Spatio-Temporal Query Processing in Smartphone Networks

Demetrios Zeinalipour-Yazti

Department of Computer Science, University of Cyprus
P.O. Box 20537, 1678 Nicosia, Cyprus
dzeina@cs.ucy.ac.cy

***Abstract*— In this position paper, we present a powerful and distributed spatio-temporal query processing framework, coined HUB-K. Our framework can be utilized to promptly answer queries of the form: *"Report the objects (i.e., trajectories) that follow a similar spatio-temporal motion to Q, where Q is some query trajectory."* HUB-k, relies on an in-situ data storage model, where spatio-temporal data remains on the smartphone that generated the given data, as well a state-of-the-art top-k query processing algorithms, which exploit distributed trajectory similarity measures in order to identify the correct answers promptly. We present preliminary design choices, an outline of our preliminary implementation and an outlook to future challenges.**

I. Introduction

The advances in networking technologies along with the wide availability of implicit or explicit positioning technology in commodity devices, make spatio-temporal records ubiquitous in many different domains. Popular cell-phones, such as Google's Android-based[1] devices, Nokia's Maemo-based devices[2] and i-Phone[3] devices, are equipped with built-in GPS receivers that allow these devices to derive their geo-spatial coordinates over time. Additionally, these devices are already equipped with a multitude of other sensing devices such as accelerometers (which enable the derivation of orientation, vibration and shock), proximity sensors, ambient light sensors and many others. The advent of these capabilities is soon expected to provide enormous distributed spatio-temporal collections of data that can create interesting new applications in a whole new range of domains.

An important observation with these emerging spatio-temporal systems, is that the generated data remains *in-situ*, at the device that generated the data until some event of interest occurs. This happens mainly for efficiency reasons, (i.e., it would be too energy costly to continuously transmit data records over to a centralized entity), but also for economic reasons (i.e., certain regions are still solely relying on 2G networks, thus users can not take advantage "always-connected" 3G networks.)

We argue that many future applications will adhere to this in-situ data storage model for the following reasons: i) spatiotemporal data becomes available in an ever growing number of applications; ii) organizations realize that a distributed data storage and query processing model is in many occasions more practical than storing everything centrally. A category of applications for which this is particularly true, are Sensor, RFID-related, Bokode and other related technologies that try to capture the physical world at a high fidelity; and iii) many of the generated spatiotemporal records might become outdated before they are ever utilized (e.g., a cell phone user might never be involved in some query), which again shows that centralization might be a wasteful approach.

[1]http://www.android.com/
[2]http://maemo.org/
[3]http://www.apple.com/iphone/

To formalize our description, let $\{A_1, A_2, \cdots, A_m\}$ denote a collection of spatiotemporal trajectories. A spatio-temporal trajectory A_i ($i \leq m$) is defined as a sequence of l multidimensional tuples $\{a_1, ..., a_l\}$ where each tuple is characterized by two spatial dimensions and one temporal dimension (i.e. $a_j(x_j, y_j, t_j)$, $\forall j \in 1, .., l$). A *segment* or *subsequence* of a trajectory A_i ($i \leq m$), is defined as a collection of r consecutive tuples $[a_j..a_{j+r}]$ ($j + r \leq l$). In the context of a smartphone network, each trajectory A_i resides in its entirety on some arbitrary smartphone device (i.e., data is fragmented horizontally).

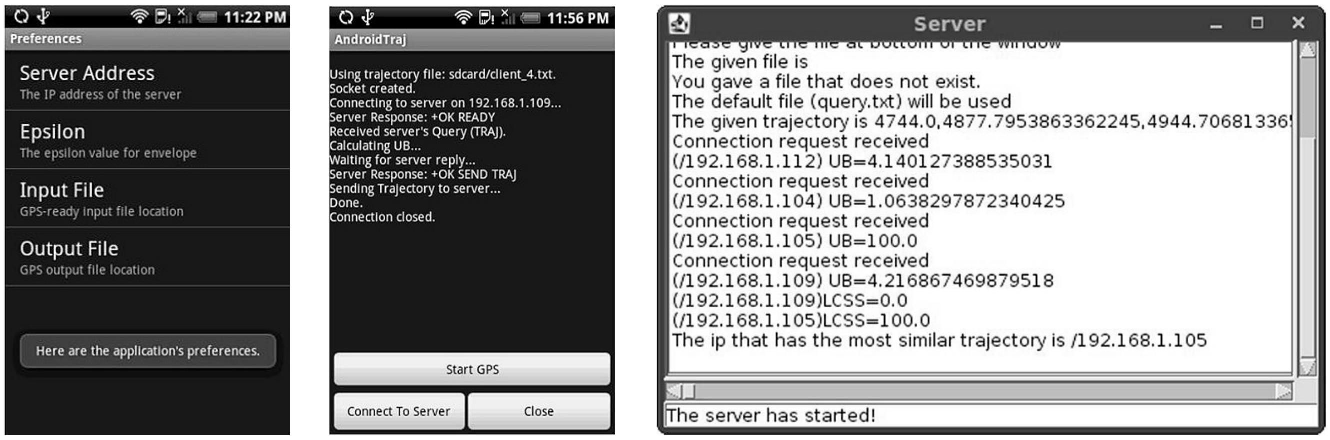
In this position paper, we present a powerful and distributed spatio-temporal query processing framework, coined HUB-K, which relies on an in-situ data storage model, as well as state-of-the-art top-k query processing algorithms [3], [2]. These algorithms exploit trajectory similarity measures in a distributed fashion to identify the correct answer promptly.

In the next section we will outline the operation of the HUB-K algorithm at a conceptual level. We shall then explore our preliminary implementation and provide an outlook to future challenges.

II. Background and Internal Algorithms

A. Preliminaries

First note that the similarity query Q is initiated by some querying node QN, which disseminates Q to all *candidate nodes* in the network (e.g., all participating smartphones in a network). A straightforward way to find the answer would be for QN to fetch all n trajectories (denoted as *DATA*) and then perform a centralized similarity computation using


```
Preferences
Server Address
The IP address of the server
Epsilon
The epsilon value for envelope
Input File
GPS-ready input file location
Output File
GPS output file location
Here are the application's preferences.
```

```
AndroidTraj
Using trajectory file: sdcard/client_4.txt.
Socket created.
Connecting to server on 192.168.1.109...
Server Response: +OK READY
Received server's Query (TRAJ).
Calculating UB...
Waiting for server reply...
Server Response: +OK SEND TRAJ
Sending Trajectory to server...
Done.
Connection closed.
```

```
Server
The given file is
You gave a file that does not exist.
The default file (query.txt) will be used
The given trajectory is 4744.0,4877.7953863362245,4944.70681336
Connection request received
(/192.168.1.112) UB=4.140127388535031
Connection request received
(/192.168.1.104) UB=1.0638297872340425
Connection request received
(/192.168.1.105) UB=100.0
Connection request received
(/192.168.1.109) UB=4.216867469879518
(/192.168.1.109)LCSS=0.0
(/192.168.1.105)LCSS=100.0
The ip that has the most similar trajectory is /192.168.1.105
The server has started!
```

Fig. 1. **Screenshots of the HUB-K prototype system** Left: The configuration panel on the Android-based smartphone client; Middle: The message exchange panel on the client; Right: The *Log Panel* of the server written in java and executing HUB-K (i.e., node QN).

some known technique such as LCSS, DTW or other L_p-Norm distance measures presented in [3]. Let us denote the execution of these centralized techniques as $FullM(Q, A_i)$ ($\forall i \leq m$). Although a centralized execution of LCSS is effective in coping with the temporal and spatial distortions in trajectories, as it performs local stretching in both time and space, it is expensive in terms of data transfer and delay as it requires the transfer of all trajectories prior execution.

B. The HUB-K Algorithm

The *HUB-K* algorithm is a new iterative algorithm for retrieving the K most similar trajectories to a query Q without pulling to Q all trajectories a' priori. The algorithm minimizes the number of *DATA* entries transferred towards QN by exploiting an upper bound on the *Longest Common Sub-Sequence (LCSS)* similarity metric [1]. In particular, each node compares its local trajectory A_i to a bounding envelope of the query, i.e.,

$$LCSS(MBE_Q, A) = \sum_{i=1}^{n} \begin{cases} 1 & \text{if A[i] within envelope} \\ 0 & \text{otherwise} \end{cases}$$

The above yields an upper bound score, per moving object A, that bounds above the LCSS matching between Q and A (i.e., for any two trajectories Q and A, $LCSS(A, Q) \leq LCSS(MBE_Q, A)$). The HUB-K algorithm is exploiting the Upper Bound (UB) in conjunction with an iterative acquisition process (i.e. rather than a monolithic transfer of all trajectories a priori, which minimizes tremendously the transfer of data.

III. Indicative Execution Plans

In this section we outline indicative modes of operation that are supported by our preliminary implementation.

Interactive Mode: In this mode, users start moving around, outdoors, by enabling the "GPS-logging"-mode, supported from within the user interface of HUB-K. At some arbitrary point, one of the devices A can ask the server QN to identify who is moving similar to A. The server will compute the answer and transmit it back to the user. The result will show up on an interactive Google-maps plot, so that user A can compare the result to its own trajectory. Preliminary tests have shown that such an action is taking only a few seconds, as opposed to the lengthy process of percolating each and every reading to QN, so we expect this feature to be of particular interest.

Trace-driven Mode, With this mode, researchers are able to realistically evaluate the given framework by select among a number of available traces (e.g., spatio-temporal data generated with the Network-based Generator of Moving Objects [4] and other datasets that we are currently underway of acquiring). Based on these datasets, researchers are able to sample out one query trajectory Q and query the network. In particular, it will be possible to identify the K most similar trajectories to Q quickly and efficiently (i.e., by showing how much time it would take to compute the answer with the centralized approach).

IV. Future Challenges

In this section we outline some challenges that arise in realizing fully distributed query processing frameworks in Smartphone Networks.

A. Challenge #1: Data Vastness

The WWW currently holds approximately 48 billion pages that change "slowly". On the other hand, we have more than 1 billion handheld smart devices (including mobile phones and PDAs) by 2010, according to the Focal Point Group[4], while ITU estimated 4.1 billion mobile cellular subscriptions by the start of 2009. This raises certainly many problems in analyzing and querying these massive distributed spatio-temporal repositories at regular intervals.

[4]According to the same group, in 2010, sensors could number 1 trillion, complemented by 500 billion microprocessors, 2 billion smart devices (including appliances, machines and vehicles).

B. Challenge #2: Uncertainty

Uncertainty is also inherent in Smartphone networks due to the following reasons: Integrating data from different smartphone devices might yield ambiguous situations (i.e., vagueness). Consider for instance the discrepancy that might occur in comparing spatial data (e.g., Triangulated Access Points vs. GPS) between different devices. Additionally, consider the discrepancy that might occur due to faulty electronics that generate outliers and errors (inconsistency) or even hacked sensor software might intentionally generate misleading information (deceit).

C. Challenge #3: Privacy

Frequent node migrations and disconnections in Smartphone Networks, as well as resource constraints raise severe concerns with respect to security, privacy and trust. Additionally, the cost of traditional secure data dissemination approaches (e.g., using encryption) may be prohibitively high in volatile mobile environments.

D. Challenge #4: Testbeds

Currently, there are no testbeds for emulating and prototyping Smartphone Networks applications and protocols at a large scale. The MobNet project (funded by the University of Cyprus between 2010-2011), is developing an innovative cloud testbed for mobile sensor devices using the Android Operating System. Consequently, users will be able to develop and deploy smartphone applications massively rather than individually on these devices.

Acknowledgments

This work is supported by the Startup Grant of the last author, funded by the University of Cyprus during 2010-2011.

References

[1] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, E. Keogh, "Indexing multi-dimensional time-series with support for multiple distance measures" In *ACM SIGKDD'03*, Washington, D.C., pp. 216-225, 2003.

[2] D. Zeinalipour-Yazti, Z. Vagena, D. Gunopulos and V. Kalogeraki, V. Tsotras, M. Vlachos, N. Koudas, D. Srivastava, "Finding the K Highest-Ranked Answers in a Distributed Network", Computer Networks, Volume 53, Issue 9, 25 June 2009, Pages 1431-1449

[3] D. Zeinalipour-Yazti, S. Lin, D. Gunopulos, "Distributed Spatio-Temporal Similarity Search" The 15th ACM Conference on Information and Knowledge Management (CIKM'06), Arlington, VA, USA, November 6-11, to appear, 200

[4] T. Brinkhoff: "A Framework for Generating Network-Based Moving Objects", GeoInformatica 6(2): 153-180 (2002)