algorithm is used to discover all TCGs dynamically, and the MHs in same TCG manage their cached data items cooperatively. In the simulated experiments, GroCoca is shown to reduce the access latency and server request ratio effectively.*

1. Introduction

A mobile system is formed with a wireless network connecting mobile hosts (MHs) to mobile support stations (MSSs). The MHs correspond to mobile clients equipped with portable devices and MSSs represent stationary servers providing information access for the MHs residing in their service areas. This kind of network topology, with the presence of MSSs, is known as an *infrastructure network*. It is a commonly deployed architecture. In this paper, we consider the infrastructure network as a pull-based mobile environment. In a pull-based mobile system, an MH sends a request to the MSS via an uplink channel, and the MSS processes the request and sends the required data item back to the requesting MH via a downlink channel. As mobile environments are characterized by asymmetric communication, in which the downlink channels are of much higher bandwidth than the uplink channels, the MSS would potentially be a scalability bottleneck in the system, as it serves an enormous number of MHs [1].

The other network topology in a mobile environment is called a *mobile ad hoc network* (MANET). In MANETs, an MH can directly communicate with other MHs residing in its transmission range; however, it has to communicate with other MHs who are not residing in the range through multi-hop routing. MANET is practical with many environments with no fixed infrastructure, such as battlefield, rescue operation, etc. In MANETs, the MHs can move freely, and disconnect themselves from the network at any instant. These two characteristics lead to dynamic network topology changes. As a result, the MHs may suffer from long access latency or access failure, when the peers holding the required data items are far way or unreachable. The latter situation may be caused by network partition [16].

The inherent shortcomings of a pull-based and MANET mobile system lead to a result that systems adopting these architectures would not be as appropriate in most real commercial environments. In reality, long data access latency or access failure could possibly cause the abortion of valuable transactions or the suspension of critical activities, so that it is likely to reduce user satisfaction and loyalty, and potentially bring damages to the enterprises involved.

COCA is a mobile cooperative caching framework that combines the P2P communication technology with a conventional pull-based mobile system, wherein the MHs can obtain the required data items either from the MSS or its neighboring peers [6].

In the recent years, cooperative caching in mobile environments has been drawing increasing attention [8, 9, 12, 14, 15]. Our work is different from previous works in that we extend COCA with a GROup-based COoperative CAching scheme (GroCoca), which can take into account the specific communication topology of the "tightly-coupled groups". We propose the formation of *tightly-coupled groups* (TCGs) that respect the geographical vicinity property of *communication groups*, as well as the operational vicinity property of *computation groups*. It is not difficult to define whether two peers are geographically close, based upon their locations. Geographically close peers can communicate in ad hoc mode. Two peers are said to be operationally close, if they perform similar operations and

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

access similar set of data items. Since we are more interested in data management and caching issues in GroCoca in this paper, we consider two peers to be operationally close based on the set of data items they access. Operationally close peers can share data items with minimal help of MSS. To determine the membership of TCGs, we propose an incremental clustering algorithm to construct the TCGs dynamically, making use of information about the MHs' mobility and data access patterns.

Several distributed mobility-based clustering algorithms have been proposed for MANETs [3, 11, 13]. The algorithms cluster the MHs into groups by considering only their mobility patterns. Our work is unique in that we propose an incremental clustering algorithm that not only considers the client mobility pattern, but also the data access pattern. We study the communication overheads incurred by our clustering algorithm in terms of access latency and power consumption as well.

The advantages of COCA are twofold. First, as some of client requests can be handled by the peer, it reduces the number of requests sent to the MSS. Thus, the system workload, can be shared among the MHs. This desirable feature can be considered as a load sharing technique in mobile environments. The reduction on the number of server requests eases the traffic load in the scarce uplink channel. Second, when an MH that is not residing in the service area of the system encounters a local cache miss, it still has a chance to obtain the required data item from its peers before constituting an access failure. Our simulation results show that GroCoca can further improve the system performance by increasing data accessibility in TCGs identified dynamically.

The rest of this paper is organized as follows. Section 2 gives an overview of the COCA model. Section 3 delineates the group-based COCA extension, GroCoca, based on an incremental clustering algorithm. Section 4 describes the simulation model of COCA and GroCoca. Section 5 studies the performance of COCA and GroCoca through a number of simulated experiments. Finally, Section 6 offers brief concluding remarks on this work.

2. COCA

We assume that each MH is equipped with two wireless network interface cards (NICs) with a transmission range ($TranRange$), in which one is dedicated for communicating with the MSS, while the other one is devoted to communicate with other MHs via their wireless NICs. When an MH cannot find the required data item in its local cache, i.e., a *local cache miss*, it broadcasts a *request* message to its neighboring peers via P2P broadcast communication. The peers caching the required data item replies to the MH with a *reply* message via P2P point-to-point communication. After a timeout period, if the MH receives replies from some neighboring peers, i.e., a *global cache hit*, it selects the closest peer as a target peer. The MH then sends a *retrieve* request to the target peer via P2P point-to-point communication. Finally, the target peer receiving the *retrieve* request sends the requested data item to the MH also via P2P point-to-point communication. On the other hand, if no neighboring peer turns in a *reply* message throughout the timeout period, i.e., a *global cache miss*, the MH has to obtain the required data item from the MSS.

The timeout period is initially set to a default value that is defined as a round-trip time of a P2P communication scaled up by a congestion factor (φ), i.e., $\frac{size\ of\ request + size\ of\ reply}{BW_{P2P}} \times \varphi$, 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 of σ_τ, i.e., timeout period = $\overline{\tau} + \varphi'\sigma_\tau$.

3. GroCoca

In this section, we present the group-based cooperative caching scheme, GroCoca, extended from COCA. Client mobility is a key factor to the system performance of cache management in mobile environments. In a cooperative cache environment, if an MH arbitrarily forwards its cached data items to its neighboring peers for cooperative cache replacement, there may not be much benefit to the receiving MH, since the data items may not be useful to the latter. Furthermore, it does not make good use of the cache space of the receiving MH, since it may well have moved far away from the original MH after a while. Likewise, if an MH does not cache the data item returned by a neighboring peers in order to conserve its cache, thinking that the peer would still be accessible in the future, it may have regretted if the peer moves far away. In addition, when an MH forwards a data item to another peer and they possess different data access patterns, the action will reduce the peer's local cache hit ratio, and the "alien" data item will be removed very soon, as the peer is not interested in that data item. The decision of whether a data item should be cached thus depends on both factors of the access affinity on the data items and the mobility for each MH. To develop an effective cooperative caching scheme, GroCoca is proposed which defines and makes use of TCG which is defined as a group of MHs that are geographically and operationally close, i.e., sharing common *mobility* and *data access* patterns. In GroCoca, the common mobility pattern is discovered with an incremental clustering algorithm and the similarity of access patterns is captured by frequency-based similarity measurement.

3.1. Mobility Pattern

The MSS performs the incremental clustering algorithm to cluster the MHs into TCGs based on their mobility patterns. The mobility pattern is modeled by a weighted average distance between any two MHs. The MHs need not explicitly update their locations, but they embed the location information in the request sent to the MSS for the

requesting data items. The location information is represented by a coordinate $(x,\ y)$ that can be obtained by global positioning system (GPS) or indoor sensor-based positioning system. For two MHs, m_i and m_j, the distance between them is calculated as the Euclidean distance, $|m_i m_j| = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$, where $(x_i,\ y_i)$ and $(x_j,\ y_j)$ are the coordinates of m_i and m_j respectively. An exponentially weighted moving average (EWMA) [5] is used to forecast the future distance of each pair of MHs based on their mobility histories. The weighted average distance between two MHs, m_i and m_j, is denoted by $\|m_i m_j\|$. It is initially set to $+\infty$. After the MSS receives both the location information of m_i and m_j, $\|m_i m_j\|$ is set to $|m_i m_j|$. Then $\|m_i m_j\|$ is updated when either m_i or m_j sends their new location to the MSS, based on the equation: $\|m_i m_j\|^{new} = \omega \times |m_i m_j| + (1 - \omega) \times \|m_i m_j\|^{old}$, where ω $(0 \leq \omega \leq 1)$ is a parameter to weight the importance of the most recent distance. The weighted average distance of each pair of MHs is stored in a two-dimensional matrix, called a *distance matrix* (DM).

3.2. Data Access Pattern

Other than the mobility pattern of the MHs, the incremental clustering algorithm also considers the similarity of their accesses to the data items. In the MSS, each data item is associated with an identifier from 1 to n, where n is the number of data items stored in the server. The MSS maintains a counter vector (V) with a length n for each MH to store the access frequency of every data item. When an MH accesses a data item, the MSS increments the corresponding counter for the MH. The similarity score of two MHs, m_i and m_j, is calculated by the equation: $\text{sim}(m_i, m_j) = \frac{\sum_{k=1}^{n} V_i(k) \times V_j(k)}{\sqrt{\sum_{k=1}^{n} V_i(k)^2} \times \sqrt{\sum_{k=1}^{n} V_j(k)^2}}$, where $V_i(k)$ is the total number of times that an MH, m_i, accesses the data item k. Note that $0 \leq \text{sim}(m_i, m_j) \leq 1$. The similarity score of each pair of MHs is stored in a two-dimensional matrix, called an *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 its data access pattern from the received requests sent by the MH. If the measured similarity score of two MHs is larger than or equal to a threshold δ, they are considered to possess a similar access pattern. The threshold δ is set to the average non-zero similarity score in ASM, and adjusted with the standard deviation to adapt to client access behavior in different types of systems, i.e., $\overline{s} = \frac{\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \text{sim}(m_i, m_j)}{\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \lceil \text{sim}(m_i, m_j) \rceil}$, $\delta = \overline{s} + \phi \sigma_s$, where ϕ is a system parameter. Since the measured similarity score only takes into consideration a sample of the access pattern of each MH, it is smaller than the actual similarity score. The MSS would not start the clustering algorithm until δ grows up to non-zero.

3.3. Incremental Clustering Algorithm

The incremental clustering algorithm clusters the MHs into TCGs by considering their mobility and access patterns. The MHs in the same TCG possess a tight relationship because they share the common mobility and data access patterns.

In the clustering algorithm, a distance threshold Δ is adopted to determine whether an MH should be assigned to one of existing clusters or a new cluster should be created. The two matrices DM and ASM are used as input to the algorithm. In a TCG, any two MHs, m_i and m_j, possess two properties: $\|m_i m_j\| \leq \Delta$ and $\text{sim}(m_i, m_j) \geq \delta$. The first MH classified into a new cluster is considered as a leader of that cluster, called the *cluster leader*. Each cluster has only one leader, and the cluster leader may be changed when a new MH is classified into the cluster.

The clustering algorithm considers the MHs one at a time and either assigns them to the existing clusters or creates new clusters for them. Let there be $NumClient$ MHs in the system, $\mathcal{M} = \{m_1, m_2, \ldots, m_{NumClient}\}$, in c existing clusters, $\mathcal{C} = \{C_1, C_2, \ldots, C_c\}$ with c corresponding cluster leaders, $\mathcal{L} = \{l_1, l_2, \ldots, l_c\}$. Initially, a new cluster C_1 is created for the first MH, m_1, and m_1 becomes the cluster leader of C_1. Then, the other MHs are considered one by one for admission. For each newly admitted MH, m, the algorithm finds out the closest cluster leader by looking up the weighted average distance in DM. m is assigned to the closest cluster on condition that the two properties hold, i.e., the weighted average distance between the MH and the cluster leader is less than or equal to Δ and their similarity scores are larger than or equal to δ; otherwise, the algorithm finds out the next closest cluster leader. This procedure is repeated until an appropriate cluster is found or no existing cluster satisfies the conditions. If there is no suitable cluster for the MH, the MH becomes the cluster leader of a new cluster.

When an MH is assigned to one of the existing clusters, the algorithm has to decide whether the newly admitted MH should be the leader of that cluster. The decision is based on their connectivity. The connectivity of an MH, m, as denoted by $\text{Conn}(m)$, is defined as the number of MHs whose weighted average distance between the MH and m is less than or equal to Δ. For instance, assume that m is assigned to an existing cluster C_h. If $\text{Conn}(m) > \text{Conn}(l_h)$, m takes over the role as the cluster leader from l_h; otherwise, no transition is required.

When a transition of cluster leader occurs, the algorithm checks whether any neighboring clusters should be merged together. If an MH, m, becomes the cluster leader of a cluster, C_h, the algorithm finds out the closest cluster leader, l_k. If the weighted average distances between l_h and all MHs, including the leader, in cluster C_k are less than or equal to Δ and their similarity scores are larger than or equal to δ, C_h and C_k are merged together. The cluster leader of the merged cluster is the one with the highest connectivity. If

Algorithm 1 The continuous clustering algorithm

```
1:  ClusterUpdate(Cluster C, Leader L, MH M)
2:  while true do
3:    for all m ∈ M do
4:      cid ← ClusterID(m); // returns the ID of the assigned Cluster
        of an MH.
5:      classified ← false;
6:      if l_cid = m then
7:        CheckClusterMerge(C, L, m);
8:      else
9:        while min_{C_j ∈ C} ||m l_j|| ≤ Δ do
10:         if sim(m, l_j) ≥ δ then
11:           if m ∉ C_j then
12:             // reassign m to C_j
13:             C_cid ← C_cid − {m}; C_j ← C_j ∪ {m};
14:           end if
15:           if Conn(m) > Conn(l_j) then
16:             // change leadership
17:             l_j ← m;
18:             CheckClusterMerge(C, L, m);
19:           end if
20:           classified ← true;
21:         end if
22:       end while
23:       if not classified then
24:         // form a new cluster for m
25:         k ← newClusterID(); // returns a new ID
26:         C_cid ← C_cid − {m};
27:         C_k ← {m}; C ← C ∪ {C_k};
28:         l_k ← m; L ← L ∪ {l_k};
29:       end if
30:     end if
31:   end for
32: end while
```

there is a tie, the initiator, l_h, becomes the leader. The algorithm repeats the checking to the next closest cluster, until the weighted average distance between the next closest cluster and m is larger than Δ.

The MSS postpones the announcement of any changes in the cluster, e.g., an MH is assigned to a new cluster or an MH joins/leaves, to any affected MH until the MH sends a request. The MSS then piggybacks the up-to-date member list to the MH along with the required data item. In the member list, the first MH is the cluster leader. In effect, we are performing an asynchronous group view change, without enforcing stringent consistency among group members.

The incremental clustering algorithm consists of four components. Algorithm 1 is a continuous clustering algorithm that is periodically executed to keep track of any changes in the MH mobility and data access patterns and hence the TCG properties. Algorithm 2 is executed when the system detects a new MH. Algorithm 3 is invoked when an MH leaves the system. Algorithm 4, which is invoked by Algorithm 1, is dedicated for performing cluster merging operations.

In the incremental clustering algorithm, Δ is a key factor to the clustering result. If we consider a case that all MHs in a cluster can connect to each other in single or two-hop communication, the required distance set to Δ is now derived. Figure 1 illustrates a cluster wherein m_1 is the

Algorithm 2 The MH join algorithm

```
1:  MHJoin(Cluster C, Leader L, MH m)
2:  classified ← false;
3:  if C ≠ φ then
4:    while min_{C_j ∈ C} ||m l_j|| ≤ Δ do
5:      if sim(m, l_j) ≥ δ then
6:        C_j ← C_j ∪ {m};
7:        if Conn(m) > Conn(l_j) then
8:          l_j ← m;
9:          CheckClusterMerge(C, L, m);
10:       end if
11:       classified ← true;
12:     end if
13:   end while
14: end if
15: if not classified then
16:   // form a new cluster for m
17:   k ← newClusterID(); // returns a new ID
18:   C_k ← {m}; C ← C ∪ {C_k};
19:   l_k ← m; L ← L ∪ {l_k};
20: end if
```

Algorithm 3 The MH leave algorithm

```
1:  MHLeave(Cluster C, Leader L, MH m)
2:  cid ← ClusterID(m);
3:  C_cid ← C_cid − {m};
4:  if C_cid = φ then
5:    C ← C − {C_cid}; L ← L − {l_cid};
6:    return;
7:  end if
8:  if l_cid = m then
9:    k ← arg_j max_{m_j ∈ C_cid}(Conn(m_j)); l_cid ← m_k;
10: end if
```

Algorithm 4 The cluster merge algorithm

```
1:  ClusterMerge(Cluster C, Leader L, MH m)
2:  cid ← ClusterID(m);
3:  while min_{C_j ∈ C ∧ cid ≠ j} ||l_cid l_j|| ≤ Δ do
4:    if sim(l_cid, l_j) ≥ δ then
5:      merge ← true;
6:      for all m_h ∈ C_j do
7:        if ||l_cid m_h|| > Δ or sim(l_cid, m_h) < δ then
8:          merge ← false; break;
9:        end if
10:     end for
11:     if merge then
12:       if Conn(l_cid) < Conn(l_j) then
13:         l_cid ← l_j;
14:       end if
15:       C_cid ← C_cid ∪ {C_j};
16:       C ← C − {C_j};
17:       L ← L − {l_j};
18:     end if
19:   end if
20: end while
```

leader of the cluster, and m_2 and m_3 are two MHs classified into m_1's cluster. Let the distance between m_2 or m_3 and m_1 be equal to Δ. The distance l between m_1 and m_2 can be computed as: $l = \sqrt{\Delta^2 + \Delta^2 - 2\Delta^2 \cos\gamma} = \Delta\sqrt{2(1-\cos\gamma)}$, where $0 < \gamma \leq \pi$. When $\gamma = \pi$, $l = 2\Delta$, i.e., l is maximal. To ensure a single-hop communication for the peers in a cluster, l should be less than or equal to $TranRange$, i.e., $2\Delta \leq TranRange$; hence, $\Delta = \frac{TranRange}{2}$. If Δ is set to $TranRange$, all MHs

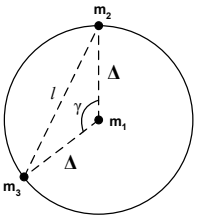


Figure 1. Threshold Δ selection

in a cluster can connect to one another in single- or two-hop communication. After presenting the clustering algorithm, the cooperative cache management scheme will be described in the next section.

3.4. Cooperative Cache Management Scheme

For every TCG, the MHs work together to manage their cache space as a whole, i.e., *global cache*. When an MH encounters a local cache miss, it sends a request to its neighboring peers. If one of its peers can turn in the required data item 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 the data item can be readily available from the peer if needed. Finally, 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 data item, since the providing peer may move away in future. After a peer sends the required data item to the requesting MH, if they are in the same TCG, the peer updates the last accessed time stamp of the data item, so that the data item can have a longer time-to-live (*TTL*) in the cache.

In GroCoca, the least valuable data item should be first removed from the global cache. To avoid incurring more communication overheads on forwarding data items among the MHs in TCG, an approximate LRU is implemented to perform cache replacement for peers in a TCG. The MH removes the data item with the least utility value from its local cache. The delay on removing the least valuable data item in TCG can be bounded by the time when each MH in the same TCG replaces its least valuable data item from their local cache once.

4. Simulation Model

In this section, we present the simulation model that is used to evaluate the performance of COCA and GroCoca. The simulation model is implemented in C++ using CSIM. The simulated mobile environment is composed of an MSS and $NumClient$ MHs. The MHs move in a 1000 m $\times$ 1000 m ($Area$) space, which constitutes the service area of the MSS. There are 20 wireless communication channels between the MSS and the MHs with a total bandwidth 10 Mbps in downlink channels and 100 kbps in uplink channels. If all the point-to-point communication channels are busy, the MH has to wait until one of them is available for transmission. Also, there is a wireless communication channel for an MH to communicate with other MHs with a bandwidth 2 Mbps and with a transmission range of $TranRange$. When an MH sends a message to another peer, it has to wait if either its channel or the peer's channel is busy.

4.1. Client Model

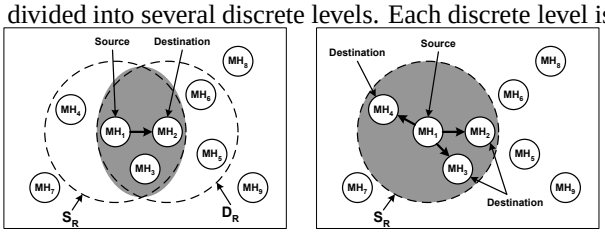
The mobility of the MHs is based on "reference point group mobility" model [10] which is a random mobility model for a group of MHs and for an individual MH within its group. Each group has a logical center which represents a motion reference point for the MHs belonging to the group. The group mobility is defined according to the "random waypoint" model [4]. The group randomly chooses its own destination in $Area$ with a randomly determined speed s from a uniform distribution $U(v_{min}, v_{max})$. It then travels with the constant speed s. When it reaches the destination, it comes to a standstill for one second to determine its next destination. It then moves towards its new destination with another random speed $s' \sim U(v_{min}, v_{max})$. All groups repeat this movement behavior during the simulation. In the simulation, the MHs are divided into several motion groups, and each group has $GroupSize$ MHs.

The MHs generate accesses to the data items following a Zipf 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 group share a common access range $AccessRange$. The access range of each group is randomly selected.

4.1.1. Power Consumption Model

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. The wireless NICs of the non-destination MH operates in an idle mode during the transmission. Furthermore, when an MH communicates with other MHs, it is able to adjust the transmission range by controlling the transmission power to the minimum range that can cover the target peers [2, 17].

In the power control mechanism, the transmission power is divided into several discrete levels. Each discrete level is



(a) P2P point-to-point communication (b) P2P Broadcast communication

Figure 2. Power consumption model.

associated with a predefined equi-width transmission range. The minimum and maximum power levels produce the shortest transmission range R_{min} and the longest transmission range R_{max} respectively. The distance of each equi-width transmission range is R_{range}. When an MH, m_i, establishes a P2P point-to-point connection to another MH, m_j, the distance between them is d, $d = |m_i m_j|$. The required minimum transmission range can be calculated by:

$$R_{ij} = \begin{cases} R_{min}, & \text{for } d \leq R_{min} \\ \lceil d/R_{range} \rceil \times R_{range}, & \text{for } R_{min} < d \leq R_{max} \end{cases} \quad (1)$$

The communication power consumption measurement is based on [7], 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 and the destination MH, D_R, Figure 2(a). The P2P point-to-point power consumption is measured by Equation 2.

$$P_{p2p} = \begin{cases} (v_{send} \times |msg|) + f_{send}, & \text{for } m = S \\ (v_{recv} \times |msg|) + f_{recv}, & \text{for } m = D \\ (v_{sd\_disc} \times |msg|) + f_{sd\_disc}, & \text{for } m \in S_R \wedge m \in D_R \\ (v_{s\_disc} \times |msg|) + f_{s\_disc}, & \text{for } m \in S_R \wedge m \notin D_R \\ (v_{d\_disc} \times |msg|) + f_{d\_disc}, & \text{for } m \notin S_R \wedge m \in D_R \end{cases} \quad (2)$$

where f is the fixed setup cost for a 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 2(b), can be measured by Equation 3.

$$P_{bc} = \begin{cases} (v_{bsend} \times |msg|) + f_{bsend}, & \text{for } m = S \\ (v_{brecv} \times |msg|) + f_{brecv}, & \text{for } m \in S_R \end{cases} \quad (3)$$

Although the power consumption model in [7] 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 wireless NICs.

4.2. Server Model

There is a single MSS in the simulated mobile environment. A database connected to the MSS contains 10000 equal-sized ($DataSize$) data items. It processes the request sent by all MHs residing in $Area$ with a first-come-first-serve policy. An infinite queue is used to buffer the requests from the MHs when the processor is busy.

Table 1 shows the default parameter settings used in the simulated experiments. Table 2 and Table 3 show the parameter settings for MHs playing different roles, as used in power consumption measurement for P2P point-to-point and broadcast communication respectively [7].

Table 1. Default parameter settings.

Parameters	Default Values	Parameters	Default Values
NumClient	100	R_{min}, R_{max}	10, $TranRange$
GroupSize	5	R_{range}	10 m
DataSize	1 kbytes	Δ	$\frac{1}{2}TranRange$
AccessRange	1000 items	φ, φ'	10, 3
CacheSize	100 items	θ	0.5
$TranRange$	100 m	ω	0.25
Speed ($v_{min} \sim v_{max}$)	1 $\sim$ 5 m/s	ϕ	-1

Table 2. Parameter settings for power consumption measurement 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 3. Parameter settings for power consumption measurement in P2P broadcast communication

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

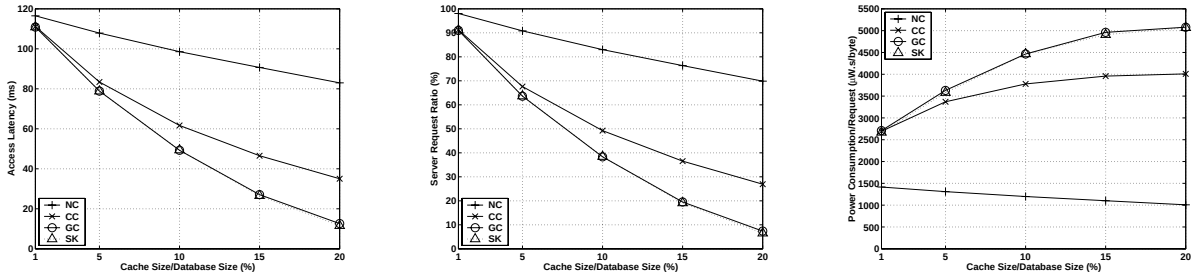
5. Simulated Experiments

In our simulated experiments, we compare the performance of COCA with a conventional caching scheme that does not involve any cooperation among MHs. This serves as a base case for comparison. LRU cache replacement policy is adopted in the base case (non-COCA or NC) and COCA (CC). Likewise, the performance of GroCoca (GC) is evaluated by comparing against the standard COCA. A perfect static knowledge of group classification (SK) is used as a yardstick to evaluate the effectiveness of GC in dynamically determining the TCG membership through incremental clustering. In the perfect static knowledge case, each MH knows the predefined mobility and data access patterns of *all* MHs so that it can correctly identify all the members in its TCG. All simulation results are recorded after the system reaches a stable state. Each MH generates about 20000 requests, where 2000 are treated as warm-up requests in order to avoid a transient effect. We conduct the experiments by varying several parameters: cache size, number of MHs and group size. The performance metrics are the access latency, server request ratio and power consumption on communication. The access latency is defined as a sum of the transmission time and the time spent on waiting for a required communication channel, if it is busy.

5.1. Effect of Cache Size

Our first experiment studies the effect of cache size on the system performance by varying the ratio of cache size to databases size from 1 percent to 20 percent. The results are shown in Figure 3.

Figures 3(a) and 3(b) show that all schemes exhibit better access latency and server request ratio with increasing cache size. This is because more required data items can be found in the local cache as the cache gets larger. In terms of the access latency and server request ratio, COCA schemes outperform the non-COCA scheme. For an MH adopting



(a) Access Latency (b) Server Request Ratio (c) Power Consumption/Request

Figure 3. Performance studies on various cache size.

COCA schemes, other than achieving a higher local cache hit ratio as the cache size gets higher, it also enjoys a higher global cache hit ratio because the chance of some neighboring peers turning in the required data item increases with the larger cache size. GC can further improve the access latency and the server request ratio, as it can effectively increase the data availability in TCG. Thus, the MHs adopting GC can achieve a higher global cache hit ratio than COCA.

The cost of adopting CC and GC is higher power consumption, as depicted in Figure 3(c). When the MHs enjoy higher local cache hit ratio, they can reduce the power consumption in broadcasting request to their neighboring peers. However, when the global cache hit ratio increases, more peers turn in the required data item to the requesting MHs. Therefore, the MHs have to consume much more power for sending the data item to their neighboring peers, and discarding unintended messages as they are residing in the transmission range of a source MH, destination MH or both. It would have saved much power if there is a mechanism for MHs to filter off unintended messages early on.

The performance difference between GC and SK increases as the server request ratio decreases. In GC, the accuracy of the clustering algorithm depends on the update rate of the mobility and data access information that is embedded to the request sent to the MSS. The update rate reduces with decreasing server request ratio so that the precision of the clustering algorithm is reduced.

5.2. Effect of Number of MHs

This experiment studies the effect of system workload on the system performance by varying the number of MHs in the system from 100 to 500 with the same default $GroupSize$. The results are depicted in Figure 4.

The system workload becomes higher with increasing number of MHs in the system, so that access latency increases in NC, as depicted in Figure 4(a). On the other hand, performance of CC and GC improves slightly as the number of MHs increases because there is a higher chance for the MHs to obtain the required data items from their neighboring peers. In other words, the system workload caused by increasing number of MHs is distributively shared by the group of MHs. The COCA schemes can thus be used as a load sharing technique in a heavily-loaded mobile environment. However, CC and GC are power intensive protocols, as depicted in Figure 4(c), the power consumption of the MHs increases with the increasing number of MHs.

In this series of experiment, the client density of the system gets higher, as the number of MHs increases. In a high density situation, the chance of multiple motion groups possessing common mobility and data access patterns increases. Since GC is able to take such dynamic mobility behavior into consideration, the performance of GC is better than SK, when the number of MHs in the system is larger than 100. This result shows that the incremental clustering algorithm is effective in determining the proper TCG dynamically. The server request ratio is slightly reduced as the number of MHs increases, so the MSS can obtain sufficient location and data access information from the MHs via the server requests.

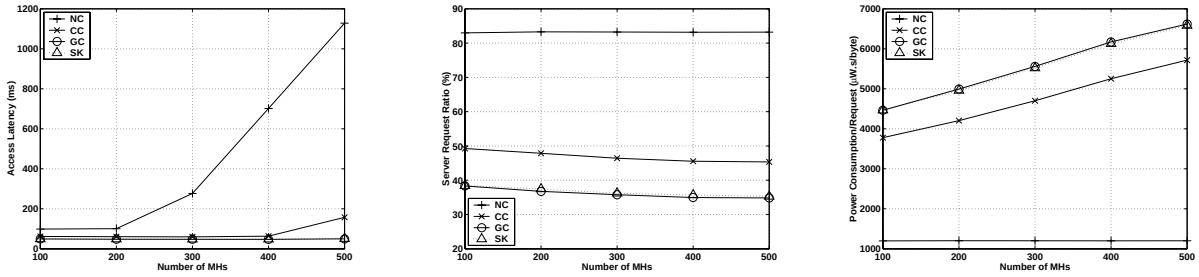
5.3. Effect of Group Size

In this series of simulated experiment, we examine the influence of system performance on various group size: 1, 2, 4, 5, 10, 20 and 25 and the results are illustrated in Figure 5.

In Figures 5(a) and 5(b), it can be observed that CC and GC outperform NC as the group size increases. Since there is no collaboration among peers in NC, its performance is not affected by varying the group size. Similar to the study of the cache size, GC performs better than CC in this series of simulated experiments. Since GC can improve the data availability in a TCG, the probability that the MHs can obtain the required data items from their neighboring peers is higher. Thus, access latency and server request ratio can be reduced. However, in CC and GC, the MHs have to consume much more power to communicate with other peers, as depicted in Figure 5(c). As the group size increases, the MHs spend more power to receive broadcast requests and to discard unintended messages sent by their neighboring peers. GC performs worse than SK when the server request ratio is less than 5 percent, due to the fact that the MSS does not have sufficient information to maintain the accuracy of the clustering algorithm.

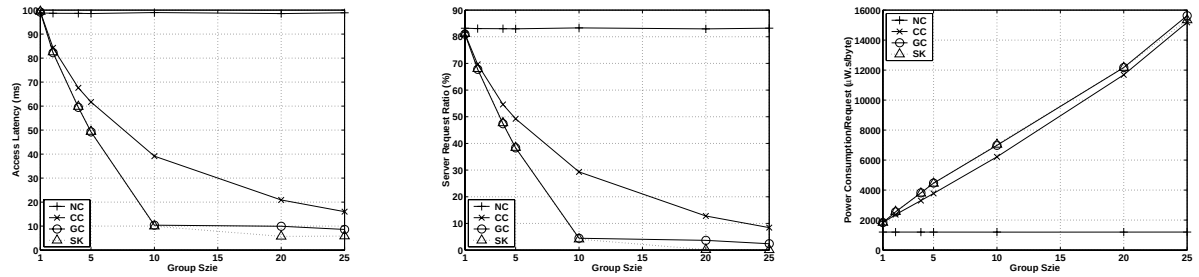
6. Conclusion

In this paper, we have described a cooperative caching scheme, called COCA, in a pull-based mobile environment, and proposed a group-based cooperative caching scheme, called GroCoca, which adopts a centralized incremental



(a) Access Latency (b) Server Request Ratio (c) Power Consumption/Request

Figure 4. Performance studies on various number of MHs.



(a) Access Latency (b) Server Request Ratio (c) Power Consumption/Request

Figure 5. Performance studies on various group size.

clustering algorithm to determine all *tightly-coupled groups* (TCGs) by considering the combination of MHs' mobility and data access patterns. In a TCG, the MHs work together to make cache placement decision to improve data availability. The performance of COCA and GroCoca is evaluated through a number of simulated experiments, which show that COCA substantially reduces the access latency and the server request ratio, but it consumes much more power than a conventional caching scheme. GroCoca further improves the access latency and the server request ratio, but with extra power consumption.

References

[1] S. Acharya, M. Franklin, and S. Zdonik. Balancing push and pull for data broadcast. In *Proc. of the SIGMOD*, pages 183–194, May 1997.

[2] S. Agarwal, S. V. Krishnamurthy, R. H. Katz, and S. K. Dao. Distributed power control in ad-hoc wireless networks. In *Proc. of the 12th PIMRC*, pages 59–66, September 2001.

[3] 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*, 11(6):387–395, November 2001.

[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 *Proc. of the 4th MobiCom*, pages 85–97, October 1998.

[5] J. H. P. Chim, M. Green, R. W. H. Lau, H. V. Leong, and A. Si. On caching and prefetching of virtual objects in distributed virtual environments. In *Proc. of the 6th ACM International Conference on Multimedia*, pages 171–180, September 1998.

[6] C.-Y. Chow, H. V. Leong, and A. Chan. Peer-to-peer cooperative caching in mobile environments. In *Proc. of the 24th ICDCS Workshops on MDC*, pages 528–533, March 2004.

[7] L. M. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Proc. of the 20th INFOCOM*, pages 1548–1557, April 2001.

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

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

[10] X. Hong, M. Gerla, G. Pei, and C.-C. Chiang. A group mobility model for ad hoc wireless networks. In *Proc. of the 2nd MSWiM*, pages 53–60, August 1999.

[11] J.-L. Huang, M.-S. Chen, and W.-C. Peng. Exploring group mobility for replica data allocation in a mobile environment. In *Proc. of the 12th CIKM*, pages 161–168, November 2003.

[12] W. H. O. Lau, M. Kumar, and S. Venkatesh. A cooperative cache architecture in support of caching multimedia objects in MANETs. In *Proc. of the 5th ACM WoWMoM*, pages 56–63, September 2002.

[13] C. R. Lin and M. Gerla. Adaptive clustering for mobile wireless networks. *IEEE JSAC*, 15(7):1265–1275, September 1997.

[14] M. Papadopouli and H. Schulzrinne. Effects of power conservation, wireless coverage and cooperation on data dissemination among mobile devices. In *Proc. of the 2nd MobiHoc*, pages 117–127, October 2001.

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

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

[17] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides. Energy-efficient broadcast and multicast trees in wireless networks. *MONET*, 7(6):481–492, December 2002.