Performance Comparison of Cache Invalidation Strategies for Internet-based Mobile Ad Hoc Networks *

Sunho Lim Wang-Chien Lee Guohong Cao Chita R. Das
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802.
{*slim, wlee, gcao, das*}*@cse.psu.edu*

Abstract

*Internet-based mobile ad hoc network (*IMANET*) combines a mobile ad hoc network (*MANET*) and the Internet to provide universal information accessibility. Although caching frequently accessed data items in mobile terminals (MTs) improves the communication performance in an* IMANET*, it brings a critical design issue when data items are updated. Prior cache invalidation strategies, proposed for cellular networks, are not suitable for an* IMANET*, where frequent link disconnection and changing network topology due to mobility of MTs can cause serious cache consistency problem. In this paper, we analyze several push and pull-based cache invalidation strategies for* IMANETS*. A global positioning system (GPS) based connectivity estimation (GPSCE) scheme is first proposed to assess the connectivity of an MT for supporting any cache invalidation mechanism. Then, we propose a pull-based approach, called aggregate cache based on demand (ACOD) scheme, to find the queried data items efficiently. In addition, we modify two push-based cache invalidation strategies, proposed for cellular networks, to work in* IMANETS*. These are a modified timestamp (MTS) scheme, and an MTS with updated invalidation report (MTS+UIR) scheme. Simulation results indicate that our proposed strategy provides high throughput, low query latency, and low communication overhead, and thus, is a viable approach for implementation in* IMANETS*.*

Keywords— Aggregate Cache, Cache invalidation strategy, Internet-based mobile ad hoc networks (IMANETs), Global positioning system (GPS), Search algorithm.

1. Introduction

Internet-based Mobile Ad hoc Networks that combine the wired Internet and wireless *Mobile Ad hoc Networks* (MANETS), called IMANETS [3], are emerging to provide a ubiquitous communication infrastructure. In an IMANET, a *mobile terminal*[1] (MT) can connect to the Internet directly through selected access points (APs) or indirectly through message relay via other MTs in wireless MANETS (e.g. IEEE 802.11 in ad hoc mode). An IMANET is well suited for many Internet applications in terms of facilitating flexible accessibility and information availability. However, frequent link disconnections and changes of network topologies (due to mobility of users) may reduce the information accessibility and increase the communication latency.

Caching frequently accessed data items in MTs is an effective techniques to enhance the communication performance in IMANETS. In the previous work [9], we proposed an *aggregate cache* for IMANETS where, by storing the data items in the local caches of the MTs and making them available to other MTs, members of the IMANET can efficiently access the data items. Thus, the aggregated local cache of the MTs can be considered as a unified large cache for the IMANET. By using a data item cached in the aggregate cache, we can reduce the average access delay, and thus, save the scarce wireless bandwidth. However, a critical design issue, *cache invalidation*, needs to be addressed for applications requiring data consistency with the server. When a data item in a server is updated, it is necessary to make sure that the cached copies of this data item are validated before they can be used.

The cache invalidation problem has been well studied in mobile computing environments [1, 2, 6, 7, 8, 14]. However, these studies are all limited to cellular wireless net-

* This research was supported in part by NSF grants CCR-9900701, CCR-0098149, CCR-0208734, and EIA-0202007.

1 In this paper, we use the term mobile terminal (MT) to refer to a portable device (e.g. a laptop computer, a personal digital assistance (PDA), a mobile phone, a handheld computer, etc) or a person who carries it.

works in which, the MTs only communicate with the base stations (BSs) but not with each other. For MANETS, Hara [5] suggested a replication scheme for periodically updated data items, based on the previously suggested replica allocation method [4]. Replicated data items are reallocated periodically depending on the access frequency, remaining time to next update, and network topology. However, estimation of optimal reallocation period and remaining time to the next update are not practically feasible. In addition, for the IMANET environment, Papadopouli et al [12] proposed the 7DS architecture, in which several protocols are defined to share and disseminate information among users. Unlike our work, they mainly focus on data dissemination, and thus, cache invalidation and cache update issues are not explored. For a comprehensive survey, please refer to [10].

Our study of cache invalidation in an IMANET environment is motivated by the following observations: 1) the communication model in an IMANET is different from that in the cellular environment - in an IMANET, a multi-hop communication is employed; 2) the cost for a broadcast in an IMANET is different from that in the cellular environment - a broadcast may lead to flooding of the whole network; 3) the connectivity to a server is less stable than that in the cellular environment - due to the user mobility, an MT may be disconnected or isolated from the server or other MTs. Thus, the cache invalidation problem in an IMANET is unique from that in a cellular network due to multi-hop message relay, operation cost model, and uncertainty in message delivery.

In this paper, we examine several push and pull-based cache invalidation schemes for IMANETS. In order to adapt and enhance these schemes for IMANETS, we propose a scheme, called *global positioning system-based connectivity estimation (GPSCE)*, for assessing the connectivity of an MT to a server (i.e., AP). With this enhancement, an MT can check whether it can access a server directly or indirectly through a multi-hop relay. In addition, we extend the search protocol used in [9] to better meet the requirements of cache invalidation schemes. With GPSCE and the modified search protocol, we adapt three existing cache invalidation schemes for IMANETS: the first one, called *aggregate cache-based on demand (ACOD)*, is based on the *pull* strategy; while the other two, namely *modified timestamp (MTS)* and *MTS with updated invalidation report (MTS+UIR)* are based on the *push* strategy.

We conduct simulation-based performance evaluation using the *ns-2* package to observe the impact of query interval, cache update interval, aggregate cache size, and the search algorithm on system performance. The results indicate that the pull-based ACOD strategy provides high throughput, low query latency, and low communication overhead. The revised search protocol, applicable to all cache invalidation schemes, helps in reducing the number of communication messages significantly. The overall results show that the ACOD cache invalidation scheme is superior in all aspects, and thus, is a viable approach for implementation in IMANETS.

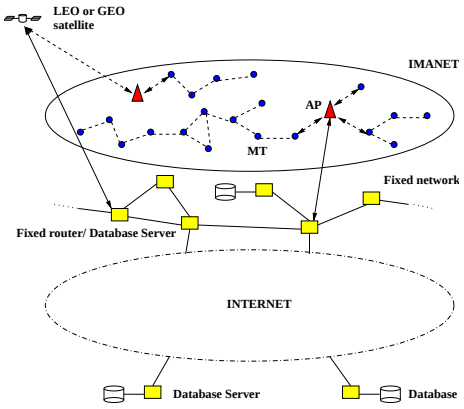


Figure 1. A system model of IMANET.

The rest of paper is organized as follows. The system model is introduced in Section 2. A GPS-based connectivity estimation scheme is described in Section 3. The search algorithm and three cache invalidation schemes are presented in Section 4. Section 5 is devoted to performance evaluation and the conclusions are drawn in the last Section.

2. System Model

As illustrated in Figure 1, an IMANET consists of a set of MTs that can communicate with each other using an ad hoc communication protocols (illustrated by the dashed-line). An MT can move in any direction and make information search and access requests from anywhere in the covered area. Among the MTs, some of them can directly connect to the Internet, and thus serve as APs[2] for the rest of MTs in the IMANET. An MT located out of the communication range of the AP has to access the Internet via relays through one of the APs. Thus, an AP is a gateway for the Internet and is assumed to have access to any information. An AP is located in a communication area and is connected to a database server[3]. Thus, an MT can not only connect to the Internet, but also can forward a message for communication with other MTs via a wireless LAN (e.g. IEEE 802.11), as used in most recent studies [9, 11, 12].

2 An AP here is a logical notation. An AP equipped with appropriate antennas can directly communicate with the Internet through wireless infrastructures including cellular base stations (BSs), Low Earth Orbit (LEO), or geostationary (GEO) satellites.

3 From an MT's point of view, since an AP is transparent to a database server, we use the terms AP and a database server (later in short, server) interchangeably.

A database may be attached to an AP, a fixed router or a database server. We assume that the database consists of a total of n data items (DB_{max}). A data item (d) is the basic unit of an update or query operation. The database is only updated by the server, while a query is generated by the MTs for read-only requests. An MT can cache a limited number of data items since the size of the cache space in an MT is relatively small compared to the total number of data items in the database. Depending on the cache invalidation strategy, a server can broadcast the list of *ids* of updated data items to the MTs to ensure data consistency, either synchronously (periodically) or asynchronously.

3. A GPS-based Connectivity Estimation Scheme

In an IR-based cache invalidation scheme for cellular networks [1, 2, 6, 7, 14], even if a query can be answered by a cached data item, an MT always waits for the next IR. Then it invalidates the cached data items accordingly, and answers the query if the cached copy is still valid. Otherwise, the MT sends a request to the server for a valid copy. Here, the basic assumption is that an MT is supposed to receive an IR regularly. However, in an IMANET, an MT may not receive an IR due to link disconnections and time-varying network topologies. Without knowing whether an MT can receive an IR from the server directly or indirectly through multi-hop relays, it may wait for the next IR, which may not arrive at all. It implies that, unlike a cellular network, connectivity with the server is not always assured.

To address this issue, we propose a *GPS-based connectivity estimation (GPSCE)* scheme, which is used as a base algorithm for supporting any cache invalidation mechanism in IMANETS. Thus, we assume that an MT is equipped with an on-board GPS receiver and is aware of its current location. In this scheme, by using the location information obtained by the GPS, an MT can know whether it is currently connected to a server. When an MT is unable to send (receive) a message to (from) a server, a query cannot be processed, because an MT neither can check the validity of a cached data item nor can receive a valid data item from a server. Once the MT is aware of its status, it can decide what operation should be followed, such as wait for an IR, send a request message, or deny a query.

3.1. The Proposed GPSCE Scheme

Using the GPS, we develop a connectivity estimation algorithm, called GPSCE. This algorithm estimates the maximum communication period of an MT, which is located a single-hop or multi-hop away from a server, based on the distance. Then, the MT is aware of its communication status to a server.

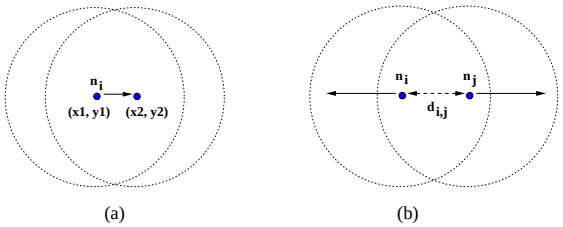


Figure 2. A vector of n_i in (a), and the distance between n_i and n_j in (b).

The algorithm works as follows: Let us assume that an MT i (n_i) samples its location periodically. In Figure 2(a), the locations of n_i at time t and $t + \Delta t$ are $i(t)$ = (x_1, y_1) and $i(t + \Delta t)$ = (x_2, y_2), respectively. Based on these two locations, a vector of n_i is given by

$$\vec{i} =< x_2 - x_1, y_2 - y_1 > . \quad (1)$$

By using the vector $\vec{i}$, we can predict a new location of n_i after time t' by $i(t') = (a_i \cdot t' + x_i,\ \ b_i \cdot t' + y_i)$, where a_i (= $x_2 - x_1$) and b_i (= $y_2 - y_1$) are the x and y vectors of n_i, respectively.

Let us assume that the current locations, and the vectors of two MTs n_i and n_j are $i = (x_i, y_i)$ and $\vec{i} =< a_i, b_i >$, and $j = (x_j, y_j)$ and $\vec{j} =< a_j, b_j >$, respectively. In Figure 2(b), if n_i and n_j move in opposite directions, they will be out of each other's communicate range. To estimate the period in which both n_i and n_j are located within each other's communication range, first we calculate the distance between them. The new locations of n_i and n_j after time t' are $i(t') = (a_i \cdot t' + x_i,\ \ b_i \cdot t' + y_i)$ and $j(t') = (a_j \cdot t' + x_j,\ \ b_j \cdot t' + y_j)$. Thus, the distance between them, $d_{i,j}$, is given by

$$d_{i,j} = \sqrt{((a_j - a_i) \cdot t' + (x_j - x_i))^2 + ((b_j - b_i) \cdot t' + (y_j - y_i))^2}. \quad (2)$$

During communication, both n_i and n_j should be located within each other's communication range (R). Here, we assume that all the MTs in a network use the same communication range, which is already known to them. Thus, $d_{i,j}$ should satisfy the following condition.

$$d_{i,j} \leq R. \quad (3)$$

Since $d_{i,j}$ is positive, we have $d_{i,j}^2 - R^2 \leq 0$. From this, we can estimate the maximum period ($P > 0$) in which both n_i and n_j are located within each other's communication range as

$$\begin{aligned} P &= \frac{-e + \sqrt{e^2 - 4 \cdot c \cdot h}}{2 \cdot c}, \quad (4) \\ \text{where} \quad c &= (a_j - a_i)^2 + (b_j - b_i)^2, \\ e &= 2 \cdot ((a_j - a_i) \cdot (x_j - x_i) + \\ &\quad (b_j - b_i) \cdot (y_j - y_i)), \text{ and} \\ h &= (x_j - x_i)^2 + (y_j - y_i)^2 - R^2. \end{aligned}$$

Based on the above equation, an MT n_i that is a single-hop away from a server can obtain the P_i to the server. Once n_i knows the P_i, it can calculate the connection expire time (T_i), the time when it is out of the communication range of a server. This is given by

$$T_i = t_{cur} + P_i, \tag{5}$$

where t_{cur} is the current time. For the MTs, which are multi-hop away from a server, let us assume that an MT n_j accesses the server by relaying a message through n_i (refer to Figure 3). Also, T_i and T_j are the connection expire times between the AP and n_i, and n_i and n_j, respectively. If any one of these connections is broken, n_j loses its connection to the server. Thus, we choose the minimum value of T, and T_j is given as

$$T_j = \text{Min}(T_i, T_j). \tag{6}$$

Once n_j calculates T_j, it can decide whether it is currently connected to a server. In this paper, the value of T is used as an initial condition to proceed with a query for any cache invalidation strategy.

3.2. A Communication Protocol for the GPSCE Scheme

In this subsection, we describe the communication protocol of the GPSCE scheme. When the connection expire time is estimated, a quintet ([id, l, v, h, t_{event}]), called a *mobility quintet* (Q), is attached to a message. Here id is the identification of a server or an MT. l and v are the current location and vector, respectively. h is the number of hops from a server, and t_{event} is the time when a message is sent. Each MTs keeps a status bit (f) to indicate whether it is accessible to a server. Initially, all the MTs set f to 0.

We now describe three situations to estimate the communication period; when an MT is single-hop away from a server, multi-hop away from a server, and when an MT changes its direction. First, we explain the case when an MT n_i is at a single-hop from a server.

Single-hop Protocol

1. When f_i is 0, n_i broadcasts an one-hop *hello* packet to the adjacent MTs, periodically.
2. When a server, r, receives a *hello* packet, it replies a *hello_ack* packet with a quintet, Q_r = [id_r, l_r, v_r, h_r, t_{event}]. Here, h_r is 0.
3. When n_i receives the *hello_ack* packet, it calculates P_i based on the received Q_r (l_r and v_r), and its current location and vector by using Eqs. 2,3, and 4 . Then n_i obtains T_i using Eq. 5, sets f_i to 1, and caches Q_r. Also, n_i sets h_i to 1 by increasing h_r.

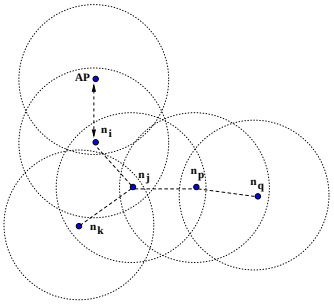


Figure 3. A sample network topology.

Note that if n_i does not move ($v_i = 0$), T_i is infinite. After setting f_i to 1, n_i stops broadcasting the *hello* packet. To keep a seamless connection with a server, we proactively broadcast a *hello* packet before t_{cur} becomes T_i. Otherwise, the MTs cannot response during the $hello$ message interaction to estimate a new T, even though they are connected to a server. When n_i receives a *hello_ack* packet, it repeats step 3. If n_i does not receive a packet within a specified timeout period, n_i sets f_i to 0 and initiates to broadcast a *hello* packet periodically.

Next, we describe the protocol when an MT is multi-hop away from a server. Let us assume that an MT n_i and the server are within each other's communication range ($f_i = 1$), as shown in Figure 3, and n_j tries to decide whether it is accessible to the server.

Multi-hop Protocol

1. When f_j is 0, n_j broadcasts a *hello* packet.
2. When n_i receives a *hello* packet from n_j, it checks the f_i value to decide whether it can reply. Since f_i is 1, n_i replies a *hello_ack* packet including both Q_i and T_i to n_j. If f_i is 0, then n_i ignores the *hello* packet.
3. When n_j receives a *hello_ack* packet from n_i, it calculates T_j in the same manner described in the single-hop protocol. Then n_j finds the minimum value of T_j by using Eq. 6 (T_j = Min(T_i, T_j)), caches the *hello_ack* packet sender's quintet (Q_i), sets f_j to 1, and stops broadcasting the *hello* packet.

Since n_j accesses the server through n_i, if n_i is out of the communication range of the server, n_j can no longer communicate with the server regardless of its connectivity with n_i. Also, n_j may receive multiple *hello_ack* packets, but it chooses the packet in a first come first serve (FCFS) manner.

Finally, we describe the protocol when an MT changes its direction that leads to the change of the position vector, and its connection expire time. When an MT is a single-hop away from a server, it recalculates T based on the server's

location and vector that are fixed values, and its new location and vector. When an MT is multi-hop away from a server, it should recalculate T and broadcast a message to the MTs that access the server through it. Let us assume that n_i, n_j, n_k, n_p and n_q have connection with a server directly or indirectly through multi-hop relays (refer to Figure 3). Assume that, n_j changes its direction.

1. When n_j changes its direction, it updates its position vector and recalculates T_j. To recalculate T_j, n_j requires the location of n_i that replied a *hello_ack* packet before. For this, we utilize the Q_i cached in n_j to predict the location of n_i. Since n_j knows the vector and location of n_i at t_{event}, it can estimate the current location of n_i at t_{cur} ($i(t_{cur}) = (v_{i,x} \cdot (t_{cur} - t_{event}) + l_{i,x}$, $v_{i,y} \cdot (t_{cur} - t_{event}) + l_{i,y})$, where $l_{i,x}$ and $l_{i,y}$, and $v_{i,x}$ and $v_{i,y}$ are the location and vector of n_i, respectively.). Based on the new location of n_i, n_j can recalculate T_j by using Eqs. 2, 3, and 4. Then n_j broadcasts an one-hop *loc_update* packet, attached with the updated Q_j to the MTs that rely on accessing the server through n_j (e.g. n_k, n_p and n_q).
2. When n_k receives a *loc_update* packet from n_j, it compares the id of the packet sender with the id cached in the quintet. If it matches, n_k recalculates T_k based on the received quintet by using Eqs. 2, 3, and 4. Then n_k chooses the minimum value (T_k = Min(T_j, T_k)), and caches the received quintet. Similarly, n_p and n_q update their Ts. If the id does not match, the *loc_update* packet is ignored.

When n_i receives a *loc_update* packet, it ignores the packet because any direction change does not affect n_i's connectivity to the server as long as it is in the communication range.

4. Cache Invalidation Strategies for IMANETs

In this section, we first discuss a *modified simple search (MSS)* algorithm for implementing the invalidation policy. Then we present the three cache invalidation schemes.

4.1. A Modified Simple Search Algorithm

In an IMANET, the data items from a server may be cached in some of the MTs. Thus, a search scheme needs to be deployed for finding a data item whether from the server or from an MT. In this paper, for efficient search of queried data items, we propose a *modified simple search (MSS)* algorithm that is an extended version of a *simple search (SS)* scheme [9], in which the cache update is not considered. To reduce the query delay and number of communication messages, we use an aggregate cache concept in which a data item can be received from either the local cache of the MTs or from a server.

For implementing a cache invalidation scheme, when a query is generated which can be answered by a cached data item in an MT, the MT sends a message to a server to verify whether the cached data item is a valid copy. If the data item is invalid, the MT receives the valid copy from the server and uses it. Basically, the SS algorithm broadcasts a request to all neighboring MTs for a response. Since the cache update is only received from a server, in the MSS scheme, the broadcasted requests are sent only to the MTs, which are along the path closer to the server. This is different from the SS scheme, where a request is broadcasted to all neighboring MTs irrespective of the direction. The MSS scheme is thus a directed broadcast, and the direction can be obtained from the hop count, encoded in a packet. The MSS scheme can be implemented using any existing routing protocol. The detail steps are described below. Let us assume that an MT n_p generates a request for a data item as shown in Figure 4.

1. When a query is generated, n_p checks its local cache. If the queried data item is available, n_p immediately broadcasts a *request* packet to a server. A *request* packet contains the requester's id (id), id of the queried data item (d_s, where $0 \leq s < DB_{max}$), the most recently updated timestamp of d_s (t_s), and number of hop (h) from the server.
2. When n_j receives the request, it checks its local cache. If the queried data item is cached, n_j compares its updated timestamp with t_s attached in the packet. n_j updates t_s with the most recent value, and attaches its id to indicate that it has a cached copy. Then n_j forwards the request to the server to check the validity of the data item.
3. When n_q receives the *request* packet, it compares the current number of hops from the server with h attached in the packet. Since n_q is further away from the server, it does not forward the packet, and discards it. Thus, the MSS algorithm attempts to find the shortest path to the server.
4. When a server receives the *request* packet, it compares the updated timestamp of the data item stored in the database with t_s, attached in a packet. The server replies with an *ack* packet attached with the response (either the updated data item or a valid bit telling that the data item is valid in the cache) to n_j, which subsequently sends it to the requester.
5. When n_p receives an *ack* packet, it uses the cached data item to answer the query, if it has a valid copy. Otherwise, it sends a *confirm* packet for a valid data item to n_j. Here, n_p may receive multiple *ack* packets, but it

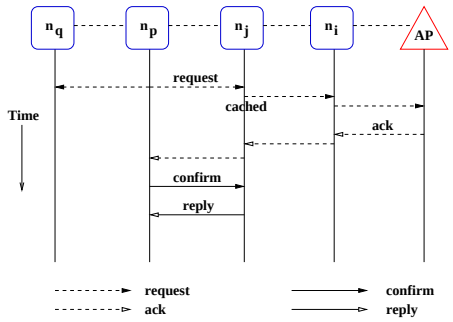


Figure 4. A *modified simple search (MSS)* algorithm for querying data items in the IMANET.

chooses the packet which arrives first. Since the packets are forwarded through different paths, n_j selects the path for a *confirm* packet based on the first receipt of the *ack* packet, and discards rest of the *ack* packets.

6. When n_j receives a *confirm* packet, it sends a *reply* packet to the *confirm* packet sender using the known route with the cached data item, which is already verified by a server. When a server receives a *confirm* packet, it attaches the valid data item to the *reply* packet.

When an MT forwards a *request* packet, the id of the MT is appended in the packet to keep the route information. Also, when an MT receives multiple copies of a *request* packet, due to broadcasting, it only processes one of them and discards the rest. For an *ack*, *confirm*, or *reply* packet, the MT also checks if its id is included in the path, which is appended to the packet. Since these packets are supposed to travel only along the assigned path that is established by the *request* packet, if the MT's id is not included in the path, the packet is discarded. When the server receives a *request* packet, it does not send the data item immediately, but sends an *ack* because other MTs, which are located closer to the sender might cache the valid data item. This helps in reducing network congestion and bandwidth consumption by multiple replies.

4.2. An Aggregate Cache based On Demand (ACOD) Scheme

In contrast to the prior push-based schemes [1, 2, 6, 7, 8, 14], where the server periodically/non-periodically broadcasts a message, we propose a *pull-based* approach. In this approach, whenever a query is generated, an MT sends a message to a server to verify a cached data item before it is used for answering the query.

The basic mechanism is described as follows. When a query is generated, an MT checks its f value, which is obtained by the GPSCE scheme. If f is 0, which implies that the MT cannot access a server, the query fails. If f is 1, the MT checks its local cache, and broadcasts a *request* packet by using the proposed MSS algorithm. When an MT receives the request, it checks its f value. If f is 1, the MT checks its local cache, and forwards the packet to the MTs closer to the server. Thus, the sender receives either an acknowledgment from a server to indicate that its cached data item is valid or the queried data item from a server or an adjacent MT, who has the valid data item. The pseudo code for the ACOD scheme is given in Figure 5.

Notations:
d, t: A data item and its timestamp, respectively.
C_p: A local cache in MT n_p.
d^c: A cached data item.
t^c: A timestamp of the cached data item d^c.

(A) When n_p generates a query q for a data item d,
if (f_p == 1) {
if ($d \in C_p$) { /* the data item is locally cached. */
Attach id of n_p, id of d, and t^c to a *request* packet;
Broadcast the *request* packet by using the MSS algorithm to the server/other MTs to validate the data item;
}
else /* the data item is not locally cached. */
Broadcast a *request* packet by using the MSS algorithm to the server/other MTs for the data item;
}
else
Fail to answer the query;

(B) When n_j (f_j = 1) receives a *request* packet for data item d,
if ($d \in C_j \ \wedge \ t < t^c$) {
Attach id of n_j, and update t with t^c to the *request* packet;
Rebroadcast the *request* packet by using the MSS algorithm to the server/other MTs to validate the data item;
}
else
Rebroadcast the *request* packet by using the MSS algorithm to the server/other MTs for the data item;

Figure 5. The pseudo code of the ACOD scheme.

4.3. A Modified Timestamp (MTS) Scheme

In the TS scheme [1] proposed for cellular networks, a server periodically broadcasts an IR that carries the current timestamp (t_{cur}) and a list of tuples (d_s, t_s) such that $t_s > t_{cur} - w \cdot L$, where d_s is a data item id and t_s is the most recently updated timestamp of d_s ($0 \leq s < DB_{max}$). Here, w is an invalidation broadcast window and L is the broad-

cast interval. If an MT is in the active mode, it receives the IR and invalidates its cached data items accordingly. If an MT is disconnected or is in the sleep mode for more than $w \cdot L$, it invalidates the entire cached data items because it cannot decide which data items are valid. In this paper, we do not consider the long disconnection problem for the sake of simplicity.

We modify the MTS scheme for IMANETS. The basic mechanism is described as follows. When a query is generated, an MT checks its f value. If f is 1, then the MT either waits for the next IR, or broadcasts a *request* packet by using the MSS algorithm. After the MT receives the IR, it invalidates the cached data items accordingly, and forwards the IR to adjacent MTs. If a queried data item is still valid, the MT uses it to answer the query. Otherwise, the MT broadcasts a *request* packet. When an MT receives a *request* packet, it forwards the packet to a server rather than waiting for the next IR. Although an MT can reply an *ack* packet while waiting for the next IR, it forwards the packet to a server. Otherwise, the query delay may increase. Here, we use a hop limit for an IR to prevent flooding of the IR packets over the network. An MT does not rebroadcast an IR, when it already has received the same IR, or the number of forwarded hops of the IR exceeds the hop limits. The pseudo code for the MTS scheme is shown in Figure 6.

A major drawback of the MTS scheme is the unavoidable query delay due to periodic broadcast of IRs.

4.4. An MTS with Updated Invalidation Report (MTS+UIR) Scheme

This scheme is the same as the MTS protocol except that an *updated invalidation report (UIR)*, which is partially derived from [2], is added. An UIR packet contains a timestamp of the last broadcasted IR from a server (t_{last_bcast}), and a list of tuples (d_s, t_s) that have been updated after the last IR has been broadcasted, where $t_s > t_{last\_bcast}$. The server inserts a number of UIRs into the IR intervals. Thus, an MT can process a query after receiving either an IR or an UIR. In this paper, we use four UIR replicates within each IR interval.

The MTS+UIR scheme follows the same procedure as the MTS scheme except followings. When an MT receives an UIR, it compares $t^c_{last\_bcast}$ saved in the local cache with t_{last_bcast} received from the UIR. If $t^c_{last\_bcast} > (t_{last\_bcast}$ - $L)$ is true, the MT invalidates the cached data items accordingly, and answers the query or sends a *request* packet using the MSS algorithm. Otherwise, the MT ignores an UIR and waits for the next IR.

While this scheme reduces the query delay compared to the MTS scheme, it has higher message overhead than that of the MTS scheme due to additional UIR broadcasts.

Notations:
t_{cur}: Defined before.
Q_p: A query queue in MT n_p.
t_{last_bcast}: A timestamp of the last arrival time of IR.

(A) When n_p generates a query q for a data item d,
if (f_p == 1) {
if ($d \in C_p$)
Queue d into Q_p and wait for next IR;
else
Broadcast a *request* packet by using the MSS algorithm to the server/other MTs for the data item;
}
else
Fail to answer the query;

(B) When n_p (f_p = 1) receives an IR,
if ($t_{last\_bcast} < t_{cur} - w \cdot L$)
Invalidate the entire cached data items;
else {
for $\forall d \in$ IR **do** {
if ($t^c < t$)
Invalidate d^c;
else
$t^c = t_{cur}$;
}
}
for $\forall d \in Q_p$ **do** { /* dequeue the data item. */
if ($d \in C_p \ \wedge \ d^c$ is marked as valid)
Use d^c to answer the query;
else
Broadcast a *request* packet by using the MSS algorithm to the server/other MTs for the data item;
}
$t_{last\_bcast} = t_{cur}$;
Rebroadcast the IR;

(C) When n_j (f_j = 1) receives a *request* packet for the data item d,
if ($d \in C_j \ \wedge \ t < t^c$) {
Attach id of n_j, and update t with t^c to the *request* packet;
Rebroadcast the *request* packet by using the MSS algorithm to the server/other MTs to validate the data item;
}
else
Rebroadcast the *request* packet by using the MSS algorithm to the server/other MTs for the data item;

Figure 6. The pseudo code of the MTS scheme.

5. Performance Evaluation

5.1. Simulation Testbed

We use a discrete event network simulator *ns-2* with the CMU's wireless extension, to conduct the experiments. The radio model simulates Lucent's Technologies WaveLAN [13] with a nominal bit rate of 2Mbps and a nominal range of 250m. The radio propagation model is based on the free-space model. For the link layer, the IEEE 802.11 MAC protocol and the distributed coordination function (DCF) are used.

To examine the proposed idea, we use a 2000m $\times$ 750m rectangular area. We assume that an AP is located in the

Parameter	Values
Number of MTs	50
Number of APs	1
Database Size (items)	1000
Cache size (items/MT)	10 to 100
Data item size (KByte)	10
Mean query interval time (sec)	5 to 200
Mean update interval time (sec)	10 to 1000
Hot data items	1 to 50
Cold data items	remainder of DB
Hot data item update prob.	0.8
Broadcast interval (sec)	20
Broadcast window (w)	10 intervals
Pause time (sec)	100

Table 1. Simulation parameters

center of an area. The MTs are randomly located in the network. The speed of an MT is uniformly distributed from 0.0 to 2.0 (m/sec). The *random waypoint mobility* model is used to simulate mobility here.

The query arrival pattern follows the Poisson distribution with a rate of λ. The update inter arrival time to the database is assumed to be exponentially distributed. The entire data items in the database are classified into two subsets: cold and hot data. We assume 80% of the update requests are for hot data items, and an update request is uniformly distributed within the hot and cold subsets. The least frequently used (LRU) cache replacement policy is applied to the MT's local cache. To model the data item access pattern, we use the Zipf distribution which is often used to model a skewed access pattern [2], where θ is the access skewness coefficient. Setting θ = 0 corresponds to the Uniform distribution. Here, we set θ to 0.9. The important simulation parameters are summarized in Table 1.

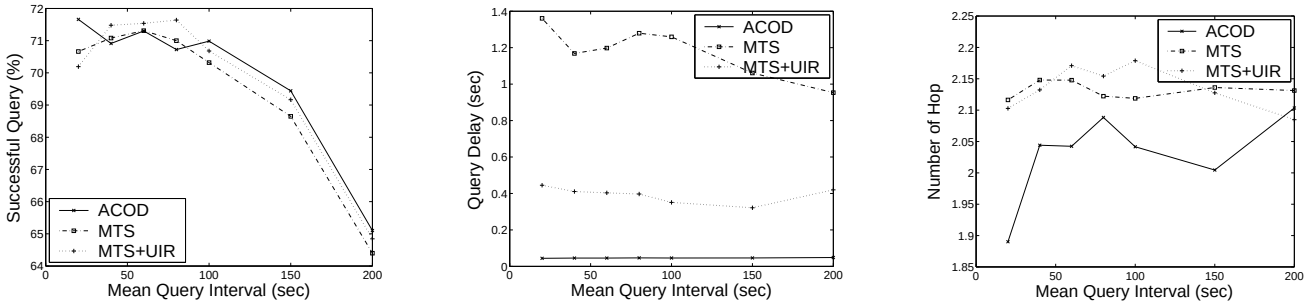
5.2. Simulation Results

In this section, we use four performance parameters: successful query rate (M_{suc}), average query delay (M_{delay}), average number of hop (M_{hop}), and average number of message (M_{msg}) for request. Based on these parameters, we evaluate the impact of query interval (T_{query}), cache update interval (T_{update}), and the search algorithm on the cache invalidation strategies. Also, the overhead of cache invalidation strategies is examined. Here, we include a subset of the results due to space limitation. For additional results, please refer to [10]

5.2.1. Impact of Query Interval First, we evaluate the M_{suc}, M_{delay}, and M_{hop} of the cache invalidation strategies as a function of T_{query}. In Figure 7(a), as the query interval increases, the percent of successful query decreases. Since all the cache invalidation strategies answer a query after either verification of a cached data item or arrival of an IR, they show almost similar performance under the same query interval and network topology. Figure 7(b) shows the impact of T_{query} on the query delay. The ACOD scheme shows the lowest M_{delay}, followed by the MTS+UIR and MTS schemes. The MTS scheme shows the longest M_{delay}, due to the long IR broadcast interval. In the MTS+UIR scheme, M_{delay} is reduced compared to that of MTS due to UIRs, which arrive during an IR broadcast interval. Figure 7(c) shows the impact of T_{query} on the number of hops. The ACOD scheme shows slightly lower M_{hop} than that of the MTS and MTS+UIR schemes. Since a data item can be received from an intermediate MT, the number of hops is less compared to the IR-based scheme.

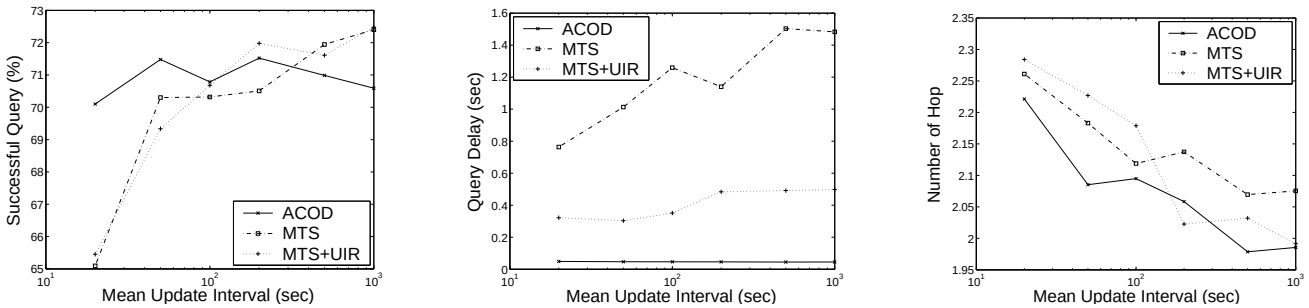
5.2.2. Impact of Update Interval Next, we examine the impact of T_{update} on M_{suc}, M_{delay}, and M_{hop}. In MTS and MTS+UIR schemes, from Figure 8(a), M_{suc} increases as the update interval increases. However, the ACOD scheme shows little change of M_{suc} over the entire update intervals, because it checks validity of a cached data item or receives a queried data item from the server, and thus, its performance is not affected by T_{update} much. In Figure 8(b), the ACOD scheme shows the lowest M_{delay} over the entire update intervals, followed by the MTS+UIR and MTS schemes. As the update interval increases, the M_{delay} of both MTS and MTS+UIR schemes is increased. Since less number of cached data items are updated at a server, an MT finds more number of cached data items marked as valid. Thus, in the MTS and MTS+UIR schemes, the MT waits for the next IR or UIR arrival for validation frequently, and query delay increases. The MTS scheme has the longest M_{delay}, because it has longer interval to receive the next IR than that of the MTS+UIR scheme. In Figure 8(c), as the update interval increases, M_{hop} of all the schemes is proportionally reduced, because more number of cached data items are valid, and thus an MT receives more queried data items from MTs rather than from a server.

5.2.3. Impact of Search Algorithm We compare the proposed MSS algorithm with the SS algorithm, and evaluate their performances in terms of M_{msg}, M_{suc}, and M_{delay}. Figure 9 shows the impact of MSS and SS algorithms on M_{msg}, M_{suc}, and M_{delay} for different cache invalidation strategies. In Figure 9(a), when the MSS algorithm is used, M_{msg} is reduced for all the cache invalidation strategies. In the MSS algorithm, unlike the SS algorithm, a request packet is broadcasted toward the MTs, which are located along the path closer to the server. Thus, the MSS algorithm eliminates unnecessarily broadcasted messages. In Figures 9(b) and (c), since the SS algorithm generates more messages over the network, it may cause traffic conges-



(a) Successful query (b) Query delay (c) Number of hop

Figure 7. Successful query, query latency, the number of hops as a function of mean query interval (T_{update} = 100, and C_{size} = 10).



(a) Successful query (b) Query delay (c) Number of hop

Figure 8. Successful query, query delay, number of hops as a function of mean update interval (T_{query} = 100, and C_{size} = 10)

tion and larger message delay. The MSS algorithm achieves higher M_{suc} and lower M_{delay} than that of the SS algorithm for all the cache invalidation strategies. In summary, the proposed MSS algorithm reduces the number of messages and increases communication performance, and thus, it is suitable for implementing cache invalidation strategies for IMANETS.

5.2.4. Message Overhead Finally, we evaluate the overhead of each cache invalidation strategy in terms of average number of messages, including broadcast, sent, and forwarded messages from the server and MTs. Figure 10 shows the impact of update interval on M_{msg} with different cache sizes. The MTS+UIR scheme shows the highest M_{msg} because both IR and UIR are periodically broadcasted over the network, regardless of the update interval. The ACOD scheme shows the lowest M_{msg} for the entire update intervals.

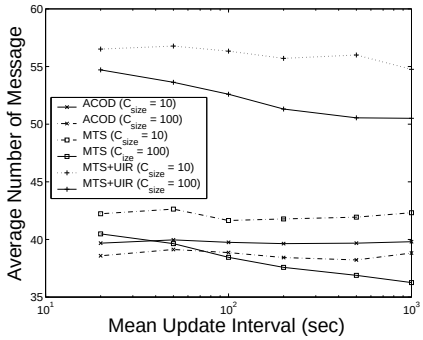
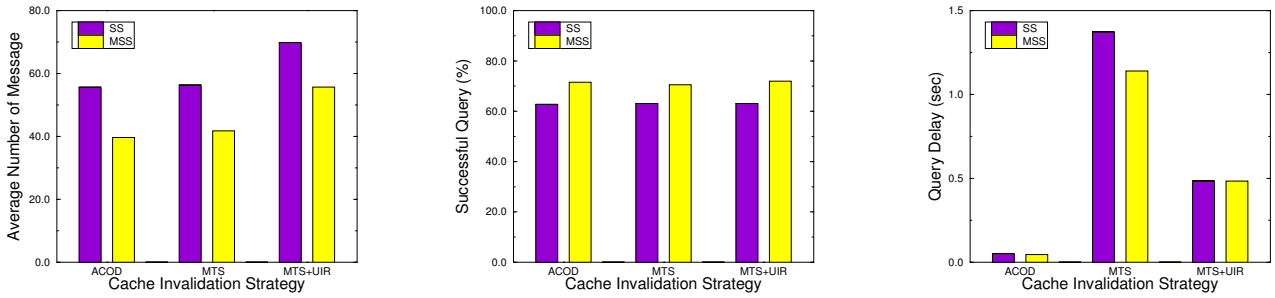


Figure 10. Average number of messages as a function of mean update interval (T_{query} = 100, and C_{size} = 10 or 100).



(a) Average number of message

(b) Successful query

(c) Query delay

Figure 9. Average number of message, successful query, and query delay with different search algorithms are shown against different cache invalidation strategies (T_{query} = 100, T_{update} = 200, and C_{size} = 10).

6. Concluding Remarks

In this paper, we investigated the cache invalidation problem in an IMANET, where the issues are different from that in a cellular network due to multi-hop message relay, operation cost model, and uncertainty in message delivery. In light of this, several push and pull-based cache invalidation strategies are carefully compared and analyzed. First, we proposed a *GPS-based connectivity estimation (GPSCE)* scheme as a base algorithm to support any cache invalidation strategy in IMANETS. Then, we proposed several push and pull based cache invalidation strategies.

We conducted a simulation based performance study to examine the performance of different cache invalidation strategies. In the MTS and MTS+UIR schemes, there is an unavoidable delay before processing a query, since a request has to wait for the next IR/UIR. Also the message overhead due to broadcasted IR/UIRs leads to waste of wireless bandwidth. The ACOD scheme provides a high throughput (successful query), the lowest query delay, and a minimal communication overhead for different workload configurating. Thus, it is a viable approach for implementation in IMANETS.

References

[1] D. Barbara and T. Imielinksi. Sleepers and Workaholics: Caching Strategies for Mobile Environments. In *Proc. ACM SIGMOD*, pages 1–12, 1994.

[2] G. Cao. A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments. In *Proc. ACM MOBICOM*, pages 200–209, 2000.

[3] M. S. Corson, J. P. Macker, and G. H. Cirincione. Internet-Based Mobile Ad Hoc Networking. In *IEEE Internet Computing*, pages 63–70, July–August 1999.

[4] T. Hara. Effective Replica Allocation in Ad Hoc Networks for Improving Data Accessibility. In *Proc. IEEE INFOCOM*, pages 1568–1576, 2001.

[5] T. Hara. Replica Allocation in Ad Hoc Networks with Period Data Update. In *Proc. 3rd International Conference on Mobile Data Management (MDM)*, pages 79–86, 2002.

[6] Q. Hu and D. Lee. Cache Algorithms based on Adaptive Invalidation Reports for Mobile Environments. *Cluster Computing*, pages 39–48, 1998.

[7] J. Jing, A. Elmagarmid, A. Helal, and R. Alons. Bit-Sequences: An Adaptive Cache Invalidation Method in Mobile Client/Server Environment. *Mobile Networks and Applications*, pages 115–127, 1997.

[8] A. Kahol, S. Khurana, S. Gupta, and P. Srimani. An Efficient Cache Management Scheme for Mobile Environment. In *Proc. 20th International Conference on Distributied Computing System (ICDCS)*, pages 530–537, April 2000.

[9] S. Lim, W. Lee, G. Cao, and C. R. Das. A Novel Caching Scheme for Internet based Mobile Ad Hoc Networks. In *Proc. 12th International Conference on Computer Communications and Networks (ICCCN)*, pages 38–43, 2003.

[10] S. Lim, W. Lee, G. Cao, and C. R. Das. Cache Invalidation Strategies for Internet-based Mobile Ad Hoc Networks (IMANETs). Technical Report CSE–04–011, Dept. of CSE, The PennState Univ., April 2004.

[11] H. Luo, R. Ramjee, P. Sinha, L. Li, and S. Lu. UCAN: A Unified Cellular and Ad-Hoc Network Architecture. In *Proc. ACM MOBICOM*, pages 353–367, 2003.

[12] M. Papadopouli and H. Schulzrinne. Effects of Power Conservation, Wireless Converage and Cooperation on Data Dissemination among Mobile Devices. In *Proc. MobiHoc*, pages 117–127, 2001.

[13] B. Tuch. Development of WaveLAN, an ISM band wireless LAN. *AT&T Techical Journal*, 72(4):27–33, 1993.

[14] K. Wu, P. Yu, and M. Chen. Energy-Efficient Caching for Wireless Mobile Computing. In *Proc. 20th International Conference on Data Engineering*, pages 336–345, 1996.