A Cooperative Game Theoretic Approach for Data Replication in Mobile Ad-Hoc Networks

Dan Hirsch
Department of Computer Science
Missouri S&T
Rolla, MO 65409, USA
Email: daniel.hirsch@mst.edu

Sanjay Madria
Department of Computer Science
Missouri S&T
Rolla, MO 65409, USA
Email: madrias@mst.edu

***Abstract*—The mobile computing environment provides many benefits such as ubiquitous access to computing but include constraints on resources such as: available bandwidth and battery life. Replication is a widely recognized method for balancing the demands of storage space with bandwidth and battery life. We propose a novel scheme that seeks to strategically balance these constrained resources through a cooperative game-theoretic approach for replication in a mobile environment. Our replication strategy relies on the cooperation of the nodes within the network to make replica caching decisions which are spatiotemporally local-optimal for the network from an energy and bandwidth conservation standpoint. In cooperative altruistic data replication, *CADR*, each node calculates the net global benefit, *NGB*, for caching a replica of the requested data, as the result data is returned from the responding node to the requesting node, where it is then determines the spatiotemporally local-optimal node for replicating the data item. Performance results from our research indicate that our scheme, CADR, improves the query response time by 25% and 45%, mean hop count is improved by 26% and 46%, query error is reduced by 30% and 48%, while energy utilization is reduced 30% and 57% when compared with both another game theoretic replication approach and standard cooperative caching respectively.**

***Index Terms*—mobile, replication, game-theoretic**

I. Introduction

The key to overcoming mobile constraints is to increase the utility of the mobile computing device and network through the efficient management of these resources. A mobile device is unable to store every point of data it might require; therefore a caching mechanism is needed to distribute the storage burden, cooperatively, in the mobile ad-hoc network. However, with a distributed storage mechanism, such as caching, an increase in bandwidth utilization becomes an issue. If a mobile device doesn't hold a needed data item locally, in cache, it must fetch the data from another device in the network; increasing the communication burden significantly. In a mobile device, wireless communication is the single largest consumer of energy [3]. It follows that an increase in communication causes an increase in the consumption of energy and corresponding decrease in the lifespan of the device.

Replication is a widely recognized method for balancing the demands of storage space with bandwidth and battery life. Replication distributes additional copies of primary data items into the network in order to increase accessibility and decrease communication costs. While the goals of various approaches to replication are similar, there are significant differences in both implementation and result. In this paper, we propose a novel scheme that seeks to strategically balance energy, bandwidth, and storage space through a cooperative game-theoretic approach for replication in a mobile environment.

Our replication strategy relies on the collaboration of the nodes within the network to make replica caching decisions which are spatiotemporally local-optimal for the network from an energy and bandwidth conservation standpoint. This strategy takes on the nature of reciprocal altruism and is very similar to the Volunteer's Dilemma [8][13][1] in game theory. Though our methods do not conform to the standard definition of game theory, they do represent a game-theoretical methodology through a dependence upon a utility function which has the goal of mutual optimization. Reciprocal altruism, in the context of replication, is best defined as a node making a caching decision that will temporarily reduce its own resource availability, through a decrease in storage space, while increasing the overall viability and lifespan of the network, as a result of the conservation of both energy and bandwidth, while expecting that the other nodes will make similar decisions in the future. The expectation for reciprocity is the motivating factor that overcomes the cost of the sacrifice. In game theory, the Volunteer's Dilemma is a game where each player makes a decision that will benefit the whole by making a small sacrifice, similar to the principal of reciprocal altruism. Given the similarities, we have named our replication strategy Cooperative Altruistic Data Replication.

The game, therefore, in Volunteers Dilemma is one of mutual benefit. In the end, the *game*, contrary to traditional selfish games, is to make caching decisions which mutually benefit the others. This back-and-forth altruistic nature, is at its core a collaboration game with the result being node survival and data availability long term. While the cooperative behavior of nodes in our replication strategy are strikingly different from the selfish behavior found in other implementations, such as [7], there are many applications where a cooperative/altruistic caching strategy does have an advantage. Close-system implementations for such fields as military, search and rescue, as well as mobile collaborative environments, where there isn't the opportunity for selfish caching behavior would exemplify

COLLABORATECOM 2011, October 15-18, Orlando, United States

DOI 10.4108/collaboratecom.2011.20

applications of cooperative caching, where nodes collaborate towards a common goal. The goal of such systems is to evenly distribute the energy, bandwidth, and storage burden in an effort to keep as many nodes available, for the health and viability of the network as a whole.

In cooperative altruistic data replication, *CADR*, each node calculates the net global benefit, *NGB*, for caching a replica of the requested data, as the result data is returned from the responding node to the requesting node. As the response passes through each node of the return path, the historical request information for that data item is evaluated and the global savings is calculated along with a global cost and used to derive the *NGB*.

The global savings, *GS*, is defined as the global energy resources that would be conserved in the network, through a reduction in retransmission of data, if that node is chosen to host a replica of the given data item. The global cost, *GC*, is defined as the total cost to the network, in additional resource utilization, to update the data item plus the additional overhead cost if an existing replica is removed in order to make room for the new. After both the *GS* and *GC* are determined, the node calculates the net global benefit by subtracting the *GC* from *GS*, $NGB = GS - GC$.

After these calculations, the *NGB* for caching the data item at that particular node is added to the *NGB* matrix, which is contained in the response header holding the requested data. Each node in the return path adds its own *NGB*. The requesting node then compares its own *NGB* with that of the other nodes, and decides if its NGB is spatiotemporally near-optimal. If it is, the node caches a replica of the data item. If it is not, it reverses the return path, and sends a message, along with the data item and *NGB* matrix, towards the near-optimal *NGB* node. The *NGB* optimal node then caches the replica.

Some parallels with our replication scheme, CADR, can be found in existing research [11][7][15][5]. Our work differs significantly from that found in those papers as we fully consider: 1) Finding the optimal *NGB* for caching a replica at a given mobile node, 2) The global cost for updating/cache coherency, 3) The deallocation of replicas when their utility has expired, 4) The continuity of read/write access to a primary data item through proper safe leader election of a new control node when the legacy control node is either turned off, loses connectivity to the network, or simply runs out of energy.

The performance results from our simulations indicate that our scheme, *CADR*, when compared with standard cooperative caching [10], improves the query response time by 45%, the mean hop count by 46%, the query error rate is reduced by 45%, and energy utilization is reduced by 57%. When the simulation results from *CADR* are compared with another game theoretic replication scheme, *Data Replication Game* [7], *CADR* improves query response by 25%, mean hop count by 26%, and query error is reduced by 30%.

The remainder of this paper is organized in this fashion: Section II will review some of the related work. In Section III, we formulate and define the problem and supporting algorithms. The full algorithm for our replication scheme, *CADR*, is detailed in section IV. Section V provides an overview of the implementation of our simulation and corresponding environment. Our results are reported in VI. In section VII we conclude the paper.

II. Related Research

There has been substantial research effort in the area of data replication in a mobile environment. Some work, such as in [2][4][12], pay particular attention to considering groups of nodes for replica placement, while others, such as in [15][5][11], take more of an access frequency approach in an effort to conserve energy and improve response time overall. Further, work such as in [7] take a strict game-theoretic approach for replica placement.

In [2], the authors approach the problem by forming *Zones* that consist of a node and its one-hop neighbors. The *Zone* then makes caching decisions based upon the access frequency of data items by members of their *Zone*. The work in [4], particularly the *DAFN*, and *DCG* schemes, use access frequency as a means to make replication decisions, but utilize a group of nearest neighbors or stable groups respectively. In, [12], groups are considered those which are one-hop neighbors. When a node wants to make a replication decision, it first queries the others in its group to see if the data is present.

Other approaches utilize the access frequency for data items, but not groups or zones. In [15], a replica of a data item is cached at a particular node when: 1) It is believed to be popular, and 2) The request pattern is multi-sourced, or coming from more than one particular node or branch in their delivery path. The work in, [11] proposes a few different implementation schemes some of which give more weight by considering energy availability or hop distance to other nodes. The work in [5] seeks to distribute replicas based on access frequency of the data item at the primary cache site. They consider data items which have a sustained request frequency that is statistically more significant than other data items at the same node, as candidates for replication. After a candidate has been identified, the replica is sent down what is called a *dominant request path* in order to settle into an optimum location of equalibrium in demand.

Lastly, game-theoretic schemes for replication in a mobile environment have not been significantly studied. The work in [7], named *Data Replication Game DRG*, makes replication decisions based upon a bidding process. Each node submits a bid, which is a calculation using how far they are from the nearest copy of the data item minus how far they are from the master node holding the data item. After evaluation of all bids, the *organizer* awards the ability to cache a replica to the winning node. In this scheme, a system of payments is utilized for compensating the node for taking on a replica.

The primary difference between the group and access frequency approaches and the work presented in this paper can be found in the scope of consideration the algorithms use to make replication decisions. These schemes take a very localized view in order to function. They do not generally consider the wellfare of the network nor do they seek to find balance in

the number of replicas in the system. Further, the scope of the algorithms found in this work does not focus on placing replicas in spatiotemporally near-optimal configurations. Many of these schemes also do not consider the consistency of cached items when an update is made. Lastly, the primary differences between the work in this paper and [7] lay in the different approaches to game theory and what information is used for optimization. [7] only considers distance from the nearest node holding a replica, and the distance to the node holding the master copy. It does not consider access frequencies, or any cache replacement strategies, as in our work. Further the work in [7] expects a selfish behavior from the member nodes, whereas our work focuses on closed systems that will collaborate for caching.

The biggest difference between all schemes reviewed and the work in this paper can be summarized by the fact that they do not consider the temporal nature of access frequencies resulting in the loss of utility of a replica over time. Many algorithms cache a replica, and that replica is on the node forever, while others use an arbitrary cache invalidation scheme to remove some after a set time, which may or may not be appropriate. In this paper, we utilize both access frequency and update cost to make replication decisions using a game-theoretic approach for optimization.

III. Problem Formulation and Supporting Algorithms

Within the scope of this research we consider a network comprised of mobile nodes. These mobile nodes operate independently and network together via wireless radio, with broadcast radius r, in an ad-hoc fashion. This network organization is known as a mobile ad-hoc network, or *MANET*. A specific node, denoted M_k where k is the kth mobile node. A node M_i, is said to be in the set of one-hop neighbors of node k, N^k, if, for coordinates x and y, $\sqrt{(x_k - x_i)^2 + (y_k - y_i)^2} < r$. The mobile nodes in the network coordinate in the caching of data items which are evenly distributed to each node in order to collectively share the storage burden. When an specific node, M_k, needs a specific data item D_j, it first looks first to its local cache, denoted C^k, which is the set of data items cached at mobile node k. If $D_j \in C^k$, it is said to be a *local cache-hit*. If $D_j \notin C^k$, it is said to be a *local cache-miss*. In the case of a *cache-miss*, M_k will request D_j from the network. The items in C^k can be further classified as in either the set of replicas held at k, R^k, or in the set of primary/original data items held at k, P^k. For each node k, $C^k = R^k \cup P^k$. Each primary data item, D_j is assigned to a specific mobile node as its *Authoritative Control Node ACN*, defined as A_j. The *ACN* is the node responsible for writes to the primary data item.

A. Transitory Window of Interest

When considering information specific to a node within a *MANET*, we feel it is important to realize the scope of what information is useful. Given the dynamic topology of a *MANET*, node specific information has only temporal relevance and should be time boxed into a *Transitory Window of Interest*.

TABLE I
Symbol Definitions

Symbol	*Description*
M_k	The kth mobile node
D_j	The jth data item
N^k	The nearest neighbors of mobile node k
C_k	The cache of mobile node k
R_k	Replicas in cache at mobile node k
R_j^k	Replica of data item j at mobile node k
$\tau_{R_j^k}$	Age of replica j at mobile node k
P_k	Primary data items in cache at mobile node k
A_j	Authoritative control node for data item j
$\bar{\lambda_k}$	Transitory window of interest
H^k	Log of historical requests
H_j^k	Historical requests for data item j at node k

This window defines how far back, in the life of a node, that information is considered useful. We propose a correlary relationship exists between the average time that mobile nodes are in N^k to the time span of which historical information at M_k is considered relevant.

Proposition III.1. *A correlary relationship exists between the average time that mobile nodes are in N_k to the time span of which historical information at M_k is considered relevant.*

To begin to derive a calculation that represents the *transitory window of interest* for mobile node M_k, we start with the average time that one-hop neighbors remain within one-hop of the given node. We define this average time as $\tau_{\bar{N^k}}$. Since after $\tau_{\bar{N^k}}$ time, a moving node is still likely to be within two hops of a former neighbor, we define the *transitory window of interest* as $\bar{\lambda_k}$, where $\bar{\lambda_k} = 2\tau_{\bar{N^k}}$. It is important to note that $\bar{\lambda_k}$ time is relative to the node in question and that each node in the network can potentially have different $\bar{\lambda_k}$ times. We utilize the *transitory window of interest*, or $\bar{\lambda_k}$ time, throughout both the replication algorithm as well as supporting algorithms.

B. Adaptive Replica Deallocation

In order to increase data storage availability at each node, replicas will be deallocated after a certain amount of time. We proposed in Subsection III-A that node specific information has only temporal relevance due to the dynamic topology of a *MANET*. This led to the derivation of a *transitory window of interest*, or $\bar{\lambda_k}$ time. We would further propose that decisions made by use of $\bar{\lambda_k}$ time are also transitory, and should be considered temporally relevant.

Proposition III.2. *Decisions made through the use of a transitory window of interest, $\bar{\lambda_k}$ time, following Proposition III.1, are also transitory, and should be considered temporally relevant.*

Each mobile node, M_k, has a windowed history of responses to data requests that have either delivered directly to, or have been relayed through M_k. This historical set of request responses, denoted H^k, is limited by the *transitory window*

of interest, $\bar{\lambda_k}$ time. So only relevant historical information is considered. The subset of historical requests for data item j, D_j, is defined as H_j^k. Further, let $\tau_{R_j^k}$ denote the age, in time, of a replica of data item j at mobile node k.

Replication decisions, the process of which is forthcoming, are partially made through the evaluation of historical request information, H_j^k. Given Proposition III.1, these replication decisions are considered temporally relevant as well, indicating a decrease in the potential utility of a replica over time. Because of the spacial skew of requests for certain data items from certain geographic regions *zipf*[9][15][5], and considering node travel and the dynamic topology of a *MANET*, what may have been spatiotemporally near-optimal for replica placement at one point, may not be relevant placement at a later time. Given that we consider request history, H^k, to remain relevant for $\bar{\lambda_k}$ time, we propose that replicas should also have a base time to live, *TTL*, of $\bar{\lambda_k}$. The *TTL* of a given replica is the age at which it is deallocated.

However, we realize that the relevance of a replica at M_k increases with its utilization. Because of this, the utilization of a replica should correspondingly increase its *TTL*. Instead of setting a strict *TTL* = $\bar{\lambda_k}$, it is prudent to consider both $\bar{\lambda_k}$ time as well as the relative utility of a replica of j held at a node k, R_j^k, which adds an adaptive element to extend the base *TTL* if the utilization of a particular data item is relatively high. In our previous work [5], we developed a means by which to extend the *TTL* adaptively based on utility.

In order to set the stage for the adaptive *TTL* we first need to define a utilization ratio, U_{ratio} which will be used to help determine the adaptive portion. This U_{ratio} is defined in Eq. 1, where n_j^k is the number of requests for D_j in H_j^k.

$$U_{ratio} = \frac{n_j^k}{\tau_{R_j^k}} \tag{1}$$

We then take the product of the utilization ratio, U_{ratio} in Eq. 1 and the $\bar{\lambda_k}$ time, and add the standard *TTL*, $\bar{\lambda_k}$, to derive an adaptive time to live, *aTTL*, as defined in Eq. 2.

$$aTTL = \left(\frac{n_j^k}{\tau_{R_j^k}} * \bar{\lambda_k} \right) + \bar{\lambda_k} \tag{2}$$

If we define the *aTTL* of R_j^k as *aTTL(j, k)*, $U_{ratio} = 0.75$, and $\bar{\lambda_k} = 120$, we get $aTTL(j,k) = 210$, which when $210 - \tau_{R_j^k} = 0$, we deallocate R_j^k from C^k.

When a mobile node, M_k, is ready to deallocate a replica j, R_j^k, it first pings the authority control node, A_j, for data item k. If no acknowledgement is recieved from A_j within $\bar{\lambda_k}$ time, the safe leader election, detailed below, is initiated. If an acknolwedgement is recieved, the replica is deallocated. This is done to insure that data is not lost due to the authority control node holding the primary replica of the data item, going offline, running out of energy, or simply going off grid for an extended period of time.

C. Data Consistancy for Replicas

Every request to update a primary cache item, must be made at the authoritative node, A_j, for the D_j. Upon receiving an update request, the A_j node will process the request, and make a decision whether or not to schedule the transaction. If the transaction is scheduled, the A_j sends a cache invalidation request to the network holding a replica of data item D_j, $\forall M_k$ where $D_j \in R_k$, with instructions to stop serving the data item from cache until a replacement is sent. After the write transaction has executed, the A_j will send out a replica update request with the new data.

D. Safe Leader Election - ACN Continuity

Safe leader election, or *Authoritative Control Node* A_j continuity, is a fundamental scheme that assures a successor to A_j should it disconnect, go off network, or die. Every $\bar{\lambda_k}/2$ time, each M_k where $D_j \in R_j^k$ would send a message with a set of metrics, to A_j indicative of the health of the replicating node, how much it has been serving up R_j^k, remaining battery life, and the frequency of complete disconnection from the network. The A_j, upon receiving all of the metrics from all replicating nodes, would compute the list and pick an order list of best candidates to take over as the authoritative node should it go offline. The authoritative node would then broadcast out its successor list to the replicating nodes. In the event that the authoritative node should go offline, the replicating nodes would ping the successor list in order and the top one that responds would be the new authoritative node.

IV. Cooperative Altruistic Data Replication

Our replication scheme, CADR, is based upon the *Volunteer's Dilemma* problem in game theory, [8][13][1]. The *Volunteer's Dilemma* is a game in which a player make decisions that will benefit all, while taking on a small sacrifice with the expectation that other players will make similar decisions in the future. In our algorithm, mobile nodes found to be the globally optimum placement for a replica, become the "player" that takes the small sacrifice of decreased storage capacity in order to benefit the other mobile nodes through a reductionin both bandwidth and energy utilization. The CADR algorithm uses the following steps to help decide replica placement: calculation of a net global benefit, *NGB*, for each node in the return path, the determination of global optima by the recipient node, and the decision of what to do with a displaced replica in the event that the new replica has a greater NGB than an existing replica at the data item.

The calculation of *NGB* is a multi-step process. Net global benefit is derived by subtracting the global cost incurred from the global savings realized through the placement of a replica at the specified node. Global cost, GC is a metric comprised of both the network cost to update the replica and the network cost incurred from either displacing or deallocating an existing replica to make room for the new one. Global savings, GS, as mentioned earlier, is the total network savings that will be realized from replicating at that node, through the reduction in the retransmission of data items from the sourcing node to

the node under consideration for replica placement. Along the delivery return path of a requested data item from source node to requesting node, the global cost, GC and global savings, GS is calculated, and a net global benefit is determined for each node. That net *NGB*, along with the remaining global benefit of the other replicated items on the node, are added to the message being sent back to the requesting node.

Each node, M_k maintains a windowed history, $\bar{\lambda_k}$, of requests for data item j, H_j^k, either originating from or relayed through themselves. This *transitory window of interest*, $\bar{\lambda_k}$, is disucussed in detail within Subsection III-A. Within the *CADR* scheme, this window is used both as a base metric for replica deallocation and to keep the historical request data relevant.

The global savings, GS, is determined by first evaluating the request response history records. For each record in the history, H^k, for the given data item under consideration, H_j^k, the hop count from the data source to the node evaluating the history, M_k, is determined. The hop count for all records in the history for that node are summed and multiplied by the size of the data item. This gives an indication of how much total savings in energy and bandwidth will be realized from caching the data item at that particular node.

Given j and k, $\forall H_j^k$, let the global savings, GS, of replicating D_j at M_k, be defined as:

$$GS_j^k = \sum HC(H_j^k, M_k) * S_j \tag{3}$$

, where $HC()$ is a function returning the hop count from the response node in H_j^k to M_k.

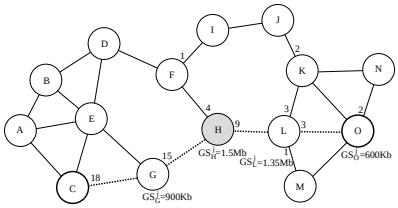

Fig. 1. MANET with Demand and Global Savings

Consider the mobile ad-hoc network pictured in Figure 1. In this network, mobile node O requested a specific data item. For this demonstration, assume the data item in question, D_j, has a size, S_j of $50Kb$. Data item D_j was found at M_C. M_C responded back with a response message containing the data item. When the message gets to M_G, it calculates a global savings of $900Kb$, which is the aggregate of all requests coming into the node, 15, plus 3 that originated at itself, times the size of D_j. Since the total requests coming through or from M_G is 18, the global savings is $900Kb$. Likewise, when M_H receives the response message containing the data item, and M_G's NGB contribution, M_H calculates its savings at $1.5Mb$, which is the aggregate of the 13 requests that mobile node H relay's through itself, plus the two that originate from mobile node H, times two hops to the source, since M_H is two hops from the source of the data, then multiplied by the size of D_j. The savings for each of mobile nodes L and O are calculated in similar fashion. The preliminary global optima for caching a replica is at mobile node H, followed by mobile node L.

The global cost, GC, is derived by first calculating how much it will cost the network to keep that data item updated when the authority node makes a change to the primary data item. This is determined by finding the hop count distance from the authority node to the node in question. That hop count is then multiplied by the size of the data item. Next, the network cost for the displacement or deallocation of an existing replica, on that mobile node, is calculated. To determine this cost metric, each replica is evaluated by looking at their stored NGB. The stored NGB is multiplied by a ratio which represents the amount of life left for that replica.

Let each replica of D_j have an update frequency ω_j in t time. Given D_j and M_k, let the global cost GC_k^j of replicating D_j at M_k be defined as:

$$GC_j^k = HC(A_j, M_k) * S_j * \lceil \frac{\bar{\lambda_k}}{\omega_j} \rceil + RGB_i^k \tag{4}$$

Where $HC()$ is a function returning the hop count from the A_j of D_j to M_k, $\lceil \frac{\bar{\lambda_k}}{\omega_j} \rceil$ gives the relative update occurances of D_j over the course of the base expected life *TTL*, or $\bar{\lambda_k}$ time, of the replica. RGB_i^k gives the remaining NGB_i^k for the replica of data item i at note k, R_i^k, which must be removed to make room for the replica of data item D_j, multiplied by the fraction representing the remaining *aTTL* for R_i^k. Let RGB_i^k be defined in 5

$$RGB_i^k = GB_i^k * (1 - \frac{R_i^k \tau}{aTTL(i,k)}) \tag{5}$$

Given the global savings, GS_j^k, and global cost, GC_j^k, let the net global benefit, NGB_j^k, for replicating D_j at M_k be defined as:

$$NGB_j^k = GS_j^k - GC_j^k \tag{6}$$

For a positive caching decision, $GS_j^k > GC_j^k$ must hold. If there is room in the cache of M_k for D_j then $RGB = 0$. If there is no room in the cache of M_k for D_j, then $S_j \leq S_i$ must hold true.

Consider the example used earlier with Figure 1 and the demonstration of the global savings, GS_j^k. The GS_j^H for replicating D_j at M_H was $1.5Mb$. Let us suppose that a candidate for replacement, R_i^H has an *aTTL* of 120, with a size, S_i, of $30Kb$, and has an age, $\tau_{R_i^H}$ of 100. Lets also assume that this particular candidate for replacement would have a remaining life ratio of $1-(100/120) = 1-0.83 = 0.17$. If the *Net Global Benefit*, NGB, value that was stored at the time of a caching decision for R_i^H at M_H was $600KB$, but the remaining life ratio is 0.17, then the RGB for the existing replica is $102Kb$. Consider the cost for updating the new replica of D_j at M_H is $4 * 2 * 50 = 400Kb$, because the data item gets updated twice every 60 time units with the *aTTL* of 120, two hops from M_C to M_H where $M_C = A_i$, times S_j. So the total global cost would be $400Kb + 102Kb = 502Kb$. Since $GC \approx 0.5Mb$ which is $< 1.5Mb$, the caching of the new replica would have a$1500Kb - 502Kb \approx 1.0Mb$ NGB.

Algorithm 1 Calculate NGB at each mobile node along the query return path. Make replication decision at request node.

$\{QFM =$ Query Found Message$\}$
$\{QRP =$ Query Return Path$\}$
if $D_j \in C_k$ **then**
  while $QRP > 1$ **do**
    $i = QRP[0]$
    $GS_j^i = \sum HC(H_j^i, M_i) * S_j$
    $GC_j^i = HC(A_j, M_i) * S_j * \lceil \frac{\bar{\lambda_i}}{\omega_j} \rceil$
    $NGB_j^i = GS_j^i - GC_j^i$
    if $NGB_j^i > 0$ **then**
      $QFM.NGBMatrix[i] \leftarrow NGB_j^i$
    end if
    $QRP - i$
    $QRP[0] \leftarrow QFM$
  end while
  $\{C =$ Candidate$\}$
  $\{CNGB =$ Candidate NGB$\}$
  for all $M_k \in QFM.NGBMatrix$ **do**
    if $QFM.NGBMatrix[M_k] > CNGB$ **then**
      $C = M_k$
    end if
  end for
  if $C \neq \emptyset$ **then**
    $C \leftarrow ReplicationRequest(D_j)$
  end if
end if

V. Simulation Environment/Parameters

For our experiment, the simulation tool developed in [5] was extended to this work. This environment implements a *Dynamic Source Routing*[6], *DSR*, protocol. Normal DSR routing is a two-phased process where a path search message goes out; returning the most efficient path, then the search request follows the path returned. However, our path discovery message contains the actual search request as a way to cut down on our message handling.

Some of the environmental variables which can be selected within the simulation tool include: query time, number of nodes, number of data items, size of cache, and run time. The simulator world space is set to either 1200m x 320m rectangle, as in [11] and [5], or 1200m x 640m. The space is also divided up into a grid of ten equal spaces which are utilized by the mobility and query algorithms.

A. Query Model

A Zipf data access pattern, similar to that found in [9][15][5], is used to build the query model. The Zipf calculation was implemented using the ten grids spaces mentioned above. Each grid space has been given a slightly different access pattern from that of its neighbor grid spaces, but a significantly different access pattern from a grid space on the other side of the world.

As in [15][5][9], the Zipf distribution has been skewed in order to create a more profound deliniation of the access pattern between grid spaces. By skewing the Zipf data access patterns, the access patterns produce a stronger likelihood of selecting certain data types for one grid space compared to another. Whichever grid space the mobile node is in, determines which access pattern they use when making a query. This simulates real-world situations where certain data demand is spatially skewed, while other data is equally queried.

B. Mobility Models

We utilized the same simulator framework as our work in [5]. In that work a modified random waypoint mobility model was developed which significantly reduced the central tendency of the traditional random waypoint scheme. Please see our work in [5] for details on our modified random waypoint mobility model.

C. Test Variables

For the purposes of the test, certain simulator variables were fixed and used repeatedly. The world size was fixed at 1200 by 320 meters or 1200 by 640 meters. The mobile node count was fixed at 100 with 300 data items available for primary caching. The wireless range, or r of the mobile devices was set to 100 meters. The total available cache size to hold data, S_j, was given a static 500KB *Kilobytes*. It is important to note that the total available cache size, S_j is not the total available data storage for the node, but simply the portion set aside for caching. The query time, or time between queries, was made random in selection from a set range around the selected query time for the test. The available query times used in the simulator include: 4, 6, and 8 seconds. The average data item size was varied for different tests and was selectable according to the relative size of the data item to the total available cache size. The relative size ratio's were: 10%, 15%, and 20%. Lastly, node failure was simuluated with settings of 0%, 5%, 10% and 20%. The bandwith of the mobile devices is fixed at 2Mbps and the total run time of each iteration of the experiment was 60 minutes; as listed in Table II.

TABLE II
Test Environment Variables

Variable	*Value(s)*
World Size	1200 x 320 meters, 1200 x 640 meters
Mobile Nodes	100
Wireless Range R	100 meters
Cache Size S_j	500KB
Data Items	300
Query Time Interval	4, 6, or 8
Bandwidth	2Mbps
Data Item Size Ratio	10%, 15%, 20%
Node Failure	0%, 5%, 10%, 20%
Run Time	60 minutes

VI. Experimental Results

A. Test Scenarios

Standard *Cooperative Caching*, *CoopCache* [10], *Tidal Replication*, *TR* [5], and *Data Replication Game*, *DRG* [7]

make up the comparison algorithms tested and compared alongside our proposed algorithm, *Cooperative Altruistic Data Replication*, *CADR*. These four algorithms were put through 72 different, independent test scenarios. Each test scenario represented a unique permutation of options for the four variable environmental settings: *Query Time*, *Data Item Size Ratio*, *Node Error Rate*, and *World Size*. The options for these settings are defined in Table II. Each scenario was executed ten times, for 60 minutes, for all four algorithms, resulting in 2,880 unique tests of 60 minutes, or 172,800 hours, with each testing algorithm seeing 43,200 hours of testing. Node placement, data item assignment, node movement, and query selection were random for each test, but consistant parameters were used for each test scenario.

B. Measurements

The primary goals of replication are to both reduce access time, increase data availability, and decrease the energy burden. Given those goals, query hop count was the primary consideration when evaluating the performance of each algorithm. Both energy savings and access time were also considered, but these measures can be, and were, derived directly from the hop count metric. Query error rate was also measured and evaluated along with a metric defining the percentage of queries answered with a replica to evaluate the data availability. The result summary is found in Table III

Mean Hop Count $\bar{HC}$: This is a metric which measures the mean number of nodes through which a requested data item must be retransmitted in order to reach the requesting node from the response node. This is one of our primary methods to determine the efficacy and performance of our algorithm versus the comparison algorithms.

Median Hop Count $\tilde{HC}$: Similar to $\bar{HC}$, except another indicator of the average hop count without the noise from larger outliers.

Hop Count Standard Deviation $\sigma_{\bar{HC}}$: This metric is an indicator of the strength of the $\bar{HC}$ statistic by showing how tightly the population fits around the mean.

Confidence Interval for Mean Hop Count $C_{\bar{HC}}$: Metric showing the bounds for the mean hop count at 95% confidence interval.

Mean Local Hit Ratio $\bar{LHR}$: This is a metric which gives an indication of how many queries were answered from a nodes local cache versus the node having the request the data from another node. Given the Zipf[11] query model utilized in the simulator, this is a strong indication of how effective replica placement was executed.

Mean Query Access Time $\bar{QAT}$: This ia a measure of the average (mean) time a query takes to arrive from source node to requesting node.

Mean Replica Utilization Ratio $\bar{RUR}$: This metric represents the ratio between the number of queries answered from a replicating node over the number of queries answered from a primary node.

Mean Query Error Ratio $\bar{QER}$: This metric gives insight into the effective placement of replicas through the showing and comparison of the reduction in query error for each tested algorithm versus the base cooperative caching algorithm.

TABLE III
SUMMARY TEST RESULTS

Metric	*CoopCache*	*TR*	*DRG*	*CADR*
$\bar{HC}$	6.20	4.14	3.74	**2.78**
$\tilde{HC}$	6.00	3.00	3.00	**2.00**
$\sigma_{\bar{HC}}$	3.69	3.50	3.29	**2.47**
$C_{\bar{HC}}\pm$	0.02	0.01	0.01	0.01
$\bar{LHR}$	0.01	**0.08**	0.06	**0.08**
$\bar{QAT}$	0.21	0.20	0.16	**0.14**
$\bar{RUR}$	0.0	0.62	0.56	**0.78**
$\bar{QER}$	0.25	0.18	0.17	**0.12**

C. Access Time

Access time is closely correlated to the hop count of the query response. To evaluate access time, we considered the mean/median hop count metrics from the various tests performed. Different variables such as data item size ratio, query time, and world size, were tested and evaluated for their effect on the performance of the algorithms. Though many permutations of different variable settings were tested, the primary settings used in most of our comparisons is: 4 second query time, 1200m x 640m world size, and 0.15 data item size ratio.

For our query response time metric, we derived the response time for each algorithm based off of the hop count for each answered query. To facilitate this derivation, we utilized the equation listed in Eq. 7, where H is the hop count for the query, $\bar{S_k}$ is the size of the data item (76KB avg), and B is the total available bandwidth of the mobile node: 2MB.

$$\bar{QRT} = (H * 0.03) + (H * \bar{S_k}/B) \tag{7}$$

When considering response time, *CADR* realized a 44% improvement in mean response time over *CoopCache*, 30% improvement over *TR*, and 25% improvement over the primary comparison algorithm *DRG*. The running *median* response time, over time, is shown in Figure 2

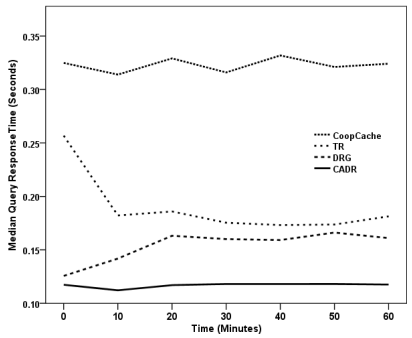

Fig. 2. Running Median Query Response Time. Settings: 1200x640 world size and .15 Data Item Size Ratio

When mean hop count was evaluated, *CADR* saw a 45% improvement over *Cooperative Cache*, 33% over *TR*, and

26% over *DRG*. While *CADR* performed very well against all comparison algorithms in the area of mean hop count, evaluation of the standard deviation of the mean hop count revealed the *CADR* algorithm to be even stronger. The mean hop count standard deviation's of *Cooperative Cache*, *TR*, and *DRG* were 3.69, 3.50, and 3.29 respectively, while the standard deviation for *CADR* measured at 2.47. We believe that this shows the performance of the replication strategy in *CADR* to be much stronger than the primary comparison algorithms, as it indicates a tighter pattern of query hop count results. The mean hop count measurements, along with the standard deviation metrics, for all tested algorithms, are illustrated in Figure 3.

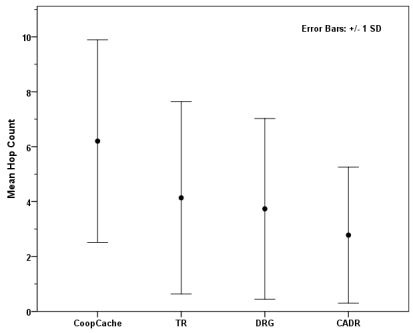

Fig. 3. Mean Hop Count w/Standard Deviation. Settings: 1200x640 world size and .15 Data Item Size Ratio

Next, the query results were evaluated considering the number of queries being answered from either a primary data source, or a replicating data source. By measuring the percentage of queries answered by either a primary source or replicating source, one can further reveal the strength of the replicating strategy as the more effective the replication strategy, the fewer queries are answered by a primary data item. The *TR* algorithm answered 38% of its queries from a primary source and 62% from a replicating source. *DRG* answered 45% and 55% from a primary and replicating source respectively. Finally, the *CADR* algorithm answered only 22% of its queries from a primary source and 78% from a replicating source. The primary/replica source statistics are illustrated in Figure 4.

An ineffective replication strategy will tend to place replicas more haphazardly in the network. An effective replication strategy, on the other hand, will place replicas where they will make the biggest impact, a spaciotemporally near-optimal placement. Given the nature of the geo-skewed query access pattern used in our simulator, found in the real world [15][9][5]: how hard the primary data sources must work in comparison with the replicating sources, is a huge indicator of the strategic strength of the replication algorithm.

The next indicator we considered was that of the average hop count for both a primary and replicating data source in comparison with the average hop count of all sources together. We believe these metrics further show the strength of the replica placement strategies that were tested. For this metric, we considered the median hop count. We used the median

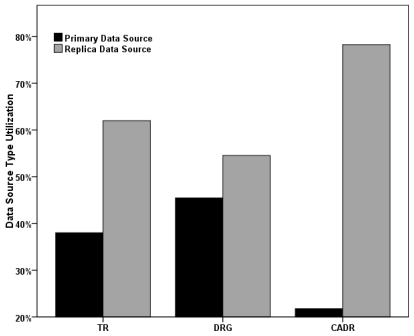

Fig. 4. Percent of Successful Queries from Primary or Replica Data Item. Settings: 1200x640 world size and .15 Data Item Size Ratio

hop count, as opposed to mean, as a way to eliminate the outlier noise and take a look at the averages from a different perspective. The most interesting outcome of this evaluation was in the fact that all three replicating algorithms had the exact same median hop count, 2, when it came to the replica data sources, yet the median combined hop counts of both *TR* and *DRG* measured at 3 and the median combined hop count for *CADR* measured at 2. The metric that seemingly made the difference was the median hop count measurements for the primary data sources. Both *TR* and *DRG* had a median primary source hop count of 5, while *CADR* measured at 3. We believe this is due to the fact that the *CADR* algorithm placed the replicas in a more balanced and globally optimal way.

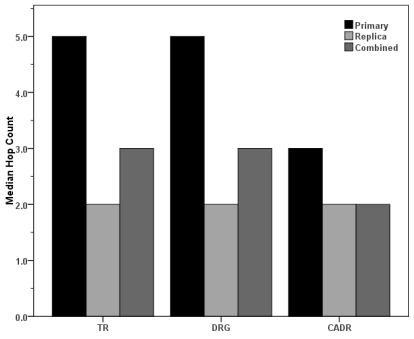

Fig. 5. Median Hop Count for 1) Primary Data Items, 2) Replica Data Items, and 3) Combined. Settings: 1200x640 world size and .15 Data Item Size Ratio

While the difference between the primary and replicating data source median hop counts of both *TR* and *DRG* were both 3, the difference between the primary and replicating data source averages for *CADR* was only 1. The median average hop count metrics for primary and replicating sources are shown in Figure 5. While evaluating this measurement, considering the previously mentioned primary/replica query count and standard metrics mentioned above, we believe the combined view shows that *CADR* places replicas in a considerably more optimal way than the comparison algorithms from a hop count/access time perspective.

In consideration of access time and hop count, the last set of metrics evaluated were the impact of both the data item size ratio as well as the node failure rate. Both were evaluated through all permutations of our tests. It was discovered that

while increasing the data item size ratio and/or the node failure rate does impact the performance of the algorithms, it does not signicantly alter their performance in relationship to each other. In other words, neither the data item size ratio nor the node failure rate exposed any inherent flaws or weaknesses in any of the algorithms such that it would cause one to be more performant than the other in certain circumstances.

D. Data Availability

The second major consideration when evaluating the efficiency of replication algorithms is that of data availability. Because of the transient nature of node relationships in a mobile network, as well as the tendency for the mobile network to become disjoint, data availability becomes an issue which replication can help mitigate. By optimally placing replicas closer to the areas of high demand, availability increases. The lower the query error rate, greater data availability is realized in the network. Greater data availability decreases overall realized access time and decreases the energy utilization as a result of fewer retries of failed queries.

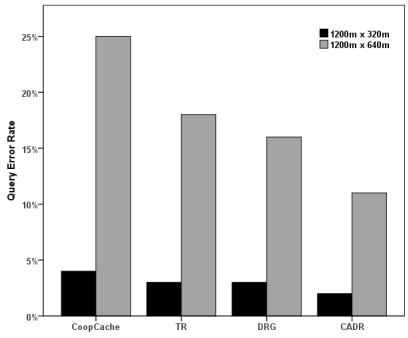

Fig. 6. Mean Query Error Rate by World Size

As shown in Figure 6, in a 1200m x 640m world space, *Cooperative Cache* had a high 25% query error rate, compared with a low 4% in a 1200m x 320m world space. *TR* had a query error rate of 3.5% in the smaller and 18% in the larger world space. *DRG* realized a 3.5% error rate in the smaller and 17% in the larger while *CADR* measured a 2.5% query error rate in the smaller world space and only 12% in the larger. The overall increase of query error rate in the bigger world space, 1200m x 640m, can be attributed to the greater tendency for the network to be disjoint, creating higher proportion of failed queries. Further, the more inefficient the replication strategy, the higher energy burden is produced, resulting in a greater node failure and higher query error rate.

E. Energy Efficiency

Lastly, energy efficiency, or gains thereof, is an important metric when considering the effectiveness of a replication algorithm. In Figure 7, the energy savings rate for each tested algorithm, in comparison with standard cooperative caching *CoopCache*, is shown for each of the three data item size ratio's used in the tests. This particular metric is another primary indicator of the efficacy of the replication strategy for the given algorithm. Higher energy savings translates into longer network up time due to the increased longetivity of each node. The more nodes that are alive and in the network, the lower the query error. Similarly, the lower the query error, the lower the access time. As you can see in Figure 7, *CADR* performed the best in all categories. The main data item size ratio used in our evaluation was 0.15. With a 0.15 data item size ratio, *CADR* saved an additional 16% more energy than *DRG*, with a total energy savings of 56%. This directly translates into a 16%, at minimum, increase in average node longevity.

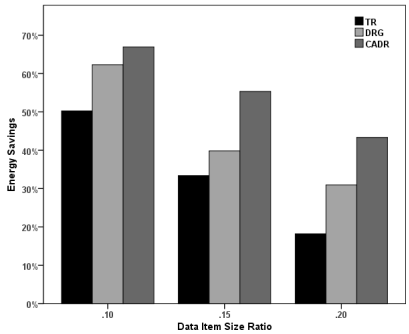

Fig. 7. Energy Savings by Data Item Size Ratio and Algorithm Settings: 1200x640 world size

The calculation used to determine energy efficiency is illustrated in Eq. (8) and derived from [14]. The Δ_{hopcnt} is calculated based on the mean hop count from traditional cooperative caching, *CoopCache*, and the mean hop count from the simulated algorithm. The result of the calculation in Eq. (8) is converted to mWh in order to derive the mean mWh used per query response. The mean mWh is the multiplied by the number of queries for each simulation, then compared with the energy used for [10] to arrive at an energy savings ratio.

$$En = \left((5.27 * 10^{-6} * 15000) + 266 * 10^{-6}\right) * \Delta_{hopcnt} \quad (8)$$

Looking towards energy efficiency as a measure of network reliability and longevity, we ran a seperate set of tests which measured the number of active nodes every twenty minutes for a run length of 6 hours. Each node was given one gigabyte of broadcast capability. As each nodes battery level was depleted, the node went offline. The results of this metric can be seen in Figure 8. In the chart, you will see that *CARD* lasted approximately 40 additional minutes than *DRG* and 90 additional minutes over standard *CoopCache*.

F. Scalability

A second set of tests were conducted in order to measure the scalability of *CADR* with different numbers of node counts. Both *DRG* and *TR* were tested along with *CADR* with 25, 100, and 250 nodes as well as both world sizes of 1200m x 320m and 1200m x 640m. It was discovered that *CADR* does scale well with different node population. However, it was found that while *CADR* outperformed *DRG* and *TR*, the delta hop count metrics between *DRG* and *CADR* decreases

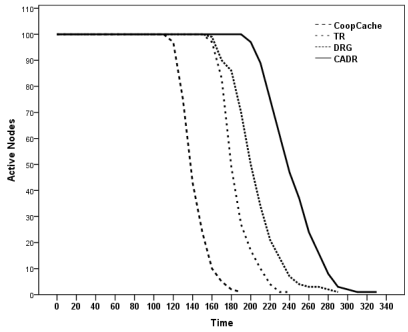

Fig. 8. Mobile Node Lifespan by Algorithm - Each node started with enough energy to broadcast $1GB$ of data. Settings: 1200x640 world size and .15 Data Item Size Ratio

significantly the more dense the network becomes relative to the world space. This can be attributed to a situation where the more dense a network is, the higher edge count, one-hop neighbors, per node, which results in a greater likelihood of queried data being closer. However, *CADR* did out perform, albeit on a smaller interval, all other tested algorithms despite changes to the network density. Due to space constraint, we have not further analyzed the results of our scalability study.

G. Result Summary

In summary, *CADR* performed extremely well under all test permutations and all comparison algorithms. *DRG* was the main algorithm our schemes were tested against, as it was also a game theoretic model. *CADR* saw a 25% decrease in response time, a 26% improvement in mean hop count and energy efficiency, and a 30% reduction in query error over *DRG*, all of which indicates a greater measure in the strategic placement of replicas. When compared with cooperative caching, *CADR* realized a 45% decrease in response time, a 46% improvement in mean hop count, a 56încreased energy efficiency, and a 48% reduction in query error.

VII. Conclusion

In this paper, we proposed a novel scheme for data replication in order to help mitigate the drawbacks inherent in a mobile environment such as high energy utilization, high access times, and low data availability. A game theoretic approach was detailed as a solution to decrease both the energy usage and access time, and increase data availability. Our cooperative game theoretic scheme took a volunteer's dilemma/reciprocal altruism approach that we have called *Cooperative Altruistic Data Replication* or *CADR*.

Our approach seeks to find the spatiotemporally local-optimal placement of replicas through the evaluation of the demand at nodes along the query response path. Nodes then make an altruistic decision to increase their own storage burden for the good of the network. As stated before, though our methods do not conform to the standard definition of game theory, they do represent a game-theoretical methodology through a dependence upon a utility function which has the goal of mutual optimization. In future work, we will take a more pessimistic/selfish approach towards the replication algorithms, outlined in this paper, through the introduction of an incentive model.

Our algorithms were thoroughly tested against three other similar replication algorithms under a wide variety of simulation parameters including: world size, data item size, query time, node count, and node failure ratios. Over all, *CADR* was found superior in energy conservation and data availability.

Our future work in this area will go two directions. One direction will look towards reducing the quantity of replicas created while increasing the usefulness of existing replicas through the application of our redistribution algorithm *Magnetic Distribution* found in [5]. The second direction will look towards adding an incentive model in an effort to move our scheme towards a more pessimistic/selfish game theory.

References

[1] M Archetti. Cooperation as a volunteer's delimma and the strategy of conflict in public. *The Journal of Evolutionary Biology*, 22(11):2192, 2009.

[2] N Chand, R.C. Joshi, and J Misra. Efficient cooperative caching in ad hoc networks. *Comsware 2006. First International Conference on Communication System Software and Middleware*, pages 1–8, 2006.

[3] Selim Gurun, Priya Nagpurkar, and Ben Y. Zhao. Energy consumption and conservation in mobile peer-to-peer systems. In *MobiShare '06: Proceedings of the 1st International Workshop on Decentralized Resource Sharing in Mobile Computing and Networking*, pages 18–23, New York, NY, USA, 2006. ACM.

[4] T. Hara and S.K. Madria. Data replication for improving data accessibility in ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(11):1515–1532, 2006.

[5] D. Hirsch and S. Madria. A resource-efficient adaptive caching scheme for mobile ad hoc networks. In *Reliable Distributed Systems, 2010. SRDS 2010. 29th IEEE Symposium on*, pages 64 –71, Oct-Nov 2010.

[6] D Johnson and D Maltz. Dynamic source routing in ad hoc wireless networks. *Mobile Computing*, pages 153–181, 1996.

[7] S.U. Khan, A.A. Maciejewski, H.J. Siegel, and I. Ahmad. A game theoretical data replication technique for mobile ad hoc networks. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1 –12, april 2008.

[8] Seungjoon Lee, Dave Levin, Vijay Gopalakrishnan, and Bobby Bhattacharjee. Backbone construction in selfish wireless networks. In *Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, SIGMETRICS '07, pages 121–132, New York, NY, USA, 2007. ACM.

[9] Christoph Lindemann and Oliver P. Waldhorst. Modeling epidemic information dissemination on mobile devices with finite buffers. In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 121–132, New York, NY, USA, 2005. ACM.

[10] Francoise Sailhan and Valerie Issarny. Cooperative caching in ad hoc networks. In *MDM 2003: Proceedings of the 4th International Conference on Mobile Data Management, 2003*, pages 13–28, 2003.

[11] Hara T. Shinohara, M. and S. Nishio. Data replication considering power consumption in ad hoc networks. In *MDM 2007: Proceedings of the 2007 International Conference on Mobile Data Management*, pages 118–125, Washington, DC, USA, 2007. IEEE Computer Society.

[12] Yi-Wei Ting and Yeim-Kuan Chang. A novel cooperative caching scheme for wireless ad hoc networks: Groupcaching. In *NAS '07: Proceedings of the International Conference on Networking, Architecture, and Storage, 2007*, pages 62–68, 2007.

[13] Jeroen Weesie. Cost sharing in a volunteer's dilemma. *The Journal of Conflict Resolution*, 42(5):600, 1998.

[14] O. Wolfson, Bo Xu, and R.M. Tanner. Mobile peer-to-peer data dissemination with resource constraints. In *MDM 2007: Proceedings of the 8th International Conference on Mobile Data Management, 2007*, pages 16 –23, May 2007.

[15] L. Yin and G. Cao. Supporting cooperative caching in ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(1):77–89, 2006.