EFFECTIVE REPLICATED SERVER ALLOCATION ALGORITHMS IN MOBILE COMPUTING SYSTEMS

Jin-Woo Song[1], Kwang-Jo Lee[1], Taek-Hun Kim[1], Sung-Bong Yang[1]

[1]Department of Computer Science, Yonsei University, Seoul, Korea
{fantaros, kjlee5435, kimthun, yang}@cs.yonsei.ac.kr

ABSTRACT

In mobile environments, mobile device users access and transfer a great deal of data through the online servers. In order to enhance users' access speed in a wireless network, decentralizing replicated servers appropriately in the network is required. Previous work regarding this issue had focused on the placement of replicated servers along with the moving paths of the users to maximize the hit ratio. When a miss occurs, they simply ignored the file request. Therefore, we suggest a solution to take care of such a miss by sending a file request to a replicated server nearby in the network.

This paper is to propose new cost-effective wireless access algorithms incorporating a present replicated server allocation algorithm with more keen analysis of the moving patterns of mobile device users. We propose four different algorithms that allocate available replicated servers in the network so as to minimize the communication costs. The experimental results show that, among the proposed algorithms, the replicated server clustering algorithm allocated replicated servers with near optimal communication costs.

KEYWORDS

Replicated Server Allocation, Shared Data, Communication Cost, Moving Pattern, Mobile Network

1. INTRODUCTION

Recently the number of mobile device users has increased explosively worldwide, owing to the technological advances in mobile devices and wireless networking (WIBRO, 3G LTE, and 4G) [1]-[3]. Mobile device users are now able to access a great deal of information through wireless communication devices such as cell phones and PDAs. Such easy accessibility through mobile communications has influenced our daily life tremendously from both economic and social perspective.

Although we are enjoying the benefits from the technological advances, the current mobile devices still have insufficient computing capability and storage. Hence, data replication is important in the design of a mobile computing system. In general, mobile computing systems rely on decentralized server systems [4], [5]. Data replication provides faster accessibility towards the shared data by reducing costly accesses to the online servers. However, maintaining replicated servers in the network requires the operating expenses due to the data storage and maintenance of the shared data. Hence, proper placement of available replicated servers in a mobile computing system is a very critical issue [6]-[7].

When a mobile device user tries to access some shared data, the user can get the data either from the replicated server in the cell at which the user is currently located or from the replicated server nearest to the current cell if there is no replicated server at the current cell. We assume that the online servers do not provide any services for accessing the shared data, due to the fact

that the communication cost for accessing them is much more expensive than that for accessing a nearby replication server.

In this paper we propose new cost-effective replica server allocation algorithms incorporating a present algorithm with more careful analysis of the moving patterns of mobile device users. The proposed algorithms minimize the communication cost in accessing the shared data at the replicated servers in a wireless network. First, we propose *the vertex occurrence count algorithm* (VOC) which is based on the number of visits, call it *the occurrence count*, to each cell made by all the users. VOC maps the network into a graph in which a vertex is a network cell and an edge represents adjacency between a pair of neighboring cells in a network. VOC selects greedily one vertex at a time that has the largest occurrence count among the unselected vertices and allocates a replicated server into the selected vertex. The ties are broken in such a way that the vertex with the largest sum of the occurrence counts of its adjacent vertices is selected for a replicated server.

We modify VOC by lowering the vertex occurrence counts of the neighbors of the cells selected by VOC. We reduce the occurrence counts of the neighboring cells in the hope that replicated servers are not likely to be allocated to the adjacent cells. We call this modified algorithm *the VOC-neighbor reduction algorithm*. A similar algorithm, called the *greedy set-cover algorithm*, also allocates replicated servers so that they are not allocated to the cells adjacent to each other. Finally we present *the replicated server clustering algorithm* that utilizes the *k*-means clustering. The algorithm returns a set of clusters; the center of each cluster becomes the location for a replicated server.

In order to compare the performances of the proposed algorithms, we have implemented two *edge-based occurrence count* algorithms that were modified from the algorithms in [6]. Their algorithms utilize the user occurrence and the movement occurrence counts under the condition that when a file request is not satisfied the request simply fails. Hence we have modified their algorithms so that no failure occurs; that is, we search a nearby cell with a replicated server in the network.

We have also implemented randomly guided search techniques for finding the locations for replicated servers: *a simulated annealing algorithm* and *a genetic algorithm*. Their solutions are used to compare the performances of the proposed algorithms. Note that although the qualities of their solutions are better than those of the proposed algorithms, they cannot be implemented in practice due to their long execution time. The experimental results showed that all the proposed algorithms outperformed the edge-based occurrence count algorithms, and the replicated server clustering algorithm allocated the replicated servers with near optimal communication costs.

The rest of the paper is organized as follows. Section 2 discusses the basic knowledge on mobile computing systems and related work. Section 3 describes the edge-based occurrence count algorithms, the proposed algorithms, and the random guided search techniques in detail. The experimental results are given in Section 4. Finally, Section 5 concludes the paper.

2. BACKGROUND

2.1. Replicated Servers in Mobile Computing Systems

In mobile computing systems, a server determines a bounded area called *a service area* that consists of generally more than one cell. If a mobile device user enters into a service area, data communications are made through the server in the service area. So they can access the shared data from the online server through the infra-network consisted of servers. Fig. 1 shows a sample path on which a mobile device user moves around the service areas. In this paper, we

assume that a service area consists of a single cell. This assumption is not unrealistic because we can always map a cell into a group of cells that are actually covered by a service area.

In Fig.1, a couple of replicated servers are decentralized at cells *D* and *I*, respectively. The shared data from a replication server at cell *I* can be accessed only while the user is within cell *I*. The user cannot access the data from the cells that do not have replicated servers. It is obviously profitable to allocate replicated servers at the cells where a lot of mobile device users are expected to visit. Recently researchers presented various research results on this issue by exploiting the moving patterns of the users [6]-[9].

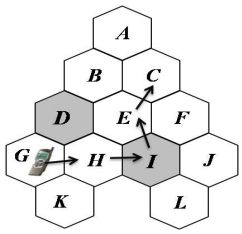


Fig. 1. A sample user path on a mobile computing system.

2.2. Wireless Access to Replicated Servers in a Mobile Computing System

Traditionally, data replication has been used for decentralized systems to speed up the shared data access [10]-[12]. In mobile computing systems, mobile devices suffer from restricted memory space, shortage of battery as well as slower system performance; because users ever demand smaller size devices with more added functionality and enhanced system performance. Therefore, data replication for mobile computing systems should be dealt in a different way from the traditional one and can be generally classified into two types.

In the first type, some of the mobile devices themselves serve as replicated servers. The mobile devices hold the shared data for faster access and for less server communication traffic. However, the sizes of replicated data are quite limited because the storage size of a mobile device is relatively small. Therefore, a system of this type can hardly support large-sized data [4], [5].

The second type distributes replicated servers throughout the network as in Fig. 1. A mobile device user can access quickly the shared data from a replicated server at the cell in which the user is currently located. A *hit* is said to occur for such a case. When a user in a cell without a replicated server requests the shared data, the request can be processed through an online server. But usually a limited number of replicated servers are deployed in a network due to the cost factor. Thus available replicated servers should be decentralized appropriately. [6]-[8]

Peng and Chen [6] proposed a shared data allocation algorithm that utilizes the moving patterns of mobile device users. Their algorithm focuses on the placement of replicated servers along with the moving paths of the users to maximize the hit ratio. Lim and Kim [13] proposed server replication schemes to reduce the location management cost in cellular networks. Their algorithm simply divides a network into sub-networks uniformly and places replicated servers in the center of each sub-network. They did not consider the moving patterns of mobile users.

3. COMMUNICATION COST-BASED REPLICATED SERVER ALLOCATION

3.1 Problem Definition

In this paper, we study the replicated server allocation problem in a wireless network. For the problem we adopt a data replication system of the second type with a slight modification. When a user is at a cell without having a replicated server and requests some shared data, we say that a *miss* has occurred. A user should request an online server when a miss occurs. In this paper, however, we let the user get the data from a replicated server at the nearest cell instead of from an online server. We assume that the server communication cost is more expensive than the neighbor communication cost for the problem, since an online server is usually available for several service areas.

Fig. 2 gives an example that a user at cell *23* tries to access the shared data from a replicated server when four replicated servers are located at cells *1*, *10*, *13*, and *20*, respectively. It is obvious that it takes the least communication cost when the user communicates with the nearest replicated server. Therefore, in this example, the user should access the replicated server at cell *13*. The ties can be broken arbitrarily.

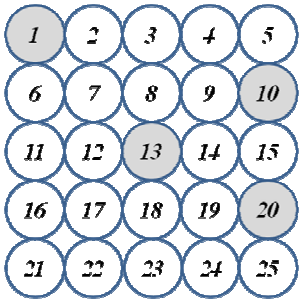


Fig. 2. Access to a nearest replicated server

We assume that the communication cost for accessing a replicated server from the current cell is the distance between the cell with the replicated server and the current cell. For example, in Fig. 2 the communication cost between cells *13* and *23* is 2 units, while the cost between *20* and *23* is 3 units. Note that the cells located at the diagonal directions are not supposed to be adjacent to the current cell. In accordance with these assumptions, we now define the problem of allocating replicated servers for minimizing the communication cost.

Assume that there are n users and each user i, i=1, … , n, has a sequence of operations (or requests), $op_i(C_1)$, $op_i(C_2)$, …, $op_i(C_m)$, where C_1, C_2, …, and C_m are the cells that user i moved in sequence. For example, if user 8 made 10 file requests at cell A, 6 requests at cell D, 20 requests at cell G, and 9 requests at cell E, then C_1=A, C_2=D, C_3=G, and C_4=E. Also, $op_8(C_1)$=10, $op_5(C_2)$=6, $op_5(C_3)$=20, and $op_5(C_4)$=9. Let $opCost_i(C_j)$ denote the communication cost of user i for processing $op_i(C_j)$. Then

$$opCost_i(C_j) = op_i(C_j) \times dist(C_j, C_r),$$

where $dist(C_j, C_r)$ is the shortest distance between C_j and C_r in the network and C_r is a nearest cell with a replicated server. We treat $dist(C_j, C_r)$ as the communication cost between C_j and C_r for a single access. Although the communication cost could be calculated with the number of hops along with other factors in a network, *dist*() does reflect the communication cost, since it is proportional to the communication cost between a pair of cells. $opCost_i(C_j)$ is 0, if cell C_j has a replicated server, i.e., $dist(C_j, C_j)$=0.

Let *commCost*(i) be the communication cost of user i for processing all the operations in the request operation sequence of the user. Then

$$commCost(i) = \sum_{j=1}^{m} opCost_i(C_j)$$

Therefore, the total communication cost *totalCommCost* for all the users can be calculated as follows

$$totalCommCost = \sum_{i=1}^{n} commCost(i)$$

The replicated server allocation problem can now be defined as finding the cells into which the available replicated servers should be located in order to minimize *totalCommCost* for satisfying the requests of all the users in the network.

3.2 The Edge-based Occurrence Count Algorithms

Peng and Chen presented a couple of replicated server allocation algorithms that consider *the user occurrence count* and *the movement occurrence count* [6]. Note that their algorithms do nothing when a miss occurs and concern only maximizing the hit ratios. Hence we modify their algorithms so that when a miss occurs for a request, the algorithms should search a nearest replicated server in the network.

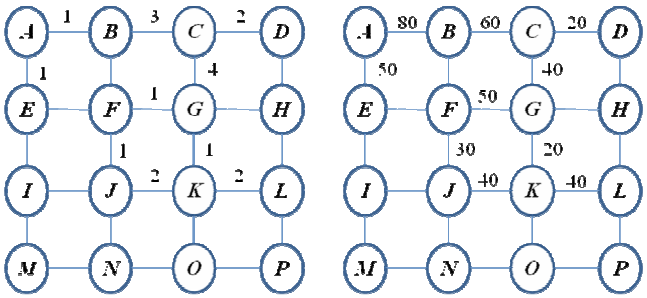
The user occurrence and movement occurrence counts are extracted from the users' moving patterns given as input. Each count can be regarded as the weight of the edge in the graph that corresponds to a given network. For example, if the moving pattern of a user is {*ABCG*, *AE*}, where *A*, *B*, …, and *G* are cells in the network and if the user passes through edge *AB* 80 times, *BC* 40 times, *CG* 5 times, and *AE* 50 times, then the user occurrence count of each edge of *AB, BC*, *CG* and *AE* is 1 since the user traversed from cell *A* to *B*, *B* to *C*, and then to *G* as well as from *A* to *E*, regardless of the frequency of traversals. The movement occurrence count of an edge is the number of times that a user traverses the edge. So the movement occurrence counts for edges *AB*, *BC*, *CG*, and *AE* are 80, 40, 5, and 50, respectively, for the user.

Among their algorithms, the first algorithm, called *the EUOC algorithm*, is based on the user occurrence counts. EUOC counts the total number of occurrences of each edge for all *n* users and then selects greedily two vertices (cells) that are the endpoints of the edge with the largest user occurrence count, and allocates a replicated server to each of the selected cells. EUOC selects the next vertices in the same manner repeatedly until there is no more replicated server left. The ties are broken by selecting the vertex of the edge whose other endpoint has already a replicated server. The second edge-based occurrence count algorithm, called *the EMOC algorithm*, behaves in the same way as EUOC except it utilizes the movement occurrence counts.

Table 1 shows the moving patterns of six users as an example. In the table, the moving pattern of user 1 is {*AB*, *BC*, *CG*, *AE*} and the numbers of moves are 80, 40, 5, and 50, respectively. Fig. 3 depicts two graphs representing a given network with the user and movement occurrence counts, respectively. Suppose that we have four replicated servers to be allocated, EUOC selects {*B, C, D, G*}, while EMOC selects {*A, B, J, K*}.

Table 1. Sample mobile users' moving patterns

	user 1	user 2	user 3	user 4	user 5	user 6
moving patterns of users	*AB* 80 *BC* 40 *CG* 5 *AE* 50	*BC* 10 *CD* 10 *CG* 20	*BC* 10 *CG* 10	*FG* 50 *FJ* 30 *JK* 20	*JK* 20 *KL* 30	*CD* 10 *CG* 5 *GK* 20 *KL* 10



(a) (b)

Fig. 3. (a) User occurrence counts (b) Movement occurrence counts with respect to Table 1.

3.3 The Proposed Replicated Server Allocation Algorithms

We now propose various replicated server allocation algorithms that utilize the occurrence counts on vertices instead of on edges.

3.3.1 The Vertex Occurrence Count Algorithm

VOC utilizes the movement occurrence counts on vertices, not on edges, since users request while they stay within the cells, not on the edges. The *vertex occurrence count* of each vertex is calculated by summing all the movement occurrence counts of the edges incident with the vertex. In Fig. 4, a number on each vertex indicates the vertex occurrence count of the vertex according to the moving patterns in Table 1.

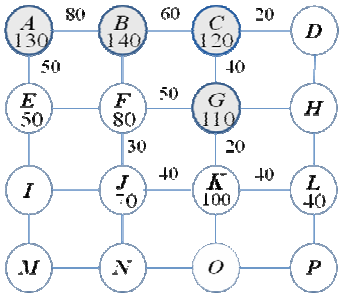


Fig. 4. The vertex occurrence counts for the moving patterns in Table 1.

VOC selects greedily a vertex with the largest vertex occurrence count among the unselected vertices and allocates a replicated server to the vertex, and repeats such greedy selection until there is no more replicated server available. The ties are broken in such a way that the vertex with the largest sum of the vertex occurrence counts of its adjacent vertices is selected for a replicated server. Suppose that we have four replicated servers to be allocated in the network. Then VOC will select the cells *B, A, C,* and *G* in sequence as shown in Fig. 4.

3.3.2 The VOC-Neighbor Reduction Algorithm

In order to improve the performance, we propose a modified VOC algorithm called *the VOC-neighbor reduction algorithm* (VOC-NR). There are tendencies that the selected cells by VOC are close to each other in the network. Hence we decrease on purpose the vertex occurrence counts of the neighboring cells of the cell chosen by VOC in the hope that the replicated servers

are allocated to the cells more dispersedly. The decrease amount d of a neighboring cell X with respect to the cell Y chosen by VOC is determined by the following equation; note that v_X and v_Y are the vertex occurrence counts of cells X and Y, respectively.

$$d = \begin{cases} v_{\max} \times \dfrac{1}{|\, v_X - v_Y \,|}, & \text{if} \quad v_X \neq v_Y \\ v_X \qquad\qquad\qquad , & \text{otherwise} \end{cases} \qquad \text{(Eq.1)}$$

where v_{max} is the largest vertex occurrence count among all the cells in the network. Hence the new value v_X becomes $v_X - d$. As the value of $|v_X - v_Y|$ gets smaller, we subtract a larger value from the vertex occurrence count. With such decrements, the adjacent cells to the cells chosen by VOC are not likely to be selected for replicated servers. In case when the value of $|v_X - v_Y|$ is large, d becomes so small that the algorithm cannot change the original vertex occurrence count too much.

3.3.3 The Greedy Set-Cover Algorithm

The third algorithm, called *the greedy set-cover algorithm* (GSC), utilizes the greedy approximation algorithm in [14] for selecting the cells. GSC first selects a cell with the largest vertex occurrence count, and allocates a replicated server into the cell. Then GSC ignores the neighboring cells to the selected cell, even if one of the neighbor cells has the second largest vertex occurrence count. Note that GSC works as if VOC-NR decreases the vertex occurrence counts of all the neighbors to zeroes. The algorithm greedily selects the cell with the largest count among the rest of cells and so on. The ties are broken arbitrarily.

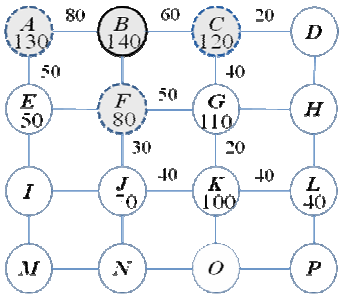


Fig. 5. The first cell selection of the greedy set-cover algorithm.

Fig. 5 shows that GSC selects cell B as the first cell for a replication server since it has the largest count. Then the neighboring cells, A, C, and F, won't be considered any more by the algorithm, even if one of those has the second largest count in the network. GSC then selects G, J and E in sequence.

3.3.4 The Replicated Server Clustering Algorithm

Both VOC-NR and GSC try to allocate replicated servers considering only neighboring cells in order to scatter the replicated servers in the network. However, such attempts may not fully contribute toward the performance improvement as the size of a wireless network increases. That is, both algorithms merely look at adjacent cells myopically (locally) rather than globally.

The fourth algorithm, called *the replicated server clustering algorithm* (RSC), is proposed to solve the problem globally by utilizing the k-means clustering method in [15]. The algorithm first selects k centers randomly but places them as far apart as possible. Such placement can be obtained with the k-clustering approximation algorithm in [16]. And then the algorithm

recalculates a new center for each cluster with the *k*-means clustering method repeatedly until no center is changed. RSC uses the following objective function to be minimized.

$$\delta = \sum_{X \in G_i - \{Y\}}^{i} dist(X,Y) \times v_X, \qquad \text{(Eq.2)}$$

where Y is the presumed center of cluster G_i, $i = 1, 2, ..., k$, and $X, Y \square G_i$.

That is, we try each cell Y in G_i as if it were the center, compute δ for Y, and then we choose the cell with the lowest δ as the new center of G_i. Notice that a new clustering with (Eq.2) may not guarantee to get better communication cost than the previous clustering, because the operation count on each cell was not considered. Therefore, the algorithm will change the current clustering only if the new clustering improves the communication cost. The algorithm is described formally below.

//Input: the moving patterns of the users and the number *k* of available replicated servers
//Output: the set *R* of cells to which the replicated servers are allocated

1. Calculate the vertex occurrence count and operation count of each cell based on the input moving patterns;
2. *R* ← the set of *k* cells selected as the centers with the *k*-clustering approximation algorithm;
3. Repeat the following until no centers are changed;
   a) Partition the cells in the network into *k* clusters whose centers are in *R*;
   b) *R*' ← the recalculated centers with the *k*-means clustering method using (Eq.2);
   c) If the communication cost of *R*' is less than that of *R*, then *R* ← *R*';

Algorithm 1. The replicated server clustering algorithm.

3.4 The Random Guided Search Techniques

3.4.1 The Simulated Annealing Algorithm

Simulated annealing is a well-known search technique as one of the generic randomized searches for global optimization problems [17]. The next algorithm, called *the simulated annealing algorithm* (SA), solves the problem with the simulation annealing technique. SA selects the first 'candidate' solution with VOC which provides a good starting position in the search space. Each step of SA replaces the current solution *R* with a 'nearby' solution randomly; that is, SA tries to change each cell in *R* to a cell in the left, the right, up, or down only if the new solution has a smaller cost than that of *R*. On the other hand, we unconditionally replace *R* with a new solution using the following probability

$$P_{temp} = e^{-\Delta cost / kT} > rand(0,1), \qquad \text{(Eq.3)}$$

where $\square cost$ denotes the difference between the communication costs of *R* and the new solution, *T* is a *synthetic temperature*, and *rand*(0,1) is a random number chosen from the interval [0,1]. Such a change provides chances to escape from local optima during the search. SA uses a cooling schedule T_i that is the temperature for cycle *i*, where *i* increases from 0 to *z*, and *z* is the number of cycles.

$$T_i = T_0 (\frac{T_z}{T_0})^{\frac{i}{z}} \qquad \text{(Eq.4)}$$

The initial temperature T_0, final temperature T_z, and the value of z are determined through various experiments. SA is described in detail below and its control parameters used for the experiments are given in Table 2.

```
//Input: the moving patterns of users and the number k of available replicated servers
//Output: the set R of cells to which the replicated servers are allocated
1. Calculate the vertex occurrence count and operation count of each cell based on the
   input moving patterns;
2. Initialize R with the output returned from VOC with the input;
3. Repeat the following until the termination condition is met;
   a) Obtain a new solution R' by selecting one of the neighboring (left, right, up, down)
      cells with respect to each cell in R only if the selected cell improves the cost most;
      // If none of the neighboring cells improves the solution, no changes in the cells will
      be made.
   b) R ← R' with P_temp in (Eq.3);
   c) If the communication cost of R' is less than that of R, then R ← R';
```

Algorithm 2. The simulated annealing algorithm.

Table 2. Control parameters for the simulated annealing algorithm

parameter	value
number of cycles (z)	70,000
initial temperature (T_0)	20,000
final temperature (T_z)	1
Boltzmann's constant (k)	1

3.4.2 The Genetic Algorithm

The genetic algorithm [18] is yet another powerful randomized search technique for optimization problems. The genetic algorithm might be useful in the problem domains that have complex fitness landscapes and is designed to escape from local optima, while a traditional hill climbing might get stuck in. The last algorithm is called the *genetic algorithm* (GA).

First, GA prepares the initial population, in which each individual candidate solution is constructed with randomly chosen k cells. Next, GA evaluates the fitness (the communication cost) of each individual solution in the population. The calculation of the communication cost is done as if we have assigned k replicated servers at the cells in each candidate solution. GA selects a set of candidate solutions from the current population whose fitness values are lower than those of others for the succeeding generation by a roulette wheel method. Next, crossover and mutation operations are performed with the probabilities of P_c and P_m, respectively. GA uses an one-point crossover as follows. For example, if {2, 4, 9, 11} and {5, 8, 13, 15}, are a pair of candidate solutions for crossover when k=4 and if the crossover point is 2, all cells beyond the second cell are swapped. Therefore, the results of the crossover are {2, 4, 13, 15} and {5, 8, 9, 11}. In case when crossover causes duplicate cells, the duplicate cells are not swapped. Mutation is performed by replacing a cell in individual solutions with the one that leads to less communication cost. GA is described in detail below.

```
//Input: the moving patterns of users and the number k of available replicated servers
//Output: the set R of cells to which the replicated servers are allocated
1. Calculate the vertex operation count of each cell based on the input moving patterns;
2. Construct the initial population randomly;
3. Repeat the following until no more progress is expected;
```

a) Evaluate the fitness of each individual solution in the population;
b) Perform a roulette wheel selection;
c) Perform the one-point crossover for individuals with probability P_c;
d) Perform mutation with probability P_m;

4. $R \leftarrow$ the individual solution with the lowest cost;

Algorithm 3. The genetic algorithm.

In the algorithm Step 3 is repeated for a fixed number of times. When the loop is terminated, the individual solution with lowest communication cost is returned as the final solution. The control parameters for GA are given in Table 3.

Table 3. Control parameters of the genetic algorithm

parameter	value
population size	400
number of generations	1,000
crossover rate (P_c)	0.6
mutation rate (P_m)	0.008

4. EXPERIMENTAL RESULTS

4.1 Simulation Model and Results

Each algorithm was implemented in C++ and executed on a PC with Core2 Quad 2.4 GHz and 4 GB main memory. In generating the inputs, the round trip model in [8] was used for the moving behaviors of mobile device users. The length of each moving path was chosen randomly with a uniform distribution on the interval [8, 12]. The number of data accesses (operation counts) made by a user was also chosen randomly with a uniform distribution on the interval [30, 70]. A user is supposed to move to one of its adjacent cells right after the user completes all the data accesses at the current cell. We tested 300 users and compared the communication costs of the proposed algorithms with those of EUOC and EMOC. We also compared the results with the optimal communication costs that were computed exhaustively.

Fig. 6 compares the average communication costs as the number of replicated servers varies and when the number of cells is 64. The average communication costs of all the algorithms are decreased as the number of replicated servers increases, because more replicated servers in the network help reduce the communication costs. VOC showed better performance than both EUOC and EMOC. VOC-NR and GSC further improved the average communication costs while they showed almost the same performance; both algorithms selected almost the same outputs. VOC-NR and GSC showed almost the same results throughout the experiments. Since lowering the vertex occurrence counts of the neighboring cells in VOC-NR showed the same effect as lowering the counts to zeroes in GSC. RSC demonstrated outstanding results compared with other algorithms, since it selects the cells globally; other algorithms are greedy algorithms that select a cell myopically in each iteration.

Fig. 7 compares the average communication costs as the number of the cells increases from 25(=5x5) to 81(=9x9). The number of the replicated servers was fixed to four during the experiments. The average communication costs increase for all the cases as the number of cells increases. It means that the coverage with four replicated servers is getting more difficult as the number of cells increases. However, VOC still showed better performance than EUOC and EMOC consistently. The results also showed that the algorithms performed in the almost same order of the communication costs as in Fig. 6.

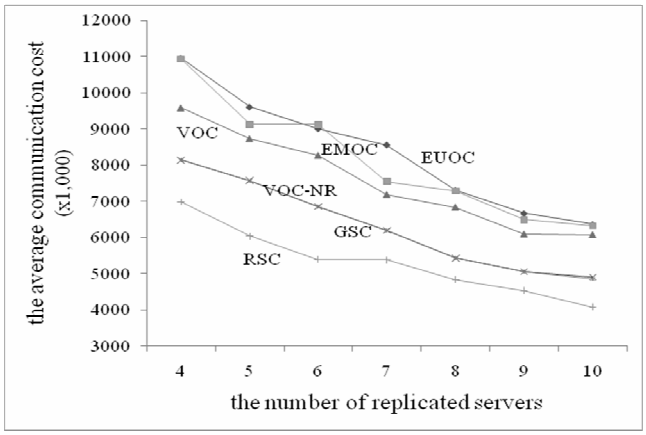


Fig. 6. The average communication costs as the number of replicated servers varies and when the number of cells is 64.

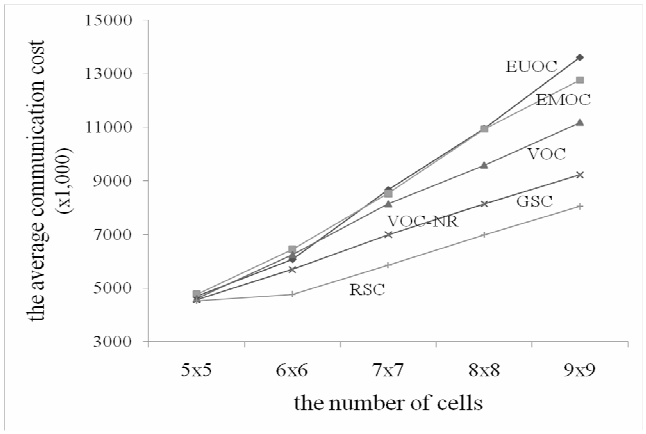


Fig. 7. The average communication costs as the number of cells increases and the number of replicated servers is 4.

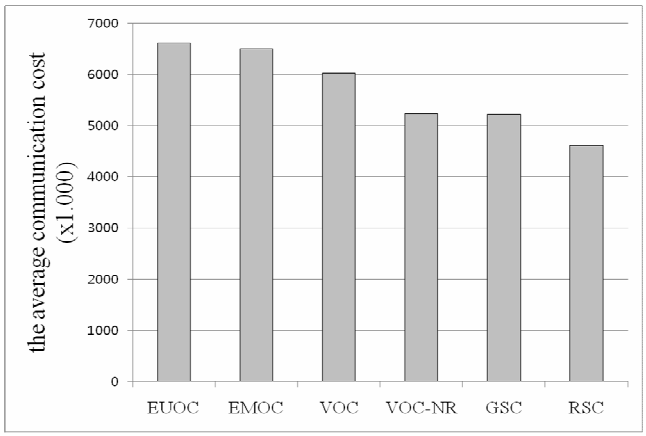


Fig. 8. The average communication costs for the algorithms.

Fig. 8 shows the average communication costs while the number of replicated servers varies from 4 to 10 and the number of cells from 25 to 81. EUOC and EMOC showed larger average

communication costs than VOC. It is verified that the vertex occurrence counts give more valuable information for replicated server allocation than the edge-based occurrence counts, since the file requests of each user are made at the cells, not on the edges (in-between cells). However, VOC could not produce better solutions than other proposed algorithms, because it does not consider the occurrence counts of the neighboring cells to the cells in its solution. Both VOC-NR and GSC improved the performances over VOC by trying not to allocate replicated servers to adjacent cells. RSC showed better performance than both VOC-NR and GSC, because RSC adjusted its solutions more globally.

Fig. 9 shows the average percentage of the gaps between the optimal cost and the cost obtained by each algorithm over all the inputs. Such a gap exhibits how accurately each algorithm allocated the replicated servers in the network to minimize the communication cost. Note that the optimal costs were obtained exhaustively for comparisons. RSC showed near optimal performance except SA and GA. Such performance was possible since RSC obtains the solutions with more global consideration. Both SA and GA found the solutions whose costs were almost the same as the optimal ones.

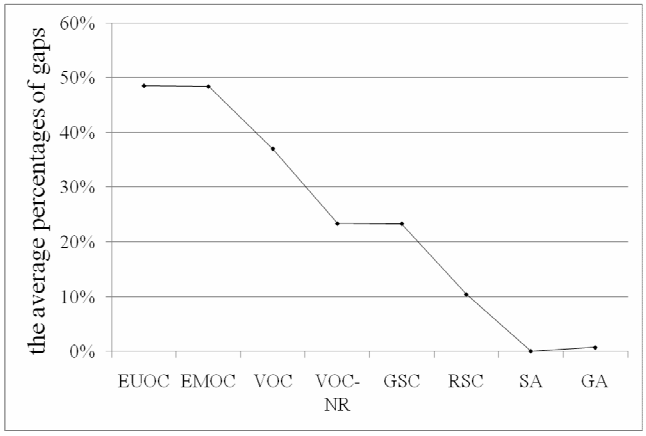


Fig. 9. The average percentage of a gap between the optimal values and the communication cost of each algorithm.

We compared the average execution times of SA and GA while the number of replicated servers varies from 4 to 10 and the number of cells from 5x5 to 9x9. SA and GA took about 6.4 seconds and 4.3 seconds, respectively. The average execution time of SA is longer than that of GA by about 2.1 seconds. This is because that SA searches candidate solutions more extensively than GA. The simulation results showed that SA and GA are not suitable for realistic environments since they take too much time. Note that each of other algorithms took less than one second.

5. CONCLUSIONS

In mobile computing system environments, decentralizing the replicated servers is a very critical issue for the system performance. However, previous researches on this issue did not consider the communication when a miss occurs. It is more natural that users are allowed to access the shared data through the replicated servers nearby in the network, rather than through the online servers.

We had proposed four replicated server allocation algorithms that utilize the moving patterns of the mobile device users to minimize the communication cost in accessing the shared data at the replicated servers in a wireless network. VOC selects greedily to allocate replicated servers based on the vertex occurrence counts. The experimental results showed that the vertex

occurrence counts give more helpful information for allocating replicated servers. VOC-NR and GSC improved successfully the performances over VOC by considering the neighbors to the cells in the solutions of VOC. RSC showed better performance among the aforementioned proposed algorithms, since it considers other cells throughout the entire network.

SA and GA allocated replicated servers with very near optimal communication costs since they search the solution space globally. However, they are not suitable solutions in practice due to long execution time. From the comparative research, we have found that RSC allocated replicated servers with near optimal communication costs.

ACKNOWLEDGEMENTS

This work was supported in part by the Korea Science and Engineering Foundation (KOSEF) for the research (2009-0073072).

REFERENCES

[1] S. Frattasi, H. Fathi, F.H.P. Fitzek, R. Prasad, M.D. Katz, "Defining 4G technology from the users perspective," *IEEE Network,* 2006.

[2] Changi Nam, Seongcheol Kim, and Hyeongjik Lee,"The role of WiBro: Filling the gaps in mobile broadband technologies," *Vehicular Technology Magazine*, 2008

[3] Kan Zheng, Lin Huang, Gang Li, Hanwen Cao, Wenbo Wang, Dohler, M.,"Beyond 3G Evolution," *Vehicular Technology Magazine*, 2008.

[4] Takeshita, K., Sasabe, M., and Nakano, H. 2008. "Mobile P2P Networks for Highly Dynamic Environments", In *Proceedings of the 2008 Sixth Annual IEEE international Conference on Pervasive Computing and Communications*, 2008.

[5] Jung-Suk Han , Kwang-Jo Lee, Jin-Woo Song, Sung-Bong Yang, "Mobile Peer-to-Peer systems using Super Peers," *The International Conference on Information Networking 2008*, June 2008.

[6] W. Peng and M. Chen, "Shared Data Allocation in a Mobile Computing System: Exploring Local and Global Optimization," *IEEE Transactions on Parallel Distributed Systems*, Vol.16, No.4, pp.374-384, Apr. 2005.

[7] E. Pitoura and G. Samaras, "Locating Objects in Mobile Computing," *IEEE Transactions on Knowledge and Data Engineering*, Vol.13, No.4, pp.571-592, July 2001.

[8] N. Shivakumar, J. Jannink, and J. Widom, "Per-User Profile Replication in Mobile Environments: Algorithms, Analysis and Simulation Results," *Mobile Network Applications*, Vol.2, No.2, pp.129-140, Sept. 1997.

[9] O. Wolfson, S. Jajodia, and Y. Huang, "An Adaptive Data Replication Algorithm," *ACM Transactions on Database Systems*, Vol.22, No.2, pp.255-314, June 1997.

[10] B. Ciciani, D. Dias, and P. Yu, "Analysis of Replication in Distributed Database Systems," *IEEE Transactions on Knowledge and Data Engineering*, Vol.2, No.2, pp.247-261, June 1990.

[11] S. March and S. Rho, "Allocating Data and Operations to Nodes in Distributed Database Design," *IEEE Transactions on Knowledge and Data Engineering*, Vol.7, No.2, pp.305-317, Apr. 1995.

[12] S. Jajodia, "Data Replication Gaining Popularity," *IEEE Concurrency*, Vol.7, No.2, pp.85-86, Apr. 1999.

[13] S. Lim and J. Kim, "Optimal Server Replication Schemes to Reduce Location Management Cost in Cellular Network," *IEICE Transactions on Communications*, Vol.E89-B, No.10, pp.2841-2849, Oct. 2006.

[14] V. Chvatal, "A Greedy Heuristic for the Set-Covering Problem," *Mathematics of Operations Research*, Vol.4, No.3, pp.233-235, Aug. 1979.

[15] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," *Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, University of California Press, Vol.1, pp.281-297, 1967.

[16] S. Dasgupta, C. Papadimitriou, and U. Vazirani, ed. *Algorithms*, McGraw-Hill, pp.279-281, 2006.

[17] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by Simulated Annealing*," Science*, Vol.220, No. 4598, pp.671-680, May 1983.

[18] J. Periaux, and G. Winter, ed. *Genetic Algorithms in Engineering and Computer Science,* John Wiley & Son, 1995.