Wireless Pers Commun (2013) 68:153–173
DOI 10.1007/s11277-011-0444-y

Continuous Location Dependent Queries in Mobile Wireless Sensor Networks

Liang Hong · Gang Zhou · Bo Liu · Sang Son

Published online: 15 November 2011

Abstract Query processing in mobile *Wireless Sensor Networks* (WSNs) is still a challenging problem because sensor mobility causes frequent changes of network topology. In this paper, we study the problem of processing *Continuous Location Dependent Query* (CLDQ) that retrieves the sampling data of the sensors within a specific area (i.e. query area) around a mobile sensor. Existing query processing approaches can not efficiently process CLDQs with continuously moving query areas. We propose scalable techniques to process CLDQs efficiently and accurately, including a dissemination approach, a *Contention-based Distance-aware Message Scheduling* scheme, in which each stationary sensor's data transmissions are smartly scheduled according to its distance to the mobile sensor, and an optimization scheme for continuous processing of CLDQs. Extensive experiments indicate that our techniques demonstrate better efficiency of processing CLDQs over state-of-the-art techniques while achieving high accuracy and short query latency under various network settings.

Keywords Location dependent query · Mobile wireless sensor network · Query processing · Data aggregation · Dissemination

L. Hong (✉)
School of Computer, Wuhan University, Wuhan 430072, Hubei, China
e-mail: hong@whu.edu.cn

G. Zhou
Department of Computer Science, College of William & Mary, Williamsburg, VA, USA
e-mail: gzhou@cs.wm.edu

B. Liu
College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, Hubei, China
e-mail: bo.liu@hust.edu.cn

S. Son
Department of Computer Science, University of Virginia, Charlottesville, VA, USA
e-mail: son@cs.virginia.edu

1 Introduction

As wireless sensor networks and mobile devices have been widely deployed, it is possible for mobile users to query information from physical world anywhere and at anytime. Specifically, *Continuous Location-Dependent Queries* (CLDQ) have attracted a lot of interest [9]. CLDQ is an important and useful query type which cannot be efficiently answered using current query processing approaches. In mobile WSNs, CLDQ retrieves the aggregation results of the data sampled from the stationary sensors within a specific physical area (i.e. query area, usually a circle) around the mobile sensor (defined as *reference sensor*). The answer to a CLDQ depends on the location of the reference sensor involved, and should be continuously updated due to the reference sensor's movements. Figure 1 shows an example CLDQ over metro transportation in Wuhan, China: "Reporting the nearest available bus within 1 mile around John every 1 minutes". Stationary sensors are deployed in the city to sense vehicles, and each stationary sensor has a unique identifier and is aware of its own location through positioning devices like GPS or localization techniques like [18]. Users can issue CLDQs anytime and anywhere in the city. The city is divided into several sub areas; each sub area has a base station that is responsible for injecting users' queries to the sensor network and sending the query results back to the user. Each mobile sensor registers to a stationary sensor as a proxy to access the sensor network. The data sampled from the stationary sensors in current query area are aggregated in the network, and the query results are sent back to the base station. We leave the issues on communication among base stations as future work and discuss the situation in one sub area.

The SQL-like syntax of CLDQ is as follows:
SELECT [aggregation operators] $S_1.att_1 \ldots S_1.att_i$
FROM $Sensor_1$ AS S_1
INSIDE (*Radius*, (SELECT $S_2.loc$ FROM $Sensor_2$ AS S_2))
REPORT PERIOD R_p
FOR *Duration*

CLDQ retrieves aggregation data of the stationary sensors ($Sensor_1$) within (INSIDE) the circle query area around the reference sensor ($Sensor_2$). The query results should be reported to the user every report period until the query duration expires. The aggregation operators

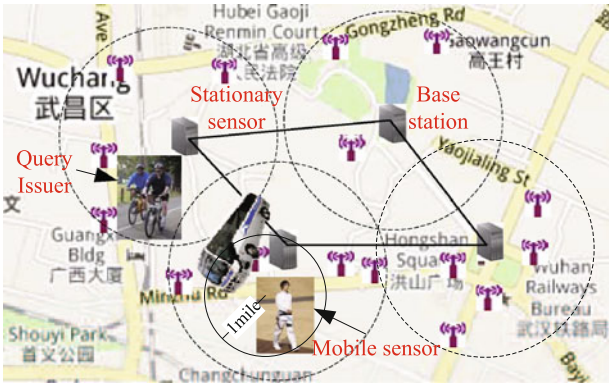

Fig. 1 Query metro transportation in Wuhan

include SUM, AVG, MAX, MIN and COUNT; att_i is the data type that the stationary sensor can sample; most of the location constraints can be expressed in terms of INSIDE [8], so only INSIDE is considered in the query.

Query processing in traditional database assumes data are available in a centralized database and focus on improving the time and I/O performance [4,8]. However, a centralized database is infeasible for a sensor network that is characterized by distributed topology and constrained resources. Answering continuous spatial queries in stationary sensor networks has been extensively studied [16,17,21,22]. These techniques typically collect the data from the sensor nodes in a fixed area without considering sensor mobility. However, sensor (especially reference sensor) mobility brings new challenges to query processing in WSNs. First, it is challenging to propagate the query and collect sensed data in a continuously changing query area. Second, it is hard to efficiently track and access mobile sensors in mobile WSNs. Finally, processing continuous query in WSNs may overwhelm the system load, hence specialized optimization techniques need to be invented to reduce overall energy consumption.

In this paper, we propose an efficient CLDQ processing approach in mobile WSNs. In our approach, the base station disseminates user's query to each mobile reference sensor along a *dissemination path* built by stationary sensor nodes. After receiving the query, the reference sensor broadcasts the query to its neighboring nodes periodically until the query duration expires. The neighboring nodes then propagate the query to all the stationary sensor nodes in the query area. Afterward, each stationary sensor aggregates the sensed data and the data received from neighbors, then sends the aggregation result to the next hop stationary sensor. In this way, stationary sensors eventually relay the query result to the base station. To summarize, we make the following contributions in this paper:

- To the best of our knowledge, we propose the first processing approach for CLDQs in mobile WSNs. Our processing approach can efficiently answer CLDQs in a fully distributed and self-organized way.
- We present an efficient approach for query dissemination from base station to mobile sensor nodes.
- We develop a *Contention-based Distance-aware Message Scheduling* (CDMS) scheme which smartly schedules each stationary sensor's transmissions during query propagation and data aggregation. CDMS can reduce the overall energy consumption while maintaining low query latency and high accuracy of query results.
- We propose an optimization scheme for continuous processing of CLDQs based on location prediction of the mobile reference sensor and *time series data forecasting* [2]. Our optimization scheme is more efficient than traditional continuous query processing approaches which simply process continuous query as a series of snap-shot queries.
- We give an extensive evaluation of the performance of processing CLDQ. Our approach exhibits a superior performance in terms of energy efficiency, query latency and query accuracy under various network conditions and outperform all comparing techniques.

The rest of the paper is organized as follows. Section 2 briefly reviews related work. Section 3 presents techniques for efficiently processing snap-shot CLDQ. In Sect. 4, we propose an optimization scheme for continuous processing of CLDQ. In sect. 5, we compare the performance of our query processing approach with the stat-of-the-art approaches. Section 6 concludes the paper and discusses the future work.

2 Related Work

Current solutions on query processing in WSNs typically collect and aggregate data from the stationary sensors residing in a fixed geographical region (e.g a rectangular or a circle). The query processing approach based on TinyDB [16] focuses on simple range queries over stationary sensors in which sensor data are aggregated in the network and sent back through the routing tree to the base station. Yang et al. propose a Two-Phase Self-Join (TPSJ) scheme [22] to efficiently process self-join query in WSNs. Liu et al. propose an energy-efficient data collection method [14] by exploring the spatial and temporal correlation of sensing data to save energy. The method dynamically partitions the sensor nodes into clusters so that the sensors in the same cluster have certain spatial and temporal correlation. A recent data collection method develops a novel TDMA schedule that is capable of collecting sensor data for any network traffic pattern [27]. However, sensor mobility makes the above solutions invalid or inefficient for processing CLDQs in mobile WSNs.

Some studies on query processing in WSNs have taken sensor mobility into account. CNFS [7] is a walk-based algorithm for retrieving data from a source node in mobile WSNs. However, the query type considered in CNFS is simpler than that of our proposal. CountTorrent [11] is a distributed mechanism for answering aggregation queries in mobile sensor networks. In CountTorrent, sensor nodes swap information with neighboring nodes as a torrent process, which introduces additional in-network transmission overhead during query processing. ICEDB [26] focuses on distributed query processing on each sensor node in mobile WSNs. However, it lacks the optimization for in-network message transmission. DRACA [10] is a replica arrangement scheme for location dependent information, which adaptively arranges replicas at positions close to nodes for frequent sending of queries. However, DRACA causes additional storage overhead for sensor nodes and prolongs query processing time.

Location dependent query processing is an important research issue in WSNs. In [15], a spatialtemporal query service named MobiQuery is proposed for mobile users to periodically gather information from their surrounding areas. Our work is inspired by the approach for processing event-based location dependent query [5]. However, these two approaches are susceptible to fast topology changes caused by sensor mobility and spend excessive cost on maintaining a sensor tree. Wu et al. present IDDA [20] for WSNs in which mobile sink collects data from the sensor nodes around it by exploiting its antenna directivity. However, IDDA only considers one-hop transmission without in-network optimization during query processing. In [6], a tree-zone-based topology management and routing scheme is proposed for energy-efficient data query processing in a large-scale sensor network. However, this scheme assumes mobile medical phones have much higher energy and larger transmission range than sensor nodes, which is not common in WSNs.

A number of approaches have been proposed to perform time series forecasting to approximate the values of sensors. Borgne et al. [1] present an adaptive scheme that allows sensor nodes to adaptively selecting the time series prediction model based on performance assessments. PAQ [19] relies on autoregressive models built at each sensor to predict local readings. PAQ technique is employed in this paper for predicting local sensor reading in query processing, as will be discussed in Sect. 4.

The novelty of our approach lies in four aspects. First, our approach is among the first that process CLDQ in WSNs in a fully distributed way. Second, our approach is purely data driven and does not make any assumption on the sensor node's transmission range. Third, our approach dynamically schedules multi-hop in-network data transmission according to mobile sensor's location, which aims to achieve a good balance among accuracy, query delay,

and energy consumption. Last, our approach exploits temporal correlation in sensory data to further optimize CLDQ processing cost.

3 Snap-Shot Query Processing

In this section, we discuss how to efficiently process the snap-shot CLDQ during each report period, which is a basis for continuous processing of CLDQ. We first present an efficient algorithm for query dissemination from the base station to mobile sensors, then propose CDMS scheme for in-network query propagation and data aggregation.

3.1 Query Dissemination

To reduce the cost of tracking and accessing mobile sensor, each mobile sensor registers to a neighboring stationary sensor that has maximum *distance projection* [5]. Such neighboring stationary sensor is called *proxy*. Distance projection is the distance between a mobile sensor and a stationary sensor projected on the mobile sensor's velocity vector. The reason for choosing such a neighboring node is to reduce the frequency that mobile sensor changes its proxy. The tie is solved by registering to the stationary sensor that is closest to base station to minimize the hops of dissemination from base station. Mobile sensor sends periodic handshake messages to its proxy. If mobile sensor moves out of the proxy's transmission range and can not receive handshake messages from proxy, it starts to broadcast *choosing proxy* message to nearby stationary sensors. Each of the stationary sensors calculates distance projection and sends it to proxy upon receiving the choosing proxy message. The proxy sends a *register* message to the stationary sensor that has maximum distance projection.

After receiving the user's query, base station disseminates the query to each mobile sensor using a dissemination path built by stationary sensor nodes. The formation of dissemination path is as the following: a mobile sensor's proxy sends a routing request from itself to the base station using the routing protocol GPSR [12], a greedy routing protocol for end-to-end routing in WSNs. Using GPSR, a stationary sensor sends the request to the neighboring stationary sensor that is closest to the base station. The receiving sensor caches the sender's address as the next hop sensor to the proxy. In this way, a dissemination path can be established from the base station to the mobile sensor's proxy.

A dissemination path should be updated as the mobile sensor changes its proxy. The update method in [13] uses a threshold on newly added path to decide the timing of update, which can not optimize the overall dissemination cost. We propose a hop-based update algorithm by considering the tradeoff between update cost and dissemination cost. This algorithm adaptively chooses one of the two update methods: conservative update (CU) and aggressive update (AU) based on a threshold of the dissemination path's hops. Using CU, the new proxy finds a route to the old proxy; then the route to old proxy together with the old dissemination path form a new dissemination path. Using AU, the new proxy initiates a routing request to the base station to establish a new dissemination path, and then deletes the old dissemination path. Since CU does not reconstruct the dissemination path, it usually causes less update cost but longer dissemination path than AU.

In hop-based update algorithm, when a mobile sensor needs to change its proxy, it checks the hops of current dissemination path. If the number of hops is larger than the hop threshold, it requires the new proxy to use AU method. Otherwise, it requires the new proxy to use CU method. The hop threshold is determined by the query arrival rate and QoS requirements.

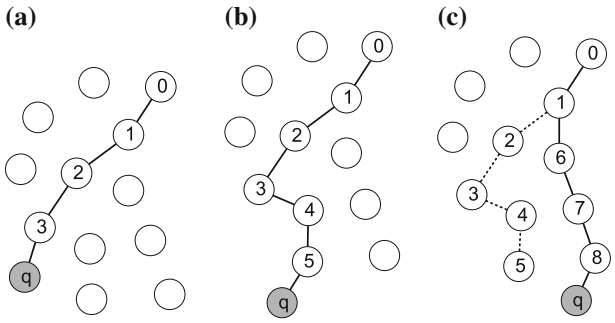

Fig. 2 Update dissemination path

An example is depicted in Fig. 2, there is a 4-hop dissemination path from base station 0 to the mobile sensor q (Fig. 2a). Stationary sensor 3 is q's proxy. The hop threshold is set to 5. After q changes its proxy to stationary sensor 5 (Fig. 2b), it requires sensor 5 to use CU method. However, after q changes its proxy to stationary sensor 8 (Fig. 2c), it requires sensor 8 to use AU method. The reason is that q finds the hops of the new dissemination path exceed the hop threshold. Sensor 8 then initiates a route request to base station. When the route request arrives at stationary sensor 1, sensor 1 directs a message from stationary sensor 2 to stationary sensor 5 to delete the obsolete dissemination path.

3.2 Basic Idea of CDMS Scheme

We propose a *Contention-based Distance-aware Message Scheduling* (CDMS) scheme that smartly schedules multi-hop in-network message transmissions. CDMS is based on the basic contention-based scheme [12] in which each stationary sensor receiving the query determines its transmission precedence independently without assistance from the mobile sensor or knowledge about its neighboring stationary sensors. Each stationary sensor sets a timer that can be calculated by: $timer = max\_delay \times \frac{\alpha}{\pi}$ where α is the angle formed by the *reference line* and the line connecting the mobile sensor and current stationary sensor, and *max_ delay* is the maximum time that the mobile node is allowed to complete the propagation. The reference line is a horizontal line emanating from the mobile sensor. An individual stationary sensor broadcasts the query to neighboring stationary sensors after its timer expires. Moreover, to prevent two stationary sensors from having the same timer, a small jitter is added to each timer.

However, the basic contention-based scheme does not consider optimizing the cost of query propagation and data collection. Using such scheme, query is flooded to every stationary sensors that reside in the query area. Different from the basic contention-based scheme, CDMS takes the distance between the stationary sensor and the mobile sensor into account: an individual stationary sensor node's timer should be determined by its hops from the mobile reference sensor.

In query propagation, the query is propagated from the stationary sensors close to the mobile sensor (*inner nodes*) to the stationary sensors far from the mobile sensor (*outer nodes*) in the query area hop by hop. In this way, we avoid *shuttle propagation* in which the query is propagated from inner nodes to outer nodes then back to inner nodes. Shuttle propagation causes unnecessary transmission overhead and extra delay.

Observe that the more data aggregated in a stationary sensor, the fewer messages transmitted. Moreover, assume sensor nodes are uniformly distributed, the larger the hop from the mobile sensor, the more the stationary sensor nodes in this hop. Therefore, to optimize the in-network aggregation cost, CDMS schedules the data flow of aggregation in a reverse direction to that of query propagation, i.e. data are collected and aggregated hop by hop from outer nodes to inner nodes and are aggregated in the network. Ideally, an individual stationary sensor aggregates the data flows through it and sends the compact aggregation result to the next hop at most once.

3.3 Query Propagation

If query radius is shorter than the mobile sensor's transmission radius, mobile sensor broadcasts the query to all the sensors in the query area by one message. Otherwise, the query needs to be propagated from the sensors in the mobile sensor's transmission range to other sensors in the query area.

Assume *prop_ delay* is the maximum time that completing query propagation in one hop is allowed. The propagation timer of the stationary sensor that is one hop from the mobile senor can be calculated as: $timer_{prop} = prop\_delay \times \frac{\alpha}{\pi}$, where α is the angle formed by the reference line and the line connecting the mobile sensor and current stationary sensor. After receiving the query, the propagation timer of the stationary sensor that is two hops from the mobile sensor can be calculated as: $timer_{prop} = prop\_delay \times \left(\frac{\alpha}{\pi} + 1\right)$. To summarize, the propagation timer of the stationary sensor that is n hops from the mobile sensor is as follows:

$$timer_{prop} = prop\_delay \times \left(\frac{\alpha}{\pi} + n - 1\right), \quad n \in N \tag{1}$$

where α is the angel formed by the reference line and the line connecting the mobile sensor and current stationary sensor. An individual stationary sensor broadcasts the query to its neighboring nodes after its timer expires. The stationary sensors that are n hops from the mobile sensor start to propagate the query in an anticlockwise order after the *n-1* hop stationary sensors finishing the propagation. If one stationary sensor finds its transmission range is covered by a neighboring stationary sensor that has already propagated, it will not transmit the query again.

As shown in Fig. 3a, mobile sensor q first broadcasts the query to the immediate neighboring stationary sensor 1, 2, 3, and 4. Then the query is propagated hop by hop from these sensor nodes to all the other nodes within the query area. The propagation order is marked with dotted line, which is determined by the angle formed by the reference line and the line connecting q and the node.

As discussed in Sect. 3.2, the number of outer nodes is larger than that of inner nodes, so outer nodes divide the *prop_ delay* into smaller time pieces, and possibility of two nodes having the same timer becomes higher. To avoid transmission collisions, a jitter is added to each timer which is set to inversely proportional to the number of this sensor's neighbors, because the stationary sensor with more neighboring sensors can broadcast the query to more sensors by one message.

Each stationary sensor maintains a neighbor list containing all its neighboring sensors' locations. If a stationary sensor finds all its neighboring nodes in the query area have already received the query during query propagation, it will stop propagating the query. Such stationary sensor is called *boundary node*.

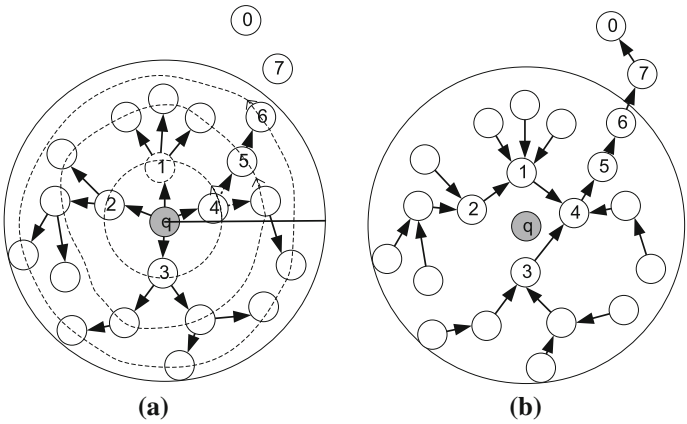

Fig. 3 In-network query processing. **a** Query propagation. **b** Data aggregation

3.4 In-Network Aggregation

Once a stationary sensor receives the query, besides setting propagation timer, it also sets the aggregation timer. A stationary sensor's aggregation timer is divided into two parts: the time for completing the propagation and the time for waiting for the data from neighboring sensor nodes. Aggregation timer is closely related to the distance between the stationary sensor and the reference sensor, i.e. the hops from the reference sensor. The maximum possible hops of the sensor nodes in the query area can be calculated as $\left\lceil \frac{R}{R_{tr}} \right\rceil$, where R is the query radius and R_{tr} is the sensor node's transmission radius. Therefore, the number of hops from the stationary sensor to the boundary of the query area can be calculated as $\left\lceil \frac{R}{R_{tr}} \right\rceil - n$, where n denotes the stationary sensor that is n hops from the reference sensor, which can be obtained during query propagation. Given the maximum time for completing propagation in one hop *prop_ delay*, the time for completing the propagation after a stationary sensor receives the query can be calculated as:

$$T_{prop} = prop\_delay \times \left(\frac{2\pi - \alpha}{2\pi} + \left\lceil \frac{R}{R_{tr}} \right\rceil - n \right) \tag{2}$$

where $prop\_delay \times \left(\frac{2\pi - \alpha}{2\pi} \right)$ is the remaining time for completing all the nodes' propagation in current hop. And the time for waiting for the data from neighboring stationary sensors can be calculated as:

$$T_{agg} = agg\_delay \times \left(\frac{\alpha}{2\pi} + \left\lceil \frac{R}{R_{tr}} \right\rceil - n \right) \tag{3}$$

where *agg_ delay* is the maximum time allowed for completing the data aggregation and transmission in one hop. From (2) and (3), the aggregation timer is as follows:

$$\begin{aligned} timer_{agg} = {} & prop\_delay \times \left(\frac{2\pi - \alpha}{2\pi} + \left\lceil \frac{R}{R_{tr}} \right\rceil - n \right) \\ & + agg\_delay \times \left(\frac{\alpha}{2\pi} + \left\lceil \frac{R}{R_{tr}} \right\rceil - n \right) \end{aligned} \tag{4}$$

If a stationary sensor is boundary node, it sets its aggregation timer to $agg\_delay \times \frac{\alpha}{2\pi}$. Otherwise, it sets its aggregation timer according to (4) and sends the aggregation results to the query sender until the aggregation timer expires. Eventually, the reference sensor's immediate neighboring nodes get the partial aggregation results and send the results to the reference sensor's proxy, where final aggregation result is computed. In case some of these stationary sensors can not transmit their aggregation results directly to proxy, they route the data packets to the proxy using GPSR. After aggregating all the data, the proxy sends the query result back to base station using GPSR.

As is shown in Fig. 3b, the data flow of in-network aggregation is reverse to that of query propagation. The data are eventually aggregated in stationary sensor 1, 2, 3 and the proxy 4. Sensor 1 and 3 send their aggregation results directly to proxy 4 while sensor 2 send its aggregation result to proxy 4 through sensor 1. Proxy 4 aggregates its own data and the data from sensor 1, 2 and 3, then sends the aggregation result to stationary sensor 5. Sensor 5 aggregates its own data and the received data, then sends the aggregation result to the boundary node 6. Finally, sensor 6 aggregates its own data and the received data, then sends the query result to base station 0 through stationary sensor 7.

The above algorithm is based on the assumption that query radius is longer than the mobile sensor's transmission radius. Otherwise, each stationary sensor can receive the query directly from the mobile sensor and sends its data to the proxy sensor where the final aggregation result is computed.

3.5 Fairness of CDMS

In traditional message scheduling approaches [14,27], a sensor close to the base station tends to achieve much higher throughput than a sensor far away. To ensure fair access to network bandwidth, all the stationary sensors should send same number of packets to the base station, irrespective of their distance to the base station.

In CDMS, each stationary sensor sets propagation timer and aggregation timer according to their distances to the proxy nodes. Each sensor in the query area only sends one data packet in each report period, since sensor forwards aggregation result to the next hop sensor after receiving the data from distant sensors. Such scheduling guarantees network-layer fairness. MAC-layer fairness in CDMS is achieved by IEEE 802.11 protocol. As a result, the message scheduling in CDMS is fair in both the network and MAC layers.

4 Continuous Processing of CLDQs

In this section, we discuss how to continuously process CLDQs based on the snap-shot query processing approach discussed in Sect. 3. We propose an optimization scheme to reduce the continuous processing cost of CLDQs. This scheme is based on location prediction of the reference sensor and time series data forecasting. The basic idea of our scheme is predicting the location of the reference sensor in the next report period, then reusing the message transmissions of the sensors in the overlapping area of current query area and predictive query area in case the location prediction is correct. Such reuse reduces the overhead of query propagation and in-network data aggregation. Otherwise, if there is no overlapping area, a CLDQ degenerates into a series of separate snap-shot queries.

4.1 Overview of the Optimization Scheme

In current report period, each reference sensor broadcasts not only its current position but also predictive position in the next report period. The predictive position is calculated based on the reference sensor's current location and speed vector. When a stationary sensor receives reference sensor's predictive position, the stationary sensor decides whether it is in the overlapping area by computing the distance between the reference sensor and itself. In the next report period, if a stationary sensor finds all its neighboring stationary sensors are in the overlapping area during query propagation, it will not propagate the query because its neighboring sensors have already received the query in current report period.

The stationary sensor in the overlapping area determines the next hop sensor to which it transmits the data in the next report period by scanning its neighbor list. In addition, the stationary sensor can predict the sampling data in the next report period using time series forecasting. Therefore, by utilizing the broadcast nature of wireless communication, the stationary sensor can send the current data and predictive data to the next hop stationary sensors in current and next report period respectively by one message. In the next report period, if the data prediction and the reference sensor's location prediction are both correct (within error bound), stationary sensors in the overlapping area need not to transmit their data again in the next report period. Otherwise, if some of the stationary sensors' readings in the next report period are outliers, these sensors should send the outliers to the next hop sensors. Note that even if some stationary sensors do not need to transmit their data, they should still transmit the aggregation result if they are not boundary nodes.

As shown in Fig. 4, q is reference sensor's current position and q_p is the predictive position in the next report period. The real circle is the query area and the dashed circle is the reference sensor's transmission range. Suppose the location prediction of the mobile sensor is correct. In the query propagation of the next report period, stationary sensor 8 finds all its neighboring stationary sensors are in the overlapping area and have already received the query, so it needs not to propagate the query again. In current query area, stationary sensors 1, 2, 3 are one hop away from q, and stationary sensors 4, 5, 6, 7, 8 are two hops away from q. Using CDMS, sensor 4 and 7 send current readings to sensor 1, sensor 8 sends reading to sensor 2, and sensor 5 and 6 send readings to sensor 3. In predictive query area, sensor 7 and 8 are one hop away from q_p, and sensor 1, 2, 3, 4, 5, 6 are two hops away from q_p. As a result, sensor 1, 4 should send readings to sensor 7, and sensor 2, 3, 5, 6 should send the readings to sensor 8 in the next report period. Using continuous processing optimization scheme, sensor 7 and 8 send current readings to sensor 1 and 2 respectively while sending their predictive data to the next hop sensors in the next report period (probably the reference sensor's proxy). Sensor 4 sends current reading to sensor 1 and predictive data to sensor 7. Similarly, sensor 5, 6 send current readings to sensor 3 and predictive data to sensor 8. Ideally, 1 propagation

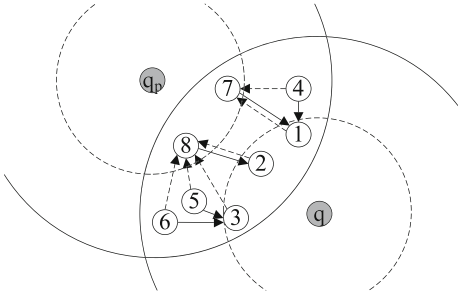

Fig. 4 Example of continuous processing optimization scheme

message and 6 aggregation messages can be saved using the above optimization scheme. Since continuous query usually has a large number of report periods, the total cost saving can be considerably high.

4.2 Correctness of Location Prediction

Before each report period, reference sensor should check its current location to determine whether location prediction of the last report period is correct. Each reference sensor calculates the difference between current location and predicative location of the last report period. In case this difference is larger than some threshold, say, ΔD, the prediction fails, otherwise the prediction succeeds. This technique is known as *dead reckoning* [3]. We use dead reckoning technique to verify the correctness of location prediction in our scheme.

4.3 Time Series Forecasting

Using time series forecasting, a stationary sensor can predict the sampling data in the next report period based on the historical sampling data. Our time series forecasting technique is based on local *autoregressive* (AR) model [19]. The predictor $P(t)$ of a time series $F(t)$ at time t is given by its mean η plus a linear combination of the increments or decrements of the last three readings with respect to η. The prediction at time $t > t_{i-1}$ is given as:

$$P(t) = \eta + \alpha(v_{i-1} - \eta) + \beta(v_{i-2} - \eta) + \gamma(v_{i-3} - \eta) \tag{5}$$

where v_{i-1}, v_{i-2}, v_{i-3} are the last three readings. To learn the local AR model, each sensor node caches three historical readings and computes the mean η and coefficient α, β, and γ. The coefficients of the best linear predictor are obtained by minimizing the following function:

$$Q(\alpha, \beta, \gamma) = \sum_{i=4}^{N} \left(\overline{v}_i - (\alpha \overline{v}_{i-1} + \beta \overline{v}_{i-2} + \gamma \overline{v}_{i-3}) \right)^2 \tag{6}$$

The coefficients α, β, γ can be computed by setting the partial derivatives of the minimum squared error to zero and solving a linear system of three equations.

If a data prediction exceeds the error bound, the stationary sensor marks current reading as an outlier. The stationary sensor keeps monitoring the local AR model, and decides whether to choose to re-learn the model based on the deviation of current reading and the number of recent outliers.

4.4 Prediction Invalidation

If data prediction fails, the stationary sensor simply transmits the readings in current report period to the next hop sensor. If location prediction fails, the query area in the next report period deviates from the predictive query area. Prediction invalidation scheme only need to invalidate the data predictions of the stationary sensors in the overlapping areas of the current query area and the predictive query area. Stationary sensors in the overlapping area that have least hops away from the reference sensor's predictive position (e.g. sensor 7 and 8 in Fig. 4) are responsible for monitoring the queries from the reference sensor. Assume the number of hops of these sensors is h, if they have not received the query within $h \times prop\_delay$ time period from the beginning of the report period, or the reference sensor is not in the predictive

position (within error bound), these stationary sensors send messages to their neighboring stationary sensors in the overlapping area (e.g. sensor 1, 2, 3, 4, 5, 6 in Fig. 4) to invalidate the data predictions. In this way, all stationary sensors in the overlapping area eventually get invalidation messages and will not send invalid data.

5 Simulation Results

We have implemented our approach and competing approaches [13,16,17] by simulation in GloMoSim [25] where MAC layer protocol is set to IEEE 802.11. In this section, we first illustrate the metrics and parameter settings used in simulation. Then we evaluate the impact of different system parameters on the performance of processing CLDQ.

5.1 Parameter Settings and Metrics

In our simulation, 200 stationary sensor nodes and 3 mobile sensor nodes are uniformly distributed in a 1,000 m × 1, 000 m square area. The mobility of sensor nodes is modeled by *Random Way Point* (RWP) model [24]: the node travels to the destination at a speed randomly chosen from a configured range [0, V_{max}]; upon arrival, it pauses for a random period and selects a new destination. The pause time is set to 30 s in our simulation. Stationary sensor, mobile sensor and base station's transmission radii are set to 100 m. The $prop\_delay$ and agg_delay in CDMS are set to 0.05 s by default. The workload CLDQ can be defined by users according to their requirements. Without loss of generality, we set the workload CLDQ's report period and duration to 60 and 600 s respectively, and the aggregation function to AVG. We ran all simulations on a 1.8 GHz Core Duo processor machine with 2 GB RAM. Each data point in the figures has a 90% confidence interval which comes from the average result of 10 runs. Table 1 summarizes the configuration parameters for the simulations.

We employ the following metrics in our simulations:

Query Accuracy: We define query accuracy as the ratio of the received query result to the exact query result.

Query Latency: The average elapsed time from the time a query being broadcasted by reference sensor to the time the base station receiving the query result.

Energy Consumption: In WSNs, energy consumption is dominated by message transmissions. Therefore, we evaluate the energy consumption by counting the average number of in-network messages.

Query Dissemination Cost: Query dissemination cost is the cost of disseminating queries to all the mobile sensors, which consists of three parts: building dissemination path, disseminating messages and updating the dissemination path.

Table 1 Simulation parameters

Num. of stationary sensors	200
Num. of mobile sensors	3
Mobility model	Random way point
Transmission radius	100 m
MAC protocol	802.11
Simulation area	1,000 m × 1,000 m
$prop\_delay$	0.05 s
agg_delay	0.05 s

Query Processing Cost: The query processing cost is the sum of the propagation cost and aggregation cost of all the stationary sensors involved in the query each report period.

Benefit Ratio: The benefit ratio is calculated by dividing the number of saving messages by the total number of messages of both dissemination and continuous query processing.

5.2 Comparing Approaches

We divide our query processing approach into two types: the approach without continuous processing optimization (denoted by CLDQ-S) and the one with such optimization (denoted by CLDQ-C). We compare CLDQ-S, CLDQ-C with TinyDB query processing approach [16,17] and tree-zone-based communication protocol BEE [6]. In addition, we compare our query dissemination approach with SEAD [13] and BEE [6].

TinyDB approach is a centralized query processing mechanism for stationary WSNs. In TinyDB approach, stationary sensors are organized into a *Semantic Routing Tree* (SRT) to allow each node to determine if any of the nodes below it will need to participate in a given query. Base station floods the query to all the mobile sensors. Sensor data are aggregated in the network and sent back through SRT to the base station. As all the comparing techniques require a beacon mechanism for sensor nodes to maintain information about their neighbor nodes, we do not measure the cost of beacon messages, which is usually of lower order than the application data traffic [21]. As a result, the cost of TinyDB approach is composed of flooding cost, SRT maintenance cost and data aggregation cost.

In BEE, query is first broadcasted to all the corresponding mobile *Medical Phones* (MPs) that have much higher power than the sensors. Each MP acts the center of a *Domain* which covers a limited area of a cell and contains several zones. Stationary sensors self-organize themselves into different zones. In each zone, the stationary sensors cooperate to form a *Minimum Spanning Tree* (MST). Each sensor is assigned a probability p to become a tree root of the zone. Each root collects data from stationary sensors in its zone and transmit them to the MP either directly or through multihop routing.

Data dissemination in mobile WSNs has been studied extensively. However, the flooding approach in [23] is proved to be not efficient compared to SEAD. Moreover, IDDA [20] focuses on interest dissemination from the mobile sink to neighbor stationary sensors, which is an opposite dissemination direction from our approach. Therefore, we use SEAD and BEE as the comparing approaches. In SEAD, the mobile sink selects its nearest neighbor as its *access node* (i.e. proxy) to the sensor network. A source node disseminates the sensor reading through a sensor tree. A mobile sink decides whether or not to change the access node based on the hop count between the mobile sink and its access node. In BEE, each root joins the domain of the nearest MP which establishes link to the nearby roots by sending periodic advertisements. MP forwards queries to the all the roots in its domain. Each root disseminates the queries to the stationary sensors in the zone.

5.3 Impact of Node Mobility

In this section, we evaluate the impact of reference sensor's mobility on the performance of query dissemination and query processing by varying V_{max} from 1 to 25 m/s. The query radius is set to 200 m.

Query accuracy is evaluated in Fig. 5a by varying maximum speed of mobile sensors. We observe that CLDQ-S and CLDQ-C maintains higher and more stable query accuracy than TinyDB and BEE. The first reason is that CLDQ involves less sensor nodes than TinyDB approach and BEE. CLDQ only involves stationary sensors in the query areas while TinyDB involves all the stationary sensors in the system and BEE involves the stationary sensors in nearby zones. The second reason is that BEE and TinyDB uses a fixed routing tree to propagate queries and relay data, which does not optimize the scheduling of in-network transmissions and therefore causes more message collisions. In contrast, as discussed in Sect. 3.2, CDMS avoids unnecessary message collisions by employing the contention-based

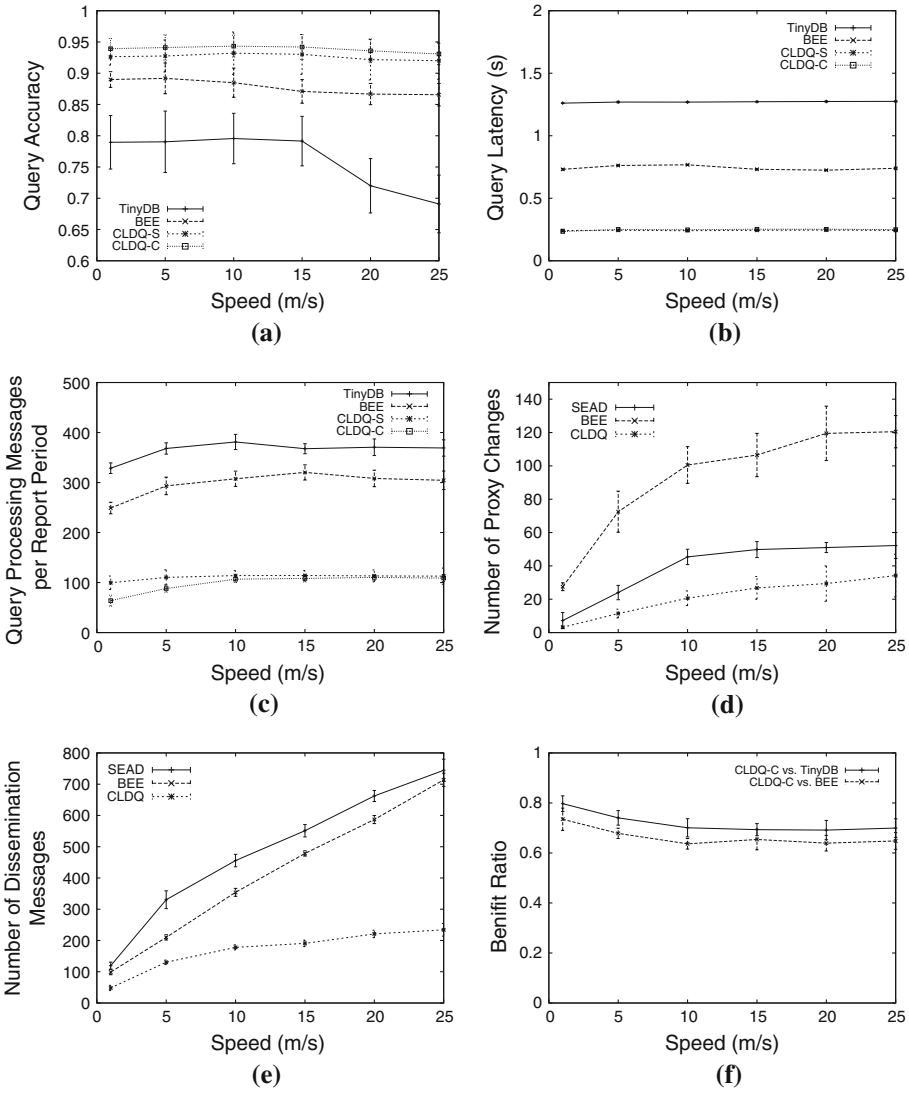

Fig. 5 Impact of node mobility. **a** Query accuracy. **b** Query latency. **c** Query processing messages. **d** Proxy changes. **e** Dissemination messages. **f** Benefit ratio

scheme. In CLDQ-C, continuous processing optimization technique reduces total number of transmissions, which decreases the probability of messages collisions and the impact of node failures. As a result, CLDQ-C has higher query accuracy than CLDQ-S.

As shown in Fig. 5b, CLDQ-S and CLDQ-C have much shorter query latency than TinyDB and BEE. This is because TinyDB and BEE rely on fixed routing trees to aggregate and transmit the data in the network which results in more transmission overhead. In contrast, CLDQ accounts for the propagation and aggregation timer set according to each stationary sensor's hops from reference sensor. This results in a significant reduction in query latency. Note that CLDQ-S and CLDQ-C have almost the same query latency. This is reasonable because query latency is determined by the largest hops from reference sensor within the query area and these two approaches have equal size query areas.

Figure 5c shows the average number of query processing messages during each report period. CLDQ-S and CLDQ-C significantly outperforms TinyDB approach and BEE. This result is attributed to the fact that unnecessary transmissions can be reduced by scheduling of CDMS. In addition, the static routing trees in TinyDB approach and BEE are not always the optimal routing path in continuously changing query area. A common trend may be observed: processing messages of CLDQ-S and CLDQ-C increase in the initial stage and then come to a stable state as maximum speed increase. This is because transmission collisions decrease rapidly as the overlapping query areas become smaller, while the impact of collisions on total processing messages is weakened as maximum speed of reference sensor further increases.

Figure 5d shows the total number of proxy changes of all the reference sensors within the query duration. As reference sensor's maximum speed increases, the number of proxy changes of both CLDQ and SEAD increases accordingly. However, our approach results in much less proxy changes compared to SEAD. The advantage becomes more obvious as maximum speed increases. As explained in Sect. 3.1, the proxy selection method based on maximum distance projection prolongs reference sensor's resident period within its proxy's transmission range, which reduces the number of proxy changes.

Figure 5e indicates that the number of dissemination messages of SEAD and BEE is larger and grows faster than that of CLDQ as maximum speed of reference sensor increases. The reason is two folds. First, our proxy selection method incurs much less proxy changes than SEAD and BEE, which saves routing request messages and update messages. Second, our hop-based update algorithm optimizes the overall cost by considering the tradeoff between update cost and dissemination cost.

As shown in Fig. 5f, the highest benefit ratio of CLDQ-C to TinyDB reaches to more than 0.8 and the highest benefit ratio of CLDQ-C to BEE reaches to more than 0.7. This is because the cost of both query dissemination and processing can be reduced using CLDQ approach. A decrease of the benefit ratio is observed as reference sensor's speed increases. This is reasonable since the overlapping query area becomes smaller as maximum speed increases, resulting in less message savings of continuous processing optimization scheme.

5.4 Impact of Query Radius

Our second set of experiments studies the impact of various query radii on the performance of CLDQ. We compare CLDQ-S, CLDQ-C with TinyDB and BEE approaches by varying query radius from 100 to 350 m. The maximum speed of the reference sensor V_{max} is set to 1 m/s.

As shown in Fig. 6a, query accuracy of all the approaches decreases as query radius increases, because more data packets are dropped due to collisions and node failures in larger query areas. Note that CLDQ-S and CLDQ-C outperforms TinyDB and BEE, and

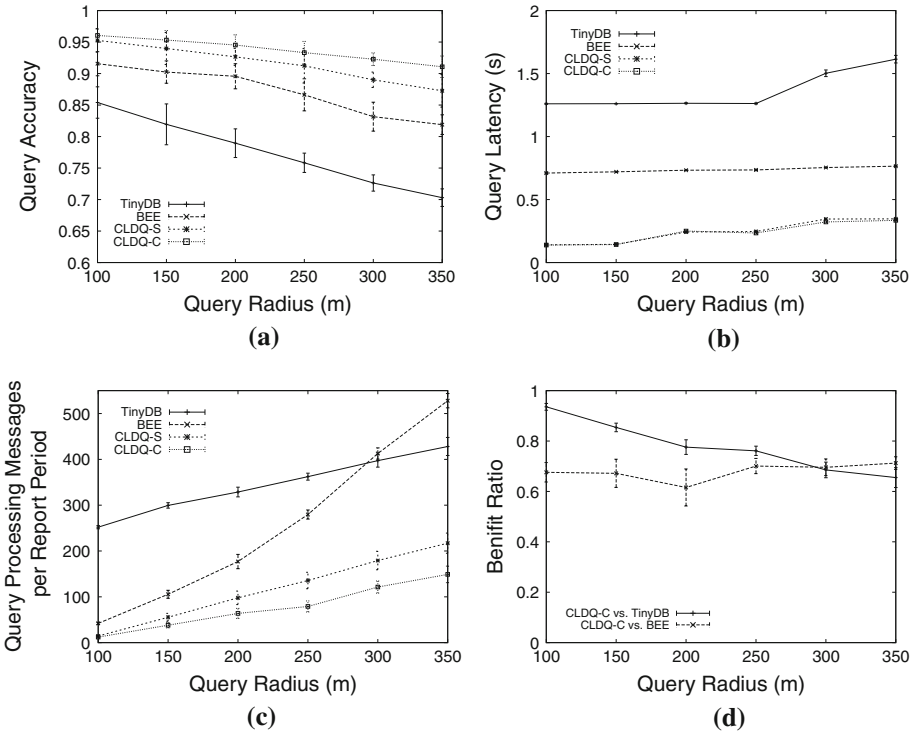

Fig. 6 Impact of query radius. **a** Query accuracy. **b** Query latency. **c** Query processing messages. **d** Benefit ratio

decrease slower than these two approaches. This is because CDMS reduces the probability of message collisions.

Figure 6b shows query latency of all the approaches increases with the query radius, because transmission hops increases as query radius becomes longer, which increases query latency accordingly. We can also observe that the query latency of CLDQ-S and CLDQ-C is close to each other but much shorter than that of TinyDB and BEE. As mentioned above, TinyDB and BEE rely on a fixed routing tree to disseminate query and collect sensor data. In contrast, CLDQ schedules propagation and aggregation by hop-based timers. In this way, CLDQ builds an optimal routing path which reduces query latency.

As shown in Fig. 6c, when the query radius is equal to the transmission radius (100 m), the number of CLDQ processing messages is very small, because stationary sensors in the query area can directly receive the query from the reference sensor by only one message. As query radius increases, query processing cost per report period of all the approaches continuously increases since more stationary sensors participate in the query. The processing cost of TinyDB and BEE increases much faster than CLDQ-S and CLDQ-C, which proves CLDQ approach is scalable and efficient. The number of query processing messages of TinyDB is always higher than 200, because the flooding cost and maintenance cost is comparably constant in a given network scenario. Further, BEE's query processing messages increase rapidly as query radius increases, because more zones are involved as query area become larger. We observe that CLDQ-C outperforms CLDQ-S, and the advantage becomes more obvious as the query radius increases. This is due to the fact that overlapping areas in successive report

periods probably become larger as query area becomes larger, facilitating our optimization scheme for continuous CLDQ processing. Given a fixed node density, the saving of our optimization scheme is proportional to the number of stationary sensors in overlapping areas, because more nodes can reuse their transmissions in previous report period by time series data forecasting.

As expected, Fig. 6d suggests that CLDQ-C maintains a high benefit ratio over both TinyDB and BEE. Note that the benefit ratio of CLDQ-C over TinyDB decreases as query radius becomes longer. The reason is that the proportion of the stationary sensors outside the query areas in TinyDB decreases as query radius increases.

5.5 Impact of Node Density

Finally, we evaluate the impact of node density on the performance of query processing by varying node density from $0.2 \times 10^{-3}\text{m}^2$ to $1 \times 10^{-3}\text{m}^2$. The query radius is set to 200 m and maximum speed of mobile sensors is set to 1 m/s.

As shown in Fig. 7a, query accuracy of all the comparing approaches decreases as the node density becomes larger. The reason for this outcome is that the collisions among sensor nodes increase as node density increases, deteriorating query accuracy. As expected, CLDQ-S and CLDQ-C outperforms BEE and TinyDB, and the highest accuracy of CLDQ-C reaches to 0.94 when node density is $0.2 \times 10^{-3}\text{m}^2$. CDMS and optimization scheme for continuous query processing contribute to the high accuracy of CLDQ-C.

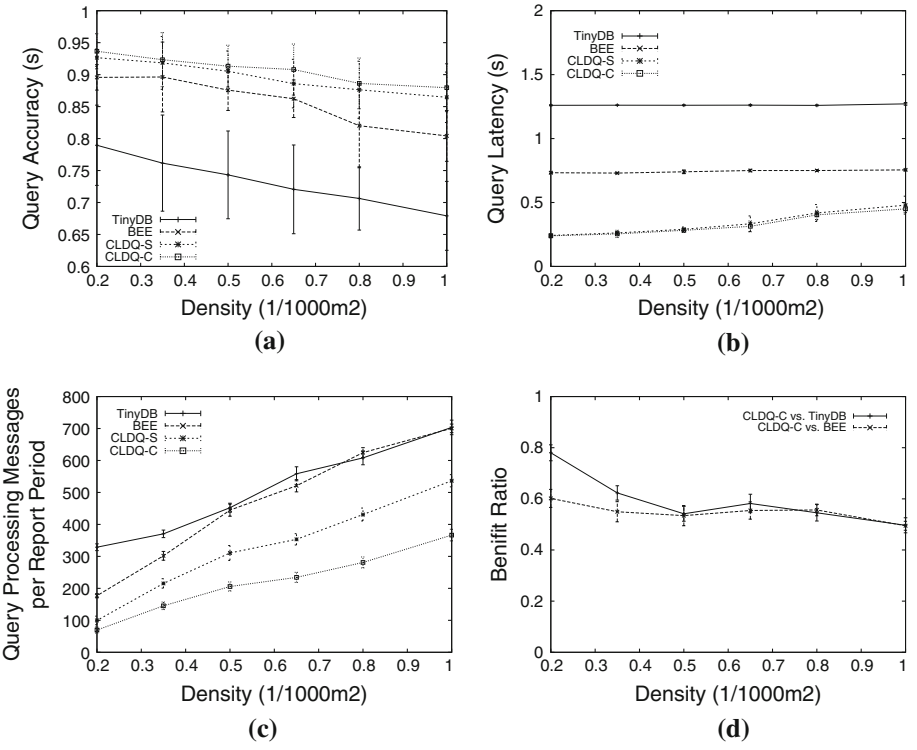

Fig. 7 Impact of node density. **a** Query accuracy. **b** Query latency. **c** Query processing messages. **d** Benefit ratio

Figure 7b shows that query latency of all the comparing approaches increases slowly with node density. Two factors contribute to this. First, as stationary sensors become denser, collisions increase, which results in more waiting time among the colliding nodes. Second, the increasing number of sensor nodes that participate in the query causes more transmission delay. We observe that CLDQ-S and CLDQ-C still result in a much shorter latency than TinyDB approach and BEE under all node densities.

Figure 7c shows the query processing cost of all the approaches increases with node density. This is due to the fact that more sensor nodes participate in the query as node density increases. We observe that CLDQ-S and CLDQ-C outperforms TinyDB approach and BEE in query processing cost under all node densities, and CLDQ-C has better performance than CLDQ-S. The increasing advantage of CLDQ-C over CLDQ-S may be explained as follows: as node density increases, more stationary sensors reside in overlapping areas; these sensors can save data transmissions if their predicted data are similar to current sampling data using our optimization scheme for continuous query processing.

Figure 7d shows the benefit ratio decreases slowly as node density increases. This is because the collisions among sensor nodes increase as node density increases, which incurs more retransmission messages in processing CLDQ. Nevertheless, CLDQ-C achieves high benefit ratio compared to TinyDB approach and BEE, which highlights the efficiency of our query processing approach.

6 Conclusions

Continuous Location Dependent Query (CLDQ) is an important and useful data service query existing in a wide range of data centric applications in mobile WSNs. This paper presents a promising processing approach for CLDQ, including an efficient dissemination approach, a *Contention-based Distance-aware Message Scheduling* (CDMS) scheme, and an optimization scheme for continuous processing of CLDQs. The dissemination approach manages the sensor mobility using the *maximum distance projection* strategy, and updates the dissemination path based on hop threshold. The CDMS scheme schedules query propagation and data aggregation by enabling each sensor node sets propagation and aggregation timer based on its hops from the reference sensor. The optimization scheme reduces continuous query processing cost by reusing data transmissions of the stationary sensors in the overlapping areas in case data prediction is correct. Our simulation results show that CLDQ achieves query accuracy at least 12% higher than TinyDB approach and 3% higher than BEE, query latency at least 65% lower than TinyDB approach and 40% lower than BEE, query processing cost at least 48% lower than TinyDB approach and 48% lower than BEE, and benefit ratio at least 50% over TinyDB approach and 49% over BEE.

As for future work, we plan to investigate the quality driven query processing by focusing on node failure and unreliable wireless transmission. Then, we will adapt our approach to optimize processing cost of multiple queries in inter-base station scenario. Finally, we plan to implement our approach in real sensor networks.

Acknowledgments The work in this paper was supported by Doctoral Fund of Ministry of Education of China (Project No.: 20100141120050) and Fundamental Research Funds for the Central Universities (Project No.:111135) and NSF China 61070011. This work was also supported in part by NRF WCU R33-2010-10110.

References

1. Borgne, Y. A. L., Silvia, S., & Bontempi, G. (2007). Adaptive model selection for time series prediction in wireless sensor networks. *Singnal Processing, 87*(12), 3010–3020.
2. Brockwell, P. J., & Davis, R. A. (2002). *Introduction to time series and forecasting*. New York: Springer-Verlag.
3. Fujimoto, M. R. (2000). *Parallel and distributed simulation systems*. London: Wiley-Interscience.
4. Gedik, B., Wu, K. L., Yu, P. S., & Liu, L. (2006). Processing moving queries over moving objects using motion-adaptive indexes. *IEEE Transactions on Knowledge and Data Engineering, 18*(5), 651–668.
5. Hong, L., Wu, Y., Son, S. H., & Lu, Y. (2009). Event-based location dependent data services in mobile WSNs. In *RTCSA* (pp. 331–340).
6. Hu, F., Wang, Y., & Wu, H. (2006). Mobile telemedicine sensor networks with low-energy data query and network lifetime considerations. *IEEE Transactions on Mobile Computing, 5*(4), 404–417.
7. Huang, H., Hartman, J. H., & Hurst, T. N. (2006). Efficient and robust query processing for mobile wireless sensor networks. In *GLOBECOM* (pp. 1–5).
8. Ilarri, S., Mena, E., & Illarramendi, A. (2006). Location-dependent queries in mobile contexts: Distributed processing using mobile agents. *IEEE Transantions on Mobile Computing, 5*(8), 1029–1043.
9. Ilarri, S., Mena, E., & Illarramendi, A. (2010). Location-dependent query processing: Where we are and where we are heading. *ACM Computing Surveys, 42*(3), 1–73.
10. Ishihara, S., & Suda, T. (2009). Replica arrangement scheme for location dependent information on sensor networks with unpredictable query frequency. In *ICC*.
11. Kamra, A., Misra, V., & Rubenstein, D. (2007). Counttorrent: Ubiquitous access to query aggregates in dynamic and mobile sensor networks. In *Sensys* (pp. 43–57).
12. Karp, B., & Kung, H. T. (2000). GPSR: Greedy perimeter stateless routing for wireless networks. In *MobiCom* (pp. 243–254).
13. Kim, H. S., Abdelzaher, T. F., & Kwon, W. H. (2003). Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks. In Sensys (pp. 193–204).
14. Liu, C., Wu, K., & Pei, J. (2007). An energy-efficient data collection framework for wireless sensor networks by exploiting spatiotemporal correlation. *IEEE Transactions on Parallel and Distributed Systems, 18*(7), 1010–1023.
15. Lu, C., Xing, G., Chipara, O., Fok, C. L., & Bhattacharya, S. (2005). A spatiotemporal query service for mobile users in sensor networks. In *Proceedings of the 25th IEEE international conference on distributed computing systems* (pp. 381–390)
16. Madden, S., Franklin, M. J., Hellerstein, J. M., & Hong, W. (2002). TAG: A tiny aggregation service for ad-hoc sensor networks. In *OSDI* (pp. 131–146).
17. Madden, S., Franklin, M. J., Hellerstein, J. M., & Hong, W. (2005). TinyDB: An acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems, 30*(1), 122–173.
18. Stoleru, R., Stankovic, J. A., & Son, S. (2008). On composability of localization protocols for wireless sensor networks. *IEEE Network Magazine, Special Issue on Composable Context Aware Services, 22*(4), 21–25.
19. Tulone, D., & Madden, S. (2006). PAQ: Time series forecasting for approximate query answering in sensor networks. In *EWSN* (pp. 21–37).
20. Wu, Y., Zhang, L., Wu, Y., & Niu, Z. (2006). Interest dissemination with directional antennas for wireless sensor networks with mobile sinks. In *Sensys* (pp. 99–111).
21. Xu, Y., Lee, W. C., Xu, J., & Mitchell, G. (2006). Processing window queries in wireless sensor networks. In *ICDE* (pp. 70–80).
22. Yang, X., Lim, H. B., Ozsu, M. T., & Tan, K. L. (2007). In-network execution of monitoring queries in sensor networks. In *SIGMOD* (pp. 521–532).
23. Ye, F., Luo, H., Cheng, J., Lu, S., & Zhang, L. (2002). A two-tier data dissemination model for largescale wireless sensor networks. In *MobiCom* (pp. 148–159).
24. Yoon, J., Liu, M., & Noble, B. (2003). Sound mobility models. In *MobiCom*.
25. Zeng, X., Bagrodia, R., & Gerla, M. (1998). GloMoSim: A library for parallel simulation of large-scale wireless networks. *ACM SIGSIM Simulation Digest, 28*(1), 154–161.
26. Zhang, Y., Hull, B., Balakrishnan, H., & Madden, S. (2007). ICEDB: Intermittently-connected continuous query processing. In *ICDE* (pp. 166–175)
27. Zhao, W., & Tang, X. (2011). Scheduling data collection with dynamic traffic patterns in wireless sensor networks. In *INFOCOM*.

Author Biographies

Dr. Liang Hong is an Assistant Professor in the School of Computer at Wuhan University, Wuhan, China. He received his Ph.D. degree from Huazhong University of Science and Technology, China in 2009. He was a visiting scholar in the Department of Computer Science at the University of Virginia from 2007 to 2008. He has published more than 10 research papers in conferences and journals. He served local arrangement chair of IEEE WAIM 2011. He is also a reviewer for several conferences. He is funded by NSF China and Nokia Research Center, Beijing. He is a member of ACM and CCF.

Dr. Gang Zhou is an Assistant Professor in the Computer Science Department at the College of William and Mary. He received his Ph.D. degree from the University of Virginia in 2007. He has published 37+ wireless communication and sensor networking papers in prestigious conferences and journals and the total citations of his papers are 1,000+ according to Google Scholar, among which the MobiSys'04 paper has been cited 460+ times. He served chair and technical program committee positions for 30+ academic conferences, including IEEE INFOCOM, IEEE ICDCS, IEEE RTSS, IEEE RTAS, IEEE IPDPS, etc. He is a reviewer for 25+ journals, such as ACM Transactions on Sensor Networks, ACM Transactions in Embedded Computing Systems, IEEE Transactions on Mobile Computing, IEEE Transactions on Parallel and Distributed Systems, etc. He is also a reviewer for 40+ conferences, such as ACM SENSYS, ACM/IEEE IPSN, IEEE INFOCOM, IEEE ICDCS, IEEE RTSS, IEEE IPDPS, etc. Dr. Zhou served as NSF and also GENI proposal review panelist in 2008 and 2010. He also received an award for his outstanding service to the IEEE Instrumentation and Measurement Society in 2008. He is a recipient of the Best Paper Award of IEEE ICNP 2010. He is a member of IEEE and ACM.

Bo Liu received the Ph.D. degree in computer science from Huazhong University of Science and Technology, Wuhan, China in year 2009. He was a visiting scholar at Deparment of Electrical Engineering and Computer Science in Vanderbilt University from August 2007 to August 2008. Since 2009, he has been with the School of Computer Science and Technology, HUST, where he is currently an Assistant Professor with the Service Computing Technology and System Laboratory and also with the Cluster and Grid Computing Laboratory. His research interests include P2P computing, social network analysis and social media management. He has published over 10 research papers. Dr. Liu serves as a reviewer for IEEE Transactions on Multimedia and ETRI Journal.

Sang Son received his Ph.D. in Computer Science at the University of Maryland in 1986. He then joined the University of Virginia as Assistant Professor of Computer Science and was promoted to Associate Professor in 1992, and to full Professor in 1999. He is serving as an Associate Editor of IEEE Transactions on Computers, since Sept. 2007, and served as an AE of the IEEE Transactions on Parallel and Distributed Systems. He is also serving on the editorial boards of the Real-Time Systems Journal, Journal of Information Processing Systems, and Journal of Mobile Communications, Networks, and Computing. He is the chair of the IEEE Technical Comiittee on Real-Time Systems, and has served as the Program Chair and/or General Chair of several real-time and database conferences, including IEEE Real-Time Systems Symposium, IEEE Conference on Parallel and Distributed Systems, International Workshop on Real-Time Database Systems, IEEE Conference on Electronic Commerce, and International Conference on Networked Sensing Systems. He was President of the Korean Computer Scientists and Engineers Association (KOCSEA), and in 2004, he received the IEEE Outstanding Contribution Award. He is the editor of three books and author or co-author of one book and over 260 refereed papers.