2008

Managing continuous k-nearest neighbor queries in mobile peer-to-peer networks

Patricio A. Galdames S
Iowa State University

Managing continuous k-nearest neighbor queries in mobile peer-to-peer networks

by

Patricio A Galdames S

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Ying Cai, Major Professor
Wensheng Zhang
Daji Qiao

Iowa State University

Ames, Iowa

2008

DEDICATION

This thesis is dedicated to my parents, Carmen and Humberto, who taught me that even the largest task can be accomplished if it is done one step at a time and dedicated work. To my dear Lilian for her unconditional love and inspiration throughout the course of this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost, Dr. Ying Cai for his extraordinary guidance, patient encouragement and insightful criticisms throughout this research and the writing of this thesis. He has been an inspiration to me and his technical insight, multi-disciplinary knowledge and personality have presented me with precious pearls of wisdom. I am also grateful to Dr. Wensheng Zhang and Dr. Daji Qiao for serving on my committee.

I thank my dear friend Jose Ayala, Lizandro Solano and Kihwan Kim for sharing their knowledge and time towards discussions surrounding this thesis. I am especially grateful to Mrs. Bessie Fick and many friendly people from the Memorial Lutheran Church that have made my life in Ames easier. Last, and by no means the least, I wholeheartedly appreciate the Fulbright Commission and CONICYT - Chile for funding my studies in Iowa State University and the Department of Computer Science at Iowa State University for giving me the opportunity to improve my education.

ABSTRACT

A continuous k nearest neighbor (CKNN) query retrieves the set of k mobile nodes that are nearest to a query point, and provides real-time updates whenever this set of nodes changes. A CKNN query can be either stationary or mobile, depending on the mobility of its query point. Efficient processing of CKNN queries is essential to many applications, yet most existing techniques assume a centralized system, where one or more central servers are used for query management. In this thesis, we assume a fully distributed mobile peer-to-peer system, where mobile nodes are the only computing devices, and present a unified platform for efficient processing of both stationary and mobile CKNN queries. For each query, our technique computes a set of safe boundaries and lets mobile nodes monitor their movement with respect to these boundaries. We show that the result of a query does not change unless a node crosses over a safe boundary. As such, our technique requires a query to be re-evaluated only when there is a crossing event, thus minimizing the cost of query evaluation. For performance study, we model the communication cost incurred in query processing with a detailed mathematical analysis and verify its accuracy using simulation. Our extensive study shows that the proposed technique is able to provide real-time and accurate query results with a reasonable cost.

CHAPTER 1. OVERVIEW

Given a set of mobile nodes, a *continuous k-nearest-neighbor* (CKNN) query retrieves the set of k nodes that are nearest to a query point p, and provides real-time updates on the query results whenever this set of nodes changes. Depending on whether or not the query point moves during a query's life time, CKNN queries can be classified into two categories, *stationary* and *mobile*, which we will refer to as *S-CKNN* and *M-CKNN* queries, respectively. Unlike traditional KNN queries that retrieve the set of k nearest neighbor at some snap of time point, a CKNN query is a continuous query; as mobile nodes move continuously, the query results may keep changing. By allowing one to monitor the k nodes nearest to a point of interest, CKNN queries are useful in many applications. For instance, in a battlefield, a commander may want to alert his soldiers as soon as they become nearest to an enemy post. In this example the query point is fixed at the enemy post location. Similarly, in an emergency, a police officer may want to track the ambulance that is closest to his/her current position. In this case, the query is mobile because the query point is bound on the police officer.

As a primitive operation in mobile object database systems, efficient processing of CKNN queries has been investigated intensively in the context of a centralized environment, wherein mobile nodes can communicate with a central server. When the query point is fixed, the proposed techniques rely on the server for query management, and can be classified into two categories. The techniques in the first category (e.g., [19], [29], [30]) let mobile nodes report the server their velocity information. At the server side, the trajectories of mobile nodes are indexed using some spatial data structure to minimize disk I/O and CPU costs in query evaluation. In these schemes, the server handles all query evaluation and mobile nodes just need to report their location information. In contrast, the techniques in the second category

(e.g., [28], [20], [10]) are distributed in the sense that they let mobile nodes participate in query evaluation. Specifically, each mobile node needs to monitor their movement with respect to some monitoring boundaries. When a node crosses over a boundary, it reports the server. In response, the server checks if any query result needs to be updated.

Recently, Kim et al. [17] presented a fully distributed approach, called Safe-Time, for processing stationary CKNN queries. The technique assumes that the maximal speed of the mobile nodes is known and based on such knowledge it estimates the minimal time duration during which the $k+1$-th nearest neighbor and the k-th nearest neighbor to a query point do not change. However this approach has some drawbacks. First, When the speeds of mobile nodes vary significantly, this approach would incur frequent location update. Indeed, it requires location update even when the $k+1$-th and k-th nearest nodes do not change. Second, if the k-th nearest neighbor node fails, the query result may be outdated and the query itself may be lost. Finally, the proposed technique can support only S-CKNN queries.

In this thesis, we present a generic solution that support efficient processing of both S-CKNN and M-CKNN queries. Our key innovation is the concept of *safe boundary*. For each CKNN query, we compute a set of safe boundaries, which are circles centered on the query point, and guarantee that as long as no mobile node crosses over those boundaries, the set of K nearest nodes to the query point does not change. Since the query needs to be re-evaluated only when there is a crossing event, a significant amount of communication overhead can be saved. To further reduce the cost, we partition the network into a number of disjoint grid cells, and make each node aware of only its relevant queries, defined to be those whose safe boundaries overlap with the cell a node resides. As a node moves, it monitors its movement against the boundaries of its relevant queries, and if it crosses over a boundary, it initiates a query evaluation and computes a new safe boundary for the corresponding query. When a node moves into a new cell, it can retrieve its new relevant queries from any other nodes in the cell. In the case that the new cell does not have any other nodes, our technique ensures that its relevant queries can be retrieved from some node in a neighboring cell, thus minimizing the cost of query management. We show that when the network is fully connected, our technique

is able to provide real-time and accurate query results.

The main contributions of our work are as follows. 1) We propose a generic solution for efficient processing of both S-CKNN and M-CKNN queries. Instead of relying on any stationary server, our technique stores queries among mobile hosts and does so dynamically to make each host aware of its nearby queries. 2) We present an analytical model that estimates the communication cost incurred by the proposed technique. The accuracy of this model is validated using simulation. The rest of the thesis is organized as follows. We present our proposed work in detail in Chapter 2. In Chapter 3, we present and validate an analytical model based on a fully connected ad hoc network. Then, in Chapter 4 we demonstrate the effectiveness of our scheme by simulating and ad hoc network under different network conditions. Also, we present related work in Chapter 5. Finally, we offer some concluding remarks in Chapter 6.

CHAPTER 2. PROPOSED TECHNIQUE: SAFE-BOUNDARY

We consider a mobile ad hoc network formed by a set of location-aware mobile nodes, each having a unique ID. Since many MANET communication protocols (e.g., LAR [18], DREAM [22], GPSR [15]) have been developed, we will simply assume these nodes can communicate with each other through packet relaying and will not concern how this is actually implemented. Without causing ambiguity, we will use terms node and user exchangeably. In the following subsections, we first present our technique for efficient processing of S-CKNN queries, and then extend it to support M-CKNN queries.

2.1 S-CKNN query management

When a user issues a CKNN query on point p, the system needs to return the k nodes that are nearest to the query point p, and provide continuous update whenever the query result changes. Let N_k be the k-th nearest node to p with a distance of d_k, and N_{k+1} be the $(k+1)$-th nearest node to p with a distance of d_{k+1}. Let b be the circular boundary that centers on p with a radius of $\frac{d_k+d_{k+1}}{2}$. These notations are illustrated in Figure 2.1. We observe that the set of k nodes nearest to p does not change as long as no node moves across b. As such, to process the CKNN query, we just need to make mobile nodes aware of b, which we will referred to as the query's *safe boundary*. When a node moves, it can monitor its movement against b. If it crosses over the boundary, it computes a new safe boundary by identifying the $k+1$ nodes nearest to p and informs the query creator of the new set of k nearest nodes.

The above approach converts the processing of a CKNN query into monitoring the crossing events on its safe boundary. Given a set of n active CKNN queries, there are n corresponding safe boundaries in the system. The problem now is how to manage these boundaries. A simple

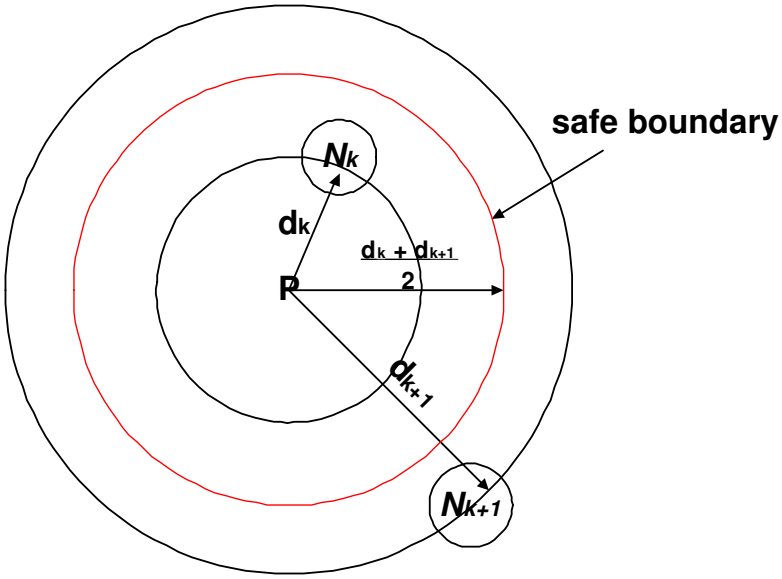


Figure 2.1 Safe Boundary

solution is to let each node cache the entire set of boundaries, but it will incur high network costs. Each time a new safe boundary is computed, a network-wide broadcast is needed. In the next section, we propose a cost-effective solution. We first explain the basic idea and then present the solution in a more formal way.

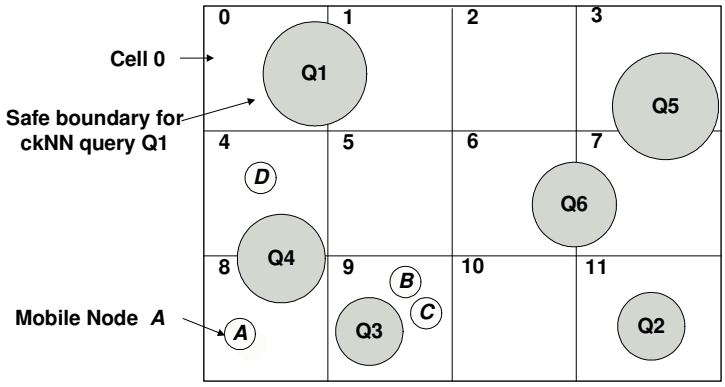


Figure 2.2 Query management

2.1.1 Basic idea

Our basic idea is to let each mobile node cache only its nearby boundaries. Let r is the transmission radius of mobile nodes. We partition the network domain into a set of disjointed cells, each being $\frac{r}{\sqrt{2}} \times \frac{r}{\sqrt{2}}$. An example of such partitioning is illustrated in Figure 2.2. The network partitioning is made known to all mobile nodes. We say a cell is a node's home cell if the node is currently inside the cell, and a query is relevant to a cell if the query's safe boundary overlaps with the cell. A query is relevant to more than one cell if its safe boundary spans over several cells. For instance, Q_3 in Figure 2.2 is relevant only to cell 9 while Q_5 is relevant to cells 3 and 7. Moreover, we say a query is relevant to a node if the query is relevant to the node's home cell.

When a node N creates a CKNN query on point p, it first determines the cell that contains point p, and then broadcasts a message to all nodes in the cell to search the initial set of $k+1$ nodes nearest to p. If the cell does not have all these nodes, N expands the search scope to include the cell's neighboring cells. This step is repeated until the $k+1$ nodes nearest to p are all located. N then computes the safe boundary b and broadcasts the query to all nodes inside the relevant cells. In our approach, each mobile node caches all queries relevant to its home cell. Thus, when a node moves into a new cell, it can find its relevant queries from any node in the cell.

One technical problem here is how to handle the situation when a cell does not have any node, in which case a node moving into the cell may not find its relevant queries. To address this problem, we use the following approach. When a node moves out of its home cell, it checks if it is the last node in the cell. If there are some other nodes in the cell, the node can simply delete the queries that are relevant to this cell. Otherwise, it keeps these queries, and becomes the cell's *retaining node*. For example, if node A in Figure 2.2 is moving out of cell 8, it will become the retaining node to this cell. Note that a cell can have at most one retaining node; and this happens only when there is no node inside the cell. When a node becomes a cell's retaining node, it keeps the cell's relevant queries until they are retrieved when some node moves into the cell.

The concept of retaining node allows a node to retrieve a cell's relevant queries from its retaining node, if the cell contains no other node. However, the cost of locating a cell's retaining node can be high if the node has moved far away from the cell. This cost can be minimized by keeping a cell's retaining node as close as possible to the cell. Suppose node H is the retaining node to cell i, and H is moving from its current cell j into a new cell k. If the distance from k to i is farther than that from j to i, then H can send i's relevant queries to another node, if any, in cell j, which then becomes i's new retaining node. As an example, suppose node A in Figure 2.2 moves from cell 8 to 9, and then to 10. If A is the retaining node to cell 8, it may unload the cell's relevant queries to either B or C.

When the network is fully connected, the above technique ensures that a node moving into a new cell can always retrieve its relevant queries, either from another node currently in the cell or from the cell's retaining node (if the cell contains no other node), which is likely to be nearby. This scheme also guarantees real-time and accurate query results. Since each node is made aware of the queries relevant to its home cell, it can monitor its movement against them and re-evaluate a query whenever necessary. In reality, the network may be disconnected due to various factors such as node failures. When this happens, no technique can provide accurate query results. To minimize the impact of network disconnection, we make some minor revisions on the original design. When a node fails to create a query, it can choose to repeat the operation until it is successful. When a node becomes a cell's retaining node, it can periodically try to geocast the relevant queries back to the cell. In this way, the queries can be restored as soon as the network path to the cell is reconnected.

Another issue to consider is synchronization. For instance, two nodes may simultaneously cross over a query's safe boundary and both starts to re-evaluate the query. This problem can be addressed by dynamically assigning each cell a coordinator to synchronize query evaluation and other events. When the network is initially deployed (e.g., all nodes are airlifted to some region in a battlefield and then start to move), a cell's coordinator is randomly chosen among the nodes. If a node moves into a cell containing no other nodes, it locates the node's retaining node and becomes the cell's coordinator. If a cell's coordinator is moving out of the cell, it

randomly assigns another node in the cell as its new coordinator. If there is no other nodes in the cell, the node remains as the cell's coordinator (and also a retaining node).

2.1.2 Detailed design

We now describe the proposed technique in a more formal way. Because of relevance, we focus on *MessageListener* and *RegionMonitor*, two daemon processes running concurrently on each mobile node. To facilitate our discussion, we define the following notations for the data structures maintained by each mobile node:

- *myID*: The node's unique identifier;
- *myPos*: The node's current position;
- *myCell*: The node's current home cell;
- *myQueries*: The list of queries relevant to *myCell*;
- *myRetains*: The list of cells to which the node is currently a retaining node and their relevant queries, each defined as (N, p, k, b), where N is the query creator, p the query point, k the number of nearest nodes, and b the current safe boundary;

MessageListener: The mobile node listens to these messages and processes them as follows:

- $SearchNode(cell)$: If $myPos$ is inside $cell$, or $cell$ is listed in $myRetains$, reply with a $candidate$ message including $myID$ and $myPos$.
- $RetrieveQueries(cell)$:
  - Send all queries relevant to $cell$ to the requester;
  - If $cell$ is listed in $myRetains$, remove the cell and its relevant queries from $myRetains$.
- $SetRetainingNode(cell, queries)$:
  - Add $cell$ and $queries$ to $myRetains$.

- *CreateQuery*(q):
  - If q's safe boundary overlaps with *myCell*, add q to *myQueries*;
  - If q's safe boundary overlaps with any cell listed in *myRetains*, add q to the list of the cell's relevant queries.
- *RemoveQuery(q)*:
  - If query q overlaps with *myCell*, remove q from *myQueries*;
  - If q's safe boundary overlaps with any cell listed in *myRetains*, remove q from the list of the cell's relevant queries.
- *UpdateSafeBoundary(q, b)*:
  - If q is included in *myQueries*, remove q if b does not overlap with myCell; otherwise update q with the new safe boundary b;
  - For each cell listed in *myRetains* that was relevant to q, remove q if b does not overlap with *myCell*; otherwise update q with the new safe boundary b.

RegionMonitor: When the mobile node moves, it monitors its movement and does the followings:

1. If it moves into a new cell, say *newCell*, do the followings:
   - Set $CandidateList = null$;
   - Broadcast message $SearchNode(myCell)$ within *myCell*;
   - Collect all *candidate* messages and add them to $CandidateList$;
   - If $CandidateList = null$, add *myCell* and *myQueries* to *myRetains* (becoming the cell's retaining node);
   - Otherwise, do the followings:
     - For each cell c (if any) in *myRetains*, check its distance to *myCell* and to *newCell*;

  - If c is closer to $myCell$ than to $newCell$, then find the node in the $CandidateList$ that is nearest to c, and send a message $SetRetainingNode(c, queries)$ to the node, where $queries$ is the list of queries relevant to c.
- Set $myCell = newCell$;

2. If it crosses over the safe boundary of any query $q(N, p, k, b)$ listed in the $myQueries$, do the followings:
   - Locate the $k + 1$ nodes nearest to p;
   - Notify N with the new set of k nearest nodes;
   - Compute a new safe boundary b', according to the positions of the kth and $(k+1)$th nodes nearest to p;
   - Broadcast $UpdateSafeBoundary(q, b')$ to all cells overlapping with b.

2.2 M-CKNN query management

In this section we extend the proposed technique described in section 2.1 to process M-CKNN queries.

2.2.1 Basic idea

The difference between an S-CKNN and an M-CKNN query is that in the latest, the geographic coordinates that describe its query point depend on the location of its focal node or query's owner. Thus, the query result of an M-CKNN not only may change because of the movement of nearby mobile nodes, but also because of the movement of its associated focal node. Our basic idea is to associate a new safe boundary to the focal node. This new safe boundary termed as the *inner safe boundary* will encircles a region around the focal node, in which the set of the k nearest neighbor nodes will not change because of its movement. The safe boundary that encloses the k nearest neighbor nodes to the query point is denoted as the *outer safe boundary* of a M-CKNN query.

Let N_k be the k-th nearest node to the query point p with a distance of d_k, and N_{k+1} be the $(k+1)$-th nearest node to p with a distance of d_{k+1}. Let b_S be the outer safe boundary that centers on p with a radius denoted as $R_S = \frac{d_k + d_{k+1}}{2}$. If we consider that the mobile nodes move at the average speed v, and assuming the worst case situation where node N_k is moving away from the position of the focal node (p) and node N_{k+1} and the focal node are moving toward each other as illustrated in Figure 2.3. Thus we define the radius of a inner safe boundary (b_F) as the maximum distance that the focal node (p) can move without affecting the current query result as $R_F = \frac{d_{k+1} - d_k}{4}$. Therefore, given a set of n active M-CKNN queries, there are $2n$ corresponding safe boundaries in the system.

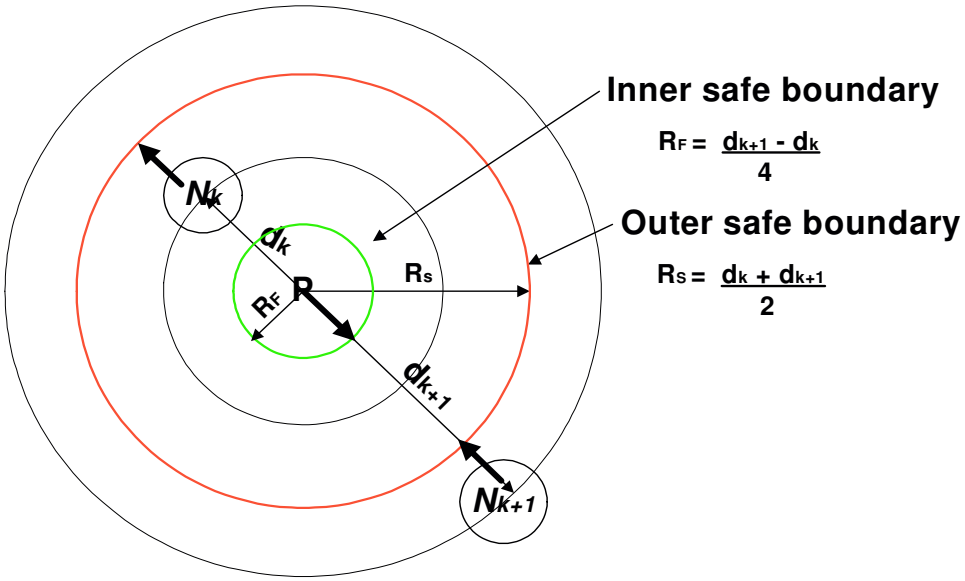


Figure 2.3 Safe Boundaries for a M-CKNN Query

When a mobile node moves, it monitors its movement against a inner safe boundary, b_S, only if the node is the query's owner. If the node crosses the inner safe boundary, then it computes a new pair of safe boundaries b_F and b_S by identifying the k+1 nodes nearest to its new position p. However, if the node is not the query's owner, it will only monitor its location against the outer safe boundary of the query.

Because of the fact that the query point can be constantly changing, the position of the

center of the outer safe boundary b_S also should be updated. However, in order to avoid unnecessary updates, a mobile node monitors its movement against a modified version of the outer safe boundary B_S of the query. In this case, the node monitors its position either against an outer safe boundary of radius $R_S + R_F$ if the node is moving outside of the region delimited for such a radius or against an outer safe boundary of radius $R_S - R_F$ if the node is member of the query result. In either case, when a node crosses one of these modified safe boundaries, it will geocast a request for a safe-boundary update to the region where the query's owner or its focal node is probably located. Such a region is considered to be the one defined by the previous known position of p and the radius R_S. If the corresponding focal node receives such a request, it will identify the k+1 nodes nearest to its current position p.

Due to the processing of M-CKNN requires the use of two safe boundaries, it is necessary to differentiate between the safe boundaries of an S-CKNN and an M-CKNN. This is done by including a new field, in the data structure that describes the safe boundary of a query, that indicates to which type of query is associated.

2.2.2 Detailed design

We know describe in a more formal way the necessary extension to the design explained in section 2.1.2 to the daemons *MessageListener* and *RegionMonitor*. To facilitate our discussion, beside of the notations explained in section 2.1.2, we include the followings:

- b_F: is the inner safe boundary of a query q defined as (p,R_F), where p is the query point and R_F is the radius of the inner safe boundary.
- b_S: is the outer safe boundary of a query q defined as (p,R_S), where p is the query point and R_S is the radius of the outer safe boundary.

MessageListener: The mobile node listens also to these messages and processes them as follows:

- $RequestSafeBoundaryUpdate(q)$:
  - if owner of query q is $myID$ do:

  - ∗ Locate the $k+1$ nodes nearest to p;
  - ∗ Notify N with the new set of k nearest nodes;
  - ∗ Compute a pair of new safe boundaries b'_F and b'_S, according to the positions of the kth and $(k+1)$th nodes nearest to p;
  - ∗ Broadcast $UpdateSafeBoundary(q, b'_F, b'_S)$ to all cells overlapping with b'_S.

- *UpdateSafeBoundary(q, b_S,b_F)*:
  - If q is included in *myQueries* and q's owner is not $myID$, remove q if b_S does not overlap with myCell; otherwise update q with the new safe boundaries b_S and b_F;
  - For each cell listed in *myRetains* that was relevant to q, remove q if b_S does not overlap with *myCell*; otherwise update q with the new safe boundaries b_S and b_F.

RegionMonitor: When the mobile node moves, it monitors its movement and does the followings:

1. For query $q(N, p, k, b_S, b_F)$, whose owner is $myID$, listed in the *myQueries*
   - If it crosses over the inner safe boundary, b_F, of q, do the followings:
     - Locate the $k+1$ nodes nearest to p;
     - Notify N with the new set of k nearest nodes;
     - Compute a pair of new safe boundaries b'_F and b'_S, according to the positions of the kth and $(k+1)$th nodes nearest to p;
     - Broadcast $UpdateSafeBoundary(q, b'_F, b'_S)$ to all cells overlapping with b'_S.
2. For any query $q(N, p, k, b_S, b_F)$, whose owner is not $myID$, listed in the *myQueries*
   - If it crosses into the outer safe boundary whose center is p and radius $R_S + RF$, do the followings:
     - Broadcast $RequestSafeBoundary(q)$ to all cells overlapping with b_F.
   - Else if it crosses out of the outer safe boundary whose center is p and radius $RS-RF$, do the followings:

– Broadcast $RequestSafeBoundary(q)$ to all cells overlapping with b_F.

CHAPTER 3. ANALYTICAL MODELS

3.1 Mobility model

In order to simulate or analysis a new protocol for an ad hoc network it is important to use a mobility model that represents the mobile hosts that will use the given protocol in the future. During the last few year many mobility models for MANET have been proposed [5] but the most referenced models are the Random Walk Mobility Model [7] and the Random Waypoint Mobility Model [12]. Even tough these mobility model may not represent accurately the movement of mobile hosts, but because of their simplicity, they have been used extensively to evaluate many protocols as a first step testing. Although it is preferable to test a new protocol with real traces, in most of the cases it is quite difficult to obtain them [3]. Although our work can be easily adapted to operate under different mobility models, because of its simplicity and its broad usage in modeling the mobility of mobile hosts, we have chosen to use the Random Walk Mobility Model [11]. In this model, a mobile host moves from its current location to a new location by randomly choosing a direction and speed in which to move and the speed is chosen randomly from a pre-defined range $[Speed_{min}, Speed_{max}]$.

3.2 Analytical model for managing S-CKNN queries

As we mentioned in chapter 2, our technique divides the network domain in a set of adjacent cell of size $\frac{r}{\sqrt{2}}$ x $\frac{r}{\sqrt{2}}$, where r is the host transmission radius. Thus, when a host broadcasts a message, it covers the entire cell where the host resides. Therefore the communication cost will be dependent of the value of r. The communication cost of the safe-boundary comes from two sources: 1) when a host moves into a new cell, and 2) when a new safe-boundary of a query has to be geocasted. In the first case, the host needs to retrieve the new cell's relevant queries,

while in the second case, the safe-boundary needs to be broadcasted to its relevant cells. In this subsection, we analyze the communication cost of a network partition that depends on a fixed value of r and we propose an analytical model, which is posteriorly validated by some simulations.

We assume the network is dense enough that all hosts are connected. Many techniques can be used for regional broadcast (e.g., broadcasting within a fixed region or a number of hops) and they can result in very different communication costs. In fact, the cost itself can have different measures, e.g., the number of packets sent and/or received, or the size of total network traffic, etc. To avoid assuming some particular technique and/or specific metric, we simply assume the cost of 1-hop broadcast is K and the cost of broadcasting a fixed region is proportional to the area of the region. Thus, the cost of broadcasting within a range of i hops is $C_i = i^2 \cdot k$.

Based on the analytical model developed by Cai et al. [4]. We summarized the assumptions considered in our analytical model:

- Mobile nodes are uniformly distributed in the service area.
- All mobile nodes are reachable by one or more hops communication. Thus, nodes must relay packets that are not sent to them.
- The area of every cell is covered in one hop transmission.
- The mobility pattern of a node is modeled by a random walk.

3.2.1 Cost of retrieving relevant queries

When a host h moves into a cell g, it needs to retrieve the cell's relevant queries. We call this host a requesting host. If there are other hosts inside the cell, any of them can supply the queries. Since the cell is covered entirely by a host's transmission range, the search cost in this case is just one-hop broadcast:

$$Cost_{retrieval} = k \tag{3.1}$$

However, if the the supplying host is not located at one hop broadcast, we will need to locate the retaining host. Suppose the retaining host is d-hops away from the requesting host. Then it will take d rounds of search to locate this retaining host. The total cost can be calculated according to this formula:

$$cost(d) = \sum_{i=1}^{d} i^2 \cdot c = \frac{d(d+1)(2d+1)}{6} \cdot k \tag{3.2}$$

Thus, we can conclude that the cost of broadcasting a message d-hops away is proportional to d^3. Beside, we need to estimate how frequent such cost is incurred by the mobile nodes. Such frequency depends on the mobility pattern of the mobile hosts. If we assume that the mobility pattern of a mobile host is represented by a random walk in 2D [26], we could assume according to Thomas et al. [25], the residence time for a random walk in two dimensions is:

$$\bar{t} = \frac{\pi A}{E[v]L}, \tag{3.3}$$

where A is the area of the cell, L is the perimeter of the cell, and $E[v]$ is the average speed of the mobile unit.

Since there are N mobile hosts uniformly distributed in the network area and each cell is a closed region of area A then the average number of mobile host per cell is $N_g = N\frac{A}{A_T}$, where A_T is the area of the entire network region. According to equation 3.3, and since there are N_g mobile hosts per cell, then the frequency of the events that some host moves out its current cell is:

$$Freq_{retrieval} = \frac{4\sqrt{2}E[v]N_g}{\pi r} \tag{3.4}$$

Each time a host moves into a new cell, it needs to request for the cell's relevant queries. Thus, the average cost per time unit for a query retrieval at the network wide can be calculated as:

$$Cost_{retrieval\_ptu} = Cost_{retrieval} \times Freq_{retrieval} \times H_{rc}^3 \tag{3.5}$$

Where H_{rc} is the average number of hops needed to reach the supplying host. The previous equation can also be expressed as:

$$Cost_{retrieval} = k\frac{4\sqrt{2}E[v]N_g}{\pi r} \times H_{rc}^3 \tag{3.6}$$

Therefore the total cost of retrieving relevant queries per time unit incurred in the entire network region is:

$$Cost_{retrieval} = \sum_{i=1}^{M}(k\frac{4\sqrt{2}E[v]N_{g_i}}{\pi r} \times H_{rc_i}^2) = k\frac{4\sqrt{2}E[v]N}{\pi r} \times H_{rc}^3 \tag{3.7}$$

Where M is the total number of cells used to divide the service area. In our case M can be calculated as $= \lceil \frac{A_T}{A} \rceil$. Because of the uniformity assumption, the average number of hops needed to reach the supplying or retaining host is H_{rc_i}, which is the same for every cell. Therefore $H_{rc_i} = H_{rc}\ \forall i \in [1, .., M]$. In this work, we use the model developed by Cai et al. [4], which computes the probability to find the retaining host in i hops.

3.2.1.1 Cost of updating S-CKNN queries

We now analyze the communication cost incurred in updating a safe boundary. Suppose the diameter of a safe boundary of a $\bar{k}NN$ query is l and the safe boundary is inscribed inside a square of size $l \times l$. The cell that contains the center position of this boundary can be seen as a concatenation of a number of rectangles. Let $a = \lceil \frac{l}{d} \rceil$ and $b = \lceil \frac{l}{d} \rceil - \frac{l}{d}$, where d is the cell size. Figure 3.1 shows these rectangles and their sizes.

These rectangles can be classified into three categories: R_{1X} (including R_1 along), R_{2X} (including R_{21}, R_{22}, R_{23}, and R_{24}), and R_{3X} (including R_{31}, R_{32}, R_{33}, and R_{34}). The number of cells that overlaps with the query depends on where the query point is located:

- Case 1: If the position is inside R_1, thequery overlaps with $(a+1)^2$ cells;
- Case 2: If the position is inside some R_{2X} rectangle, the query overlaps with a^2 cells;
- Case 3: If the position locates in some R_{3X} rectangle, the number of cells that overlap with the query is $a \cdot (a+1)$.

Suppose the safe boundaries (or its related queries) are uniformly distributed in the network domain. Then the probability of each case is proportional to the size of its corresponding area. Specifically, the probability of case 1 is b^2; the probability of case 2 is $(1-b)^2$; and the probability of case 3 is $2b(1-b)$. Thus, the average number of cells that overlap with a query is:

$$\begin{aligned} n_o &= a^2 \cdot b^2 + \\ &\quad (a+1)^2 \cdot (1-b)^2 + \\ &\quad a(a+1) \cdot 2b(1-b) \\ &= (a-b+1)^2 \\ &= (\tfrac{d+l}{d})^2 \end{aligned} \tag{3.8}$$

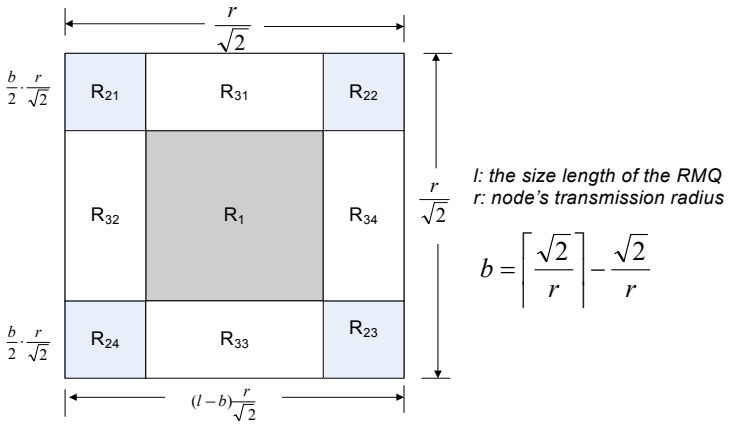


Figure 3.1 Possible regions for a query point

We can estimate the average radius ($R = \frac{l}{2}$) of a safe boundary as follows. Assuming that N mobile hosts are uniformly distributed and the average value of K of a nearest neighbor query is $\bar{k}$. If we denote the average distance from the query point to the $\bar{k}$-th nearest node as $\bar{d}_k$ and the respective average distance to the $\bar{k}+1$-th nearest node as $\overline{d_{k+1}}$. Then we can estimate those values from the probability distribution of the $\bar{k}$-th and $\bar{k}+1$-th order statistics [1] of the probability distribution of the distance to the query point using an approximated formula [24] or using a Monte Carlo method [23]. Then the average radius of the a safe boundary is

estimated as:

$$R_S = \frac{\bar{d_k} + \overline{d_{k+1}}}{2} \tag{3.9}$$

Thus, based on the fact that a cell of size $d \times d$ can be geocasted in one hop-broadcast, then the cost of sending the location request to the cells that may contain the the $k+1$ nearest nodes is approximately:

$$\begin{aligned} Cost_{locate} &= n_o \cdot Cost_{geocast}(d) \\ Cost_{locate} &= n_o \cdot c \end{aligned} \tag{3.10}$$

In order to collect the positions, we can use any location-based data aggregation technique, like the ones studied by Chen [6]. If we assume that the positions are aggregated per cell and P_o is the probability that a cell is empty, then the cost of collecting the positions of the nodes located in the searching region is approximately proportional to the number of nodes in a cell:

$$Cost_{collect} = c \cdot (1 - P_o) \cdot n_o \cdot \frac{N}{M} \tag{3.11}$$

The cost of geocasting the new safe boundary to mobile nodes located in the cells overlapped by this safe boundary is approximately:

$$\begin{aligned} Cost_{geocasting} &= c \cdot (1 - P_o) \cdot n_o \cdot Cost_{geocast}(d) \\ Cost_{geocasting} &= c \cdot (1 - P_o) \cdot n_o \end{aligned} \tag{3.12}$$

Therefore the cost of updating a safe boundary can be estimated as:

$$Cost_{updating} = Cost_{locate} + Cost_{collect} + Cost_{geocasting} \tag{3.13}$$

3.2.2 Overall cost per time unit

Now we will conclude with some formulas to estimate the retrieval and update cost per time unit incurred in the entire system.

Each time a node moves into a new cell, it needs to ask for the cell's relevant queries. According to equation 3.4 the average cost per time unit to retrieve the cell's relevant queries at the network wide can be calculated as:

$$Cost_{retrieval\_ptu} = Cost_{retrieval} \times Freq_{retrieval} \times M \tag{3.14}$$

When a mobile node crosses a safe boundary, the current safe boundary may be shrink because this node may become the new k nearest node or may be expanded if this node was included in the query answer and is moving away from the query point. Such adjusting of the size of the safe boundary increases the frequency expected by equation 3.4 because a mobile node that crosses the safe boundary may cross it again in less time that predicted by the equation as illustrated by figure 3.2. If we assume that a mobile node at least needs to cross twice safe boundary in order to become either member of the query's answer or the $k+1$-th nearest node, then we estimate the frequency of the events that some host moves into a safe boundary as:

$$Freq_{updating} = \frac{2 \cdot 4\sqrt{2}E[v]K}{\pi R_S} \tag{3.15}$$

If we assume that there are N_q CKNN being processed, and K is the number of nodes moving within a safe boundary. Then the total cost incurred in updating all the safe boundaries per time unit is

$$Cost_{updating\_ptu} = Cost_{updating} \times Freq_{updating} \times N_q \tag{3.16}$$

Now we estimate the overall cost incurred in the entire network domain by using equations 3.7 and 3.16:

$$Cost_{ptu} = Cost_{retrieval\_ptu} + Cost_{updating\_ptu} \tag{3.17}$$

3.2.3 Validation analytical model

We validated Equation 3.17 using simulation. In this study, we fixed the network domain to be 5000×5000 $meter^2$ and placed on it a number of mobile hosts. These hosts are uniformly distributed in the network domain. From simulations we found out that the mobile ad hoc network becomes connected with high probability for long period of time when the number of

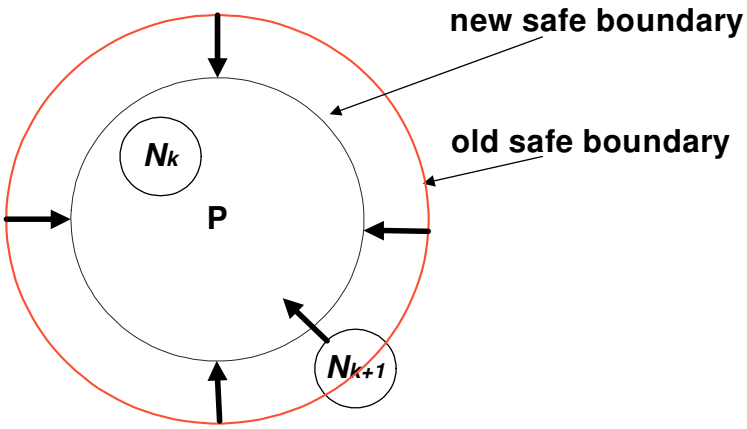
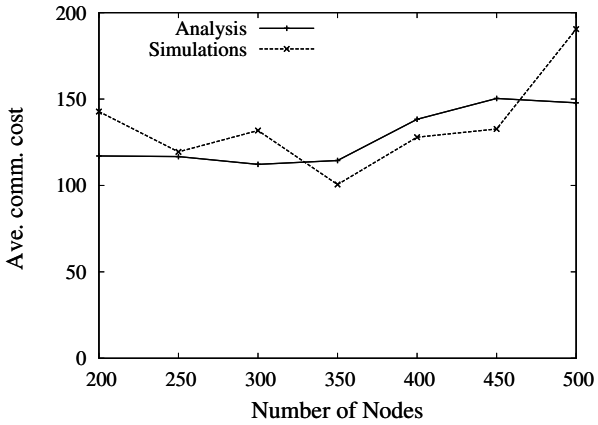


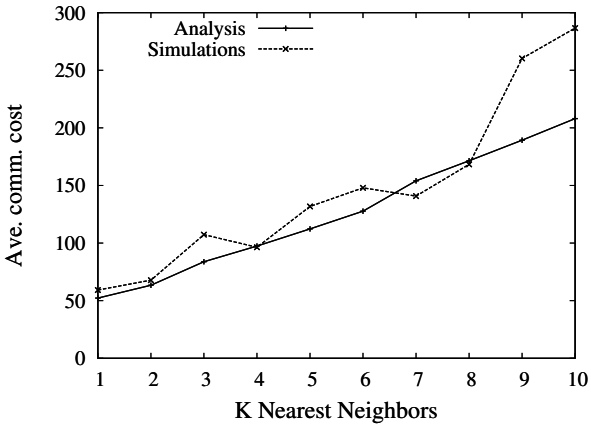
Figure 3.2 Adjusting of the size of a safe boundary

nodes is greater than 200 and the host transmission radius is set at 500 *meter*. The network area is evenly partitioned in cells whose size is approximately 350. The location of 10 CKNN queries are distributed uniformly within the network area and the number of K nearest neighbor ranging from 1 and 10. To broadcast a packet to all mobile hosts in some geographic region, we used an existing technique called *Priority Forwarding* [16]. The performance of this technique has been shown to be little sensitive to the host density. As indicated in [16], the number of hosts participating in forwarding a broadcast message is close to be proportional to the intended broadcast area. In our study, we count the cost of broadcasting a message as the number of hosts that rebroadcast this message. Thus, the cost of 1-hop broadcast is $c = 1$.

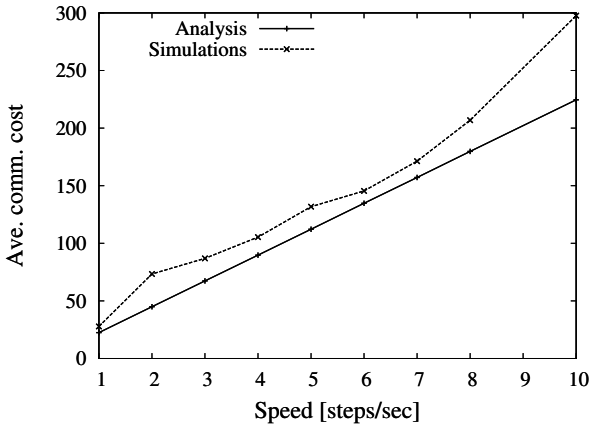
The results illustrated in figure 3.3(a) were obtained by ranging the number of nodes from 200 and 500 and each mobile node moves at the average speed of 5 $meter/sec$ and the value for K nearest neighbor value is set at 5. Those results illustrated in figure 3.3(b) were obtained by ranging the value of K from 1 to 10, fixing the number of nodes at 300, an their average speed at 5 $meter/sec$. Finally, the results illustrated in figure 3.3(c) were obtained by ranging the node speed from 1 to 10, fixing the number of nodes at 300, an the K nearest neighbor parameter at 5. During each simulation run, we periodically compute the average communication cost incurred during a fixed time interval, and collect the performance data when the average cost per time unit becomes stabilized. Figure 3.3 shows both the simulation results and the results



(a) Comm. Cost vs Number of Hosts



(b) Comm. Cost vs. K Nearest Neighbors



(c) Comm. Cost vs Speed

Figure 3.3 S-CKNN query management: Validation of the analytical model

computed using Equation 3.17. We observe that the formula in many cases is quite accurate in predicting the average cost per time unit in each simulation setting. But there are cases where a significative error occurs because of the difficulty in predicting the frequency at which a node crosses a safe boundary. Those cases occur when the mobility of the nodes increases significantly because either the number of nodes is greater than 450 or the speed is greater than 8. Similar effect occurs when the K nearest neighbor parameter is becoming large, because the probability to have more crossing events is increased as a consequence.

3.3 Analytical model for managing M-CKNN queries

In this section we extend the analytical model developed in section 3.2 in order to predict the average communication cost of our technique, when it processes M-CKNN queries. The cost of retrieving the relevant queries for a cell is the same as the one analyzed before. Therefore, we focus our analysis in predicting the average communication cost incurred in updating the safe boundaries of a mobile CKNN.

3.3.1 Overall cost per time unit

A mobile CKNN query is processed by using two safe boundaries as we explained in section 2.2. The first safe boundary, termed simply as the *outer safe boundary* is computed based on equation 3.9. The second safe boundary termed as the *inner safe boundary* is estimated as follows:

$$R_F = \frac{\overline{d_{k+1}} - \overline{d_k}}{4} \tag{3.18}$$

Every time a focal nodes crosses one of its inner safe boundaries, a new pair of safe boundaries are computed based on the distances of the k-th and $k+1$-th nearest nodes to its current location. Considering that the focal node is always located at the center of the safe boundaries every time they are recomputed, it will take on average half of the time predicted by formula 3.3 to arrive to the borders of the inner safe boundary. If we assume that there are K

mobile nodes moving within the outer safe boundary. Thus the frequency of events that the safe boundaries are recomputed can be estimated as:

$$Freq^{m}_{updating} = \text{Frequency of outer safe boundary crossings} + \text{Frequency of inner safe boundary crossings} \quad \frac{2 \cdot 4\sqrt{2}E[v]K}{\pi R_S} + \frac{2 \cdot 4\sqrt{2}E[v]}{\pi R_F} \tag{3.19}$$

If we assume that there are N_{mq} mobile CKNN being processed. Then the total cost incurred in updating all the safe boundaries per time unit is:

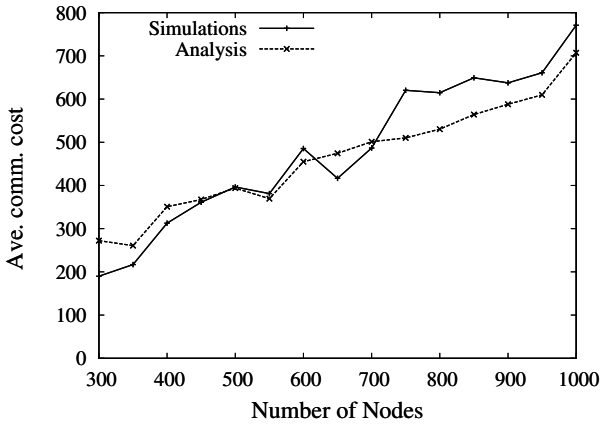
$$Cost^{m}_{updating\_ptu} = Cost_{updating} \times Freq^{m}_{updating} \times N_{mq} \tag{3.20}$$

Therefore, we estimate the overall cost incurred in the entire network domain by using equations 3.7 and 3.20:

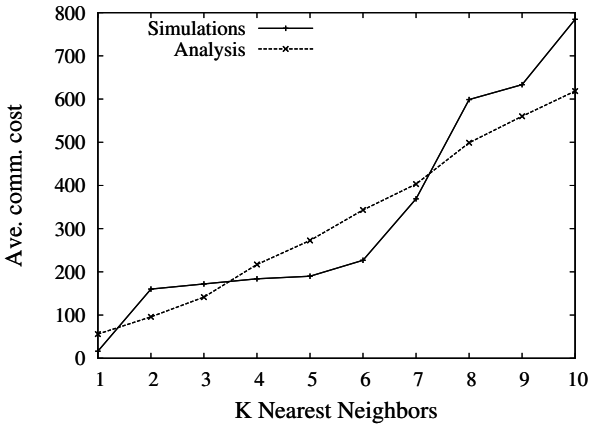
$$Cost_{ptu} = Cost_{retrieval\_ptu} + Cost^{m}_{updating\_ptu} \tag{3.21}$$

3.3.2 Validation analytical model

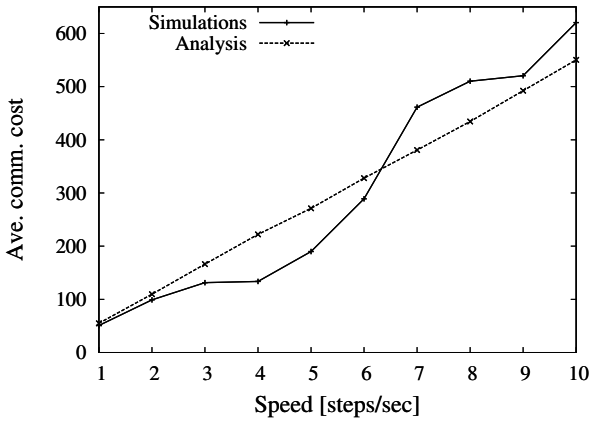
The results illustrated in figure 3.4(a) were obtained by ranging the number of nodes from 300 and 1000 and each mobile node moves at the average speed of 5 $meter/sec$ and the value for K nearest neighbor value is set at 5. Those results illustrated in figure 3.4(b) were obtained by ranging the value of K from 1 to 10, fixing the number of nodes at 300, an their average speed at 5 $meter/sec$. Finally, the results illustrated in figure 3.4(c) were obtained by ranging the node speed from 1 to 10, fixing the number of nodes at 300, an the K nearest neighbor parameter at 5. During each simulation run, we periodically compute the average communication cost incurred during a fixed time interval, and collect the performance data when the average cost per time unit becomes stabilized. Figure 3.4 shows both the simulation results and the results computed using Equation 3.21. We observe that formula 3.4 provides an approximated estimation of the communication cost incurred by our technique. The differences



(a) Comm. Cost vs Number of Hosts



(b) Comm. Cost vs. K Nearest Neighbors



(c) Comm. Cost vs Speed

Figure 3.4 M-CKNN query management: Validation of the analytical model

come from our estimation of how frequent an outer safe boundary is crossed, which we assumed as twice the frequency predicted by formula 3.3. In general the formula agrees with the growing tendency exposed by the average communication cost obtained from simulations.

CHAPTER 4. PERFORMANCE EVALUATION

We have implemented a detailed simulator to evaluate the performance of the proposed technique when the network is fully or partially connected. The simulator is built on top of an existing MANET testbed JIST/SWANS [2]. To broadcast a packet to all mobile nodes in a given geographic region, we used *Priority Forwarding* [16], a technique we developed early for efficient MANET communications. We simulate a small battlefield where the movement of mobile nodes follows the random walk model [11]. The parameters used in our simulation is summarized in Table 4.1.

Table 4.1 Parameters: Simulation of S-CKNN query management

Parameter	range	default	unit
Cell size	N/A	350×350	$meter^2$
Network area	N/A	$5,000 \times 5,000$	$meter^2$
Number of nodes	100 - 500	300	*nodes*
Transmission radius	500	500	*meter*
Average node speed	1 - 10	5	*meter/sec*
Number of queries	N/A	10	*query*
k value of CKNN	1 - 10	5	*nodes*

For performance comparison, we have also implemented a *baseline* approach, which periodically executes a query to find out its KNN. In each execution, the creator of a query sends request to the nodes around the query point, sequentially from 1 hop, 2 hops, ..., until it locates at least k nodes. In our study, we consider two performance metrics:

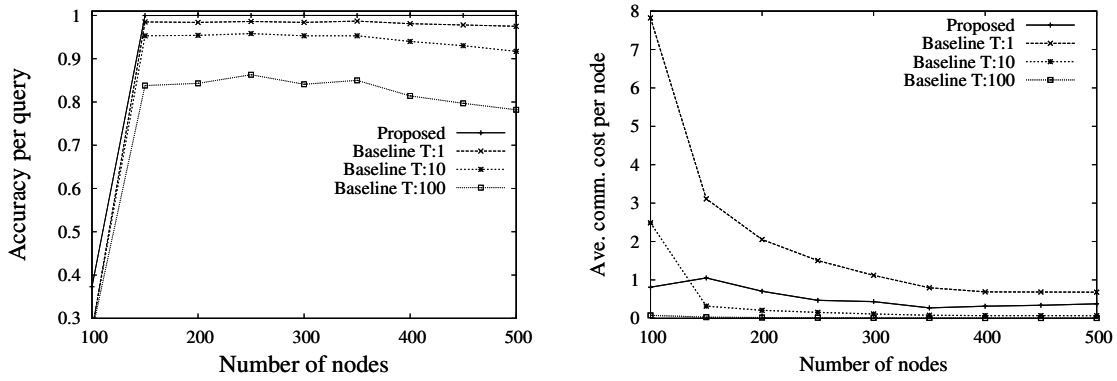
- *Average communication cost*: The average number of packets transmitted by each mobile node during the simulation.

- *Average accuracy of query results*: For each query, we compute its actual result at each simulation time point and compare it with that returned by a particular query processing technique. The average accuracy of a query result is defined to be percentage of the results that are correct.

In the next subsections, we investigate how the performance of the two techniques (i.e., proposed and baseline) is affected by a number of factors, including number of nodes, node mobility, and k value of queries.

4.1 Performance of S-CKNN query management

4.1.1 Effect of the number of nodes



(a) Accuracy (b) Total cost

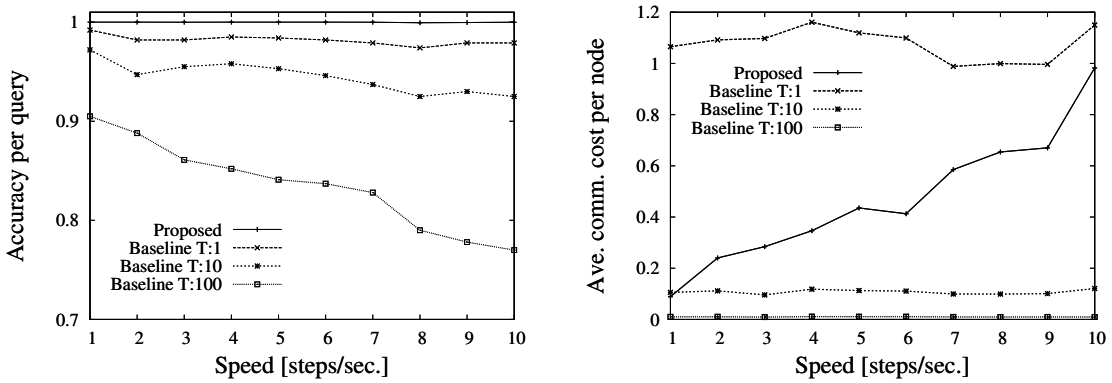
Figure 4.1 S-CKNN query management: Effect of the number of nodes

In this study, we generate a number of mobile nodes, from 100 to 500, in each simulation and placed them on the network domain. For all simulations, the average of node speed is set to be 5 meters/second, and the number of queries is fixed at 10, where the query points are randomly selected. The average K value of S-CKNN queries is 5. In the baseline approach, each query creator periodically computes the query results, at three different periods (T): 1, 10, and 100 seconds per update. The performance results are plotted in figure 4.1. When the

number of node is 100, the network is mostly disconnected. In this case, the baseline approach needs to expand its search scope to a large extend in order to find enough k nodes for a query. As such, the cost is very high. As the number of nodes increases, the network becomes more and finally fully connected. The cost of answering a query reduces accordingly, because it becomes more and more likely that k nodes can be found within 1-hop to the query point. As compared to the baseline approach, our proposed technique performs fairly stably. When the network density is low, a node crossing a safe boundary also needs to search a large area in order to recompute a query, but the chance of having such events is less, thus reducing the performance penalty.

As for the accuracy of query results, all techniques improves when the network density increases. When the network is partially connected, no technique can achieve 100% of accuracy. However, when the network is fully connected, the proposed technique can provide accurate query results, as confirmed in the performance figure. In contrast, the baseline approach, regardless the frequency of query execution, never achieves 100% of accuracy.

4.1.2 Effect of node movement



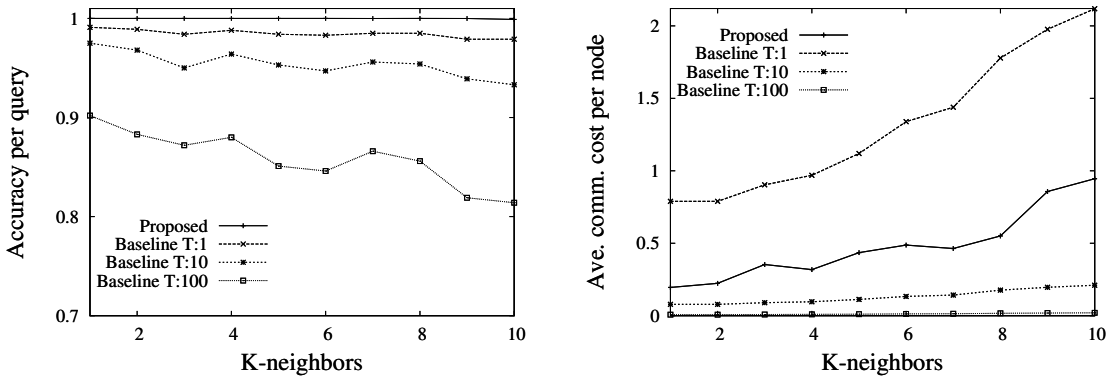
(a) Accuracy (b) Total cost

Figure 4.2 S-CKNN query management: Effect of node movement

In this study, we generate 300 mobile nodes and randomly place them on the network

domain. We also create 10 S-CKNN queries on randomly selected query points. We vary the average speed of node movement from 1 to 10 *meter/second*. The performance results are plot in Figures 4.2. It shows when the node mobility increases, the proposed technique incurs more communication costs. This is not surprising because the query results change more frequently. Each time a node crosses a query's safe boundary, the query is re-evaluated to ensure accurate query results. As for the baseline approach, the incurred communication cost is not affected by the node mobility. However, the average accuracy of query results deteriorate when the node mobility increases.

4.1.3 Effect of the K value



(a) Accuracy (b) Total cost

Figure 4.3 S-CKNN query management: Effect of the K value

This study investigates how the K value of S-CKNN queries affects the performance of the two techniques. In each simulation, we generated 300 nodes and placed them randomly on the network domains. The average speed of these nodes is set to be 5 *meter/second*, and we varied the K value of S-CKNN queries from 1 to 10. The performance results are illustrated in Figure 4.3. It shows that when the value of K increases, the communication costs incurred by both the baseline and proposed techniques increase. For the baseline approach, executing a query with a larger value of K needs to search a larger area in order to locate the initial set of mobile nodes to determine the query results. As such, the communication costs increase. As

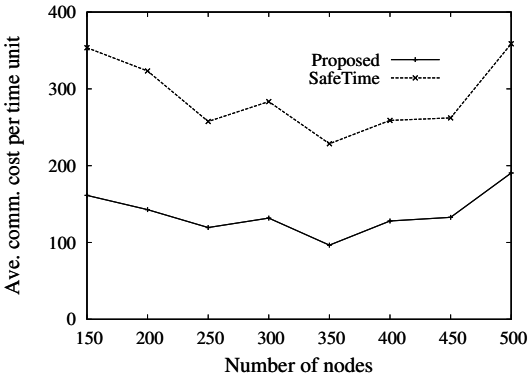
for the proposed technique, a S-CKNN with a larger value of K means a larger safe boundary (when the number of nodes is fixed). This increases the probability of having some node crossing the boundary. Since for each crossing event, the query is re-evaluated to ensure its accuracy and therefore the communication costs increase. As for the accuracy of query results, the baseline approach performs worse when the K increases. This is not surprising because the query result changes more frequently, yet the baseline approach evaluates a query periodically in a pre-determined time interval. In contrast, the proposed technique ensure the accurate query results as long as the network is fully connected.

4.2 Comparison of Safe Boundary and Safe Time

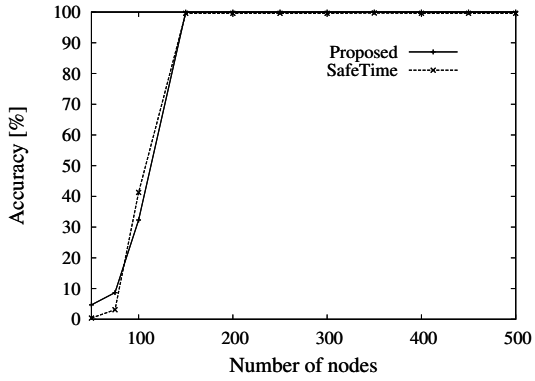
We compare our safe boundary technique with an existing technique, called *SafeTime* developed by Kim et al [17]. In this technique, the K-th nearest node to a query point is responsible to verify if the query result has been modified. When threshold time, termed as safe time, is reached, the K-th node searches for the location of the $K+1$-th nearest node. When this latter node is found, then a new safe time is computed based on the cross distance between these two nodes. Then the new K-th nearest node is responsible to locate again the $K+1$ nearest node after the new safe time is reached. The results plotted in figures 4.4(a) and (b) were obtained by ranging the number of nodes from 150 and 200 nodes, the K nearest neighbor value is set at 5 and the average speed of the nodes is set at 5 [mts/secs.].

The accuracy of safe time is based on the assumption that the maximum speed of the nodes is known. It is illustrated in figure 4.4(b), that our technique and safe time reach 100% of accuracy in their query results as long as the network becomes fully connected with high probability, which occurs when the number of nodes is approximately greater than 200 nodes. Figure 4.4(a) shows the communication cost incurred by both techniques and we observe that our technique incurred in less communication cost. This redundancy comes from the fact that safe time requires that the K-th nearest node needs to locate $K+1$-th nearest node, even tough the cross distance of both nodes remains almost the same.

Also it is illustrated in figures 4.5(b) and (c), the communication cost incurred by both

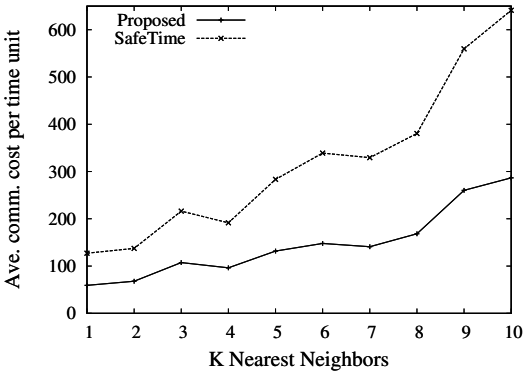


(a) Comm. Cost vs Number of Nodes

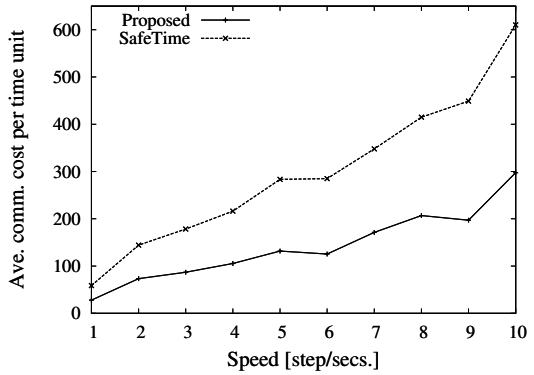


(b) Accuracy vs. Number of Nodes

Figure 4.4 Safe Boundary vs Safe Time: Effect of the number of nodes



(a) Comm. Cost vs K Nearest Neighbors



(b) Comm. Cost vs Speed

Figure 4.5 Safe Boundary vs Safe Time: Effect of node movement and K value

techniques when the number of nodes is fixed at 300 nodes. Particularly, in figure 4.5(b) the K nearest node value ranging from 1 to 10 and the average speed is fixed at 5 meters/secs and in figure 4.5(c) the average speed of the node ranging from 1 to 10 and the value for K is set at 5. From both figures we can observe that our technique incurs in less communication cost than safe time. We can see that both techniques are affected by the mobility, which means that if either the average speed of the nodes, the number of nodes or the K value are increased, the communication cost is increased. This dependency on the mobility of the nodes comes from the fact, both techniques needs to know the location of the K-th and $K+1$-th nearest nodes to a query point at some instant of time. The accuracy results of both techniques are not plotted because both techniques reached 100% of accuracy in their query results.

4.3 Performance of M-CKNN query management

Table 4.2 Parameters: Simulation of M-CKNN query management

Parameter	range	default	unit
Cell size	N/A	350×350	$meter^2$
Network area	N/A	$5,000 \times 5,000$	$meter^2$
Number of nodes	100 - 500	300	*nodes*
Transmission radius	500	500	*meter*
Average node speed	1 - 10	5	*meter/sec*
Number of queries	N/A	10	*query*
k value of CKNN	1 - 10	5	*nodes*

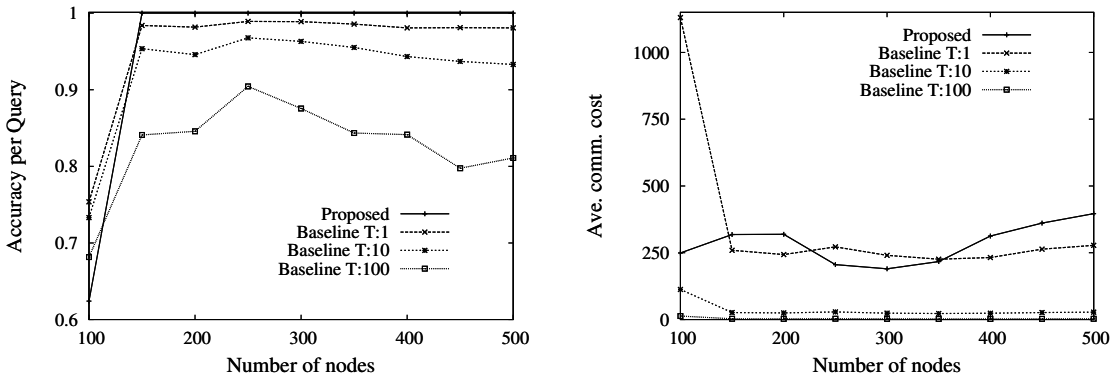
For performance comparison, we have also implemented a *baseline* approach. In this technique, each query's owner periodically executes its queries to find out their KNN respect to its new position. In each execution, the owner or creator of a query sends request to the nodes around the query point, sequentially from 1 hop, 2 hops, ..., until it locates at least k nodes. In our study, we consider two performance metrics:

- *Average communication cost*: The average number of packets transmitted during the simulation.

- *Average accuracy of query results*: For each query, we compute its actual result at each simulation time point and compare it with that returned by a particular query processing technique. The average accuracy of a query result is defined to be percentage of the results that are correct.

In the next subsections, we investigate how the performance of the two techniques (i.e., proposed and baseline) is affected by a number of factors, including number of nodes, node mobility, and the K value of queries.

4.3.1 Effect of the number of nodes



(a) Accuracy (b) Total cost

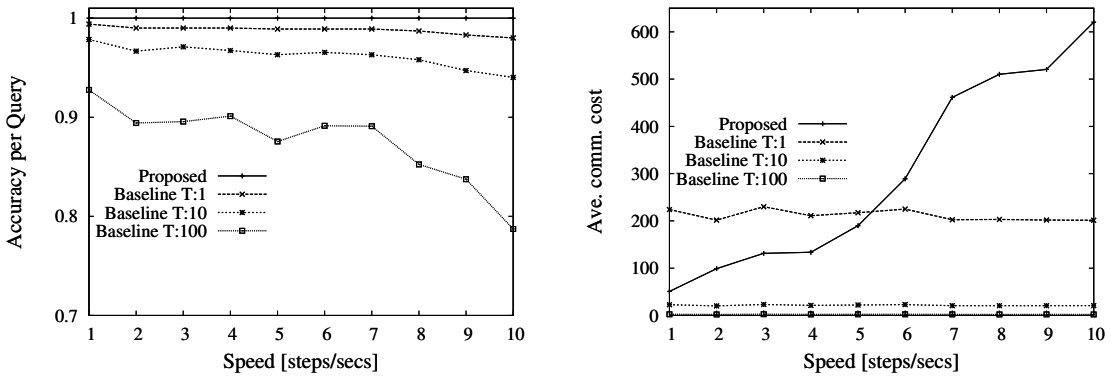
Figure 4.6 M-CKNN query management: Effect of the number of nodes

In this study, we generate a number of mobile nodes, from 100 to 500, in each simulation and placed them on the network domain. For all simulations, the average of node speed is set to be 5 meters/second, and the number of queries is fixed at 10, where the query points are randomly selected. The average K value of M-CKNN queries is 5. In the baseline approach, each query creator periodically computes the query results, at three different periods (T): 1, 10, and 100 seconds per update. The performance results are plotted in figure 4.6. When the number of node is 100, the network is mostly disconnected. In this case, the baseline approach and our proposed technique need to expand its search scope to a large extend in order to find

enough k nodes for a query. As such, the cost of the baseline approach with T=1 and our approach are similar. As the number of nodes increases, the network becomes more and finally fully connected. The cost of answering a query reduces accordingly, because it becomes more and more likely that k nodes can be found within 1-hop to the query point. However as the number of nodes increases, the probability of a node crosses an outer safe boundary also is increased and considering the fact that on average a focal node has less free space to move without affecting the query result, then it was not surprising to observe an increase in the communication cost. After 350 nodes, our approach becomes more costly than the baseline approach with T =1.

As for the accuracy of query results, all techniques improves when the network density increases. When the network is partially connected, no technique can achieve 100% of accuracy. However, when the network is fully connected, the proposed technique can provide accurate query results, as confirmed in the performance figure. In contrast, the baseline approach, regardless the frequency of query execution, never achieves 100% of accuracy.

4.3.2 Effect of node movement



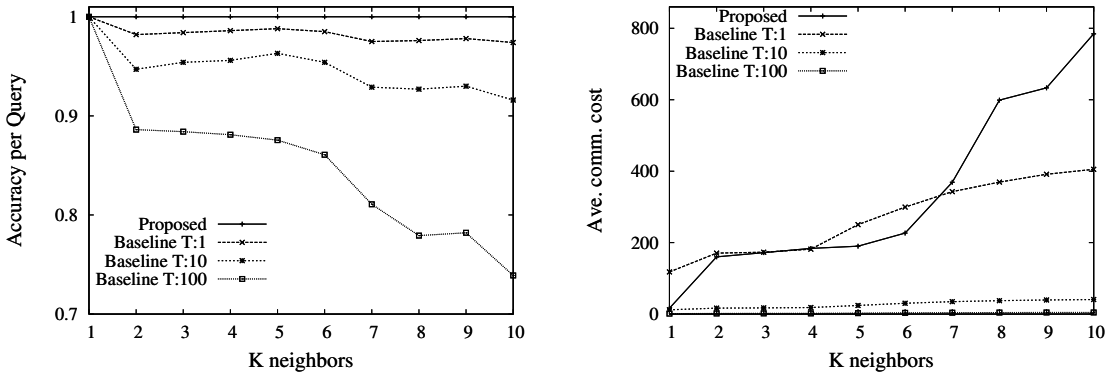
(a) Accuracy (b) Total cost

Figure 4.7 M-CKNN query management: Effect of node movement

In this study, we generate 300 mobile nodes and randomly place them on the network

domain. We also create 10 M-CKNN queries on randomly selected query points. We vary the average speed of node movement from 1 to 10 *meter/sec.*. The performance results are plot in Figures 4.7. It shows when the node mobility increases, the proposed technique incurs more communication costs, and when the speed is greater than 5 [meter/secs.] it is more costly than the baseline approach with T =1. This is not surprising because the query results change more frequently. Each time a node crosses a query's outer safe boundary, the query is re-evaluated to ensure accurate query results and also the query's focal node may cross the associated inner safe boundary in lesser time. As for the baseline approach, the incurred communication cost is not affected by the node mobility. However, the average accuracy of query results deteriorate when the node mobility increases.

4.3.3 Effect of the K value



(a) Accuracy (b) Total cost

Figure 4.8 M-CKNN query management: Effect of the K value

This study investigates how the K value of M-CKNN queries affects the performance of the two techniques. In each simulation, we generated 300 nodes and placed them randomly on the network domains. The average speed of these nodes is set to be 5 *meter/second*, and we varied the K value of M-CKNN queries from 1 to 10. The performance results are illustrated in Figure 4.8. It shows that when the value of K increases, the communication costs incurred by both the baseline and proposed techniques increase. For the baseline, executing a query

with a larger value of K needs to search a larger area in order to locate the initial set of mobile nodes to determine the query results. As such, the communication costs increase. As for the proposed technique, a M-CKNN query with a larger value of K means a larger safe boundary. This increases the probability of having some node crossing an outer safe boundary. If more nodes are probably near the outer safe boundary, it is also expected that the average size of its associated inner safe boundary becomes small. Therefore, considering these two factors and for each crossing event, the query is re-evaluated to ensure its accuracy, then the communication costs increase. We can see that when the value of K is smaller than 5, the baseline and our approach incur in similar communication cost, however when the area of an outer safe boundary is increased because of the increasing of the value of K, our approach becomes more costly. As for the accuracy of query results, the baseline approach performs worse when the K increases. This is not surprising because the query result changes more frequently, yet the baseline approach evaluates a query periodically in a pre-determined time interval. In contrast, the proposed technique ensures the accurate query results as long as the network is fully connected.

CHAPTER 5. RELATED WORK

To the best of our knowledge, the work by Kim et al in [17] is the only one that investigate efficient processing of CKNN queries in the context of a mobile peer-to-peer environment. As mentioned early, this scheme supports only S-CKNN queries. Most existing techniques assume a centralized architecture, where a server is used to track the movement of mobile nodes and handle queries. These works aims at minimizing either mobile communication cost or server processing cost, or both, and in general, can be classified into two categories.

The techniques in the first category supports only S-CKNN queries. Yu et al and Zhang et al [30, 31] propose several algorithms for processing S-CKNN queries. The proposed algorithms do not consider how the clients update their location, but simply assume the server knows their location and focus on minimizing disk I/O incurred in query processing by indexing both moving objects and queries. In [9], Hsueh et al consider how to approximate the results of S-CKNN queries. The idea is to minimize the location update cost while ensuring the query results are reasonably accurate. The technique proposed by Iwerks et al in [13] converts a CKNN query into a window query to facilitate the indexing of mobile objects. This strategy was later improved by Li et al [19]. The new scheme needs to monitor only the trajectories of the $k+1$-th and k-th nearest nodes instead of all k nearest nodes. Those trajectories are represented by a sequence of bounding boxes in a transformed time-distance space.

The techniques in the second category supports both S-CKNN and M-CKNN queries. In [8], Gedik et al propose an approach called *Motion Adaptive Indexing (MAI)* which levarage the dynamic motion of the nodes to optimize the continuous query evaluation. Assuming the knowledge of the speed and direction of mobile nodes, it uses the concept of motion sensitive bounding boxes to model both mobile nodes and mobile CKNN queries. The objects and

queries are indexed based on these bounding boxes, which are modified automatically based on the mobility of the nodes. Although this approach is initially intended to process monitoring range queries, it can also support CKNN queries with a new notion called *safe radius*. The safe radius for a query is computed as the radius of a circular region centered at the query point that encloses at least k-nearest nodes during a pre-computed period of time. When this time period expires, a new period and radius are calculated. This strategy allows a CKNN query to be converted into a monitoring range query. In [29], Xiong et al propose a technique called *SEA-CNN*, which indexes the mobile nodes in secondary memory with regular cell index. Unlike [8], this scheme does not require the server to know the velocity information of mobile nodes. In [27], Wu et al propose a technique which distribute the queries among mobile nodes and lets them evaluated directly. This scheme, however, still requires a central server to coordinate query distribution and evaluation.

All the above techniques assume the nodes' distance is Euclidean. In the context of road networks, wherein the distance measurement depends on the structure of the road segments, Jensen et al [14] propose an algorithm that allow mobile nodes to collaborate in query processing. This technique also relies on a central server for query management, and it can provide only approximate query results. In [21], Mouratidis et al proposed two algorithms, namely IMA and GMA, which manage mobile objects and query movement patterns in road networks. Another work is by Liu et al [20]. The proposed scheme offloads some computation from the server to the mobile nodes to reduce server computation cost and communication cost. For each query point, the server initially computes the query result and a monitoring area on the road network and knows the speed and direction of the mobile nodes. The monitoring area of a query contains the k-nearest neighbor nodes to a query point. Subsequently, the server transmits the monitoring area, the position of the query point and the k-th nearest node to the mobile nodes moving on the the paths overlapped by the monitoring region. Each node, within the monitoring area, monitors its position against the query point and the k-th nearest node. If the $(k-i)$-th nearest node is moving far away from the query point than the k-th nearest node, a location update message is sent to the server. Similar update occurs, if another

node not included in the KNN set is moving closer to the query point than the k-th nearest node.

CHAPTER 6. SUMMARY AND CONCLUSION

We have proposed a unified solution for efficient processing of both S-CKNN and M-CKNN queries in mobile ad hoc networks. Our technique is efficient because of the concept of safe boundary: for each CKNN query, we compute a set of circular boundaries and perform query evaluation only when some node crosses over any of these boundaries. In addition, our technique makes mobile nodes aware of their nearby boundaries and therefore minimizes the cost of query management. We assess the performance of the proposed technique through simulation and compare it with a baseline approach. The results confirm that the proposed technique is able to achieve accurate query results when the network is fully connected at a reasonable communication cost.

In our current solution, the network partitioning is pre-determined with a fixed cell size. While this simplifies our technique design, it may not achieve optimal system performance. A larger cell size reduces the chance of having a node moving out of it, and so decreases the communication costs incurred in retrieving relevant queries. However, it increases the cost of updating a safe boundary, which needs to broadcast to all nodes inside a cell. Clearly, various factors such as node mobility and query activities need to be considered in determining a good cell size. As these parameters may change from time to time, dynamic network partitioning is necessary to ensure consistent good performance. Additionally, we plan to use the concept of safe boundary to process mobile range monitoring queries. We leave these as our future work.

BIBLIOGRAPHY

[1] M. Ahsanullah and V. Nevzorov. *Order Statistics: Examples and Exercises*. Nova Science Publishers, Inc., 2005.

[2] R. Barr, Z. Haasand, and R. van Renesse. JIST/SWANS. Java in Simulation Time / Scalable Wireless Ad hoc Network Simulator. Web page at http://jist.ece.cornell.edu/, May 2005.

[3] D. Bhattacharjee, A. Rao, C. Shahand M. Shah, and A. Helmy. Empirical modeling of campus-wide pedestrian mobility: Observations on the USC campus. In *Proceedings of the IEEE Vehicular Technology Conference (VTC'04)*, volume 4, pages 2887– 2891, Los Angeles, CA, USA, September 26 - 29 2004.

[4] Y. Cai, K. Kim, T. Xu, and P. Galdames. Geotasking: A New Type of Computing. In *Technical Report 06-16, Department of Computer Science, Iowa State Univeristy*, http://archives.cs.iastate.edu/documents/disk0/00/00/04/27/00000427-00/mainfile.ps, June 2006.

[5] T. Camp, J. Boleng, and V. Davies. A Survey of Mobility Models for Ad Hoc Network Research. *Wireless Communication and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.

[6] Chi Chen. Location-based data aggregation in mobile ad hoc networks. Thesis nr 2111, Universitat Stuttgart, Faculty of Computer Science, Electrical Engineering and Information Technology, Germany, November 2003.

[7] K.-H. Chiang and N. Shenoy. A random walk mobility model for location management in wireless networks. In *12th IEEE Int'l Symposium on Personal, Indoor and Mobile Radio Communications*, volume 2, pages E43–E48, San Diego, CA, USA, September 30 - October 03 2001.

[8] B. Gedik, K-L. Wu, P. Yu, and L. Liu. Processing Moving Queries over Moving Objects Using Motion-Adaptive Indexes. *IEEE Transactions On Knowledge and Data Engineering*, 18(5):651–668, May 2006.

[9] Y.-L. Hsueh, R. Zimmermann, and M.-H. Yang. Approximate continuous k nearest neighbor queries for continuous moving objects with pre-defined paths. In *Proc. of Perspectives in Conceptual Modeling (ER'05)*, pages 270–279, Klagenfurt, Austria, October 24-28 2005.

[10] H. Hu, J. Xu, and D. Lee. A generic framework for monitoring continuous spatial queries over moving objects. In *Proc. of the 2005 ACM Int'l Conf. on Management of Data (SIGMOD'05)*, 2005.

[11] B.-D. Hughes. *Random Walks and Random Environments*, volume 2. Oxford University Press, 1996.

[12] T. Hyytia and J. Virtamo. Random waypoint mobility model in cellular networks. *Wireless Network*, 13(2):177 – 188, 2007.

[13] G. Iwerks, H. Samet, and K. Smith. Continuous k-nearest neighbor queries for continuously moving points with updates. In *Proc. of the 29th Int'l Conf. on Very Large Data Bases (VLDB'03)*, Berlin, Germany, Spetember 9-12 2003.

[14] C.S. Jensen, J. Kolar, T.B. Pedersen, and I. Timko. Nearest neighbor queries in road networks. In *Proc. of the 11th ACM Int'l Symposium on Advances in Geographic Information Systems(GIS'03)*, pages 1–8, New Orleans ,LA , USA, September 03 - 07 2003.

[15] B. Karp and H. T. Kung. GPSR: Greedy Perimeters Stateless Routing for Wireless Network. In *Proc. of ACM Int'l Conf. on Mobile Computing and Networking (MOBICOM00)*, pages 243 – 254, Boston, MA, U.S.A, 2000.

[16] K. Kim, Y. Cai, and W. Tavanapong. A Priority Forwarding Technique for Efficient and Fast Flooding in Wireless Ad Hoc Networks. In *14th Int'l Conf. on Computer Communications and Networks (IC3N'05)*, pages 223–228, San Diego, CA, U.S.A, October 17-19, 2005.

[17] K. Kim, Y. Cai, and W. Tavanapong. Safe-Time: Distributed Real-time Monitoring of cKNN in Mobile Peer-to-Peer Networks. In *Proc. of the 9th Int'l Conf. on Mobile Data Management (MDM'08)*, Beijing, China, April 27-30 2008.

[18] Y. Ko and N. H. Vaidya. Location-Aided Routing (LAR) Mobile Ad Hoc Networks. In *Proc. of ACM Int'l Conf. on Mobile Computing and Networking (MOBICOM98)*, pages 66–75, Dallas, TX, U.S.A, 1998.

[19] Y. Li, J. Yang, and J. Han. Continuous k-nearest neighbor search for moving objects. In *Proc. of the 16th Int'l Conf. on Scientific and Statistical Database Management (SSDBM04)*, page 123, Santorini Island, Greece, June 21 - 23 2004.

[20] F. Liu, K. Hua, and T. Do. A p2p technique for continuous knearest-neighbor query in road networks. In *Proc. of the 18th Int'l Conf. on Database and Expert Systems Applications (DEXA'07)*, pages 264–276, Regensburg, Germany, September 03 - 07 2007.

[21] K. Mouratidis, M. Yiu, D. Papadias, and N. Mamoulis. Continuous Nearest Neighbor Monitoring in Road networks. In *Proc. of the 32nd Int'l Conf. on Very Large Data Bases (VLDB06)*, pages 43–54, Seoul, Korea, September 12-15 2006.

[22] E. Royer and C. E. Perkins. Multicast Operation of the Ad-hoc On-Demand Distance Vector Routing Protocol. In *Proc. of ACM Int'l Conf. on Mobile Computing and Networking (MOBICOM99)*, pages 207–218, Seattle, WA, U.S.A, August 1999.

[23] B. Schmeiser. The generation of order statistics in digital computer simulation: A survey. In *Proceedings of the 10th conference on Winter simulation*, volume 1, pages 137– 140, Miami, FL, USA, December 04 - 06 1978.

[24] M. Spiegel, J. Schiller, and A. Srinivasan. *Schaum's Outline of Theory and Problems of Probability and Statistics*. McGraw-Hill, second edition, 2000.

[25] R. Thomas, H. Gilbert, and G. Mazziotto. Influence of the Moving of the Mobile stations on the Performance of a Radio Cellular Network. In *Proc. of Third Nordic Seminar*, Copenhagen, Denmark, September 1988.

[26] E. Weisstein. Random Walk–2-Dimensional.
Web page at http://mathworld.wolfram.com/RandomWalk2-Dimensional.html, 2008.

[27] W. Wu, W. Guo, and K.-L. Tan. Distributed Processing of Moving K-Nearest-Neighbor Query on Moving Objects. In *Proc. of the 23rd Int'l Conf. on Data Engineering (ICDE'07)*, pages 1116–1125, April 15-20 2007.

[28] W. Wu and K.-L. Tan. isee: Efficient continuous knearest-neighbor over moving objects. In *Proc. of the 19th Int'l Conf. on Scientific and Statistical Database Management (SSDBM'07)*, page 36, Banff, Canada, July 09-1 2007.

[29] X. Xiong, M. F. Mokbel, and W. G. Aref. Sea-cnn: Scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases. In *Proc. of the 21st Int'l Conf. on Data Engineering (ICDE05)*, pages 643–654, Tokyo, Japan, April 5 - 8 2005.

[30] Xiaohui Yu, Ken Q. Pu, and Nick Koudas. Monitoring k-nearest neighbor queries over moving objects. In *Proc. of the 21st Int'l Conf. on Data Engineering (ICDE'05)*, Tokyo, Japan, 2005.

[31] W. Zhang, J. Li, and H. Pan. Processing continuous k-nearest neighbor queries in location-dependent application. *Int'l Journal of Computer Science and Network Security*, 6(3):1–9, 2006.