Computer Communications 116 (2018) 132–146

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

# VRSense: Validity region sensitive query processing strategies for static and mobile point-of-interests in MANETs

Byungkwan Jung[a], Sunho Lim[*,a], Jinseok Chae[*,b], Cong Pu[c]

[a] T[2]WISTOR: TTU Wireless Mobile Networking Laboratory, Department of Computer Science, Texas Tech University, Lubbock, TX 79409, USA
[b] Department of Computer Science and Engineering, Incheon National University, Incheon 22012, Korea
[c] Division of Computer Science, Marshall University, Huntington, WV 25755, USA

ARTICLE INFO

*Keywords:*
Location-based services
Mobile ad hoc networks
Point-of-interest
Query processing
Validity region

ABSTRACT

Querying any point-of-interest (POI) in anywhere is a major part of location-based services (LBS) and has been applied to diverse wireless and/or mobile networks for realizing seamless services and mobile and ubiquitous computing. In particular, designing an efficient query processing scheme is admittedly challenging in mobile ad hoc networks (MANETs) because of the lack of centralized coordination, limited computing and communication capabilities, and time-varying network topologies. Unlike traditional stationary POIs, it also becomes challenging to consider mobile POIs that can invalidate prior query result. To address these challenges, we propose a set of query processing strategies based on a validity region to efficiently update the freshness of the queried POI and reduce the query traffic in MANETs. We first present and analyze time- and location-sensitive query types in the presence of static and mobile POIs and identify their corresponding query processing operations and implications. A time-sensitive query targets a POI containing a time-varying information while a location-sensitive query retrieves a location information. In this paper, we focus on location-sensitive query for both static and mobile POIs. Unlike a static POI, which is stationary and never changes its location, a mobile POI refers to a moving object and its location information is time varying. Then we propose basic rectangle and convex hull based validity regions and their corresponding query processing operations, and extend them by combining both techniques and considering an opportunistic overhearing. We also propose two more techniques in forming validity region for mobile POIs and their corresponding query processing operations to flexibly approximate the validity region. We conduct extensive simulation experiments using the OMNeT++ for performance evaluation and analysis, in which an infrastructure-based query processing approach is modified to work in MANETs for performance comparison. The simulation results indicate that the validity region based query processing strategies can reduce the number of queries and increase the time staying in the validity region.

## 1. Introduction

With increasingly popular in recent wireless, mobile, and location-aware devices, users are able to query any point-of-interest (POI) as a major part of location-based services (LBS) supported by ubiquitous communication infrastructure. For example, a user walking in a street queries a coffee shop, or a driver in a vehicle equipped with an on-board global positioning system (GPS) queries the nearest gas-station. They are all supported by 3G/4G or roadside unit (RSU) (e.g., an access point, an infostation [1], or a message relaybox [2]) located along the road that acts as a gateway to an infrastructure-based network, where a centralized server searches the queried POI and answers a set of query results back. A good deal of effort has been devoted into the designing of query processing strategies [3–15] applied to diverse wireless and/or mobile networks to realize seamless services and mobile and pervasive computing.

Although 3G/4G high speed wireless networks are increasingly popular, many areas are still remained uncovered, such as an urban or remotely isolated area. The current infrastructure could even be collapsed and unavailable due to the disasters including recent hurricanes and earthquakes. For example, emergency communications and queries among rescue people can be conducted through single- and multi-hop relays without the help of infrastructure [3–5,9]. In this paper, we investigate query processing strategies in mobile ad hoc networks (MANETs) facing with two key challenges. First, due to the lack of centralized coordination, limited computing and communication capabilities, and time-varying network topologies, it is admittedly challenging to design an efficient query processing scheme in MANETs. In

* Co-corresponding authors.
*E-mail addresses:* byung.jung@ttu.edu (B. Jung), sunho.lim@ttu.edu (S. Lim), jschae@inu.ac.kr (J. Chae), puc@marshall.edu (C. Pu).

https://doi.org/10.1016/j.comcom.2017.11.017
Received 30 June 2017; Received in revised form 29 September 2017; Accepted 27 November 2017
Available online 05 December 2017

query processing, when a user (later node) broadcasts a query to collect an object information what it is interested (e.g., POI), it may receive more than one query reply as a query result if multiple nodes can answer the query. Unlike an infrastructure-based network, where the query can directly be sent to and responded from a centralized server, the query is often flooded to the rest of nodes through multi-hop relays in the network. Thus, a blind query broadcast operation followed by a series of unconditional query forwarding operations is inefficient and even harmful because of redundant query retransmissions, causing query contentions and collisions [16].This could consume a significant amount of battery energy because wireless communication could be responsible for more than half of the total energy consumption [17]. In addition, when a node intends to reply a query result by unicasting it back to the query sender, unlike a wired network, all one hop neighbor nodes can still overhear the query result, as if it is a broadcast packet [16]. Thus, it is critical to reduce the number of broadcasted queries without degrading the validity of query result. Second, unlike traditional stationary POIs, it is also challenging to consider mobile POIs in designing query processing schemes in MANETs. This is because the mobility of POIs may invalidate prior query result. The query generating node in fact even does not aware whether the query result becomes obsolete. Thus, how to efficiently update the freshness of query results and when to judiciously re-generate a new query also become critical.

To address these challenges, we propose a set of query processing strategies based on a validity region, called *VRSense*, to efficiently update the freshness of queried data and reduce the query traffic in the presence of static and mobile POIs in MANETs. A validity region is defined as a virtual area where a query result remains the same as long as a query issuing node is located within the area. Several validity regions and variants based querying techniques [15,18–20] have been proposed to improve query performance and reduce query traffic. However, most approaches still rely on an infrastructure-based network with a centralized server in forming a validity region. They may not directly be applied to an infrastructure-less network, where each node is required to build its region in a distributed manner. Our contributions are briefly summarized in three-fold:

- We first present both time- and location-sensitive query types in the presence of static and mobile POIs without deploying a validity region based approach in MANETs. Then we provide a set of query scenarios, identify its corresponding query processing operations, and analyze its query processing implications.
- Second, we investigate a validity region based query processing approach and propose basic rectangle (*Rect*) and convex hull (*Conv*) based validity regions and their corresponding query processing operations. We also extend them by adaptively combining both techniques (*Adapt*) and considering an opportunistic overhearing (*Adapt:Ovhr*). In addition, we modify the safe exit algorithm [20], deployed in an infrastructure-based query processing scheme (*Greedy*), to work in MANETs for the purpose of performance comparison.
- Third, we observe the mobility of POIs and its impact on the validity region in MANETs. Then we propose both reduced (*Reduced*) and probabilistic (*Prob*) techniques and their corresponding query processing operations (*Reduced – Rect/Conv/Adapt/Adapt:Ovhr* and *Prob – Rect/Conv/Adapt/Adapt:Ovhr*) to flexibly approximate the validity region in the presence of mobile POIs.

We conduct extensive simulation experiments using the OMNeT++ [21] for performance comparison and analysis in terms of the number of queries and the time spent in the validity region. The simulation results indicate that the convex hull based approaches, Adapt and Adapt:Ovhr, can reduce the number of queries and increase the time spent in the validity region during the query processing operations. The Reduced and Prob applied to the proposed query processing strategies can further improve the query performance with mobile POIs. Note that this paper is significantly extended[1] based on our initial work [22].

The rest of paper is organized as follows. Prior query processing approaches are comprehensively reviewed and analyzed in Section 2.The basic query operation and time- and location-sensitive query operations are presented in Section 3. A set of validity region-based query processing strategies with static and mobile POIs is presented in Sections 4 and 5, respectively. Section 6 is devoted to performance evaluation and analysis. Finally, we further discuss the potential issues of query processing in MANETs followed by conclusion in Sections 7 and 8, respectively.

## 2. Related work

In this section, we comprehensively review prior non-region based querying approaches in terms of top-*k* query and *k* nearest neighbor (*k*NN) query, and then analyze region based querying approaches in diverse wireless and/or mobile networks.

**Top-k query:** Top-*k* query retrieves the *k* number of data items, ordered by the score based on a target attribute, from adjacent nodes in the network. A query issuing node floods a query to the rest of nodes in the network. When a node receives the query, it evaluates the score of locally stored data items and replies them that meet the queried score back to the query issuing node. The query issuing node selects the *k* highest scored data items from received data items. Since replying data items from multiple nodes may incur a reply storm problem, a two-phase top-*k* query method is proposed to reduce the query reply traffic without hurting the query accuracy in MANETs: search and data collection [3]. In the search phase, a query issuing node floods a query, collects the score information of data items, and sets the *k*-th highest score as a threshold. In the data collection phase, the query issuing node floods the query piggybacking the threshold, and each node replies its corresponding data items scored higher or equal to the threshold. In [4], each node maintains a routing table containing a rank of data items based on the score in MANETs. A query is not blindly flooded but can be forwarded to a set of replying candidate nodes, reducing the traffic of queries and replied data items. Due to the frequent link disconnections and score changes, however, there is a non-negligible communication overhead to update the routing table. A query routing scheme is proposed to handle top-*k* query incorporating a clustering framework and a data replication [5] in MANETs. A set of cluster head nodes holding high rank data items is selected and a query is propagated between head nodes to reduce the traffic of queries and replied data items. Periodic ranking updates and corresponding re-selecting head nodes may incur a non-negligible communication overhead.

***k* Nearest neighbor query:** *k* nearest neighbor (*k*NN) query searches data items from the *k* nearest neighbor nodes located adjacent to a query point, where a query issuing node floods a query. In [6–8], several *k*NN query processing strategies are proposed either with a tree-based search structure or without structure in wireless sensor networks (WSNs), where each node is aware of its location and neighbor nodes.The basic idea is that a search space is partitioned into multiple sub-areas (i.e., sectors) in the network, where each local coordinator node collects and transmits partial data items or results to the query issuing node. In [9], variants of *k*NN query processing strategies such as *Explosion* and *Spiral* are also proposed in MANETs, where a geo-routing is deployed to forward a *k*NN query to the node located nearest to the query point. The *k*NN query is further extended by considering a

[1] The key extensions have been made by presenting time- and location-sensitive query scenarios with static and mobile POIs and their corresponding query operations for motivating the proposed research, proposing two region-based techniques in forming a validity region and applying them into four proposed query operations for flexible validity region, completely re-evaluating all the proposed schemes for performance evaluation and analysis, and identifying two potential research issues for discussion.

distance constraint, $l - k$NN, in mobile sensor networks (MSNs) [10], where a query is searched by at least $k$ sensors located at least an $l$ distance apart each other. This approach is proposed to avoid in receiving query results collected in a skewed area.

**Region-based query:** Spatial query collects a location information of static objects (e.g., POIs) located around mobile nodes in the network. A query issuing node continuously sends a query to a server to refresh the information whenever its location is updated, but this can cause high query traffic. To reduce redundant queries that request the same information, the server sets a validity region and return it to the query issuing node. The validity region is a geographical area where the information is not updated as long as the query issuing node is located within the area. Thus, the query issuing node is able to decide whether to issue an additional query by verifying whether its current location is still within the validity region. A validity region-based spatial query approach [11] and its variants have been proposed in diverse networks. In particular, several spatial query strategies have been proposed based on the mobility of query issuing node and/or POIs in vehicular ad hoc networks (VANETs) [20]. Similar to the validity region, a server sets a safety region bounded by a set of exit points in terms of road segments. When a vehicle moves and passes an exit point, it retransmits a query to update the query result. The server also updates the exit points based on the mobility of query issuing vehicle. An enhanced approach is further applied to a dynamic road network environment [12], where a direct road network is considered. This approach is different from prior schemes that implicitly assume an undirected network [13] because the infrastructure-based network does not always support bi-directional movements in real. Unlike a query for stationary POIs, a continuous spatial query monitors moving POIs and refresh query result. A generic framework is proposed to efficiently reduce the cost of location update and query evaluation using a rectangular shaped safe region, where the query result is not affected, in infrastructure-supported networks [14]. Since the shape of safe region directly impacts the query performance, a range safe region is proposed for moving range queries to reduce the frequency of query location [15]. To realize this, the size of the original safe region of query issuing node is extended by overlapping multiple safe regions.

In summary, prior region-based query processing approaches may not directly be applied to MANETs, where the network topology varies because of the node mobility. Due to the lack of centralized coordination, prior continuous spatial query processing approaches may not also be applicable in the presence of mobile POIs in infrastructure-less MANETs.

## 3. Query operations

In this section, we first briefly present a basic query operation and then analyze time- and location-sensitive query operations in MANETs.

### 3.1. Basic query operation

When a node generates a query, it floods the query to the network to collect the location information of interested query objects (e.g., POIs). Each node is assumed to equip an on-board global positioning system (GPS) and is aware of the current location.Each node freely moves under the mobility model (e.g., random waypoint mobility) in the network. Since a blind flooding of query incurs a non-negligible query traffic, a *controlled flooding* in terms of number of hops is deployed to reduce query traffic in this paper. For example, a query is flooded to the nodes located within the maximum three hops from a query issuing node. The POIs can be diverse including but not limited to any geographical spot, plant, animal, wildfire, and moving vehicle depending on the target applications, such as collaborative exploration, search and rescue, disaster recovery, surveillance, etc. The POIs can be classified into a set of types ($Q_{type}$) and are assumed to uniformly be distributed in the network. When a node receives the query, it searches the queried

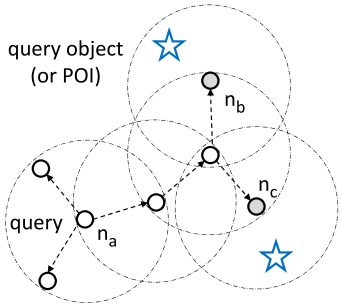


**Fig. 1.** An example of query operation in a MANET. Here, a query replying node is shaded and a queried POI is marked as a star. A direction of query propagated and the communication range of node are represented by a dashed arrow and a dash-dotted line, respectively.

POI. If the node can find or witness the queried POI, it replies the current location as a query result. Note that since the node may not return the exact location of queried POI without additional device or communication overhead, we consider that the queried POI can be detected as far as it is located within the communication range of node. A single or multiple number of POI types can be searched in a single query. In Fig. 1, a node (e.g., $n_a$) floods a query and two nodes (e.g., $n_b$ and $n_c$) closely located to queried objects reply their current location as a query result.

### 3.2. Time- and location-Sensitive query operations

Since the information of queried POI may be time- or location-sensitive, we investigate two different queries and how they affect the query operation and validity of query result. First, a time-sensitive query targets a POI containing a time-varying information and may affect the validity of prior query result that has been sent to a query issuing node. In order to maintain the validity of query result, the query issuing or replying node needs to periodically flood a query or reply the updated query result, respectively. In Fig. 2(a), both $n_b$ and $n_c$ periodically interact or measure the queried POI and forward their query result to $n_a$. For instance, a set of wireless sensors are deployed for a wildfire monitoring system, where each sensor periodically senses the current temperature and sends it to a sink through multi-hop relays. Then the sink can actively monitor the specific region where sensors reply higher temperature, suspect a wildfire, and send back additional queries. A good deal of research effort has been devoted for how to efficiently collect the information of POIs while minimizing the query traffic in wireless multi-hop networks, where $k$ nearest neighbors ($k$NNs) or top-$k$ query processing techniques [3,9] are often deployed. Note that if a query replying node moves away from the queried POI, then the query issuing node needs to flood a query again to search another query replying node to maintain the validity of query result.In Fig. 2(b), $n_a$ re-floods a query when it does not receive any further query result from $n_b$ because of its mobility.

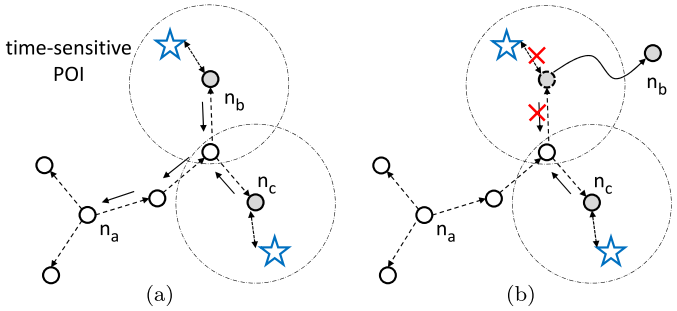


**Fig. 2.** A query for a time-sensitive POI. Here, a query reply is represented as a solid arrow.

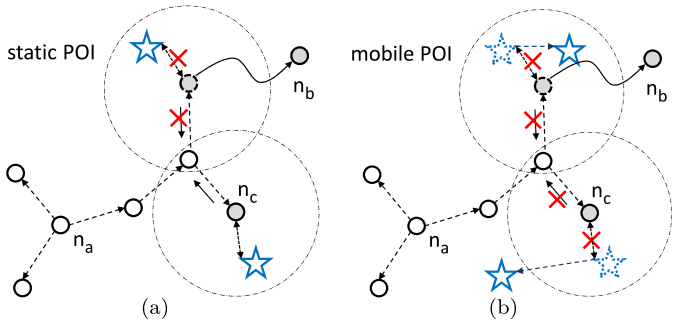


**Fig. 3.** A location-sensitive query for a static or mobile POI.

Second, a location-sensitive query is to retrieve a location information of static or mobile POI. The location information of static POI is not changed or does not change in a short period of time, such as geographical spots.When a node receives a query, it replies the current location as a query result if the queried POI can be detected.The mobility of query replying node does not affect the validity of query result. Thus, a query issuing node sends a query only when it moves away from the queried POI and needs a new query result. A query replying node also replies a query result only once. In Fig. 3(a), although $n_b$ moves away from the queried POI, $n_a$ does not issue an additional query. In fact, $n_a$ does not know whether $n_b$ has been moved away from the POI. In mobile POI, however, we refer a moving object and its location information is time-varying. In a military application, for example, a mobile POI can be a soldier, tank, or unmanned aerial vehicle (UAV). It is hard to keep track of mobile POI and update the query issuing node of location information, incurring significant communication overhead. In this paper, we consider two cases depending on whether mobile POI moves within or out of the communication range of query replying node as shown in Fig. 3(b). If a mobile POI moves within the communication range, the query result what $n_b$ has sent to $n_a$ is still valid.If a mobile POI moves out of the communication range, $n_c$ cannot detect it anymore and thus, it cannot reply a query result.

In this paper, we focus on the location-sensitive query for both static and mobile POIs and their corresponding query processing techniques in MANETs.

## 4. Validity region sensitive query

In the VRSense, we first propose four validity region-based query processing schemes with static POIs in MANETs.We also enhance and vary one of our proposed schemes for performance comparison.

### 4.1. Validity region

Although the mobility of query replying node does not affect the validity of query result in a time-insensitive query, it becomes an issue when a query issuing node needs to rebroadcast the same query. A periodic query rebroadcast may not be the best solution because of its redundancy that can lead to the flooding and significant energy consumption. In light of this, we deploy a validity region based query processing approach in MANETs. The validity region is an area where a query result remains the same as long as a query issuing node is located within the area. A network is virtually divided into a set of grids, where each square region is called *cell* and has the same size. Since each node is equipped with an on-board GPS, it knows the current location and corresponding cell.Thus, the query issuing node can rebroadcast the query only when it moves out of the validity region.

In this paper, we propose how to approximate a validity region and its corresponding query processing operation to minimize the number

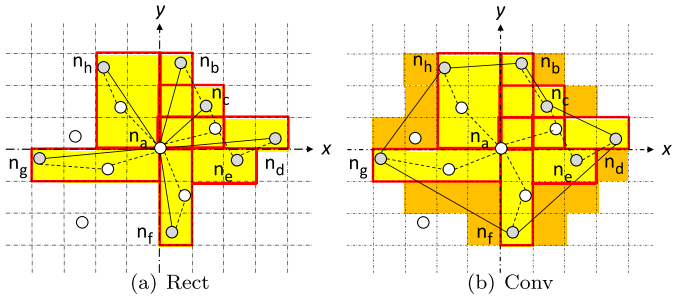


**Fig. 4.** A rectangle based validity region consists of cells (marked by yellow) located within the rectangles (marked by red) based on the locations of query replying nodes (marked by gray circle). However, a convex hull based validity region has more cells (marked by orange for additional cells) than that of the rectangle based approach. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

of queries broadcasted: (i) rectangle based (*Rect*) and (ii) convex hull based (*Conv*) approaches. First, the basic idea of Rect is to set a validity region based on the locations of query issuing node and query replying nodes.When a node receives a query broadcasted from a query issuing node, it replies a query result piggybacked with its current location.Whenever the query issuing node receives a query result, it sets a rectangle having a diagonal line connecting from the current location of query issuing node and to the location of query replying node.All the corresponding cells that cover the rectangle are considered as a part of validity region, and the query issuing nodes only store the list of cell *ids*. For example, suppose $n_a$ generates a query, broadcasts it, and receives its query results from $n_b$ to $n_h$ as shown in Fig. 4(a).Then $n_a$ constructs a set of rectangles and finds the corresponding cells that cover the rectangles.Note that the constructed rectangles based on the locations of query replying nodes, $n_b$ and $n_c$, can have the overlapped cells. The Rect is simple but its overall shape of validity region is an irregular polygon that often contains either any length of sides or size of angles.In Fig. 5(a) and (c), we snapshot the shapes of validity region when a query issuing node approximates. This implies that the query issuing node can easily be out of the validity region while it is moving and thus, it may need to frequently rebroadcast a query.

Second, in the Conv, we deploy the convex hull algorithm to approximate a validity region, where a convex hull is the smallest convex polygon containing a given set of points in a two-dimensional area [23]. Similar to the rectangle based approach, when a node receives a query and finds a queried POI, it replies a query result piggybacked with its current location. After receiving all query results, a query issuing node builds a polygon as a validity region based on the locations of query replying nodes. Note that we use the Graham-Scan based method [23] to construct a validity region.We initially choose a location with the minimum x- and y- coordinates, sort the rest of locations based on the angle to the location in counterclockwise order. Then the initially chosen location is connected with a location with the minimum angle. We keep incrementally connecting the locations only if they are located to the counterclockwise of the line connecting previous two locations. For example, $n_a$ builds a polygon with the locations of query replying nodes ($n_b$ to $n_h$) using the convex hull algorithm as shown in Fig. 4(b). All the corresponding cells that cover the polygon are considered as the validity region. The Conv is also simple but its validity region includes more cells compared to that of the Rect. More importantly, the overall shape of validity region is less irregular and closer to a circle as shown in Fig. 5(b) and (d). This may reduce the number of queries generated and broadcasted while the query issuing node is moving. The major operation of Conv is summarized in Fig. 6.

In summary, when a query issuing node generates a query, the

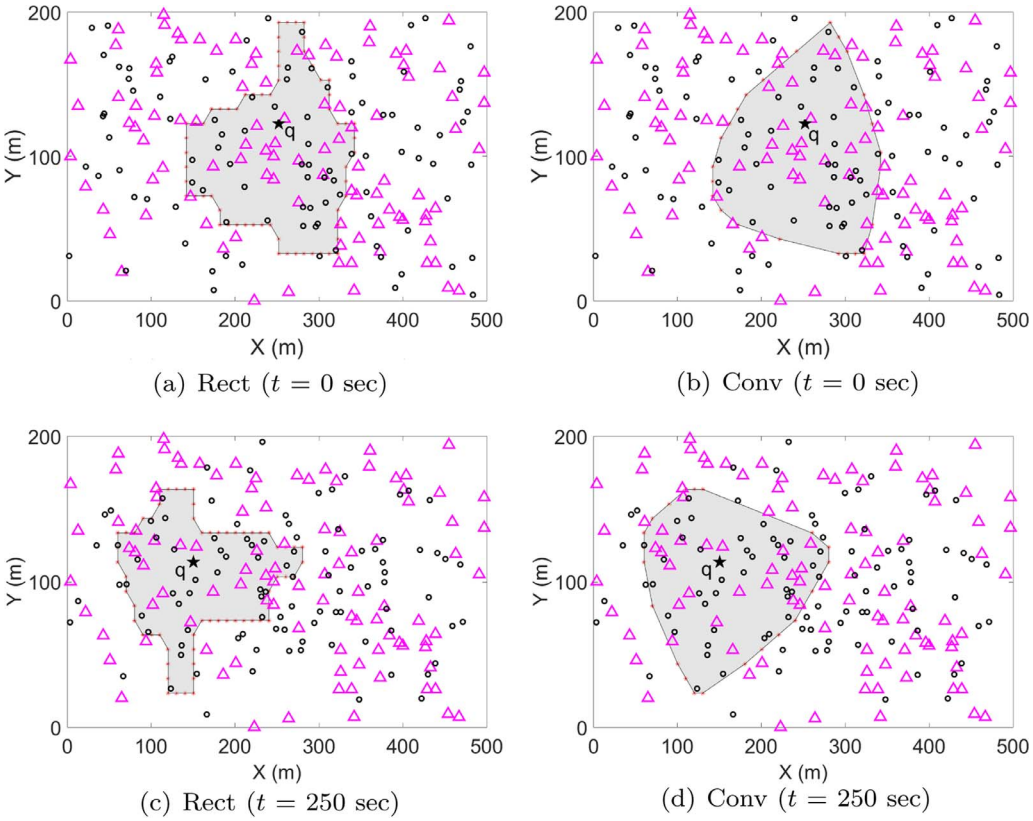


**Fig. 5.** A snapshot of validity region. We experiment with 100 nodes (marked by a circle), $q$ as the query issuing node (marked by a star) and POIs (marked by a triangle) in a network, where nodes are uniformly distributed and move according to the random waypoint mobility, 1.0 m/sec. We show a set of validity regions at the initial ($t$ = 0 sec) and after $t$ = 250 sec using rectangle and convex hull based approaches, respectively.

- $Rpy[nid', POI_{data}, l_{nid}(x,y)]$: A reply packet forwarded back to the query issuing node ($nid'$) with the queried data ($POI_{data}$) and the current location of query replying node ($nid$), $l_{nid}(x,y)$.
- $k$, $k'$, $k''$: An index incremented by one and initially set by zero.
- $B_{loc,k}$: A buffer to save the received replying node's location, $l_{nid}(x,y)$, into the $k^{th}$ slot.
- $R_{loc,k'}$: A set of locations forming a validity region.
- $G_{k''}$: A set of virtual grid ids corresponding with each location of $R_{loc,k'}$.

◇ When a query issuing node, $n_i$, receives a reply packet from a query replying node, $n_j$,
  if $l_j(x,y) \notin B_{loc}$
    Save $l_j(x,y)$ into $B_{loc,k}$;
  Sort the $B_{loc,k}$ based on the angle in counterclockwise order;
  Initiate the location with the minimum $x$- and $y$-coordinates;
  for $\forall$ $B_{loc}$ do /* Graham-Scan based [23] */
    if $B_{loc,k}$ is located to the counterclockwise of the line connecting from $B_{loc,k-2}$ to $B_{loc,k-1}$
      Add $N_{loc,k}$ to $R_{loc,k'}$;
    Replace $B_{loc,k}$ with $B_{loc,k-1}$;
  for $\forall$ $R_{loc}$ do
    Map $R_{loc,k'}$ into a virtual grid id and save it into $G_{k''}$;

**Fig. 6.** The event-driven pseudo code of convex hull based validity region.

query is limited to be flooded up to the maximum number of hops ($h_{\max}$).When a node closely located to the POI receives the query, it sends a reply directly or indirectly via multi-hop relays. Upon receiving the reply, the query issuing node initiates to build a validity region based on the locations of replying nodes using the Rect or Conv.If the query issuing node is out of bound of the validity region due to its mobility, it generates a query again. The major operation of query processing is also summarized in Fig. 7.

- $Rpy[nid', POI_{data}, l_{nid}(x,y)]$: Defined before.
- $Qry[nid, POI_{type}, h]$: A query packet flooded from a query issuing node ($n_{nid}$) with a POI type ($POI_{type}$) and the number of hops ($h$).

◇ When a node, $n_i$, generates a query,
  Broadcast a query packet, $Qry[i, POI_{type}, 0]$;
◇ When $n_j$ receives a query packet from $n_i$,
  if $++hop > h_{max}$
    Drop the query packet and exit;
  if $POI_{data}$ is found
    Send a reply packet, $Rpy[i, POI_{data}, l_j(x,y)]$, to $n_i$;
  Rebroadcast the query packet;
◇ When $n_i$ receives a reply packet from $n_j$,
  Build a validity region with the received $l_j(x,y)$ based on the Rect or Conv; /* e.g., Fig. 6 */
◇ When $n_i$ is located in out of bound of its validity region,
  Broadcast a query packet again, $Qry[i, POI_{type}, 0]$;

**Fig. 7.** The event-driven pseudo code of proposed query processing.

### 4.2. Enhancements

In the Conv, a query issuing node may not create a validity region if the number of replies is not enough. Due to the network condition, the node may experience a delay even before creating the validity region. Since the node does not know when and how many replies will arrive, it may blindly wait for the replies which are in fact dropped during the transmission. A query can also be dropped during the transmission. Thus, we propose an adaptive approach by combining both Rect and Conv, called as *Adapt*, to promptly build the validity region. In the Adapt, when a query issuing node receives the first reply, it immediately builds an initial validity region by conducting the Rect.Then the node keeps extending the validity region upon receiving the

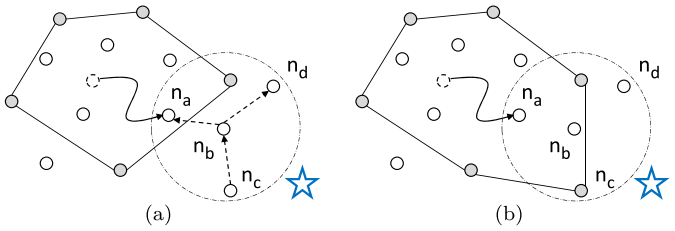


**Fig. 8.** A validity region is adaptively updated by opportunistically overhearing a reply forwarded to another querying-issuing node.

following replies while it receives enough number of replies to build a validity region based on the Conv.

Note that the proposed three approaches (Rect, Conv, and Adapt) use a static validity region and its shape does not change once a query issuing node builds the validity region. However, the node will ultimately move out of the bound of validity region because of its mobility and thus, it will generate a query again.In this paper, we also propose a scheme to update the validity region, called *Adapt:Ovhr*, to reduce the number of queries.The basic idea is to update the current validity region by opportunistically utilizing the overheard on-flying replies. In the Adapt:Ovhr, the node initially builds a validity region based on the Adapt, in which the validity region can be shaped combining by the Rect or Conv depending on the number of received replies. When the node overhears a reply forwarded back to another query issuing node, if the reply contains the same POI what the node queried, it updates the current validity region by including the location of the query replying node. Here, the node filters the location and only considers it that is out of the current validity region. The benefit of this Adapt:Ovhr is that the node can update the validity region toward the way where it moves, stay more time within the validity region, and save the number of queries.

For example, a query issuing node (e.g., $n_a$) has built a validity region and moved to a new location in Fig. 8(a).Another query issuing node (e.g., $n_d$) generates a query and a node (e.g., $n_c$) closely located to a POI sends a reply back to $n_d$. An intermediate node (e.g., $n_b$) forwards the received reply from $n_c$ to $n_d$.$n_a$ can overhear and check the reply whether it contains the same type of POI what $n_a$ queried before.If so, $n_a$ updates its validity region by including the location of query replying node, $n_c$, in Fig. 8(b). In addition, as $n_a$ moves, the location that becomes far away will be removed from the validity region.

### 4.3. Variant

In addition, we modify a safe exit algorithm [20] deployed in an infrastructure-based road network to work in MANETs, called *Greedy*, for the purpose of performance comparison. The Greedy is used as the performance lower bound. Here, the safe exit algorithm uses the Euclidean distance between a query issuing node and its queried POIs to determine when to regenerate a query for updating the query results. In the Greedy, we assume a centralized server that is aware of the network topology in a real-time manner. When a query issuing node generates a query, it sends the query to the server directly.The server ideally replies all the locations of neighbor nodes located within the maximum distance of three hops ($d_{\max}$, i.e., $h_{\max} \times r$) from the query issuing node. Here, $r$ is the radio transmission range. Whenever the query issuing node moves, it keeps recalculating the distances to the locations of query replying nodes whether it is still within the distance. In this paper, unlike the Greedy, the proposed strategies use a list of virtual cells corresponding to the validity region. Thus, a query issuing node only remembers a list of cell ids involved in its validity region, compares the current cell whether it is in the list, and determines when to generate a query again.

## 5. Validity region sensitive query with mobile point of interests

In the VRSense, we also extend the proposed query processing strategies for mobile POIs. We first observe the mobility of POIs and its impact on the validity region and then present two techniques to reset the validity region accordingly.

### 5.1. Mobile POIs vs. validity region

In the proposed query processing strategies, we implicitly assume static POIs that do not move or do not move in a short period time.A query replying node detects a queried POI located within its communication range and replies the current location as a query result to a query issuing node. Since the queried POI is stationary, a validity region built with the locations of query replying nodes is not reset until the query issuing node moves out of the validity region. In fact, the mobility of query replying nodes does not affect in building the validity region when it needs to be reset in order to update the query result.

We need to relax this assumption by considering the mobility of POIs and observe its impact on the validity region.Since each query replying node detects the queried POI located within its communication range, the basic idea is to determine whether the queried POI stays within or moves out of the communication range of query replying node. If the queried POI moves out of the boundary of communication range, the location initially sent by the query replying node is not valid anymore. Thus, the original validity region built with the location should be updated in order to refresh the query result. For example, we observe the mobility of POIs and show the snapshots of validity region in Fig. 9. We experiment with both Rect and Conv under a 200 $\times$ 500 $m^2$ rectangle network, where 100 nodes and 200 POIs with 20 POI types are uniformly distributed. A node ($q$) generates a query for one of POIs and builds a validity region with the locations replied by a set of query replying nodes. Here, all nodes and the part of POIs (e.g., 10% of total number of POIs) move according to the random waypoint mobility. Fig. 9(a) to (c) and (d) to (f) show the snapshots of validity regions built by the Rect and Conv, respectively. Initial validity regions are formed in Fig. 9(a) and (d), and they remain same because the query issuing node believes that it is still within the validity region in Fig. 9(b) and (e). However, a part of queried POIs has moved out of the communication range of the query replying nodes and thus, the locations originally sent to the query issuing node for building an initial validity region become invalid. In Fig. 9(b) and (e), we reset the validity regions based on the updated locations of node and POIs and observe that the query issuing node is already out of the validity region. This indicates that the query issuing node should have generated a new query earlier to update the query result.

In summary, it is non-trivial or requires heavy communication overhead for the query issuing node to keep track of all the locations of mobile POIs what the query replying nodes have detected. The mobility of POIs can affect the shape of validity region. For example, the size of newly built validity regions (see Fig. 9(c) and (f)) becomes smaller than that of originally built validity regions (see Fig. 9(c) and (f)). Since the query replying nodes also move, the query issuing node does not know when the validity region should be reset and whether it is still located within the validity region. Thus, the validity region built with static POIs should be refined in the presence of mobile POIs.

### 5.2. Flexible validity region

We propose how to approximate a validity region and its corresponding query processing strategies with mobile POIs. The basic idea of the first approach is to reduce the size of validity region, called *Reduce*. The rationale behind this approach is that a query issuing node may need to generate a query again to update the query result even though it is located within the validity region, which is built based on static POIs.As we can observe in Fig. 9(c) and (f)), however, the query

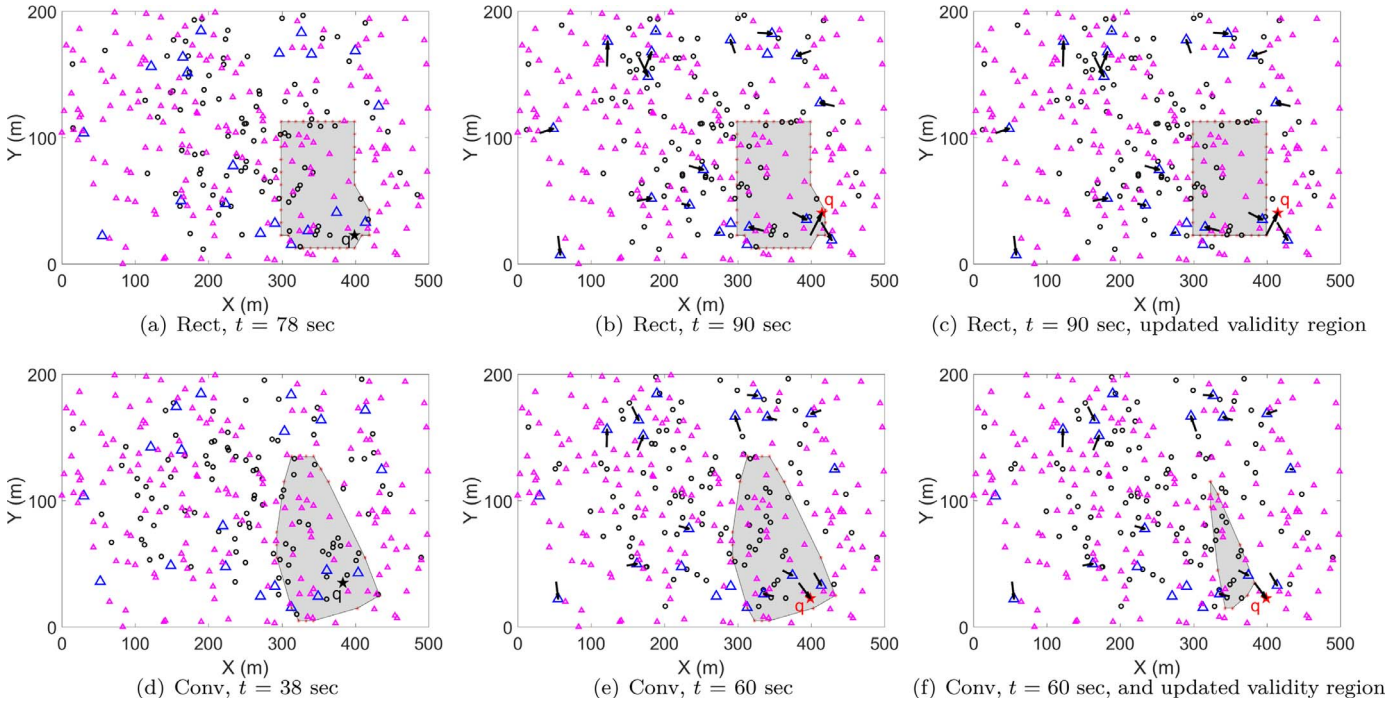


**Fig. 9.** Snapshots of initial validity region (Fig. (a) and (d)), original validity region after time elapse (Fig. (b) and (e)), and updated validity region (Fig. (c) and (f)). Here, locations of original query issuing node ($q$) and its replying nodes are marked by a black star and black circle, respectively. The current location of query issuing node is marked by a red star. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

issuing node is not aware of whether it is already located out of the validity region and misses to send a query. There is a tradeoff between the size of validity region and the accuracy of query result.With the reduced validity region, the query issuing node frequently generates a query to update the query result but the query traffic increases. For example, we depict the reduced validity regions based on the Rect and Conv, called *Reduce:Rect* and *Reduce:Conv*, respectively in Fig. 10. In Fig. 10(a), a query issuing node ($n_a$) sets a reduced validity region based on the locations of query replying nodes, single-hop away from the query issuing node. In prior validity region based on the Rect (see Fig. 4(a)), the query replying nodes located two or three hops away are involved in building the validity region.Thus, the Reduce:Rect sets a smaller validity region, where a set of cells (marked by gray) is not used in the reduced validity region. Similarly the Reduce:Conv shows a reduced validity region in Fig. 10(b), where additional cells (marked by orange) are shown compared to the validity region built by the Reduce:Rect. The Reduce:Conv loses more cells than that of the Reduce:Rect in the reduced validity region compared to the original validity region. The second approach is to assign a re-query probability ($p_{qry}$) to each cell located within the validity region, called *Prob*.The basic idea is when a query issuing node moves and becomes closer to the boundary of validity region, it has a higher probability of re-issuing

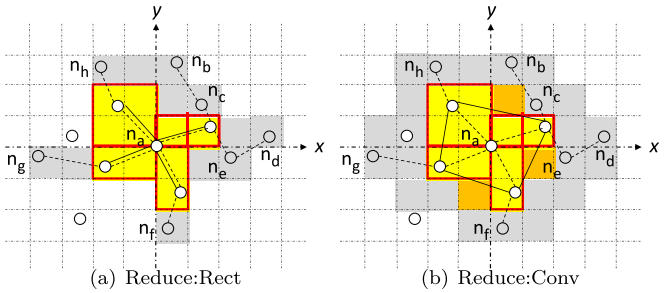


**Fig. 10.** The size of validity regions have been reduced with the Rect (*Reduce:Rect*) and Conv (*Reduce:Conv*), respectively.A set of cells (marked by gray) that was originally involved in the validity region is not belong to the reduced validity region anymore.

a query to re-build the validity region.In the Prob, the query issuing node does not blindly wait until it moves out of the validity region, but it pro-actively updates the validity region. This is because when the query issuing node is located close to the boundary of validity region, it might be already located out of the validity region due to the mobility of POIs (see Fig. 9(c) and (f)). For example, when a query issuing node ($n_a$) sets a validity region based on the Conv as shown in Fig. 11(a), it assigns $p_{qry}$ to each cell within the validity region. Whenever the query issuing node moves and visits a cell, it generates a random number (e.g., *rand[0, 1]*), compares the number with $p_{qry}$ assigned, and decides whether to reset the validity region by re-issuing a query.

In this paper, $p_{qry}$ is assigned to each cell based on its relative location within the validity region. The inner and outer located cells are assigned higher and lower $p_{qry}$, respectively. Here, $p_{qry}$ is equal to 1 for the cells located out of the validity region. This implies that the query issuing node always re-issues a query when it moves out of the validity region. In order to identify the relative location, we group the cells located from the boundary of validity region in a *ring*-based manner in Fig. 11(b). For example, a set of cells located at the boundary of validity region is considered as the same group and is assigned the same $p_{qry}$. That is, cells located in $i$th ring ($g_i$) have the same $p_{qry}$, where $i$ ranges from 1 to the total number of rings (e.g., $r = 3$). Similarly we continue to group the cells located in the next inner ring until there is no more cell left in the validity region. Then $p_{qry}$ of each cell located in $i$th *ring* is calculated as,

$$p_{qry}(g_i) = \frac{(r+1)^2 - i^2}{(r+1)^2} \cdot \log_{r+1} d, \tag{1}$$

where $d$ is $r$ - $i'$ + 1. Here, $i'$ is the ring number of cell where the query issuing node is currently located in. In Fig. 11(c), we calculate $p_{qry}$ when the query issuing node is located in the center of validity region (e.g., $i' = 3$). Overall $p_{qry}$ are low because the query issuing node is far away from the boundary of validity region. However, $p_{qry}$ increases when the query issuing node is located close to the boundary (e.g., $i' = 1$) in Fig. 11(d).

We also group cells from the center to the boundary of validity

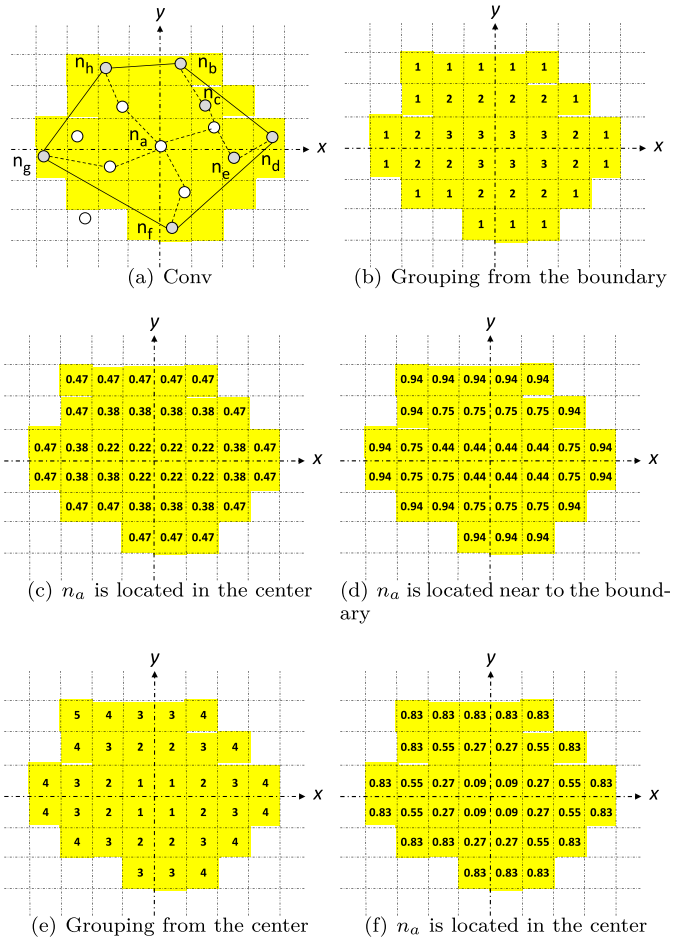


**Fig. 11.** A set of re-query probabilities is assigned to each cell located within the validity region, set by the Conv.

region in a ring-based manner in Fig. 11(e). Since the shape of validity region is often not a circle but a polygon, more number of groups with smaller number of cells are observed, e.g., $r = 5$. In this grouping, we calculate $p_{qry}$ of each cell located in $i$th ring differently using Eq. 2, where $|g_i|$ and $s$ are the number of cells located in the $i$th ring and the total number of cells located in the validity region, respectively. $w$ is a weight factor and is initially set to 0.5.

$$p_{qry}(g_i) = \begin{cases} \dfrac{\sum_{j=1}^{i} |g_j|}{s} \cdot w & \text{for } i \le i'' \\ \dfrac{\sum_{j=1}^{i} |g_j|}{s} \cdot \left( w + \dfrac{\sum_{j=i''}^{r} g_j}{s} \right) & \text{for } i > i'' \end{cases} \quad (2)$$

Here, $i''$ denotes the first ring number $i$ that satisfies $|g_i| \le |g_{i+1}|$, showing that the number of cells of outer ring is less than that of the inner ring. As the number of ring increases, the number of cells in the ring (i.e., $i$th ring) decreases because of the irregular shape of validity region. This indicates that the cells located in the outer ring are often located in the boundary of validity region, and thus higher $p_{qry}$ is assigned.In addition, the cells located in the inner ring (e.g., see 3rd ring in Fig. 11(e)) can be located in the boundary of validity region, and thus Max($p_{qry}(g_i)$) is assigned, where $i$ ranges from one to $r$. In Fig. 11(f), we calculate $p_{qry}$ when the query issuing node is located in the center of validity region. The $p_{qry}$ of inner cells is smaller compared to those shown in Fig. 11(c) and (d). In this paper, we deploy Eq. 2 to calculate $p_{qry}$.

- $Qry[nid, POI_{type}, h]$, $k$: Defined before.
- $r$: A random number between 0 and 1.
- $V_i$: A validity region what a query issuing node, $n_i$, sets.
- $c_j$: A cell located within a validity region.
- $p_{qry,c_j}$: A re-query probability assigned in a cell, $c_j$.

◇ When a query issuing node, $n_i$, receives all reply packets from query replying nodes,
  Build a validity region, $A_i$; /* Rect, Conv, Adapt, or Adapt:Ovhr */
  **for** $\forall\ c_k \in V_i$ **do**
    Calculate $p_{qry}$ and assign it to the corresponding cell, $p_{qry,c_k}$; /* Eq. 2 */
◇ When $n_i$ moves to an adjacent cell, $c_j$,
  **if** $p_{qry,c_j} == 1$ /* Move out of the validity region */
    Re-issue a query, $Qry[i, POI_{type}, 0]$;
  **else**
    if ($r \leftarrow$ rand[0, 1]) $> p_{qry,c_j}$
      Re-issue a query, $Qry[i, POI_{type}, 0]$;
  **endif**

**Fig. 12.** The event-driven pseudo code of Prob.

The major operations of Prob is summarized in Fig. 12.

## 6. Performance evaluation

We evaluate and compare the performance of proposed query processing strategies with both static and mobile POIs by changing key simulation parameters in MANETs.

### 6.1. Simulation testbed

We developed a customized discrete-event driven simulator by using the OMNeT++ [21]. A 200 × 500 $m^2$ rectangle network is deployed, where 100 nodes are uniformly distributed. The network is virtually divided into a set of grids, where each square cell size ($c_{size}$) is set to 10 × 10 $m^2$. We also vary the cell size set to 5 × 5 $m^2$ or 20 × 20 $m^2$, and use 10 × 10 $m^2$ cell size by default unless otherwise specified. Each node is equipped with an on-board GPS receiver and is aware of the current location. The radio transmission range ($r$) is assumed to be 50 m, and the two-ray ground propagation channel is assumed with a data rate of 2 Mbps. We deploy a simple CSMA/CA based medium access for the link layer. In this paper, we set the maximum number of hops for query propagation ($d_{\max}$) as three in order to limit unnecessary query propagation that may cause the broadcast storm problem [24]. Note that there is a performance tradeoff depending on the $d_{\max}$. For example, if $d_{\max}$ becomes smaller, a query issuing node may fail to receive any query reply or may not receive enough number of query replies to build a validity region. If $d_{\max}$ becomes larger, however, a query issuing node may receive redundant query replies and incur significant query traffic that can also interrupt other query operations. Here, $d_{\max}$ can be set differently and changed adaptively based on the requirement of query operations but it is out of scope of this paper.

The POIs are uniformly distributed in the network, where the total number of POIs and POI types ($POI_{type}$) are set to 200 and 20, respectively. Each node generates a query with the maximum number of POI types ($Q_{type}$) for POIs, i.e., $Q_{type}$ is set to three or eight. For example, if $Q_{type}$ is three, a query issuing node queries with three types of POIs in the network, where any node who can find a POI with any one of three types can reply. The more $Q_{type}$ is set, the more queried POIs the node receives. The query is limited and flooded up to three hops ($h_{\max}$) to reduce the network traffic and its queried data size is set to 16 Kbits. Note that since each node does not re-generate a query as far as it is

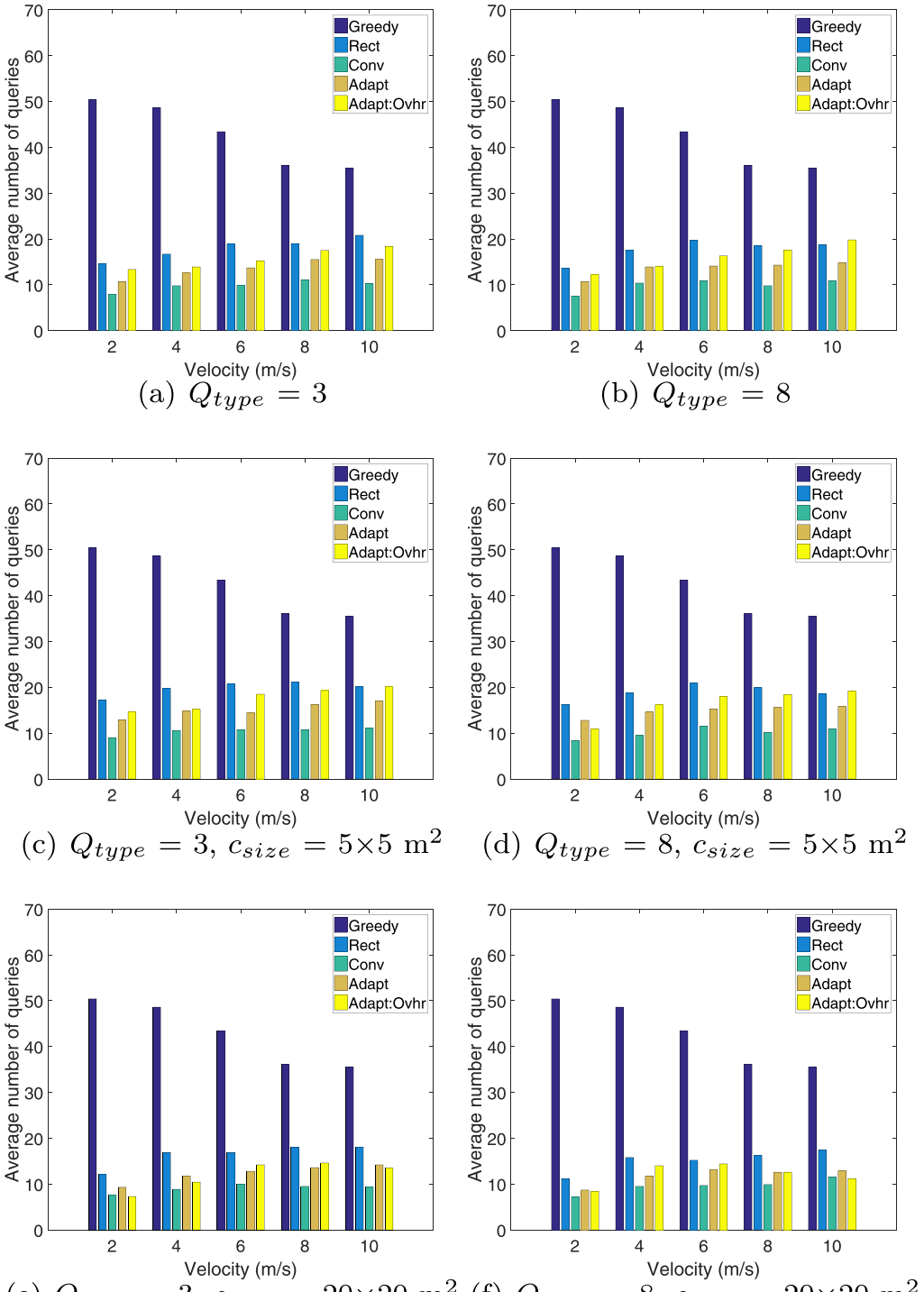


**Fig. 13.** The number of queries generated against the number of POI types, node velocity, and cell size.

located within its validity region, it does not follow a periodic query arrival pattern. The random waypoint mobility model [25] is used to simulate the node mobility with a node speed from 2 to 10 m/s for static POIs. In case of mobile POIs, the speed of both node and POI are fixed to 2 m/s.A pause time is between 0 to 50 s. With this mobility model, each node travels toward a randomly selected destination in the network.After the node arrives at the destination, it pauses for the predetermined period of time and travels toward another randomly selected destination. 10% of total number of POIs are mobile and follow the random waypoint mobility model.

For performance evaluation, we repeat the simulation 10 times by changing random seed numbers that primarily affect network topology, node mobility, and POI position. Each simulation runs up to 1000 seconds. The results are collected using 90% confidence interval and the predicted values lie within 10% of the mean. Due to the lack of space[2], we do not include the confidence interval in Figs. 13–15 but include in Figs. 17 and 18. We consider four major performance metrics in the presence of static and mobile POIs in MANETs: average number

[2] In Figs. 13–15, five approaches are compared with five different velocities presenting total 25 bar graphs. Due to the lack of space within the graphs, we do not include the confidence interval in these figures but the same performance trend is observed.

of queries; average time spent in validity region; total time spent in validity region; and number of overheard packets.

### 6.2. Impact of static POIs

We evaluate the performance of proposed four query processing strategies with static POIs in terms of the number of queries, time period spent in validity region, and percent of time spent in validity region: rectangle based (*Rect*), convex hull based (*Conv*), adaptive (*Adapt*), adaptive with overhearing (*Adapt:Ovhr*). We also modify a recent safe exit algorithm [20] deployed in an infrastructure-based network to work in MANETs, called *Greedy*, for performance comparison. We vary the number of POI types, grid size, and node velocity.

**Number of queries:** We measure the number of queries what each node generates for entire simulation time in Fig. 13. In Fig. 13(a), the Greedy shows the highest number of queries compared to other schemes.In the Greedy, a query issuing node receives the replies from more number of neighbor nodes located within $d_{\max}$ in low velocity, compared to the case in high velocity. Whenever the query issuing node moves, it re-computes the distance to each query replying nodes and regenerates a query if there is any distance over $d_{\max}$.Since the query issuing node computes the distance to more number of query replying nodes in low velocity, it is frequently over $d_{\max}$ while it moves and generates more number of queries compared to the case in high velocity.Thus, the number of queries is reduced as the velocity increases. Unlike the Greedy, all the Rect, Conv, Adapt, and Adapt:Ovhr use a validity region based on the virtual cell and they are not sensitive to the distance to individual query replying nodes. The Rect shows higher number of queries than that of the rest of three schemes because it creates a validity region with an irregular polygon. A query issuing node may move out of the bound of validity region. However, the Conv can build a validity region with a less irregular polygon than that of the Rect and shows the lowest number of queries. In Fig. 13(b), with higher $Q_{type}$, a query issuing node can receive more number of query replies. The number of queries slightly reduces in the Rect, Conv, Adapt, and Adapt:Ovhr for entire velocities. In addition, we observe the impact of cell size on the number of queries in Fig. 13(c) to (f). When the cell size becomes smaller in Fig. 13(c) and (d), the number of queries increases in all schemes because the size of validity region becomes smaller. Thus, the query issuing node may move out of the bound of validity region more often and generate a query again. In case of the larger cell size in Fig. 13(e) and (f), however, overall number of queries decrease in all schemes because the size of validity region becomes larger.

**Average time spent in validity region:** We measure the time period how long a query issuing node stays within its validity region in Fig. 14. In Fig. 14(a), the Conv shows the longest time period in low velocity, i.e., 2 m/s. Since a query issuing node receives the replies from more number of neighbor nodes in low velocity, it can create a validity region with a less irregular polygon compared to that of the Rect. Both Adapt and Adapt:Ovhr also show the similar performance because they are based on the Conv. As the velocity increases, the Adapt:Ovhr shows the longest time period because a query issuing node adaptively expands its validity region toward the way where it moves by opportunistically overhearing on-flying query replies. Thus, the query issuing node can stay in the validity region for longer period. With higher $Q_{type}$ in Fig. 14(b), the time period increases for entire velocities because a query issuing node receives more replies and it can create a validity region with regular polygon, e.g., closer to a circle. Note that the Greedy shows the shortest time period for entire velocities. Since the distance from a query issuing node to any query replying node frequently is over $d_{\max}$, the query issuing node stays less time period in its validity region and generates more queries (see also Fig. 13). In Fig. 14(c) to (f), we measure the impact of cell size on the time period spent within the validity region.When the cell size is larger (see Fig. 14(e) and (f)), overall time period increase significantly compared to that of smaller cell size (see Fig. 14(a) to (d)). This is because the query issuing node located in the validity region consisting of larger cells receives the less effect of mobility in terms of velocity. Thus, the query issuing node stays longer in the validity region.

**Percentage of time spent in validity region** We measure the percentage of time period how long a query issuing node stays in its validity region for entire simulation time in Fig. 15. The Greedy shows the highest percentage for entire velocities and $Q_{type}$ in both Fig. 15(a) and (b).In the Greedy, although a query issuing node stays in its validity region for a short period time (see also Fig. 14), it regenerates a query and quickly builds another validity region through a centralized server. This can keep the query issuing node staying almost always in the validity region. In Fig. 15(a), the Adapt:Ovhr shows the second highest percentage for entire velocities. In Fig. 15 (b), the Adapt:Ovhr shows lower performance than that of the Rect in low velocity but shows better performance as the velocity increase. This is because the validity region is adaptively extended based on the overhearing of on-flying queries. Note that the Rect shows higher percentage than that of the Conv and Adapt. In the Rect, a query issuing node can begin to create its validity region immediately after receiving the first reply. Then the query issuing node incrementally expands the validity region whenever it receives any following reply. In the Conv, however, the query issuing node may experience a delay before creating its validity region because it has to wait to receive at least three replies based on the convex hull algorithm. Fig. 15 (c) to (f) show the impact of cell size on the percentage of time period spent within the validity region. In smaller cell size (see Fig. 15 (c) and (d)), overall percentages decrease because the size of validity region becomes smaller. The Adapt:Ovhr does not affect much by the smaller cell size but the percentage increases as the velocity increases.Due to the overhearing, the Adapt:Ovhr also increases the percentage as $Q_{type}$ increases because of more chances in receiving on-flying queries. In larger cell size (see Fig. 15 (e) and (f)), overall percentages increase because the size of validity region becomes larger. In the Adapt:Ovhr, the percentage does not increase much as the velocity increases, because the validity region does not expand much. Since most locations of overheard queries are found in the current validity region, the query issuing node does not expand the region.

We also measure the number of overheard packets by changing the number of POI types and velocities in the Adapt:Ovhr. As shown in Fig. 16, the node mobility helps to overhear more number of query replies in the network, but the number of POI types does not affect much to the performance. As the velocity increases, each query issuing node can opportunistically overhear more on-flying replies and stay longer within the validity region by adaptively extending it.

### 6.3. Impact of mobile POIs

We evaluate the performance of proposed two techniques in forming validity region and their corresponding four query processing strategies with mobile POIs in terms of a number of queries, total time spent in validity region, and average time spent in validity region: reduced (*Reduce – Rect/Conv/Adapt/Adapt:Ovhr*) and probabilistic (*Prob – Rect/Conv/Adapt/Adapt:Ovhr*). Here, both $c_{size}$ and $Q_{type}$ are set by default, $10 \times 10\,m^2$ and 8, respectively. 10% of total POIs are assigned as mobile POIs. The speed of both node and mobile POI are set to 2 m/s. The Greedy is used as the performance lower bound for comparison.

First, we apply the Reduce to five different strategies and experiment them by changing the number of hops in query propagation in Fig. 17. If a query is propagated more number of hops, more number of nodes located around a query issuing node can response it, and thus the size of corresponding validity region becomes larger.In Fig. 17(a), as the number of hops increases, the number of queries decreases because the size of validity region increases. In particular, since the validity region set by the Adapt:Ovhr is affected by the number query replying nodes and overheard queries, the number of queries decreases rapidly. In Fig. 17(b), overall average time period spent within the validity region increase as the number of hops increases, because the size of

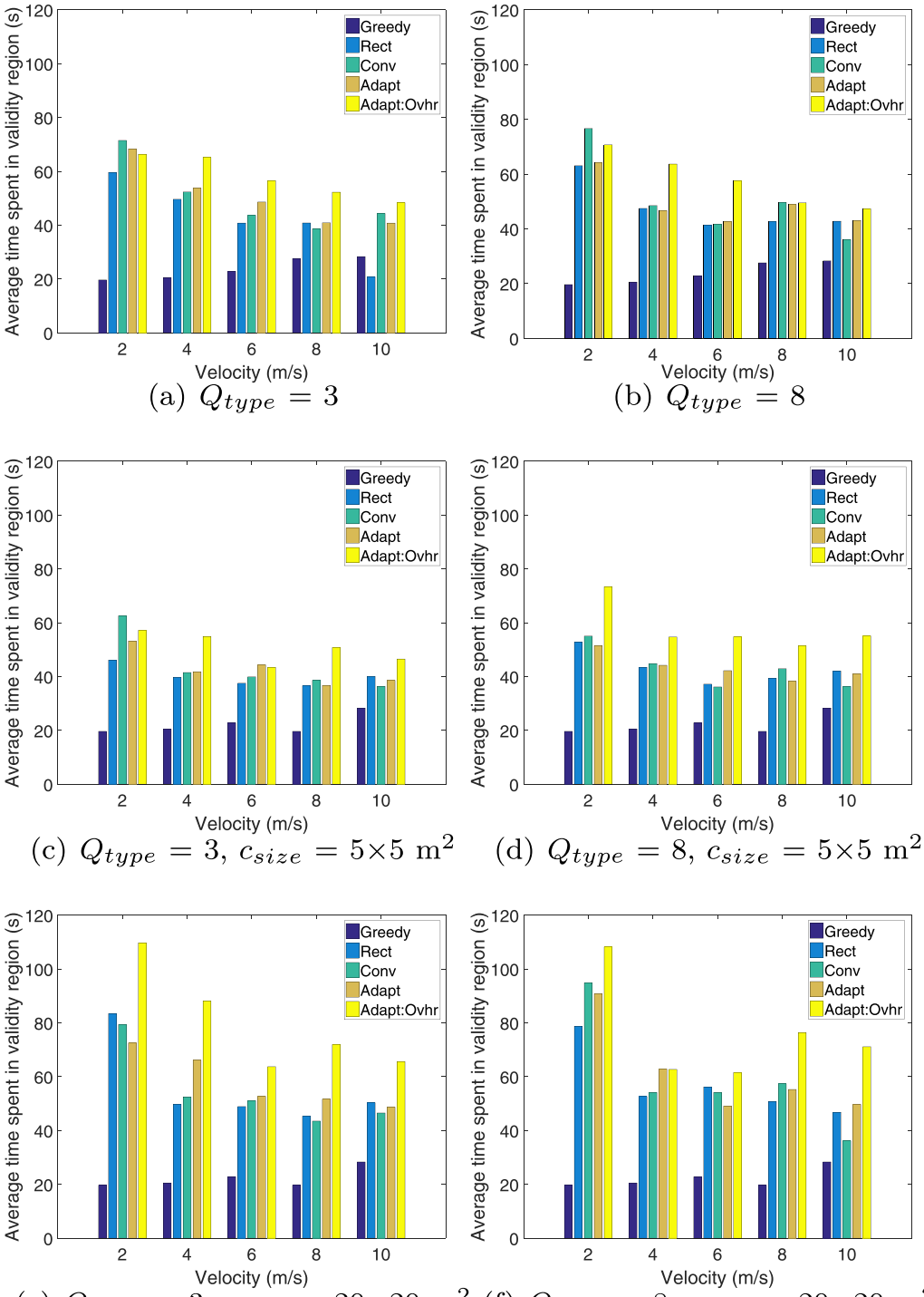


**Fig. 14.** The time period spent within the validity region against the number of POI types, node velocity, and cell size.

validity region becomes larger. The Adapt:Ovhr shows the longest time because the validity region can be expanded based on overheard queries. When the number of hops is one, all strategies except the Greedy show no difference because their sizes of validity region are similar based on the same location of query replying nodes. In Fig. 17(c), as the number of hops increases, overall percentage of time period for query issuing node staying within the validity region slightly increase.Due to the size of validity region based on the number of hops, both Adapt:Ovhr and Rect show higher percentage than that of both Adapt and Conv. Note that the performance of Adapt:Ovhr does not affect much by the number of hops because the query issuing node expands its validity region based on overheard queries during the movement.

Second, we experiment the Prob using the Eq. 2 in Fig. 18. In Fig. 18(a), the Prob increases the number of queries of all strategies, compared to the case without applying the Prob, called *Without-Prob*. Whenever the query issuing node visits new cell located in the validity region, it generates a random number and compares with the $p_{qry}$ ( $<$ 1) assigned to the cell. Thus, the query issuing node may generate a new query even without moving out of the validity region. Both Rect and Adapt:Ovhr show higher number of queries than that of both Conv and Adapt. Since both Rect and Adapt:Ovhr have larger validity region than that of both Conv and Adapt, the query issuing node has more chances to generate a new query while moving. In Fig. 18(b), the Prob decreases

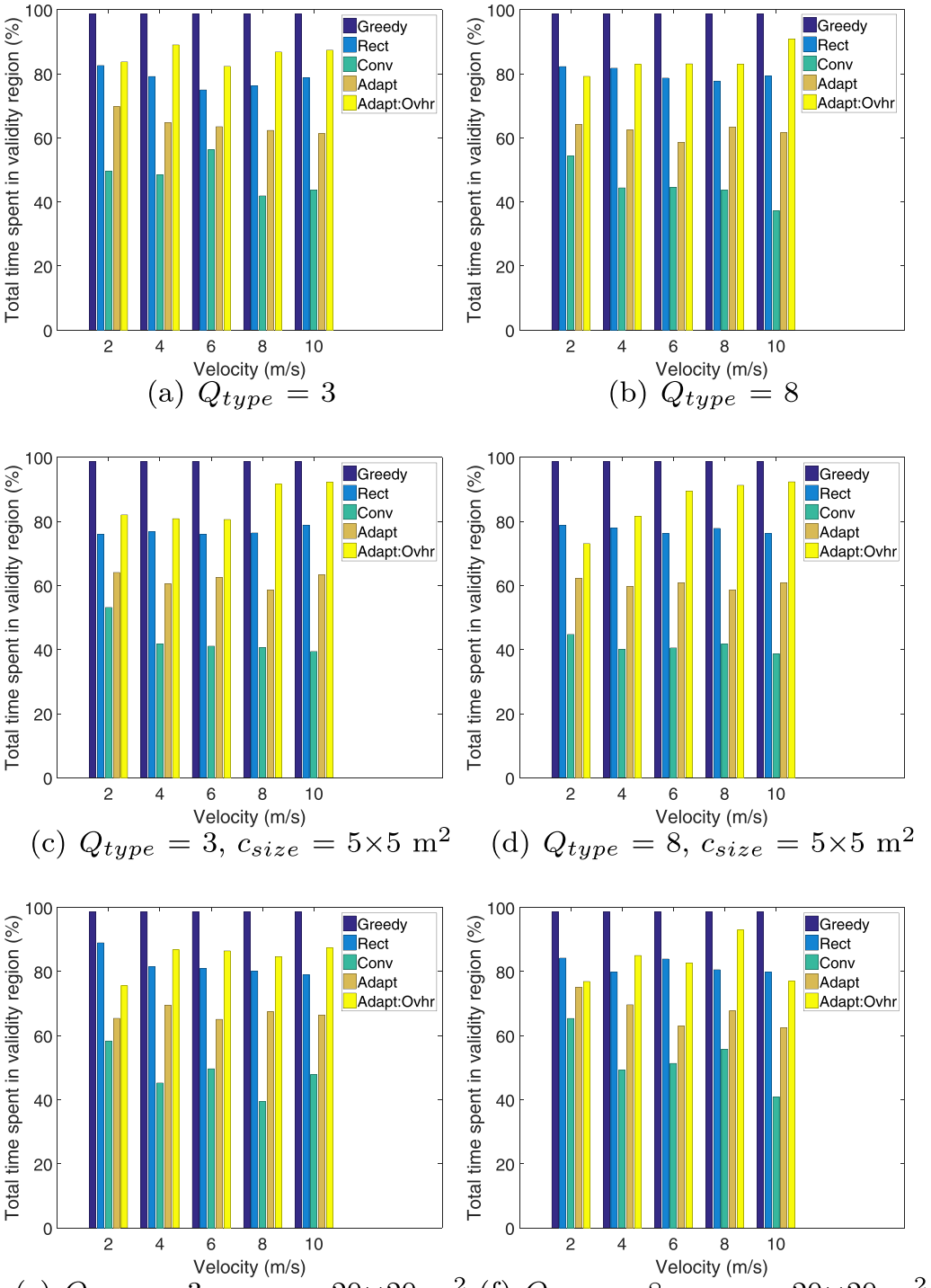


**Fig. 15.** The percentage of time period spent within the validity region against the number of POI types, node velocity, and cell size.

the time period spent in the validity region of all strategies when using the Eq. 2, compared to the Without-Prob. In particular, the time period of Adapt:Ovhr decreases significantly. Since the query issuing node expands the validity region based on overheard queries, the Adapt:Ovhr shows the longest time period in the Without-Prob. Due to the mobile POIs, however, the query issuing node may not aware of that it already has been located out of the validity region without re-issuing a query, as we observed in Fig. 9. The Prob can reduce the number of missing queries by pro-actively re-issuing a query based on the $p_{qry}$ assigned in each cell, leading to the lower time period. In Fig. 18(c), the Prob slightly decreases the percentage of time period for all strategies. Due to the expanding of validity region based on overheard queries, the Adapt:Ovhr does not affect much by the Prob but its percentage slightly decreases.

## 7. Discussion

In this section, we further identify two major issues in order to see the full potential of the proposed query processing strategies. We also summarize the cons and pros of the proposed query processing strategies, present other related issues, and compare them with prior region-based query processing strategies proposed under different environments [15,19,20,26].

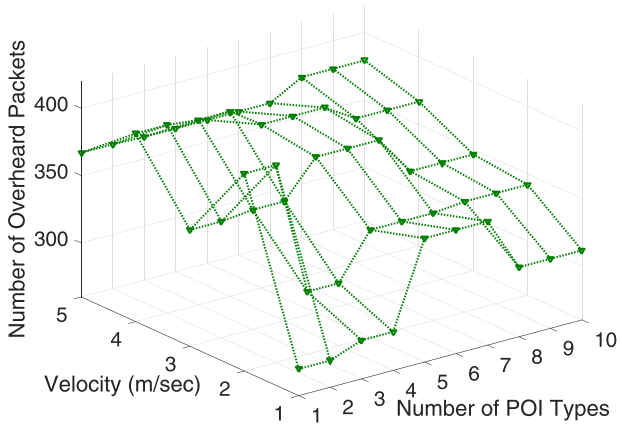


**Fig. 16.** The number of overheard packets against the number of POI types and node velocity.

### 7.1. Obsolete validity region

For mobile POIs, we approximate a validity region in best effort to reduce the number of queries and its corresponding query traffic in MANETs. Due to the non-negligible communication overhead for a query issuing node to track a queried POI, the validity region is not frequently updated according to the mobility of queried POI. As we can see in Fig. 9(c) and (f), the query issuing node might be located out of the validity region already but re-issue a query too late. Since the initially built validity region becomes obsolete, it is a critical issue either to efficiently approximate the validity region or update the query issuing node to aware when to re-issue a query in a timely manner. First, in order to enhance the accuracy of validity region, the query issuing node can periodically re-issue a query to update the validity region but this simple approach may occur a significant query traffic. Instead each query replying node can observe the direction of queried POI located with its communication range and return its location with an adjustment as a query result.For example, if the queried POI moves toward the east, the query replying node tries to estimate the location and replies by adding the adjustment. The location of queried POI would be more accurate if the query issuing node can correctly measure the speed of queried POI, but this may require additional equipment. Second, each query replying node can periodically provide an updated location of queried POI to the query issuing node by observing the queried POI even after returning its initial query result. The queried POI can be moved in and out from the communication range of query replying node. Thus, we plan to investigate how frequently the location of queried POI should be reported to the query issuing node and how adaptively adjust the validity region accordingly. We also plan to investigate the impact of validity region by considering a group mobility (e.g., platoon mobility) and diverse query patterns and types (e.g., time-sensitive).

### 7.2. Variable communication range based validity region

In this paper, we implicitly assume that all nodes have the same communication range. We need to relax this assumption by considering variable transmission power levels, often witnessed in the CISCO Aironet 340 and 350 series, Wi-Fi networks [27], and energy harvesting motivated networks (EHNets) [28]. Multiple communication ranges can lead to asymmetric links and affect in building the validity region.For example, a query issuing node may not receive all query replies from query replying nodes.A query replying node may choose a different route to deliver a query result to reach the query issuing node. The replying nodes with extended communication range may also affect the shape of validity region. Thus, the query issuing node needs to selectively choose the results of query replying nodes to efficiently build the validity region that can maximize the current query performance.

### 7.3. Other issues

We propose a framework for querying POIs based on a validity region in MANETs. Unlike prior region-based approaches [15,18–20], the proposed query processing strategies consider both static and mobile POIs and provide their associated query operations. Compared to a static validity region, both Reduce and Prob try to approximate a validity region for mobile POIs that often lead initially built validity region obsolete. Our approaches can reduce the number of redundant queries and thus reduce query traffic. Due to the broadcast nature in wireless networks, however, a query and its corresponding query reply can often be collided with any on-flying packet during the propagation in the network. Then the number of received query replies can be reduced and it can ultimately affect a validity region. In this paper, we use a simple CSMA/CA based medium access in order to clearly see the impact of validity region and its associated query operations. We believe that a better medium access control (MAC) protocol can reduce the number of collisions but it is out of scope of this paper.

We also compare the proposed query processing strategies with prior region-based approaches and summarize them in terms of five key aspects in Table 1. Note that the performance of prior region-based approaches cannot directly be compared without significant modifications because of inherently different network setup and computation method, such as VANET or centralized approach. In addition, unlike simulation-based experiments, we plan to design and develop a small-scale testbed for real experiments to explore any further potential research issue.

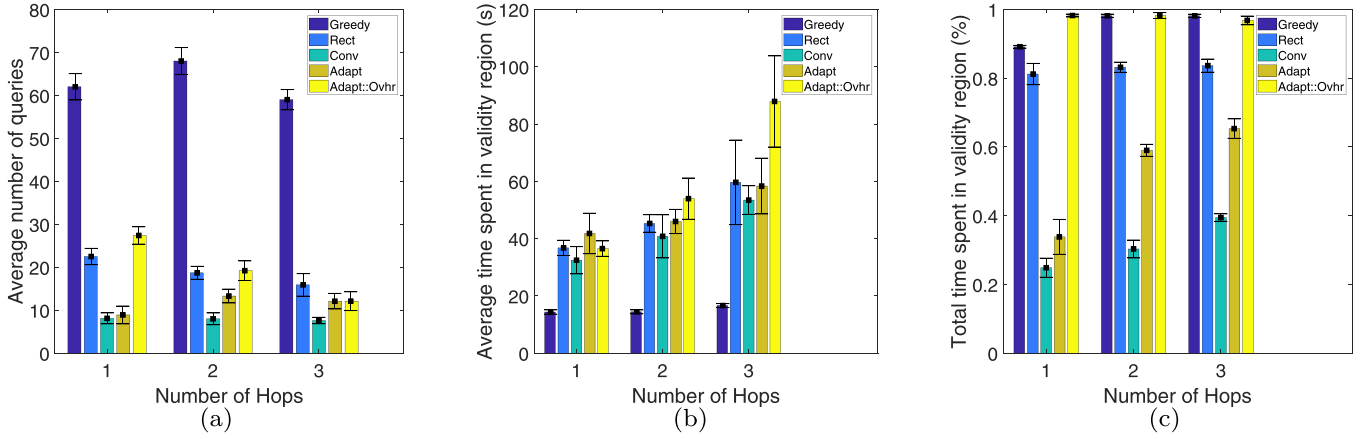


**Fig. 17.** The *Reduce* is applied to five difference strategies and their performance is measured by changing of number hops in query propagation.

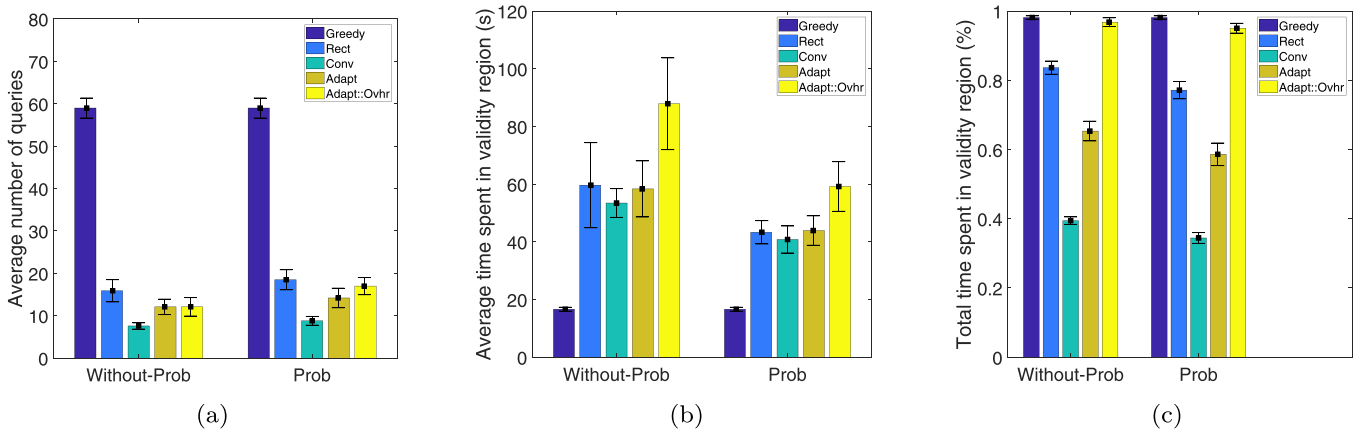


**Fig. 18.** The *Prob* is applied to five difference strategies and their performance is measured using the Eq. 2.

**Table 1**
The comparison¶ of region-based query processing strategies proposed under different environments.

| Scheme | POI Mobility | Update frequency | Computation | Formation | Network |
|---|---|---|---|---|---|
| Safe Exit Point [20] | Static | Once | Centralized | Sum of exit points | VANET |
| Extended Range Safe Region [15] | Mobile | Once | Centralized | Intersection of regions | MANET |
| MWPSR [19] | Static | Once | Centralized | Rectangle regions | MANET |
| Voronoi Diagram [26] | Static | Once | Centralized | Intersection of regions | VANET |
| Rect/Conv/Adapt/Adapt:Ovhr [22] | Static | Once | Distributed | Sum of virtual cells | MANET |
| Reduced – Rect/Conv/Adapt/Adapt:Ovhr | Static & Mobile | Once | Distributed | Sum of virtual cells | MANET |
| Prob – Rect/Conv/Adapt/Adapt:Ovhr | Static & Mobile | Multiple | Distributed | Sum of virtual cells | MANET |

¶ In this paper, we compare the proposed query processing strategies with prior approaches in terms of five aspects.
(i) *POI Mobility:*The mobility of POIs in the network, where *Mobile* implies that a part of POIs moves; (ii) *Update Frequency:*
How often does the query issuing node update its validity region before it moves out of the validity region. *Once* implies that the query issuing node initially builds a validity region and then never update it. *Multiple* implies that the query issuing node updates the validity region more than once, such as the query issuing node decides whether to reset a new validity region whenever it visits a new cell within the validity region in the Prob; (ii) *Computation:*
Who computes a valdiity region. *Centralized* implies that a centralized server or an individual node builds a validity region, but *Distributed* is when a set of nodes cooperatively builds a validity region; (iv) *Formation:*
What is a validity region consisted of. In [20], a validity region is a set of points that is a certain distance away from the query issuing node and is all located on the roads; (v) *Network:* A network type, where a validity region is deployed.

## 8. Concluding remarks

In this paper, we investigated validity region sensitive query processing strategies to efficiently reduce the number of queries and update the freshness of queried data in MANETs. We first proposed basic rectangle and convex hull based validity regions and their corresponding query processing operations for static POIs.Then we extended them by combining rectangle and convex hull techniques and considering an opportunistic overhearing. In addition, we approximated the validity region and proposed two techniques in forming validity region and their corresponding query processing operations for mobile POIs. We conducted extensive simulation based experiments and performance study and showed that the proposed query processing strategies can be a viable querying approach in MANETs.

## Acknowledgment

This research was supported in part by International Cooperative Research Grant from Incheon National University (Incheon, Korea).

## References

[1] R. Frenkiel, B. Badrinath, J. Borras, R. Yates, The infostation challenge: balancing cost and ubiquity in diverse wireless data, IEEE Pers. Commun. 7 (2) (2000) 66–71.
[2] H. Reumerman, M. Roggero, M. Ruffini, The application-Based clustering concept and requirements for intervehicle networks, IEEE Commun. Mag. (2005) 108–113.
[3] Y. Sasaki, T. Hara, S. Nishio, Two-phase top-k query processing in mobile Ad Hoc networks, Proc. Int'l Conf. on Network-based Information Systems, (2011), pp. 42–49.
[4] D. Amagata, Y. Sasaki, T. Hara, S. Nishio, A robust routing method for top-k queries in mobile Ad Hoc networks, Proc. Mobile Data Management, (2013), pp. 251–256.
[5] D. Amagata, T. Hara, Y. Sasaki, S. Nishio, Efficient cluster-based top-k query routing with data replication in MANETs, Soft. Comput. (2015) 1–18.
[6] Y. Xu, T.Y. Fu, W.C. Lee, J. Winter, Processing *k* nearest neighbor queries in location-aware sensor networks, Signal Process. 87 (12) (2007) 2861–2881.
[7] S.H. Wu, K.T. Chuang, C.M. Chen, M.S. Chen, Toward the optimal itinerary-based KNN query processing in mobile sensor networks, IEEE Trans. Knowl. Data Eng. 20 (12) (2008) 1655–1668.
[8] T.Y. Fu, W.C. Peng, W.C. Lee, Parallelizing itinerary-based KNN query processing in wireless sensor networks, IEEE Trans. on Knowl. Data Eng. 22 (5) (2010) 711–729.
[9] Y. Komai, Y. Sasaki, T. Hara, S. Nishio, *k*NN Query processing methods in mobile ad hoc networks, IEEE Trans. Mobile Comput. 13 (5) (2014) 1090–1103.
[10] Y. Han, K. Park, J. Hong, N. Ulamin, Y. Lee, Distance-Constraint k-Nearest neighbor searching in mobile sensor networks, Sensors 15 (8) (2015) 18209–18228.
[11] J. Zhang, M. Zhu, D. Papadias, Y. Tao, D.L. Lee, Location-based Spatial Queries, Proc. ACM SIGMOD, (2003), pp. 443–454.
[12] H. Cho, K. Ryu, T. Chung, An efficient algorithm for computing safe exit points of moving range queries in directed road networks, Inf. Syst. 41 (2014) 1–19.
[13] D. Yung, M.L. Yiu, E. Lo, A safe exit approach for efficient network-based moving range queries, Data Knowl. Eng. 72 (2012) 126–147.
[14] H. Hu, J. Xu, D.L. Lee, A generic framework for monitoring continuous spatial queries over moving objects, Proc. ACM SIGMOD, (2005), pp. 479–490.
[15] A.-K. Haidar, D. Taniar, J. Betts, S. Alamri, On finding safe regions for moving range queries, Math. Comput. Model. 58 (5) (2013) 1449–1458.
[16] S. Lim, C. Yu, C.R. Das, Randomcast: an energy efficient communication scheme for mobile ad hoc networks, IEEE Trans. on Mobile Comput. 8 (8) (2009) 1039–1051.
[17] R. Kravets, P. Krishnan, Power management techniques for mobile communication, Proc. ACM MOBICOM, (1998), pp. 157–168.
[18] F. Liu, K. Hua, F. Xie, On reducing communication cost for distributed moving query monitoring systems, Proc. Mobile Data Management, (2008), pp. 156–164.
[19] B. Bamba, L. Liu, A. Iyengar, S.Y. Philip, Distributed processing of spatial alarms: a safe region-based approach, Proc. IEEE ICDCS, (2009), pp. 207–214.
[20] H. Cho, S. Kwon, T. Chung, A safe exit algorithm for continuous nearest neighbor monitoring in road networks, Mobile Inf. Syst. 9 (2013) 37–53.
[21] OMNeT + +, http://www.omnetpp.org/.
[22] B. Jung, S. Lim, J. Chae, C. Pu, Validity region sensitive query processing strategies in mobile Ad Hoc networks, Proc. MILCOM, (2016), pp. 1022–1027.

[23] R.L. Graham, An efficient algorithm for determining the convex hull of a finite planar set, Inf. Process Lett. 1 (4) (1972) 132–133.
[24] S. Ni, Y. Tseng, Y. Chen, J. Sheu, The broadcast storm problem in a mobile Ad Hoc network, Proc. ACM MobiCom, (1999), pp. 151–162.
[25] D.B. Johnson, D.A. Maltz, Dynamic source routing in Ad Hoc wireless networks, Mobile Computing, (1996), pp. 153–181.
[26] B. Zheng, D.L. Lee, Semantic caching in location-dependent query processing, Proc. Int'l Symposium on Spatial and Temporal Databases, (2001), pp. 97–113.
[27] A. Akella, G. Judd, S. Seshan, P. Steenkiste, Self-management in chaotic wireless deployments, Proc. ACM MOBICOM, (2005), pp. 185–199.
[28] C. Pu, T. Gade, S. Lim, M. Min, W. Wang, Light-weight forwarding protocols in energy harvesting wireless sensor networks, Proc. MILCOM, (2014), pp. 1053–1059.