J Inf Process Syst, Vol.13, No.5, pp.1410~1429, October 2017
https://doi.org/10.3745/JIPS.03.0081

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS

Hierarchical Location Caching Scheme for Mobile Object Tracking in the Internet of Things

Youn-Hee Han*, Hyun-Kyo Lim*, and Joon-Min Gil**

Abstract
Mobility arises naturally in the Internet of Things networks, since the location of mobile objects, e.g., mobile agents, mobile software, mobile things, or users with wireless hardware, changes as they move. Tracking their current location is essential to mobile computing. To overcome the scalability problem, hierarchical architectures of location databases have been proposed. When location updates and lookups for mobile objects are localized, these architectures become effective. However, the network signaling costs and the execution number of database operations increase particularly when the scale of the architectures and the numbers of databases becomes large to accommodate a great number of objects. This disadvantage can be alleviated by a location caching scheme which exploits the spatial and temporal locality in location lookup. In this paper, we propose a hierarchical location caching scheme, which acclimates the existing location caching scheme to a hierarchical architecture of location databases. The performance analysis indicates that the adjustment of such thresholds has an impact on cost reduction in the proposed scheme.

Keywords
Internet of Things, Location Caching Scheme, Location Tracking, Mobile Computing, Mobile Object

1. Introduction

In current mobile computing systems, many users with wireless hardware are not tied to a fixed access point but move wide-area wireless network. Furthermore, mobile software, i.e., data and code which move in Internet system, constitutes a new form of distributed and mobile application [1,2]. The mobile software agents are a popular form of such application [3-5]. In Internet of Things (IoT) networks of physical objects that contain embedded technology (such as intelligent sensors), there are some efforts to support intelligence environment for controlling moving things or objects [6,7]. One of the key issues in mobile computing is to economically and quickly transfer any form of information between any desired locations at any time for mobile objects. That is the reason why the efficient location tracking mechanism becomes one of the most important problems for the Internet of Things networks as well as mobile networks [8,9].

A mobile object performs location update to explicitly create or update its location record in specific location databases. When there is a need to locate a mobile object, the network performs location lookup by querying the object's record to search the current location of the mobile object. As the

Manuscript received August 12, 2016; first revision February 21, 2017; accepted March 4, 2017.
Corresponding Author: Joon-Min Gil (jmgil@cu.ac.kr)
* Advanced Technology Research Center, Korea University of Technology and Education, Cheonan, Korea ({yhhan, glenn89}@koreatech.ac.kr)
** School of Information Technology Engineering, Catholic University of Daegu, Gyeongsan, Korea (jmgil@cu.ac.kr)

number of objects keeps increasing, the amount of signaling traffic and database operation associated with the need to track mobile objects keeps growing.

One among efforts reducing loads upon signaling network and location databases is to adopt hierarchical arrangement of the location databases [8-11]. Since the hierarchical database structures manage the mobile objects in a distributed way, these structures are more responsive and more reliable for the increasing mobile objects. In this paper, we focus on a wide-area mobile computing, i.e., the location tracking service should allow mobile objects to be located anywhere in the world, and be able to support a huge number of objects. To accommodate such vast scalability problem, the structural hierarchy of location databases need to be expanded to arbitrary N-level.

The hierarchical structure is more effective particularly when most location update and lookup are localized. In such cases, instead of contacting the remote location database that is usually existed far away from an object's current location, a small number of location databases in the object's neighborhood are accessed. In many cases, objects can be located by querying some databases in the hierarchy. However, it is noted that the execution number of database operations caused by location update and lookup can rather increase, since only one location update or lookup request to go through many databases deployed in network. Furthermore, the network signaling costs also increase particularly when the scale of the structure becomes large to accommodate a great number of objects.

Such disadvantage can be alleviated by introduction of location caching scheme in the hierarchical architecture. Location caching is based on the idea of reusing the information about an object's location from the previous lookup to that object. This leads to reduce network signaling and database loads of the basic strategy in exchange for increased CPU processing cost and hard-disk (or memory) consumption. Since technology trends are driving the costs down, deploying the caching strategy on a system-wide basis will become increasingly attractive [9,11].

The key scheme we present is the use of hierarchical location caching information. With this, the proposed hierarchical location caching scheme exploits the information to fast lookup an object and reduce the signaling traffic and the number of database accesses.

A cache entry has to be removed from the cache after it is deemed less useful. The longer the time a mobile object has not been looked up from a cache database, the higher the chance that the corresponding cache entry is not correct (i.e., the mobile object moves to another place and the cache entry points to a wrong location for it). So, the proposed scheme also supports a hierarchical cache invalidation strategy in order to invalidate less useful cache entry. In this strategy, each cache entry is invalidated after its associated threshold amount of time since its last usage. That is, the thresholds are used as a guide to select one of the hierarchical cache information.

This paper is organized as follows. In section II, we will review and describe the selected related work. In Section III, we will describe the hierarchical location caching scheme in detail. In Section IV, we will describe a new random walk model reflecting a N-level hierarchical database architecture and demonstrate our scheme's cost reduction in terms of the network signaling and database loads. Finally, in Section V, we will conclude this paper.

2. Related Works

The hierarchical location schemes [8-11] maintain a hierarchy of location databases and have been proposed in order to localize location update and lookup operations geographically. Usually, a hierarchical location management strategy arranges the location databases in a tree-like structure (see Fig. 1). From the root, the databases in each level of the hierarchy form finer and finer partitions of the total coverage area. In this case, the location database at a leaf serves a single specific region, such as registration area, and contains entries for all objects currently roaming in the region. A database at an internal node maintains information about objects roaming in the coverage area of its descendant databases. For each mobile objects, the information is a pointer to an entry at a lower level database. The hierarchical structure leads to reductions in signaling cost when most location update and lookup are localized. In such cases, instead of contacting a remote database that may be located far away from the object's current location, a small number of location databases in the object's neighborhood are accessed. In addition, there is no need for binding an object to a remote database, since the object can be located by querying one of the databases in the hierarchy.

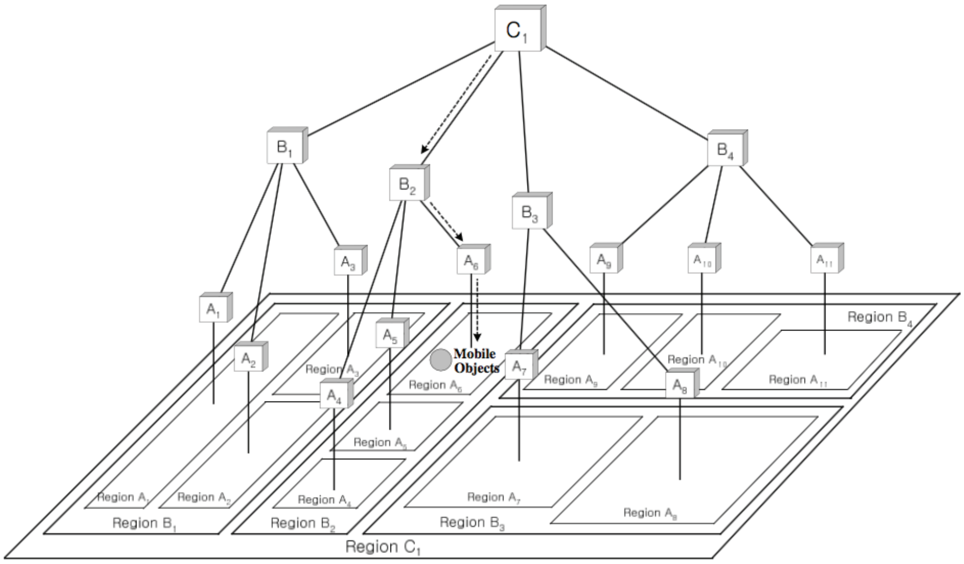


Fig. 1. Hierarchical location database architecture.

Location caching is based on the idea of reusing the information about an object's location obtained from the previous lookup. The strategy we discuss in this paper acclimates the existing studies [12-15] to the N-level hierarchical arrangement of databases. Even though the studies have been done to be applied to Personal Communication Systems (PCSs), we just use the basic concept of algorithms, and then augment and adjust them to systems supporting a huge number of mobile objects (or agents) in a hierarchical manner.

Basically, every time an object x is found, x's location is cached at its nearby database, so that any subsequent lookup to x can reuse this information. Clearly, caching is useful for those objects who receive lookups frequently relative to the rate at which they move since no traffic is associated with queries in case of cache hit. However, in case of cache miss, extra overhead is paid. Further the cache entries must be invalidated and various approaches have been proposed. In eager caching, every time an object moves to a new location, all cache entries for that object's location are updated. Thus, the cost of move operations increases for those objects whose address is cached. In lazy caching, the cached location for a given object is updated only in case of a cache miss. The basic overhead involved in lazy caching is in case of cache miss since the cached location must be visited first. An improvement to this work was presented in [13], where a caching scheme based on fully disseminating the location updates of mobiles to every node is shown to yield higher cache hit rates.

In [12], the authors investigated the classes of objects for which the caching strategy yields net reductions in signaling traffic and database loads using the notion of object's Local Call to Mobility Ratio (LCMR). LCMR is the ratio between the number of lookups originating from a local region to the number of times the object changes its service area. They derived the LCMR threshold, the minimum LCMR required for caching to be beneficial assuming incoming lookups are a Poisson process and inter-move times are exponentially distributed. In [15], caching techniques was adopted in hierarchical architecture to exploit locality. The proposed caching strategy exploited a pair of bypass pointers and presented an eager caching.

A cache entry has to be removed from the cache after it is deemed less useful. The longer the time an object has not been looked up from a cache database, the higher the chance that the corresponding cache entry, if cached, is not correct (i.e., points to a wrong location for the mobile object). In [14], a cache invalidation approach called T-threshold location caching scheme was proposed. In this scheme, a cache entry is invalidated after a threshold amount of time T since its last usage. That is, the threshold T is used as a guide to select the subset of objects to which the cache information should be applied.

3. Hierarchical Location Caching Scheme

3.1 Basic Hierarchical Location Management

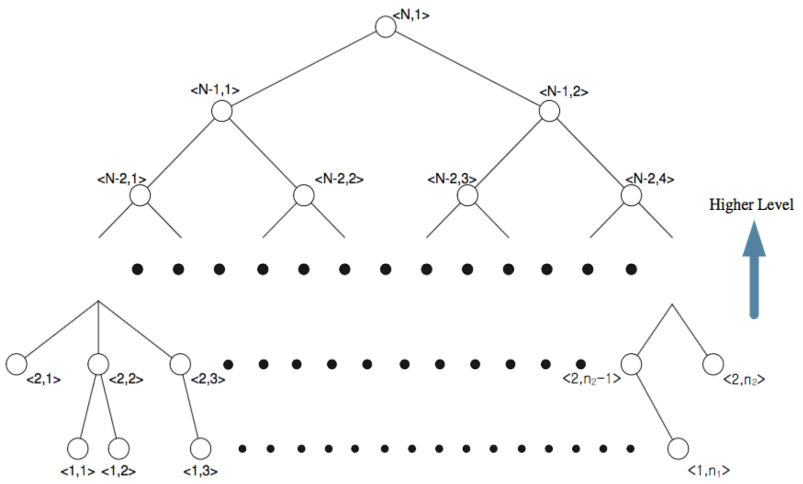


Fig. 2. Our system model.

Fig. 2 presents a N-level tree structure used as our system model. The choice of the tree structure given by the network graph may depend on traffic and mobility patterns within the network. Let $T = (V, E)$ be the tree, where $v \in V$ corresponds to a node which has a location database containing entries for objects in the subtree rooted at the node, and $e = (u, v) \in E$ is the bidirectional communication link between two connected nodes u and v. Each object is addressed by a universally unique identifier (e.g., object handles in [8,11]). Each node is of the form $\langle x, y\rangle$, where x indicates that the node is in level x of the tree (In our model, higher level of nodes is marked by higher value) and y represents the yth node in level x $(1 \le x \le N)$. Each parent knows its children nodes. The nodes in each level of the hierarchy form finer and finer partitions of the total coverage area. A node $\langle x, y\rangle$ serves a single registration area which consists of several finer registration areas managed by nodes in the subtree rooted at the node.

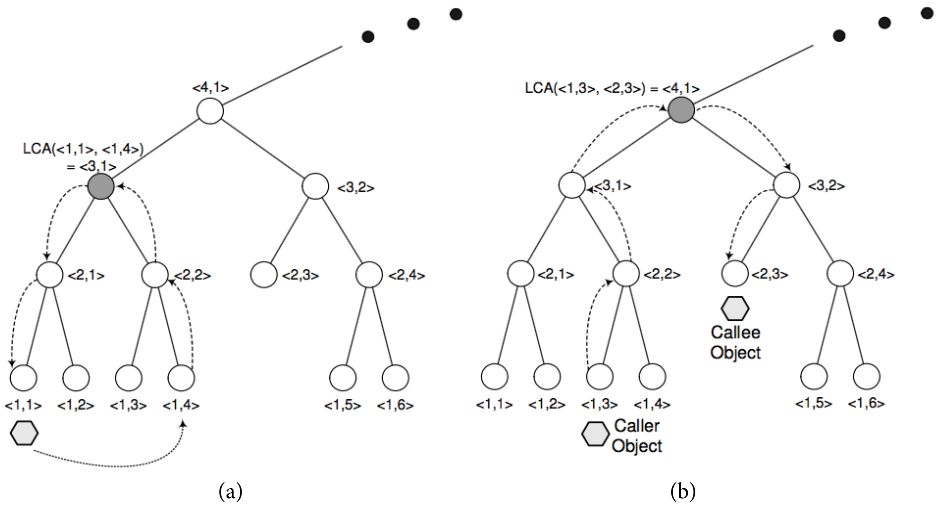


Fig. 3. Basic hierarchical location management. (a) Location update and (b) location lookup.

Movement of an object must be accompanied by updates in the appropriate databases. Let the set of ancestors and descendants of a node in the tree be defined in the usual manner. Let $LCA(\langle a, i\rangle, \langle b, j\rangle)$ denote the lowest common ancestor of node $\langle a, i\rangle$ and node $\langle b, j\rangle$. Fig. 3(a) represents the location update procedure when an object moves from node $\langle 1,1\rangle$ to node $\langle 1,4\rangle$. Node $\langle 1,4\rangle$ accepts the object by entering its identifier in the registration list of database within the node, and then, on the tree's way, a message is sent up with the information that the object has moved to node $\langle 1,4\rangle$. Each node encountered by this message, if the database in the node does not already have an entry for the object, adds an entry for the object which points to the previous node on the path of the message. This message finally encounters a node at which database already has an entry for the object. It is the $LCA(\langle 1,1\rangle, \langle 1,4\rangle) = \langle 3,1\rangle$. This node terminates the upward message, and initiates a downward message on the tree's way up to node $\langle 1,1\rangle$, telling databases encountered on the way to erase their pointer entries for the object.

Requests for location lookup can arrive, or be generated at, any leaf node $\langle a, i\rangle$ in the network, and a callee is registered at any leaf node $\langle b, j\rangle$. If the callee is registered at the originating node, location lookup occurs with only one access at the database. If the callee is not registered at the originating database (that is, $\langle a, i\rangle \neq \langle b, j\rangle$), the location lookup is routed up on the tree's way. Fig. 3(b) represents the location lookup procedure when a caller in node $\langle 1,3\rangle$ requests a location lookup into a callee in node $\langle 2,3\rangle$. Both node $\langle 2,2\rangle$ and node $\langle 3,1\rangle$ encountered along the upward route look up the entry corresponding to the callee. The node at which database holds the entry is the $LCA(\langle 1,3\rangle, \langle 2,3\rangle) = \langle 4,1\rangle$.

And then, the location lookup follows a chain of downward pointers to the node at which the callee is finally registered; the downward chain consists of nodes $\langle 4,1\rangle$, $\langle 3,2\rangle$ and $\langle 2,3\rangle$.

3.2 Hierarchical Location Caching Scheme

In this section, we propose our hierarchical location caching scheme. The application of this scheme perfectly fits the proposed hierarchical structure of location databases. When a caller sends a location lookup message for a callee and the lookup procedure is completed, the cache database of the caller's leaf node records the entire downward chain of nodes which register the callee currently. In order to make the proposed scheme efficient, we introduce the hierarchical thresholds. In the chain, each node is associated with one threshold. These thresholds are used to exploit spatial locality of a mobile object and support cache invalidation.

If a caller was located at node $\langle a, i\rangle$ and a callee p was found in node $\langle b, j\rangle$ in the latest location lookup event, the cache of the node $\langle a, i\rangle$ keeps a record (a chain of pointers) for the callee p in its database. Also, we define $l_{i,j}$ as the level of $LCA(\langle a, i\rangle, \langle b, j\rangle)$. Then, the record includes two fields as follows.

- ct: the time of the latest lookup to p
- L: the list of (N_k, T_k) pairs serving p at time ct. In the pair, N_k is the node which registers p at level k, and T_k is the threshold associated with N_k for $1 \le k \le l_{i,j}$.

The list L arranges (N_k, T_k) pairs from lower level to higher level. Note that there is no threshold correspondent to $N_{l_{i,j}}$ in the list and $T_k < T_{k+1}$ for $1 \le k \le l_{i,j} - 1$.

For example, with Fig. 4, consider that the caller at node $\langle 1,1\rangle$ requests a location lookup for callee A at node $\langle 1,4\rangle$ at time '19h 15m 57s, 2015-01-20'. After the lookup procedure has finished, the callee's cache information stored at node $\langle 1,1\rangle$ has the request time as ct's value and the list L: $\{(\langle 1,4\rangle, 10m), (\langle 2,3\rangle, 34m), (\langle 3,2\rangle, 57m), (\langle 4,1\rangle, -)\}$. After another location lookup procedure requested for callee B in node $\langle 2,7\rangle$ has finished, the cache record is also depicted at the figure.

Suppose that, at time t, a new location lookup to the callee p arrives at a leaf node. Let us assume the caller's leaf node has the cache information for the callee p and the pairs (N_s, T_s), (N_{s+1}, T_{s+1}), …, (N_{e-1}, T_{e-1}), $(N_e, -)$ have been in the list L, where $1 \le s < e \le N$. Then, the proposed scheme is described as follows:

Case 1. If $t - ct \le T_s$, the cached N_s is selected as a location hint which indicates the node serving p.

Case 1.1. After the node N_s is queried, if p is found (a location hit occurs), ct is assigned the value t.

Case 1.2. Otherwise (a location miss occurs), **Case 3** takes place.

Case 2. If $T_{k-1} < t - ct \le T_k$, where $s < k < e$, then the cached N_k is selected as a location hint which indicates the node serving the callee. Otherwise (that is, $t - ct > T_{e-1}$), the cached N_e is selected as a location hint

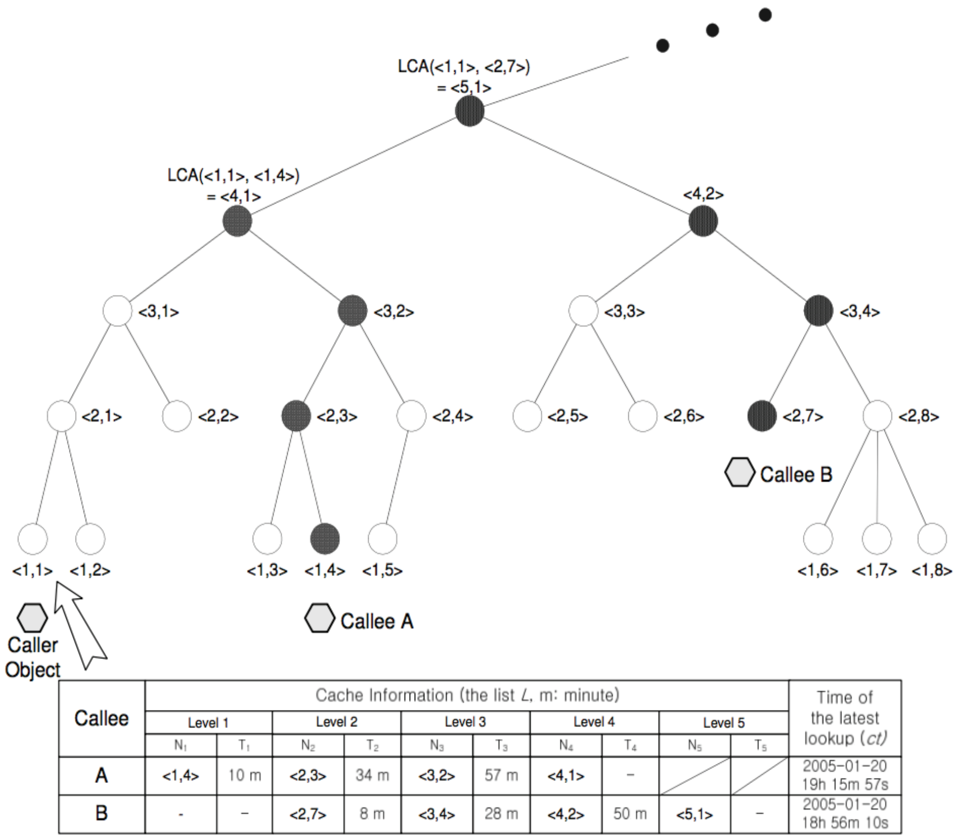
Case 2.1. After the selected node is queried, if p is found, it is assumed that p is in the subtree rooted at the node. The location lookup follows a chain of downward pointers until reaching the lowest node at which p is located. And then, ct is assigned the value t. If some of the followed nodes are different with ones in the list L of cache information, the list L is updated with the founded nodes.

Case 2.2. Otherwise (p is not found at the selected node), **Case 3** takes place.

Case 3. When a location miss occurs at **Case 1** and **Case 2**, the location lookup continues up from the node where the location miss takes place. On finding the node which registers p, the location lookup follows a chain of downward pointers from the node until reaching the lowest node at which p is located. After location lookup procedure has finished, the new downward chain including nodes registering p is assigned to the list L and ct is assigned the value t.

It is apparent that the lookup cost is low if **Case 1.1** or **Case 2.1** occurs. On the other hand, **Case 1.2** or **Case 2.2** results in an extra penalty for the cost since the additional procedure, **Case 3**, is required. If a mobile object moves infrequently and shows high spatial locality, location hits are frequent. Otherwise, location misses are frequent.

With Fig. 4, let us consider that a caller at node ⟨1,1⟩ requests a location lookup for a callee A. As depicted in Fig. 4, the cache information for the callee A is assumed to be already managed at the node ⟨1,1⟩. Suppose that the request happens at time '19h 20m 14s, 2015-01-20'. In the case, since the difference between the stored previous lookup time '19h 15m 57s, 2015-01-20' and the current request time is below the first threshold '10m', the node ⟨1,4⟩ is selected and directly queried (**Case 1**). If the callee A is found at the node (**Case 1.1**), ct is assigned the request time and the procedure comes to an end.



Callee	Cache Information (the list L, m: minute)										Time of the latest lookup (ct)
	Level 1		Level 2		Level 3		Level 4		Level 5		
	N_1	T_1	N_2	T_2	N_3	T_3	N_4	T_4	N_5	T_5	
A	<1,4>	10 m	<2,3>	34 m	<3,2>	57 m	<4,1>	–			2005-01-20 19h 15m 57s
B	-	–	<2,7>	8 m	<3,4>	28 m	<4,2>	50 m	<5,1>	–	2005-01-20 18h 56m 10s

Fig. 4. Cache information in hierarchical location caching scheme.

If the callee A has moved to a neighboring node $\langle 1,5\rangle$ prior to the lookup, it is not found at the selected node (**Case 1.2**). In this case, the node $\langle 2,3\rangle$, which is the parent of the node $\langle 1,4\rangle$, is queried. Since the callee A's entry is not found at the node $\langle 2,3\rangle$, the node $\langle 3,2\rangle$, which is the parent of the node $\langle 2,3\rangle$, is also searched. Since the entry in the node $\langle 3,2\rangle$'s database has a pointer for the callee A, the lookup procedure continues down until reaching the lowest node $\langle 1,5\rangle$ at which the callee A is located. After the procedure has finished, the list L is updated with two new nodes, $\langle 1,5\rangle$ and $\langle 2,4\rangle$, instead of the existing nodes, $\langle 1,4\rangle$ and $\langle 2,3\rangle$. Finally, ct is assigned the request time (**Case 3**).

4. Performance Analysis

In this section, we use a N-level balanced tree structure of location databases for simplicity of performance analysis, although an arbitrary tree architecture was exploited for describing our scheme in the previous section. Moreover, we add an additional condition into the system model: root node and all internal nodes have R children. So, we can define the N/R balanced tree structure of location databases, $T_{N/R}$, as the N-level balanced tree structure satisfying the additional conditions. A tree $T_{N/R}$ has total $\frac{R^N-1}{R-1}$ nodes (databases), where there are R^N-1 leaf nodes. In the $T_{N/R}$, let RA_i be a registration area in level i, $R-1$ where $1 \le i \le N$. RA_1 is the smallest registration area in level 1 and has no sub-registration area. For $2 \le i \le N$, an RA_i has R sub-registration areas. The mathematical expression for an RA_i is as follows:

$$RA_i = \bigcup_{k=1}^{R} RA_{i-1}^k \tag{1}$$

where RA_{i-1}^k is kth registration area in level $i-1$. Also, we assume the homogeneous network of which all registration areas in same level have the same shape and size.

Let t_{RA_i} be i.i.d. (independently and identically distributed) random variable representing the residence time of an RA_i. Let $f_{RA_i}(t)$ be the density function of t_{RA_i}. t_{RA_1} is random variable representing the residence time of the lowest node RA_1, and its density function is $f_{RA_1}(t)$.

We assume that t_{RA_1} has a Gamma distribution with mean $1/\lambda(= E[t_{RA_1}])$, standard deviation σ, and variance $V(=\sigma^2)$. The Gamma distribution is selected for its flexibility and generality. By selecting the proper values for the parameters, a Gamma distribution can be an Exponential, an Erlang or a Chi-square distribution. A Gamma distribution can also be used to represent the distribution for a set of measured data. The Laplace transform of the Gamma distribution is

$$f_{RA_1}^*(s) = \left(\frac{\gamma\lambda}{s+\gamma\lambda}\right)^{\gamma}, where\ \gamma = \frac{1}{V\lambda^2} \tag{2}$$

For $2 \le i \le N$, suppose that an object visits $k RA_{i-1}$s in an RA_i for a period $t_{RA_i}^k$. During $t_{RA_i}^k$, the object resides at jth RA_{i-1} for a period t_j. Then, $t_{RA_i}^k = t_1 + t_2 + \cdots + t_k$ has the following density function:

$$f_{RA_i}^{(k)}(t) = \int_{t_1=0}^{t}\int_{t_2=0}^{t-t_1}\cdots\int_{t_{k-1}=0}^{t-t_1-\cdots-t_{k-2}} f_{RA_{i-1}}(t_1)f_{RA_{i-1}}(t_2)\ldots f_{RA_{i-1}}(t_{k-1})f_{RA_{i-1}}(t-t_1-\cdots-t_{k-1})\,dt_{k-1}\ldots dt_2dt_1 \tag{3}$$

Using the Laplace transform convolution, we can get the Laplace transform for $f_{RA_i}^{(k)}(t)$ as follows:

$$f_{RA_i}^{(k)*}(s) = [f_{RA_{i-1}}^{*}(s)]^k. \tag{4}$$

where $f_{RA_{i-1}}^{(k)}(s)$ is the Laplace transform of $f_{RA_{i-1}}(t)$.

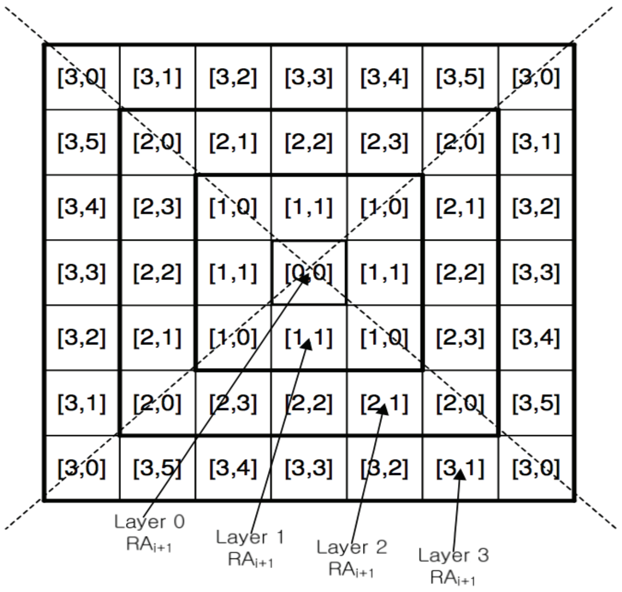


Fig. 5. Mesh $4-layer$ registration area RA_{i-1}s in an RA_i.

We describe a two-dimensional random walk model for mesh planes in order to compute the area residence time density function. Our model is similar to one proposed by [16] and considers a regular hierarchical area overlay structure. We assume that an object resides in an RA_i for a period and moves to one of its four neighbors with the same probability, i.e., with probability $1/4$. Recursively from this assumption, for $2 \le i \le N-1$, we also can say that an object resides in an RA_i for a period and moves to one of other four neighbor RA_i with probability $1/4$. For $2 \le i \le N$, an RA_i is referred to as an $r-layer$ if it overlays with $4r^2-4r+1(=R)$ RA_{i-1}s. Fig. 5 shows the $4-layer$ RA_i overlaying 49 RA_{i-1}s in $T_{N/49}$. RA_{i-1} at the center of RA_i is called the $layer\ 0$. RA_{i-1}s that surround the $layer\ s-1$ RA_{i-1}s are called $layer\ s$. In an $r-layer$ RA_i, RA_{i-1}s that surround the $layer\ r-1$ RA_{i-1}s are referred to as boundary neighbors, which are outside of the RA_i.

According to the equal moving probability assumption, we classify RA_{i-1}s in an RA_i into several types. An RA_{i-1} type is of the form $[x,y]$, where x indicates that the RA_{i-1} is in layer x and y represents the $y+1$st type in layer x. RA_{i-1}s of the same type have the same traffic flow pattern because they are at the symmetrical positions on the mesh area. Fig. 5 illustrates the type of RA_{i-1}s for a $4-layer$ RA_i.

In the random walk model of an $r - layer\ RA_i$, a state (x, y) represents that an object is in one of the RA_{i-1}s of type $[x, y]$. In $r - layer\ RA_i$, the state (r, j) represents that the object moves out of the area from state $(r - 1, j)$, where $0 \le j \le 2r - 3$. For $0 \le j \le 2r - 3$, states (r, j) are absorbing, the others are transient. For $r > 1$, the total number $S(r)$ of states for random walk of $r - layer\ RA_i$ is $r^2 + \mathrm{r} - 1$. The *transition matrix* of this random walk is an $S(r) \times S(r)$ matrix $P = (p_{(x,y)(x',y')})$. For $0 \le j \le 2r - 3$, $p_{(r-1,j)(r,j)}$ is the probability that an object moves from an RA_{i-1} of the type $[r - 1, j]$ to a neighbor out of the area RA_i in one step. The absorbing state (r, j) loops back to itself with probability 1.

For $k \ge 1$, we use the Chapman-Kolmogorov equation to compute the probability $p^{(k)}$ for the number of steps that an object moves from an RA_{i-1} type to another. An element $p^{(k)}_{(x,y)(x',y')}$ in $p^{(k)}$ is the probability that the random walk moves from state (x, y) to state (x', y') with exact k steps. For $0 \le j \le 2r - 3$, define $p_{k,(x,y)(r,j)}$ as

$$p_{k,(x,y)(r,j)} = \begin{cases} p_{(x,y)(r,j)} & ,for\ k = 1 \\ p^{(k)}_{(x,y)(r,j)} - p^{(k-1)}_{(x,y)(r,j)} & ,for\ k > 1. \end{cases} \tag{5}$$

Then, $p_{k,(x,y)(r,j)}$ is the probability that an object initially resides at an RA_{i-1} of the type $[x, y]$, moves into an RA_{i-1} of the type $[r - 1, j]$ at the k-1st step, and then moves out of the RA_i at the kth step.

Let $q_{(r-1,j)}$ be the probability that an object enters the RA_i through an RA_{i-1} of the type $[r - 1, j]$ at the first step and $\tilde{q}_{(r-1,j)}$ be the probability that an object moves out of the RA through an RA of the type $[r - 1, j]$ at the last step. Then, for $r \ge 2$, we have

$$\tilde{q}_{(r-1,j)} = \sum_{y=0}^{2r-3} \sum_{k=1}^{\infty} q_{(r-1,y)} p_{k,(x,y)(r,j)} \tag{6}$$

$$\sum_{y=0}^{2r-3} \tilde{q}_{(r-1,j)} = 1 \tag{7}$$

Each term of the right side of Eq. (6) represents the probability that an object initially resides at an RA_{i-1} of the type $[r - 1, j]$ and leaves the RA_i from an RA_{i-1} of the type $[r - 1, j]$ at the kth step. For $r > 1$, the following equations hold

$$\begin{cases} q_{(r-1,0)} = \tilde{q}_{(r-1,0)} \\ q_{(r-1,0)} = \tilde{q}_{(r-1,2r-j-2)}, & for\ 0 < j \le 2r - 3 \end{cases} \tag{8}$$

From Eqs. (6)–(8), we can get a linear system and a particular solution for $q_{(r-1,j)}$. For a $4 - layer\ RA_i$, we have $q_{(3,0)} \simeq 28.571\%$ and $q_{(3,j)} \simeq 14.285\%$ for $0 \le j \le 5$. For a $5 - layer\ RA_i$, we have $q_{(4,0)} \simeq 22.222\%$ and $q_{(4,j)} \simeq 11.111\%$ for $0 \le j \le 7$. For the two RA_i structures, we use 500 truncated terms in Eq. (6) to approximate the infinite summations. The error for the truncation is within 10^{-12}.

For $r \ge 2$ and $2 \le i \le N$, from Eqs. (3) and (4), the Laplace transform of residence time density function $f_{RA_i}(i)$ for an $r-layer\ RA_i$ is as follows.

$$\begin{aligned} f_{RA_i}^{*}(s) &= \sum_{k=1}^{\infty}\sum_{y=0}^{2r-3}\sum_{j=0}^{2r-3} q_{(r-1,y)}p_{k,(r-1,y)(r,j)}f_{RA_i}^{(k)*}(s) \\ &= \sum_{k=1}^{\infty}\sum_{y=0}^{2r-3}\sum_{j=0}^{2r-3} q_{(r-1,y)}p_{k,(r-1,y)(r,j)}\cdot\left[f_{RA_{i-1}}^{*}(s)\right]^{k} \end{aligned} \tag{9}$$

For $r \geq 2$, from the moment generation property, the expected residence time of a registration area is as follows:

$$\mathrm{E}[t_{RA_i}] = \sum_{k=1}^{\infty}\sum_{y=0}^{2r-3}\sum_{j=0}^{2r-3} q_{(r-1,y)}p_{k,(r-1,y)(r,j)}\psi(k)\,. \tag{10}$$

where

$$\psi(k) = (-1)k\left[f_{RA_{i-1}}^{*}(0)\right]^{k-1}\frac{df_{RA_{i-1}}^{*}(s)}{ds}\bigg|_{s=0} \tag{11}$$

Since the residence time density function for an $r-layer\ RA_i$ is defined recursively in Eq. (9), we can get the recursive definition of residence time density function for $r-layer$ registration areas in $T_{N/R}$, where $\mathrm{R} = 4r^2 - 4\mathrm{r} + 1$, as follows.

$$f_{RA_i}^{*}(s) = \begin{cases} \left(\dfrac{\gamma\lambda}{s+\gamma\lambda}\right)^{\gamma} & if\ i=1, otherwise \\ \displaystyle\sum_{k=1}^{\infty}\sum_{y=0}^{2r-3}\sum_{j=0}^{2r-3} q_{(r-1,y)}p_{k,(r-1,y)(r,j)}\cdot\left[f_{RA_{i-1}}^{*}(s)\right]^{k} \end{cases} \tag{12}$$

4.1 Probabilities for Each Case of the Hierarchical Location Caching Scheme

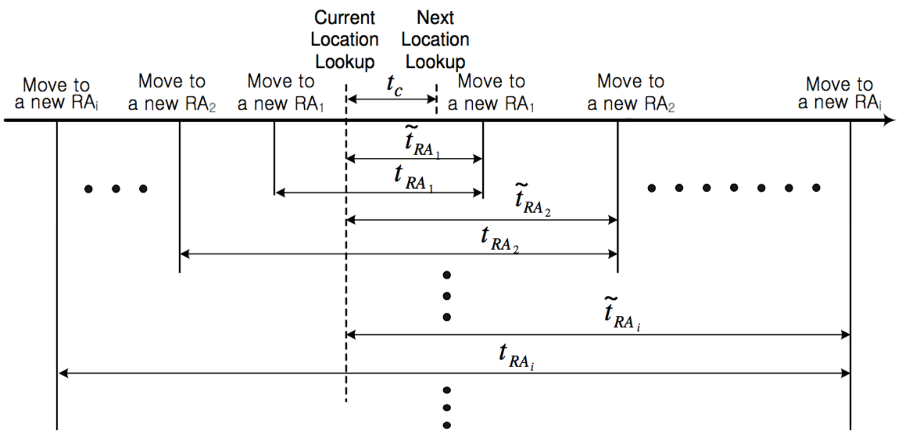


Fig. 6. The relationship among t_c, t_{RA_i} and $\tilde{t}_{RA_i}$.

This section proposes probabilities in connection with the hierarchical location caching scheme. Let t_c be an i.i.d. random variable, which represents the time interval between two consecutive location lookups directed from an RA_1 to a remote object p. Let $\tilde{t}_{RA_i}$ be the time interval between the previous location lookup for p and the time when p moves out of the RA_i. The relationship among t_c, t_{RA_i} and $\tilde{t}_{RA_i}$ is given in Fig. 6. Let $f_c(t)$ and $\tilde{f}_{RA_i}(t)$ be the density function of t_c and $\tilde{t}_{RA_i}$, respectively.

We assume that the arrivals of location lookup request for an object are a Poisson process and $E[t_c] = 1/\lambda_c$. From the assumption, we have

$$f_c(t) = \lambda_c e^{-\lambda_c t} \tag{13}$$

Since the incoming location lookup requests are random observers of t_{RA_i} for all i (which is referred to as the excess life theorem [17]), we have the density function of $\tilde{t}_{RA_i}$ as follows.

$$\tilde{f}_{RA_i}(t) = \frac{1}{E[t_{RA_i}]} \int_{\tau=t}^{\infty} f_{RA_i}(\tau)\, d\tau \tag{14}$$

From [18], we can get the Laplace transform for the $\tilde{t}_{RA_i}$ distribution as follows.

$$\begin{aligned} \tilde{f}^*_{RA_i}(s) &= \int_{t=0}^{\infty} e^{-st} \tilde{f}_{RA_i}(t) dt \\ &= \frac{1}{E[t_{RA_i}]} \left[1 - f^*_{RA_i}(s)\right]. \end{aligned} \tag{15}$$

The probability that an object moves no RA_i between two consecutive location lookups is derived as follows.

$$\begin{aligned} \Pr\left[t_c < \tilde{t}_{RA_i}\right] &= \int_{t_1=0}^{\infty} \int_{t_2=0}^{t_1} \lambda_c\, e^{-\lambda t_2} \tilde{f}_{RA_i}(t_1) dt_2 dt_1 \\ &= 1 - \frac{1}{E[t_{RA_i}]\lambda_c} \left[1 - f^*_{RA_i}(\lambda_c)\right]. \end{aligned} \tag{16}$$

Let us assume that, after the latest location lookup to an object p, the cache information at the leaf node $\langle 1, i \rangle$ indicates that the object p resides at a leaf node $\langle 1, j \rangle$ at the present time (In this analysis, we use complete balanced tree. Therefore, the level of a leaf node is always one). Also, we define $l_{i,j}$ as the level of $LCA(\langle 1, i \rangle, \langle 1, j \rangle)$.

Let μ_1 be the probability of the cache hit with threshold T_1. That is, μ_1 is the probability for **Case 1.1** presented in the hierarchical location caching scheme. From conditions given by **Case 1.1**, we have

$$\begin{aligned} \mu_1 &= \Pr[t_c < \tilde{t}_{RA_1}] \cdot \Pr[t_c \le T_1] \\ &= \left(1 - \frac{\lambda}{\lambda_c}\left[1 - \left(\frac{\gamma\lambda}{\lambda_c + \gamma\lambda}\right)^{\gamma}\right]\right) \cdot \left(1 - e^{-\lambda_c T_1}\right). \end{aligned} \tag{17}$$

Let ν_1 be the probability of the cache miss with threshold T_1. That is, ν_1 is the probability for **Case 1.2** presented in the proposed scheme. From conditions given by **Case 1.2**, we have

$$\begin{aligned}\nu_1 &= \Pr[t_c \geq \tilde{t}_{RA_1}] \cdot \Pr[t_c \leq T_1] \\ &= \frac{\lambda}{\lambda_c}\left[1-\left(\frac{\gamma\lambda}{\lambda_c+\gamma\lambda}\right)^{\gamma}\right] \cdot \left(1-e^{-\lambda_c T_1}\right).\end{aligned} \tag{18}$$

For $1 < k < l_{i,j}$, let μ_k be the probability of the cache hit with threshold T_k. From conditions given by **Case 2.1**, we have

$$\begin{aligned}\mu_k &= \Pr[t_c < \tilde{t}_{RA_k}] \cdot \Pr[T_{k-1} \leq t_c \leq T_k] \\ &= \left(1-\frac{1}{E[t_{RA_k}]\lambda_c}\left[1-f^*_{RA_k}(\lambda_c)\right]\right) \cdot \left(e^{-\lambda_c T_{k-1}} - e^{-\lambda_c T_k}\right).\end{aligned} \tag{19}$$

On the other hand, when $k = l_{i,j}$, we have

$$\begin{aligned}\mu_{l_{i,j}} &= \Pr[t_c < \tilde{t}_{RA_{l_{i,j}}}] \cdot \Pr\,[T_{l_{i,j}-1} < t_c] \\ &= \left(1-\frac{1}{E[t_{RA_{l_{i,j}}}]\lambda_c}\left[1-f^*_{RA_{l_{i,j}}}(\lambda_c)\right]\right) \cdot e^{-\lambda_c T_{l_{i,j}-1}}.\end{aligned} \tag{20}$$

For $1 < k < l_{i,j}$, let ν_k be the probability of the cache miss with threshold T_k. From conditions given by **Case 2.2**, we have

$$\begin{aligned}\nu_k &= \Pr[t_c \geq \tilde{t}_{RA_k}] \cdot \Pr[T_{k-1} \leq t_c \leq T_k] \\ &= \frac{1}{E[t_{RA_k}]\lambda_c}\left[1-f^*_{RA_k}(\lambda_c)\right] \cdot \left(e^{-\lambda_c T_{k-1}} - e^{-\lambda_c T_k}\right).\end{aligned} \tag{21}$$

On the other hand, when $k = l_{i,j}$, we have

$$\begin{aligned}\nu_{l_{i,j}} &= \Pr[t_c > \tilde{t}_{RA_{l_{i,j}}}] \cdot \Pr[T_{l_{i,j}-1} < t_c] \\ &= \frac{1}{E[t_{RA_{l_{i,j}}}]\lambda_c}\left[1-f^*_{RA_{l_{i,j}}}(\lambda_c)\right] \cdot e^{-\lambda_c T_{l_{i,j}-1}}.\end{aligned} \tag{22}$$

Since **Case 3** indicates the behaviors following after **Case 1.2** or **Case 2.2** occurs, we need not consider a probability for the case. All probabilities proposed in Eqs. (17)–(22) sum up to 1.

$$\sum_{k=1}^{l_{i,j}} (\mu_k + \nu_k) = 1. \tag{23}$$

4.2 Network Cost Analysis

This section proposes the network cost analysis of the hierarchical location caching scheme using the probabilities presented by the above subsection. We assume that there are shortcuts between interior nodes. Since the network costs of hierarchical location lookup caching scheme through shortcuts depend on the network architecture, we evaluate the benefits using diverse values of the network parameters. Also, we assume that a location notification to caller or caller's node after a successful lookup takes the same costs in both caching scheme and non-caching scheme and the costs are not included into our analysis.

Firstly, we normalize the network cost to send a message between a parent node and a child node into α. Consider the remote access to a callee p at node $\langle 1, j\rangle$ from a caller at node $\langle 1, i\rangle$. Let c_s^k be the network cost to send a message through a shortcut between $\langle 1, i\rangle$ and a node of level k, where $1 < k < l_{i,j}s$. The node of level k is one of ancestors of $\langle 1, j\rangle$. We assume that the value of c_s^k is always m times as low as that of the cost which does not use a shortcut. If a shortcut is not used, a message goes to the node of level k from $\langle 1, i\rangle$ via $LCA(\langle 1, i\rangle, \langle 1, j\rangle)$. For $m \geq 1$, each c_s^k is defined as follows.

$$c_s^k = \frac{l_{i,j} - 1 + l_{i,j} - k}{m} \cdot \alpha. \tag{24}$$

Let c_h^k be the network cost of a location lookup when N_k is used as the cache information and a cache hit occurs. When a node N_k between N_2 and $N_{l_{i,j}}$ is used as the cache information, there is additional cost $k - 1$, which brings out when the location lookup follows a chain of downward pointers from N_k until reaching the lowest node where the callee is located. Therefore, c_h^k can be defined as follows.

$$c_h^k = c_s^k + (k-1) \cdot \alpha. \tag{25}$$

Let c_m^k be the network cost of a location lookup when N_k is used as the cache information and a cache miss occurs. When a cache miss takes place, the lookup continues up from the node where the miss takes place until finding a node storing an entry for the callee, and down to the lowest node where the callee is located. Therefore, c_m^k can be defined in $T_{N/R}$ as follows.

$$\begin{aligned} c_m^k &= c_s^k + \sum_{h=k+1}^{N} Pr[\tilde{t}_{RA_{h-1}} < t_c \leq \tilde{t}_{RA_h}] \cdot (h - k + h - 1) \cdot \alpha \\ &= c_s^k + \sum_{h=k+1}^{N} \frac{1 - f_{RA_{h-1}}^*(\lambda_c)}{E[t_{RA_{h-1}}]\lambda_c} \left(1 - \frac{1 - f_{RA_h}^*(\lambda_c)}{E[t_{RA_h}]\lambda_c}\right) \cdot (h - k + h - 1) \cdot \alpha. \end{aligned} \tag{26}$$

Therefore, the network cost in the hierarchical location caching scheme can be defined as

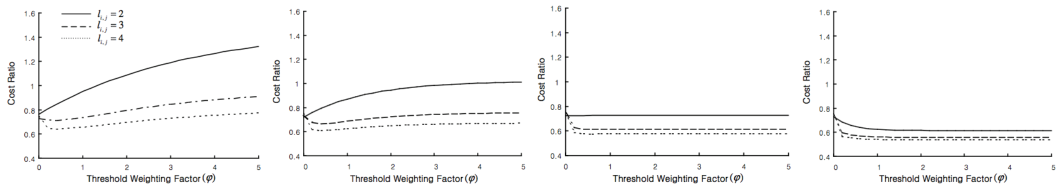
$$\Omega_1 = \sum_{k=1}^{l_{i,j}} (\mu_k \cdot c_h^k + \nu_k \cdot c_m^k). \tag{27}$$

On the other hand, the network cost of non-caching scheme is defined as follows.

$$\Omega_2 = 2\alpha \cdot (l_{i,j} - 1). \tag{28}$$

Since threshold values affect the performance of the proposed scheme, it is important to determine the values. A threshold reflects a forecasted residence time at the found area after last location lookup. In this analysis, therefore, the values are assumed to be proportional to the expected residence time of the associated registration area. For $1 \leq k < l_{i,j}$, we use the *threshold weighting factor* φ and select the values as follows.

$$T_k = E[t_{RA_k}] \cdot \varphi. \tag{29}$$



(a) (b) (c) (d)

Fig. 7. Relative network cost ratios (Ω_1/Ω_2). (a) $\lambda_c = 0.3$, (b) $\lambda_c = 0.7$, (c) $\lambda_c = 1.5$, and (d) $\lambda_c = 2.5$.

Fig. 7 depicts the variation of network cost ratio Ω_1/Ω_2 with respect to various values of φ, λ_c, and $l_{i,j}$. The values of parameters used in the numerical results are as follows: $\lambda = 1.0$, $N = 5, r = 4$ (hence, $R = 49$), $\sigma = 1/\lambda$, $m = 2$, and $\alpha = 1$. The results show that the network cost reduction of hierarchical location caching scheme becomes high when the frequency λ_c of location lookup becomes high. Also, the network cost reduction becomes relatively high when the lowest common ancestor between caller and callee is near at the root of the proposed tree architecture (i.e. when $l_{i,j}$ is high). The cost of the caching scheme even becomes higher than that of non-caching scheme when $l_{i,j} = 2$ and threshold weighting factors are more than 1.3 (when $\lambda_c = 0.3$) or 3.0 (when $\lambda_c = 0.7$). In the other hand, the cost reduction of up to 45% is achieved when $\lambda_c = 2.5$ and $l_{i,j} = 4$. In the case $\lambda_c = 0.3$ and $\lambda_c = 0.7$, the most cost reduction is shown when the weighting factors are roughly 0.3 and 0.4, respectively (although we cannot say which value makes the cost reduction maximum when $l_{i,j} = 2$). We also can see that the network cost ratio is regular irrespectively of the weighting factor when λ_c is high. That is, the efficiency of the caching scheme is high and regular regardless of the weighting factor value if lookup rate is high.

Fig. 8 shows the effect of the standard deviation σ (or variance) for the residence time t_{RA_1} distribution on network cost. Except λ_c, $l_{i,j}$ and σ, the values of other parameters used in this numerical results are the same as ones used in Fig. 7. We can observe that the network cost reduction becomes large as σ increases. That is, for the same expected value of t_{RA_1}, more long residence times will be observed as the standard deviation increases. More long residence times imply that more cache hits occur. Thus, more cost reduction is observed. Also, we can see that the threshold weighting factor has significant impact on cost reduction when lookup rate is low.

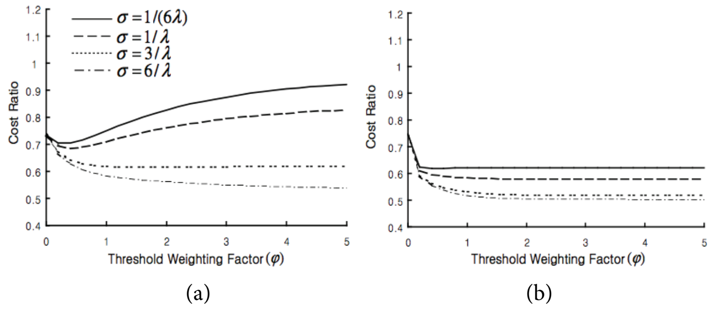


Fig. 8. The effect of the standard deviation for t_{RA_1} distribution on network cost ratio (Ω_1/Ω_2). (a) $\lambda_c = 0.5, l_{i,j} = 3$. (b) $\lambda_c = 2.0, l_{i,j} = 3$.

Fig. 8 shows the effect of the standard deviation σ (or variance) for the residence time t_{RA_1} distribution on network cost. Except λ_c, $l_{i,j}$ and σ, the values of other parameters used in this numerical results are the same as ones used in Fig. 7. We can observe that the network cost reduction becomes large as σ increases. That is, for the same expected value of t_{RA_1}, more long residence times will be observed as the standard deviation increases. More long residence times imply that more cache hits occur. Thus, more cost reduction is observed. Also, we can see that the threshold weighting factor has significant impact on cost reduction when lookup rate is low.

4.3 Database Cost Analysis

In this section, we are interested in discovering the reduction in database costs obtained by the hierarchical location caching scheme. The number of database operations caused by a location lookup increases in the hierarchical structure of location databases. In such environment, the caching scheme can reduce the number of database operations by bypassing quite many databases to lookup the location of a callee. Database costs are not under the influence of underlying network structure, which is contrasted with network costs. The only one which has impact on database costs is the number of query and write operations during a location lookup and update.

For a given database (including the cache database), we normalize both a query cost and a write cost into β and γ, respectively. Let $\hat{c}_h^k$ be the database cost of a location lookup when N_k is used as the cache information and a cache hit occurs. Prior to location database queries, it is required to query the cache database in the node where a caller exists. After querying the database in the cached node, other queries are performed to the databases in nodes within a chain of pointers from the selected node N_k to the lowest node where the callee exists. Finally, the update to the cache database is required. So, from this procedure, $\hat{c}_h^k$ can be defined as follows.

$$\hat{c}_h^k = (k+1) \cdot \beta + \gamma. \tag{30}$$

Let $\hat{c}_m^k$ be the database cost of a location lookup when N_k is used as the cache information and a cache miss occurs. When a cache miss takes place, the lookup continues up from the node where the miss takes place until finding a node storing an entry for the callee, and down to the lowest node where the callee is located. So, $\hat{c}_m^k$ can be defined in $T_{N/R}$ as follows.

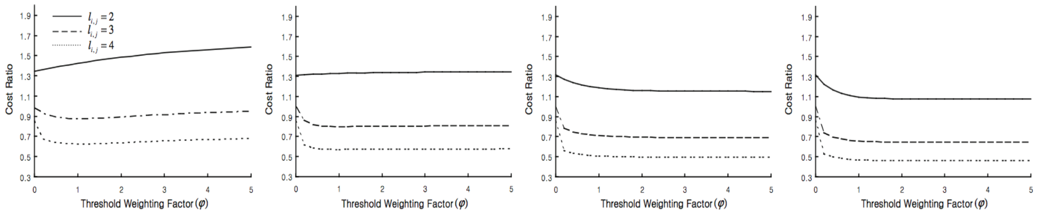
$$\begin{aligned}\hat{c}_m^k &= \beta + \gamma + \sum_{h=k+1}^{N} Pr[\tilde{t}_{RA_{h-1}} < t_c \le \tilde{t}_{RA_h}] \cdot (h-k+h-1) \cdot \beta \\ &= \beta + \gamma + \sum_{h=k+1}^{N} \frac{1 - f_{RA_{h-1}}^*(\lambda_c)}{E[t_{RA_{h-1}}]\lambda_c} \left(1 - \frac{1 - f_{RA_h}^*(\lambda_c)}{E[t_{RA_h}]\lambda_c}\right) \cdot (h-k+h-1) \cdot \beta\,.\end{aligned} \tag{31}$$

Therefore, the database cost in the hierarchical location caching scheme can be defined as

$$\theta_1 = \sum_{k=1}^{l_{i,j}} (\mu_k \cdot \hat{c}_h^k + \nu_k \cdot \hat{c}_m^k). \tag{23}$$

On the other hand, the database cost of non-caching scheme is defined as follows.

$$\theta_2 = 2\beta \cdot (l_{i,j} - 1). \tag{23}$$

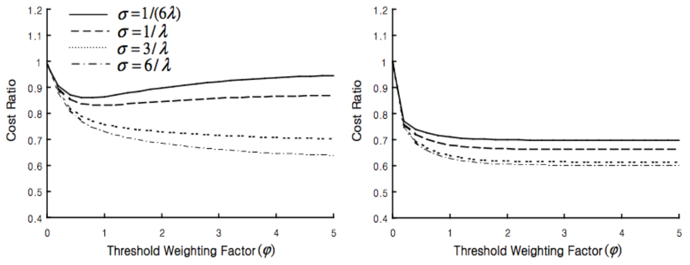


(a) (b) (c) (d)

Fig. 9. Relative database cost ratios (θ_1/θ_2). (a) $\lambda_c = 0.3$, (b) $\lambda_c = 0.7$, (c) $\lambda_c = 1.5$, and (d) $\lambda_c = 2.5$.

Fig. 9 depicts the variation of database cost ratio θ_1/θ_2 with respect to various values of φ, λ_c, and $l_{i,j}$. The values of parameters used in the numerical results are as follows: $\lambda = 1.0,\ N = 5, r = 4$ (hence, $R = 49$), $\sigma = 1/\lambda$, and $\beta = \gamma = 1$. The results are similar to ones related to network costs. It is shown that the database cost reduction of hierarchical location caching scheme becomes high when the frequency λ_c of location lookup becomes high. The cost reduction becomes relatively high when $l_{i,j}$ is high. When $l_{i,j} = 2$, for the most part of λ_c and φ, the cost of the caching scheme even becomes higher than that of non-caching scheme. This is because the caching scheme requires additional location database query when location miss occurs and non-caching scheme itself does not need to access many databases when $l_{i,j}$ is so low. On the other hand, it is noted that the cost reduction of up to 55% is achieved when $\lambda_c = 2.5$ and $l_{i,j} = 4$. In the case $\lambda_c = 0.3$ and $\lambda_c = 0.7$, the most cost reduction is shown when the threshold weighting factors are roughly 1.0 and 1.2, respectively (although we cannot say which value makes the cost reduction maximum when $l_{i,j} = 2$). We also can see that the database cost ratio is regular irrespectively of the weighting factor when λ_c is high. That is, the efficiency of the caching scheme is high and regular regardless of the weighting factor value if lookup rate is high.

Fig. 10 shows the effect of the standard deviation σ (or variance) for the residence time t_{RA_1} distribution on database cost. We can observe that the database cost reduction becomes large as σ increases. Also, we can know that the weighting factor has significant impact on cost reduction when lookup rate is low.



(a) (b)

Fig. 10. The effect of the standard deviation for t_{RA_1} distribution on database cost ratio (θ_1/θ_2). (a) $\lambda_c = 0.5, l_{i,j} = 3$. (b) $\lambda_c = 2.0, l_{i,j} = 3$.

5. Conclusions

In the future systems accommodating a vast number of potentially mobile objects in the Internet of Things, it is required to locate a mobile object to communicate. Finding locations of mobile objects contributes significant traffic to the network and puts much overload to location databases.

Two distinct but related topics were addressed in this paper. The first was the hierarchical location caching scheme supporting cache invalidation in a hierarchical architecture of location databases. The idea is to use the cache to store the location information which is used as a hint to locate the mobile objects, and to invalidate some corrupt cache by using the hierarchical threshold values. It was proposed to reduce location lookup costs in terms of the network signaling and database loads. In order to analyze performance of the proposed scheme, we designed a new random walk model reflecting hierarchical architecture. From the performance evaluation, we found the following facts: 1) the proposed scheme can reduce the network and database costs in most cases; 2) the cost reduction becomes high and regular irrespectively of various threshold values when λ_c is high; 3) the cost reduction also becomes high when the called object is located far from the originator of lookup request; 4) the cost reduction also becomes high as the standard deviation σ of the lowest area's residence time increases.

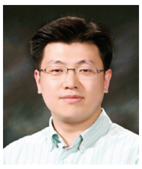
Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. NRF-2016R1D1A3B03933355), and also supported by the Graduate School Research Program of KOREATECH.

References

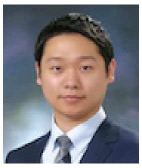
[1] K. Rothermel, S. Schnitzer, R. Lange, F. Durr, and T. Farrell, "Context-aware and quality-aware algorithms for efficient mobile object management," *Pervasive and Mobile Computing*, vol. 8, no. 1, pp. 131-146, 2012.

[2] J. Vitek and C. Tschudin, "Mobile object systems: towards the programmable Internet," in *Lecture Notes in Computer Science,* vol. 1222, Heidelberg, Germany: Springer, 1997.
[3] G. P. Gupta, M. Misra, and K. Garg, "An energy efficient distributed approach-based agent migration scheme for data aggregation in wireless sensor networks," *Journal of Information Processing Systems*, vol. 11, no. 1, pp. 148-164, 2015.
[4] P. Morreale, "Agents on the move mobile software agents," *IEEE Spectrum*, vol. 35, no. 4, pp. 34-41, 1998.
[5] L. Vasiu and Q. H. Mahmoud, "Mobile agents in wireless devices," *IEEE Spectrum*, vol. 37, no. 2, pp. 104-105, 2004.
[6] K. Gao, Q. Wang, and L. Xi, "Controlling moving object in the internet of things," *International Journal of Advancements in Computing Technology*, vol. 4, no. 5, pp. 83-90, 2012.
[7] L. E. Talavera, M. Endler, I. Vasconcelos, R. Vasconcelos, M. Cunha, and F. J. da Silva e Silva, "The mobile hub concept: Enabling applications for the internet of mobile things," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, St. Louis, MO, 2015, pp. 123-128.
[8] M. van Steen, F. J. Hauck, P. Homburg, and A. S. Tanenbaum, "Locating objects in wide-area systems," *IEEE Communications Magazine*, vol. 36, no. 1, pp. 104-109, 1998.
[9] E. Pitoura and G. Samaras, "Locating objects in mobile computing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 4, pp. 571-592, 2001.
[10] E. Pitoura and I. Fudos, "Distributed location databases for tracking highly mobile objects," *The Computer Journal*, vol. 44, no. 2, pp. 75-91, 2001.
[11] A. Baggio, G. Ballintijn, M. van Steen, and A. S. Tanenbaum, "Efficient tracking of mobile objects in Globe," *The Computer Journal*, vol. 44, no. 5, pp. 340-353, 2001.
[12] R. Jain, Y. B. Lin, C. N. Lo, and S. Mohan, "A caching strategy to reduce network impacts of PCS," *IEEE Journal on Selected Areas in Communications,* vol. 12, no. 8, pp. 1434-1445, 1994.
[13] K. Ratnam, I. Matta, and S. Rangarajan, "Analysis of caching-based location management in personal communication networks," in *Proceedings of the 7th International Conference on Network Protocols*, Toronto, Canada, 1999, pp. 293-300.
[14] Y. B. Lin, "Determining the user locations for personal communications services networks," *IEEE Transactions on Vehicular Technology*, vol. 43, no. 3, pp. 466-473, 1994.
[15] R. Jain and F. Anjum, "Caching in hierarchical user location databases for PCS," in *Proceedings of the IEEE International Conference on Personal Wireless Communication*, Jaipur, India, 1999, pp. 496-500.
[16] I. F. Akyildiz, Y. B. Lin, W. R. Lai, and R. J. Chen, "A new random walk model for PCS networks," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 7, pp. 1254-1260, 2000.
[17] S. M. Ross, *Stochastic Processes*, 2nd ed, New Delhi, India: Wiley, 1995.
[18] Y. B. Lin, "Reducing location update cost in a PCS network," *IEEE/ACM Transactions on Networking*, vol. 5, no. 1, pp. 25-33, 1997.



Youn-Hee Han http://orcid.org/0000-0002-5835-7972

He received B.S. degree in Mathematics from Korea University, Seoul, Korea, in 1996. He received his M.S. and Ph.D. degrees in Computer Science and Engineering from Korea University in 1998 and 2002, respectively. From March 4, 2002 to February 28, 2006, he was a senior researcher in the Next Generation Network Group of Samsung Advanced Institute of Technology. Since March 2, 2006, he has been a Professor in the School of Computer Science and Engineering at Korea University of Technology and Education, Cheonan, Korea. His primary research interests include theory and application of mobile computing, including protocol design and mathematical analysis. Since 2002, his activities have focused on mobility management, media independent handover, and cross-layer optimization for efficient mobility support. His research topics also include mobile sensor/actuator networks, social network analysis, and deep learning. He has published approximately 150 research papers on the theory and application of mobile computing, and has filed 30 patents on ICT (Information and Communication Technology) domain. He has been serving as an editor for Journal of Information Processing Systems (JIPS) since August 2011. In addition, he has also made several contributions in IETF and IEEE standardization, and served as the co-chair of s working group in Korea TTA IPv6 Project Group.



Hyun-Kyo Lim

He received B.S. degree in Computer Science and Engineering from Korea University of Technology and Education, in 2015. He received M.S. degree in School of Computer Science and Engineering from Korea University of Technology and Education, in 2017. His research interests include mobile communication and mobility management. Since 2016, his activities have focused on Software Defined Network (SDN), future internet, and machine learning.

Joon-Min Gil http://orcid.org/0000-0001-6774-8476

He received his B.S. and M.S. degrees in Computer Science from Korea University, Korea in 1994 and 1996, respectively. He received his Ph.D. degree in Computer Science and Engineering from Korea University, Korea in 2000. Before joining in School of Information Technology Engineering, Catholic University of Daegu, he was a senior researcher in Supercomputing Center, Korea Institute of Science and Technology Information (KISTI), Daejeon, Korea from October 2002 to February 2006. From June 2001 to May 2002, he was a visiting research associate in the Department of Computer Science at the University of Illinois at Chicago, USA. His recent research interests include cloud computing, big data computing, distributed and parallel computing, and wireless sensor networks.