Mobile P2P Queries over Temporal Data

Steven Mudda
Networking Lab
University of Applied Sciences of Southern Switzerland
Email: steven.mudda@supsi.ch

Silvia Giordano
Networking Lab
University of Applied Sciences of Southern Switzerland
Email: silvia.giordano@supsi.ch

***Abstract*—In this paper we evaluate the performance a mobile P2P query processing framework, where peers issue location-dependent range queries and receive temporal data. We consider an environment where stationary objects generate temporal data and peers issue location-dependent range queries. We evaluate this framework by varying the constraints of range queries and measure their effect on three metrics: percentage of queries satisfied, fraction of data received and query response time. From our simulations, we make two key observations: the range query constraints have no impact on the performance metrics beyond a certain threshold and the temporal nature of data drastically limits the mobility benefits (traditionally exploited by mobile P2P query systems). We also present scenarios where a mobile P2P query processing framework can be used to satisfy location-dependent range queries.**

Keywords*—*query processing; range query; spatiotemporal data

I. Introduction

With the ever increasing use of mobile devices in our daily life, the demand for location-based services (LBS) has increased exponentially [1]. The location based services provide value-added information by considering the location of mobile users. LBSs has enabled the development of many interesting applications, such as emergency services (e.g., for roadside assistance), location-dependent advertising (e.g., sending interesting offers to users near a shopping mall), or tracking services (e.g., to keep track of fleets of vehicles).

However, in order to provide location-based services we need to process location-dependent queries. The response to a location-dependent query depends on the location of querying user. For example, a query—"Get bikes available at bike stations within 5km", the set of bike stations in response depends on user's current location and the time at which the query was posed [2].

Existing systems that process location-dependent queries utilize fixed communication infrastructure and centralized servers ([3], [4], [5], [6], [7]). Scalability, bottlenecks, and low fault-tolerance are some of the critical issues in centralized systems. These systems have a central point of failure that could result in the complete failure of LBSs for all users. However, with the proliferation of mobile wireless devices these location-dependent queries can also be processed at peers using different communication technologies like IEEE 802.11 and Bluetooth. Further, increase in the computational and storage capacity have enabled these devices to maintain a repository of information obtained from several location-dependent queries. These advancements have made peer-to-peer (p2p) query processing very appealing, where mobile devices evaluate location-dependent queries on data stored in local repository.

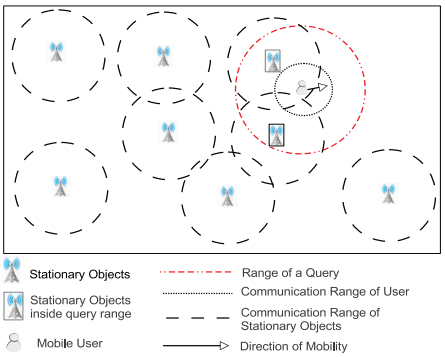

Fig. 1. Stationary Objects that fall inside the region of a *rangequery*

Among the many variants of location-dependent queries, *range query* on stationary objects has received a major focus of attention in research [8]. A *range query* aims to retrieve objects located within a certain range. Figure 1 shows an example of *range query* and stationary object(s) (objects with fixed location) located inside it. Current works on *range queries* only focus to retrieve data from stationary object(s) that produce *time invariant data*. Thus, data obtained for queries issued at same location does not change with time. In this paper, we focus on *location-dependent range queries* to retrieve data about stationary object(s) that produce time-varying or *temporal data*. For example, the number of bikes available at different bike stations in a city changes with time.

We present our mobile peer-to-peer query processing framework where every mobile user maintain a repository of temporal data received from stationary objects and execute *range* queries received from its peers. We focus on a scenario where stationary object(s) (e.g. bike stations) periodically broadcast their temporal data (e.g. number of available bikes) and mobile users (carrying wireless devices) issue range queries to obtain data from stationary objects or peers. In this scenario, the data obtained for queries issued at same location can change with time

From our simulations, we observe that by using a peer-to-peer query processing framework, users can obtain data from peers around 70% of time (less than data obtained in traditional centralized systems). The temporal nature of data limits the amount of data received. Two primary reasons behind this limitation are: requested data at peer is no longer valid due to the temporal constraint (i.e., data gets expired) and peers who do have data might be out of the communication range (i.e., their mobility is not in the direction of location where the

978-1-4799-2736-4/14/$31.00 ©2014 IEEE

query was issued). Further, we also observe that increasing the range of a query does not have any significant impact on the percentage of data obtained.

Our contributions in this paper can be summarized as follows:

- To the best of our knowledge we are the first one to present an evaluation of *range queries* over stationary objects with *temporal data* in a mobile peer-to-peer query processing framework.
- We study the impact of two constraints of a range query: distance and freshness, on data obtained.
- We show that freshness constraint reduces the benefits of mobile peer-to-peer framework and the distance constraint does not effect the results after a certain threshold.
- We consider a real world spatio-temporal data of bike stations located in Toronto city and simulate the mobility of users using a real world map of Toronto city.

The rest of the paper is structured as follows. In Section II, we give an overview of related work. We then describe our mobile peer-to-peer query processing framework in Section III. In Section IV, we present the evaluation of our query processing framework through simulations. Finally, we conclude the paper with future directions in Section V.

II. Related Work

Most of the work done on *location-dependent range queries* in mobile environments are based on fixed communication infrastructure and centralized database servers. They utilize a client-server model, where mobile clients issue *range queries* to central query processing servers [9] [10]. In these systems, the stationary objects continuously transfer their data to a central server and clients can obtain information about near-by objects by issuing *range queries* to the central server. The locations of different stationary objects are usually indexed at the central server in the form of R* Tree [11] and its variants.

Some of the recent works [12], [13], [14], [15] also address the issue of query processing in a mobile peer-to-peer environment. In [12], the authors assume that medium range connections exists between mobile devices and base-stations that interface with the wired Internet infrastructure, thereby partially relying on fixed communication infrastructure. In [14] [15], the authors deal with query processing with *time invariant* (that does not change with time) data. However, in real world scenarios the data generated by objects is usually time varying in nature. The authors of [13] address the problem of finding k-nearest objects in a scenario where every mobile device maintains a continuously updated cache of k-nearest objects. They only focus on obtaining the number of stationary objects within a certain range in a mobile peer-to-peer query processing framework. The authors assume that every mobile device has prior information of all stationary objects in its local storage and the cache is dynamically loaded with nearby stationary objects based on the current location of device. This assumption is not valid in scenarios where stationary objects produce time varying data. Finally, these works do not consider the data residing on stationary objects nor do they consider the temporal variation of data.

The work that is closely related to ours is spatio-temporal search for data in a mobile adhoc communication settings [16]. The authors also focus on querying for data that satisfies certain spatial and temporal constraints. However, they have only focused on scenarios that have single querying user. This assumption does not meet the requirement of a real world scenario, where multiple users simultaneously query for information. In our paper, we focus on scenarios with multiple querying users issuing *location-dependent range queries* to obtain temporal data of stationary objects in a mobile peer-to-peer environment. Further, we study the impact of different query constraints in obtaining temporal data of stationary objects.

III. Mobile Peer-to-Peer Query Processing

In this section, we describe our query processing framework. We first present our network scenario in Section III-A. Section III-B describes how users process data received directly from stationary objects. Finally in Section III-C, we present the different components of query processing in our mobile peer-to-peer framework.

A. Network Scenario

We consider a scenario consisting of stationary objects and mobile users. Both stationary objects and users are equipped with wireless devices like IEEE 802.11 that can communicate with each other in an adhoc manner. Further we assume that every user is aware of its location with the help of Global Positioning System (GPS) or other mechanisms.

We categorize users into two types: *Active* and *Passive*. *Active* users can be described as people who are interested in obtaining some location based information and are issuing queries to collect it. *Passive* users do not issue any query proactively but store any data received *directly* from stationary objects. The location of query is same as the location of user who is issuing it.

Now, we present some of the notations that will be used in describing our query processing framework:

N^a : Number of *Active* mobile users
N^p : Number of *Passive* mobile users
M : Number of Stationary objects
u_i^a : *Active* user with index $i, i \in 1, 2, 3, ...N^a$
$l_i^a(t)$: Location of *Active* user u_i^a at time t
u_j^p : *Passive* user with index $j, j \in 1, 2, 3, ...N^p$
O_k : Stationary object with index $k, k \in 1, 2, 3, ...M$
l_k^o : Location of Object O_k
$q_i(t)$: Range query issued by *Active* user u_i^a at time t
$d_k(t)$: Data packet transmitted by Object O_k at time t
$r_i(t)$: Query response transmitted at time t

1) Stationary Object (O_k): The data available at stationary object O_k at any time t is represented as a time series.

$$D^k = \{D_t^k : t \in T\}, T = \{t_i; i \in \mathbb{N}\}$$

Every object O_k periodically broadcast its temporal data item D^k into the network. If a mobile user (*Active* or *Passive*)

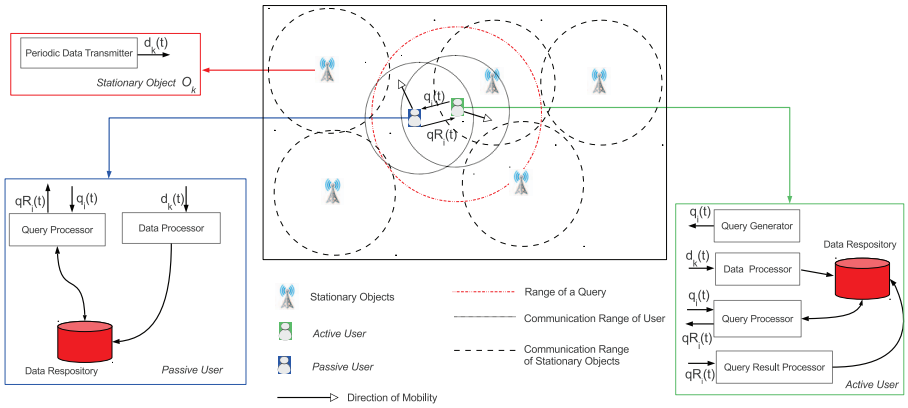

Fig. 2. Mobile Peer-to-Peer Query Processing Framework

comes within the communication range of O_k, then it might receive D^k. The structure of data packet $d_k(t)$ transmitted by a stationary object O_k at time t is given below:

$$d_k(t) := \{O_k, l_k^o, D^k, t\}$$

2) Data Repository: Both *Active* (u_i^a) and *Passive* (u_j^p) users maintains a local repository of data they receive. An *Active* user might receive data directly from an object O_k or by querying its peers (*Active* or *Passive*). However, a *Passive* user can only receive data directly from an object O_k. The format of local repository (R) and tuples (τ) that can be stored at every user is shown below:

$R = \{\tau_1, \tau_2, \tau_3 .... \tau_M\}, \tau_k := \{O_k, l_k^o, D^k, t\}$

If τ_k is received either from stationary object or peer

B. Processing of Stationary Object(s) data

Algorithm 1 describes how *Active* and *Passive* users process and store data packet $d_k(t)$ they receive directly from O_k.

Algorithm 1 Processing of data from Stationary Objects

Require: data packet $d_k(t)$, local repository R
Ensure: updated local repository R
  for all τ_i in R **do**
    if $d_k(t).O_k = \tau_i.O_i$ and $d_k(t).t > \tau_i.t$ **then**
      Replace τ_i with $d_k(t)$ in R
    end if
  end for

C. Mobile Query Processing

In this section, we first present how queries are generated by *Active* users and how they are propagated in the network. Then, we present how users execute these queries. Finally, we present how the query responses are propagated back to the querying user.

1) Query Generation: *Active* users generate queries and periodically broadcast them into the network. A query $q_i(t)$ transmitted by an *Active* user u_i^a at time t contains the following parameters:

$q_i(t) := \{u_i^a, l_i^a(t), d_{max}, d_{fresh}, t\}$
d_{max} : Maximum distance from an object
d_{fresh} : Data freshness or how old an object's data could be d_{max} and d_{fresh} are two constraints that each data item relevant to a query $q_i(t)$ must satisfy.

2) Query Propagation: The queries generated by *Active* users are propagated to only those users who are within its one hop neighborhood, i.e. when an *Active* user u_i^a broadcast its query at time t then users within its communication range could receive and process the query $q_i(t)$. Furthermore, users do not forward any queries they receive from neighbors.

3) Query Execution: When an *Active* (u_i^a) or *Passive* (u_j^p) user receives a query $q_i(t)$ from their neighboring *Active* user then two possible scenarios can arise: 1) user (u_i^a or u_j^p) has no data relevant to the query or 2) has data relevant to the query. In the first scenario, received query $q_i(t)$ is dropped, while in the second scenario query is executed over data stored in local repository.

Algorithm 2 Query Execution

Require: query $q_i(t)$, local repository R
Ensure: query result $qR \subset R$
  $qR = \emptyset$ t_{now} = *Current Time*
  for all τ_k in R **do**
    if distance($q_i(t).l_i^a$, $\tau_k.l_k^o$) $\leq q_i(t).d_{max}$ and $(t_{now} - \tau_k.t$) $\leq q_i(t).d_{fresh}$ **then**
      Add τ_k to qR
    end if
  end for

The process of executing a query $q_i(t)$ over a local repository R is described in Algorithm 2. If a user processing the query $q_i(t)$ has some relevant data (i.e., qR is not empty) then a response $r_i(t)$ is transmitted back to the querying user. The parameters of a response $r_i(t)$ is shown below:

$r_i(t) := \{u_i^a, l_i^a(t), u_x, l_x, qR, t\}$
u_i^a := Querying *Active* user with index i
l_i^a := Location of Querying *Active* user with index i

$$u_x := \begin{cases} u_j^a & \text{if an } \textit{Active} \text{ user with index } j \text{ responds} \\ u_j^p & \text{if a } \textit{Passive} \text{ user with index } j \text{ responds} \end{cases}$$

$$l_x := \begin{cases} l_j^a & \text{if an } \textit{Active} \text{ user with index } j \text{ responds} \\ l_j^p & \text{if a } \textit{Passive} \text{ user with index } j \text{ responds} \end{cases}$$

4) Query Result Processing: Once an *Active* user u_i^a receives a response $r_i(t)$ for its query $q_i(t)$ it updates its local repository. Algorithm 3 describes a simple procedure for updating the local repository.

Algorithm 3 Local Repository Update

Require: response $r_i(t)$, local repository R
Ensure: updated local repository R
  for all τ_i in $r_i(t).qR$ **do**
    for all τ_j in R **do**
      if $\tau_i.O_i = \tau_j.O_j$ and $\tau_i.t > \tau_j.t$ **then**
        Replace τ_j with τ_i in R
      end if
    end for
  end for

IV. Evaluation

In this section, we describe the metrics used to evaluate the performance of *location-dependent range* queries in mobile peer-to-peer query processing framework and also present the different simulation settings. Finally, based on our simulation results we provide some key insights for using a mobile peer-to-peer query processing framework.

A. Performance Metrics

In order to evaluate the performance of *range* queries over stationary objects (with temporal data) in a mobile peer-to-peer query processing framework, we use the following three metrics: *Average percentage of queries satisfied*; *Average fraction of data received* and *Average query response time*. We define a data as valid if it satisfies both constraints d_{max} and d_{fresh}.

1) Average percentage of satisfied queries: This metric is the average (over all querying *Active* users in the network) of the ratio of total number of satisfied queries over the total number of queries issued by an *Active* user. It measures the effectiveness of a mobile peer-to-peer query processing framework in providing data to a querying user. If the peers (or direct neighbors) of a querying user have some valid data in their local repository, then the querying user could receive multiple responses. We consider a query is satisfied if the querying user will receive *atleast* one response. We assume that, if a user receives at least one response to her/his query then they can take some action based on it.

2) Average fraction of data received: This metric is the average (over all querying *Active* users in the network) of the ratio of total unique data received by a querying user over the total unique data stored at peers. It measures the amount of valid data (of stationary objects) received by all querying users in the network. The peers within the communication range of a querying user could have valid data of multiple objects. However, due to the mobility of users, packet collisions, and wireless properties, two problems could arise: peers might not receive a query; or the response(s) might not be received by the querying user. Therefore this metric measures the quality of data obtained by a user.

3) Average Query Response Time: It measures how quick a valid data (or response) could be received by a querying user. It is the average time taken to receive the first query response.

Fig. 3. Bike Stations Located in Toronto City

B. Simulation Settings

We simulate our mobile peer-to-peer query processing framework using INET framework of OMNeT++ [17] and use SUMO [18] to simulate the mobility of users. We represent the stationary objects by a set of bike stations distributed in a city and its temporal data by the "number of available bikes at different times". For our simulations, we consider bike stations distributed in the city of Toronto, Canada [19]. By utilizing a live feed, we collect time-series data for bike stations located in Toronto city and use it to simulate the temporal variation of data at different bike stations. The bike stations periodically transmit their temporal data for the entire simulation. Figure 3 shows the spatial distribution of bike stations in Toronto city.

We use an OpenStreetMap [20] of Toronto city to generate different routes for both *Active* and *Passive* users. We chose a random source and destination for each user and find out a possible route using Dijkstra's Algorithm. In order to replicate the real world environment, users enter the simulation area at different times. Upon reaching their destination, users exit the simulation area to replicate the scenario of people not participating after they reach their home or office. Therefore the number of users present in the simulation changes with time. The *Active* users only constitute a certain percentage of total users moving around the city, while the remaining set of users are *Passive*.

Stationary objects and users can communicate with each other in an adhoc manner using a UDP networking protocol. We set the communication range of each user to 100m, a typical outdoor range of wireless devices. For bike stations, we set the communication range to 200m as they depend on fixed infrastructure and have lesser power constraints.

Table I presents different simulation parameters and their corresponding values.

TABLE I. Simulation Parameters

Parameter	Value
Simulation Area (m^2)	2200m X 2600m
Total Number of Bike Stations (M)	31
Max Number of Active Users (N^a)	160
Max Number of Passive Users (N^p)	1800
Bike Station's data transmission interval (in seconds)	30s
Active user's querying interval (in seconds)	60s
Radio communication range of user (in meters)	100m
Radio communication range of bike station (in meters)	200m
Mobility Speed of every user (in meters/second)	1.3m/s
Simulation Time (in seconds)	1800s

C. Simulation Results

The two constraints that describe every range query are: distance constraint (d_{max}) and freshness constraint (d_{fresh}). We evaluate the performance of our mobile peer-to-peer query processing framework by varying these constraints and observe their impact on performance metrics. Table II presents the set of different constraints used in our evaluation.

TABLE II. RANGE OF CONSTRAINTS

d_{max}	100m, 200m, 300m, 400m, 600m, 800m, 1000m
d_{fresh}	60s, 120s, 180s, 300s

1) Effect of query constraints on Average percentage of satisfied queries: Figure 4 shows the *Average percentage of satisfied queries* for different values of freshness when d_{max} varies from 100m to 1000m. When d_{max} grows from 100m to 300m, we observe an increase in the percentage of satisfied queries as the number of peers with valid data grows. Clearly, as the number of peers increase, the likelihood that a querying user receives atleast one response to its query also increases. This increment in peers can also be observed from the increase in total number of responses in the network (refer Figure 6).

When d_{max} is set to 300m, an ideal scenario arises due to the communication range of users and bike stations used in our setting. In this scenario, the peers of a querying user are exactly in between the querying user and potential bike stations. (i.e. peers are within the communication range of both querying user and bike stations). Therefore the likelihood of peers having valid data and responding to querying user is also high.

Further, we observe a minor increase in *Average percentage of satisfied queries* when d_{max} increases from 300m to 400m. This increment is due to additional peers who come in contact with a querying user—peers who are moving towards the querying user. But, the magnitude of this increment decreases as d_{fresh} varies from 300s to 60s. For lower values of freshness even though there is an increase in the number of peers (moving towards the querying user) the data stored in their repository becomes stale (i.e. loses its temporal validity) more quickly.

Finally, when d_{max} is increased beyond 400m, the percentage of satisfied queries does not change significantly. This is because the number of peers that participate in query processing and have valid data reaches a threshold (in range of 420-440 users).

The *Average percentage of satisfied queries* also increases when d_{fresh} varies from 60s to 300s for a given d_{max}. By increasing the value of freshness constraint, we increase the temporal validity of objects stored in peers. This enables peers to satisfy queries from users for a longer duration of time.

2) Effect of query constraints on Average fraction of data received: Figure 5 shows the *Average fraction of data received* for different values of freshness when d_{max} varies from 100m to 1000m. We observe that it varies differently in three intervals: $100 \leq d_{max} \leq 200$; $200 < d_{max} \leq 300$ and $d_{max} > 300$

When d_{max} increases from 100m to 200m, the total valid data (i.e. unique) stored at peers also increases. Thus, we observe an increase in the total number of responses (see Figure 6). This increase results in the growth of *Average fraction of data received*. However, the magnitude of increment is lower for higher values of freshness, because of multiple users responding to a query are closer to each other and this generates more collisions, effecting the fraction of data that a querying user could receive. We observe an exception when $d_{fresh} = 60s$, because the growth in valid data items does not match with a similar growth in number of peers. In this case, a significant growth in the *Average fraction of data received* happens when d_{max} moves from 200m to 300m.

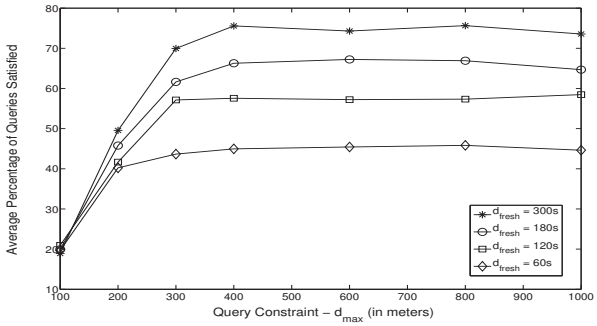

Fig. 4. Average percentage of satisfied queries when the distance constraint d_{max} varies from $100m$ to $1000m$

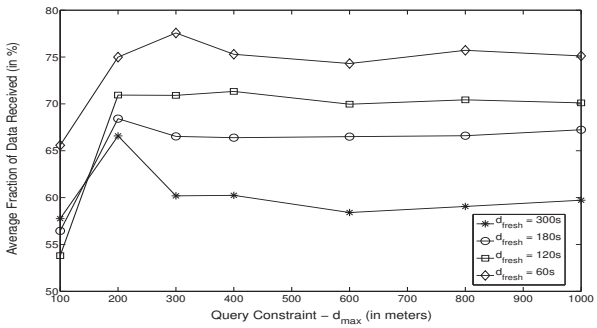

Fig. 5. Average fraction of data received when distance constraint d_{max} varies from $100m$ to $1000m$

For the other values of d_{fresh}, when d_{max} is increased to 300m, we observe a maximum increase in valid data stored at peers due to the formation of an ideal scenario (described in Subsection IV-C1). But, we do not observe a significant increase in the number of responses received successfully due to collisions. Therefore as d_{max} grows from 200m to 300m, the increase in the total data stored in peers does not correspond

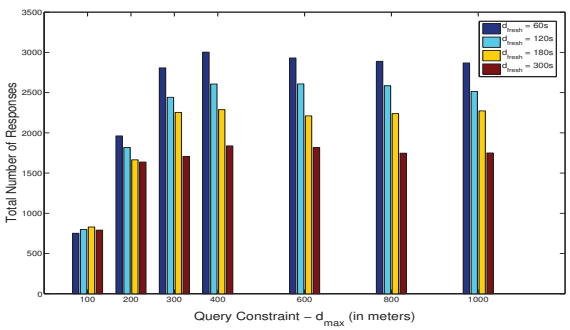

Fig. 6. Total number of responses transmitted by all users for different levels of d_{max} and d_{fresh}

to equal increment in the data received by a querying user. As a result, we find a reduction in the *Average fraction of data received*.

Finally, the *Average fraction of data received* does not vary significantly for further increments in d_{max} (beyond 300m) because the number of valid data at peers does not increase significantly. This is also reflected in Figure 6 where the total number of query responses does not vary significantly (for any level of freshness).

It is interesting to notice that the peak for *Average fraction of data received* and *Average percentage of satisfied queries* happens at different distance constraint values, as visible by comparing Figures 4 and 5. As discussed before, this difference is mainly due to collisions. In the beginning, as distance constraint increases the number of peers with valid data increases. Therefore, on one side the querying user has more probability to receive a response for her/his query, but the percentage of data received over the data stored at peers does not increase due to collisions. We also observe that after a given threshold (beyond 300 meters in our setting) both metrics do not vary significantly.

3) Effect of query constraints on Average query response: Table III shows the *Average query response* time as the query constraint d_{fresh} varies from 60s to 300s and d_{max} varies from 100m to 1000m. We observe that d_{fresh} and d_{max} have no impact of the average query response time.

TABLE III. AVERAGE QUERY RESPONSE TIME (IN SECONDS)

		d_{max}						
		100m	200m	300m	400m	600m	800m	1000m
d_{fresh}	60s	1.223	1.286	1.350	1.291	1.339	1.287	1.265
	120s	1.234	1.282	1.280	1.244	1.295	1.267	1.274
	180s	1.258	1.223	1.283	1.268	1.252	1.263	1.253
	300s	1.212	1.203	1.261	1.250	1.235	1.254	1.233

V. CONCLUSION AND FUTURE WORK

We evaluate the performance of location-dependent range queries in a mobile peer-to-peer querying processing framework. The environment consists of spatially distributed stationary objects and mobile users. Stationary objects generate temporally varying data and periodically broadcast them into the environment. From our evaluation, we derive some recommendations for using a mobile peer-to-peer querying processing framework with location-dependent range queries:

- The distance constraint d_{max} of a range query should be greater than the communication range of objects. Also, when choosing an adequately high value of d_{max}, (in our case it is sufficient more than 300 meters) the framework will enable querying users to receive responses roughly 70% of the time, but further increasing does not help.
- Similarly, when choosing a sufficiently high value of freshness constraint d_{fresh}, (in our case: more than 3 minutes) the framework will enable querying users to receive responses roughly 70% of the time.

Therefore, a mobile peer-to-peer query processing framework might not be useful in scenarios where querying users need to obtain data of all object's that satisfy distance and freshness constraints. In these scenario a traditional centralized query processing framework with infrastructure would be recommended.

In future, we would like to consider more evaluation scenarios and other data distribution strategies. We also plan to observe the impact of other parameters like user's mobility speed and data injection rate by stationary objects on the performance of range queries in mobile peer-to-peer query processing framework. We also intend to generalize our results and give a theoretical analysis of the framework.

Acknowledgment.This work was supported by the EU FP7 ERANET program under grant CHIST-ERA-2012 MACACO.

REFERENCES

[1] J. Schiller and A. Voisard, *Location-based services*. Elsevier, 2004.
[2] A. Y. Seydim, M. H. Dunham, and V. Kumar, "Location dependent query processing," in *Proceedings of the 2nd ACM international workshop on Data engineering for wireless and mobile access*. ACM, 2001, pp. 47–53.
[3] Y. Cai, K. A. Hua, and G. Cao, "Processing range-monitoring queries on heterogeneous mobile objects," in *Mobile Data Management, 2004. Proceedings. 2004 IEEE International Conference on*. IEEE, 2004, pp. 27–38.
[4] B. Gedik and L. Liu, "Mobieyes: Distributed processing of continuously moving queries on moving objects in a mobile system," in *Advances in Database Technology-EDBT 2004*. Springer, 2004, pp. 67–87.
[5] D. Yung, M. L. Yiu, and E. Lo, "A safe-exit approach for efficient network-based moving range queries," *Data & Knowledge Engineering*, vol. 72, no. 0, pp. 126 – 147, 2012.
[6] H. Hu, J. Xu, and D. L. Lee, "A generic framework for monitoring continuous spatial queries over moving objects," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. ACM, 2005, pp. 479–490.
[7] Y. Tao, D. Papadias, and Q. Shen, "Continuous nearest neighbor search," in *Proceedings of the 28th international conference on Very Large Data Bases*. VLDB Endowment, 2002, pp. 287–298.
[8] S. Ilarri, E. Mena, and A. Illarramendi, "Location-dependent query processing: Where we are and where we are heading," *ACM Comput. Surv.*, vol. 42, no. 3, pp. 12:1–12:73, Mar. 2010.
[9] H. Hu, J. Xu, and D. L. Lee, "A generic framework for monitoring continuous spatial queries over moving objects," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '05. New York, NY, USA: ACM, 2005, pp. 479–490.
[10] J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D. L. Lee, "Location-based spatial queries," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '03. New York, NY, USA: ACM, 2003, pp. 443–454.
[11] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, *The R*-tree: an efficient and robust access method for points and rectangles*. ACM, 1990, vol. 19, no. 2.
[12] W.-S. Ku and R. Zimmermann, "Nearest neighbor queries with peer-to-peer data sharing in mobile environments," *Pervasive and Mobile Computing*, vol. 4, no. 5, pp. 775–788, 2008.
[13] T. Nghiem, A. Waluyo, and D. Taniar, "A pure peer-to-peer approach for knn query processing in mobile ad hoc networks," *Personal and Ubiquitous Computing*, pp. 1–13, 2012.
[14] Z. Huang, C. S. Jensen, H. Lu, and B. C. Ooi, "Skyline queries against mobile lightweight devices in manets," in *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*. IEEE, 2006, pp. 66–66.
[15] W.-S. Ku, R. Zimmermann, and H. Wang, "Location-based spatial query processing with data sharing in wireless broadcast environments," *Mobile Computing, IEEE Transactions on*, vol. 7, no. 6, pp. 778–791, 2008.
[16] J. Michel, C. Julien, J. Payton, and G. Roman, "A spatiotemporal model for ephemeral data in pervasive computing networks," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*. IEEE, 2012, pp. 179–184.
[17] OMNET++, "http://www.omnetpp.org/."
[18] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3&4, pp. 128–138, December 2012.
[19] BikeStation-LiveFeed, "https://toronto.bixi.com/data/bikestations.xml."
[20] OpenStreetMap, "http://www.openstreetmap.org/."