SmartTrace: Finding Similar Trajectories in Smartphone Networks without Disclosing the Traces

Costandinos Costa*, Christos Laoudias*, Demetrios Zeinalipour-Yazti* and Dimitrios Gunopulos‡

* University of Cyprus, P.O. Box 20537, 1678 Nicosia, Cyprus
‡ University of Athens, 15784 Athens, Greece
cs07cc6@cs.ucy.ac.cy, laoudias@ucy.ac.cy, dzeina@cs.ucy.ac.cy, dg@di.uoa.gr

***Abstract*—In this demonstration paper, we present a powerful distributed framework for finding similar trajectories in a smartphone network, without disclosing the traces of participating users. Our framework, exploits opportunistic and participatory sensing in order to quickly answer queries of the form: *"Report objects (i.e., trajectories) that follow a similar spatio-temporal motion to Q, where Q is some query trajectory."* SmartTrace, relies on an in-situ data storage model, where geo-location data is recorded locally on smartphones for both performance and privacy reasons. SmartTrace then deploys an efficient top-K query processing algorithm that exploits distributed trajectory similarity measures, resilient to spatial and temporal noise, in order to derive the most relevant answers to Q quickly and efficiently. Our demonstration shows how the SmartTrace algorithmics are ported on a network of Android-based smartphone devices with impressive query response times.**

To demonstrate the capabilities of SmartTrace during the conference, we will allow the attendees to query local smartphone networks in the following two modes: *i) Interactive Mode*, where devices will be handed out to participants aiming to identify who is moving similar to the querying node; and ii) *Trace-driven Mode*, where a large-scale deployment can be launched in order to show how the K most similar trajectories can be identified quickly and efficiently. The conference attendees will be able to appreciate how interesting spatio-temporal search applications can be implemented efficiently (for performance reasons) and without disclosing the complete user traces to the query processor (for privacy reasons)[1]. For instance, an attendee might be able to determine other attendees that have participated in common sessions, in order to initiate new discussions and collaborations, without knowing their trajectory or revealing his/her own trajectory either.

I. Introduction

The widespread deployment of smartphone devices featuring geo-location (e.g., AGPS, Cell tower and WLAN positioning) and other sensing capabilities (e.g., proximity, ambient light, accelerometer, camera, microphone, etc.) along with Internet connectivity through WLAN, WCDMA/UMTS(3G), HSPA(3.5G) and LTE/WiMAX(4G) networks, have brought a revolution in location-oriented mobile applications and services. IMS Research and Comscore reported over 225M smartphone sales in February 2010 (i.e., RIM, Apple, Microsoft, Google and Palm) and according to the Focal Point Group, handheld smart devices (including mobile phones and PDAs) could number 1 billion in 2010. We define a Smartphone Network as "*a set of smartphone devices that communicate in an unobtrusive manner, without explicit user interactions, in order to realize a collaborative or social task.*"

There is already a proliferation of innovative applications founded on smartphone networks. One example is opportunistic and participatory sensing [6], [1], [3], where applications can task mobile nodes in a given region to provide information about their vicinity using their sensing capabilities. Another example is road traffic delay estimation [9] using WiFi beams collected by smartphone devices rather than invoking expensive GPS acquisition. On the social site, Google Latitude[7] enables users to track the places they and their social network have visited. The given service already reports over 3M enrolled users and over 1M active users, despite the controversial privacy concerns. Similarly, mobile social networking applications like Foursquare, Gowalla and Loopt enjoy enormous success in the Smartphone community and academic efforts in this direction are also underway.

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. $a_j(x_j, y_j, t_j)$, $\forall j \in 1, .., l$). Each trajectory A_i resides in its entirety in-situ, which is cheaper and more efficient for smartphone environments. Given a query Q, itself expressed as a spatio-temporal trajectory, we compare each A_i to the points of Q within some temporal and spatial window. SmartTrace, circumvents expensive and massive similarity executions by running an inexpensive linear-time computation on the smartphones in a pre-processing step. It then uses an iterative top-K processing algorithm in order to iteratively identify the K most similar trajectories to Q, without ever pulling the target trajectories to the centralized query processor.

The notion of *similarity* captures the trajectories that differ only slightly, in the whole sequence, from the given search query Q. An example query might be: (e.g., *"Find whether there is a cycling route from the Metropolitan Museum of Art in Manhattan, through central park to the Juilliard School"*), or *"Find which Zebras moved more closely to Zebra named Abby before it got injured" [8]*. There are already centralized

[1]Video Presentation available at the following URL: http://www.cs.ucy.ac.cy/~dzeina/smarttrace/

trajectory search services such as GeoLife[2], GPS-Waypoints[3], ShareMyRoutes[4], and their academic counterparts [4], to perform this kind of querying. However, these services store user's trajectories on a centralized or cloud-like infrastructure.

On the other hand, the techniques utilized by SmartTrace are decentralized and maintain the data *in-situ* (i.e., on the smartphone that generated the data). When a query emerges, we collect a set of scores from participating nodes (as opposed to collecting their location continuously) and derive the answer intelligently based on these scores only, without ever unveiling the target trajectories to the query processor. While this cannot take advantage of global knowledge structures available in a centralized setting (e.g., catalogs, indexes, etc.), our setting has numerous advantages for the environments under discussion as explained next.

In this demonstration paper, we will start out by presenting the high-level algorithms behind SmartTrace. We shall then present the complexities that arise in indoor environments, where location can not be derived by absolute means. We will finally present our demonstration setting and plan that will support both interactive scenarios and trace-driven scenarios.

II. Background and Internal Algorithms

A. Preliminaries

First note that the similarity query Q is initiated by some querying node QN (or alternatively at some smartphone that propagates its Q towards QN). QN then disseminates Q to all active smartphone users in a pre-specified spatial boundary. Upon receiving Q, each candidate smartphone executes locally an inexpensive linear-time matching function. QN then collects these scores and puts together a vector of upper bounds $UB = (ub_1, \cdots, ub_m)$. We will refer to the UB-vector constructed on QN as *METADATA* and to the actual trajectories stored locally on each smartphone as *DATA*. Obviously, *DATA* is orders of magnitudes larger than *METADATA*, thus *DATA* needs to stay on the smartphone during query resolution. Our objective is to intelligently exploit the *METADATA* scores in order to identify the K highest ranked answers without pulling *DATA* to QN.

B. The SmartTrace Algorithm

The *SmartTrace* algorithm is a novel iterative algorithm for retrieving the K most similar trajectories to a query trajectory Q. Our proposed scheme performs well both with respect to response time and energy, but also does so without ever revealing the complete target trajectories to QN (i.e., it only returns the matched subsequence, if any.)

Description: In step 1 of the SmartTrace algorithm, QN instructs all m nodes to invoke the computation of the linear-time upper-bounding function $LCSS(MBE_Q, Ai)$ $(i \leq m)$. In that way it circumvents the massive deployment of the expensive similarity function $LCSS(Q, A_i)$ [11], presented next, which performs local stretching in both time and space to overcome the temporal and spatial distortions in trajectories. In particular, each node compares its local trajectory A_i to a bounding envelope of the query, i.e.,

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

In step 2, QN retrieves all these upper bounds and adds them in descending order to a local *METADATA* vector. By doing this, QN obtains a quick summary of the trajectories similar to Q.

Steps 3 to 5 are executed iteratively until convergence. In particular, during step 3, QN adds the identities of the objects with the $\lambda + 1$ highest upper bounds to a set named S. These objects provide the first line of candidates for the answer set, as these objects have the highest $LCSS(MBE_Q, A_i)$ value. The given objects will be analyzed more carefully in the next step of the algorithm in order to determine the correct top-K set. Please notice that the objects in the S-set, do not again define the final top-K result. In particular, it is absolutely possible that some arbitrary object in the S-set with a high $LCSS(MBE_Q, A_i)$ score has a low full score $LCSS(Q, A_i)$. Consequently, the algorithm can still not converge with the desired result.

The λ parameter, mentioned previously, expresses an application-specific confidence in the *METADATA* bounds. In particular, when the *METADATA* vector contains tight bounds, then λ might be set to a small value. So this parameter defines how aggressively some application wants to determine the top-K results. It can be proven that SmartTrace will not perform more than $O(m/\lambda)$ iterations in the worst case.

In step 4, QN asks each smartphone in the S-set, to compute the full scores (if a smartphone has been contacted in a previous iteration we do not contact it again). In particular, we ask each smartphone to locally compute $FullM(Q, A_i)$, where A_i is stored locally, and only transmit the value of $FullM(Q, A_i)$ towards QN (i.e., the decentralized way). Alternatively, we could have also fetched the S-set to the sink and then compute $FullM(Q, A_i)$ $\forall A_i \in S$ (i.e., the centralized way), however this would violate both the trace disclosure factor and also degrade the response time of the algorithm to a level comparable to the centralized algorithm. Notice that the fourth step of the algorithm applies only to the elements in the S-set, as opposed to all m elements so this is really much cheaper in terms of energy consumed on the smartphone as $|S| << m$.

In our case, $FullM(Q, A_i)$ is the LCSS similarity, which has been extensively used in many 1-D sequence problems, such as string matching. The 2-dimensional adaptation of LCSS using the L_∞[5] is defined as following:

Definition: *Given integers δ and ϵ, the Longest Common Sub-Sequence similarity $LCSS_{\delta,\epsilon}(A, B)$ between two sequences A and B is defined as:*

[2]GeoLife, 11/2010, http://research.microsoft.com/en-us/projects/geolife/

[3]GPS Waypoints, 11/2010, http://www.gps-waypoints.net

[4]ShareMyRoutes.com, 11/2010, http://www.sharemyroutes.com/

[5]We could also use L_1 or L_2 for the recursion step.

$$LCSS_{\delta,\epsilon}(A,B) =$$

$$\begin{cases} \mathbf{0}, & \text{if A or B is empty} \\ \mathbf{1} + \mathbf{LCSS}_{\delta,\epsilon}(\mathbf{Tail}(\mathbf{A}), \mathbf{Tail}(\mathbf{B})) \\ \quad \text{if } |a_{x:l_1} - b_{x:l_2}| < \epsilon \text{ and} \\ \quad\quad |a_{y:l_1} - b_{y:l_2}| < \epsilon \text{ and } |l_1 - l_2| < \delta \\ \mathbf{max}(\mathbf{LCSS}_{\delta,\epsilon}(\mathbf{Tail}(\mathbf{A}), \mathbf{B}), \mathbf{LCSS}_{\delta,\epsilon}(\mathbf{A}, \mathbf{Tail}(\mathbf{B}))) \\ \quad \text{otherwise} \end{cases}$$

where δ and ϵ are application-specific parameters that allow flexible matching in the *time* (e.g., if we have two identical trajectories, but the first one moves earlier in time) and the *space* (e.g., we have two identical trajectories but the first has some slight deviation in its spatial movement) domain, respectively. LCSS deals with both aforementioned limitations of the L_p*-Norm* family of distances, because these cases are simply dropped from the matching without skewing the measure.

In step 5, we determine whether the algorithm has reached a termination condition. In particular, we check if the (λ+1)-th highest UB is smaller than the K-th highest full matching value. If this is the case, then we can safely terminate the execution of the algorithm being sure that the correct top-K has been identified. If this condition does not hold (i.e., when the UB of an object X is larger than the K-th highest full matching value Y), then we are enforced to perform another iteration as the answer is not deterministic (i.e., either X or Y can be the K-th answer). Consequently, we increase the step increment λ so that it identifies the next λ candidates in the next round.

In the final step, which occurs only once at the very end, we might ship each matched subsequence A_i^{match} ($|A_i^{match}| << |A_i|$) to QN, which can then return it to the user. Notice, that identified nodes s_i ($i \leq K$) might choose not to share the matching or share it based on some local trace disclosure profile [5], in order to preserve k-anonymity and other higher anonymity schemes. In any case, neither QN nor the querying user will ever see the complete trajectory of participating users.

III. SmartTrace in Indoor Environments

In this section we show how our algorithm can operate in an indoor environment and in the absence of absolute positioning techniques, such as GPS. In particular, we show how the SmartTrace algorithm can exploit location-related information, such as signals emitted by WiFi access points, in order to conduct the similarity search.

A. Overview

It is well known that satellite signals, used by GPS receivers, are severely attenuated or blocked inside buildings, thus failing to provide accurate location information. Yet, the massive availability of smartphone devices in conjunction with the fact that human spend over 90% of their time in indoor environments has revitalized the interest in indoor location-aware applications (e.g., applications for company premises, shopping malls, libraries, hospitals, homes, etc.)

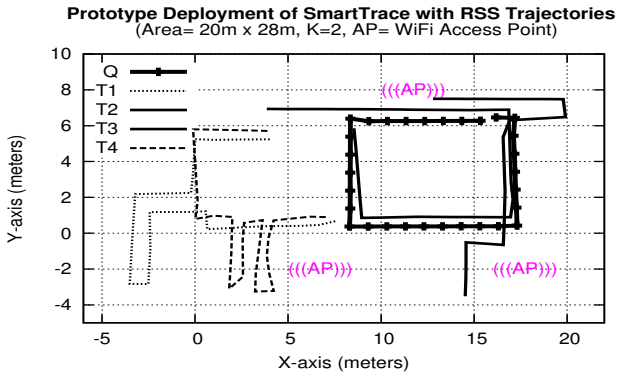

Fig. 1. Real Deployment of SmartTrace Indoors at the KIOS Research Center.

Notice that built-in Cell Tower or WLAN positioning techniques for smartphones, cannot provide an accuracy good enough for computing trajectory similarities for indoor environments. For instance, the *"My Location"* service by Google, can only provide accuracies of about 200m while Cell Tower Positioning provides accuracies of several thousand meters. Thus, both techniques can not cope with indoor spaces we consider in this section.

A spatio-temporal trajectory A_i ($i \leq m$), denoted as RSS Trajectory, is a sequence of l multidimensional tuples $\{a_1, ..., a_l\}$. Each tuple is characterized by n spatial dimensions and one temporal dimension, i.e. $a_j(t_j, , S_j^1, S_j^2, \ldots, S_j^n)$, $j \leq l$, where t_j denotes the timestamp on which the record was generated and S_j^k ($k \leq n$, $j \leq l$) the RSS value from the k-th Access Point (AP), measured by A_i at t_j. Using the above model, the similarity between two queries can now be computed using the LCSS metrics on individual attributes and combining them linearly.

We have conducted an example validation experiment in an area that is roughly 28m x 20m containing office rooms, open plan workstations and meeting rooms connected with corridors. In the above scenario we have conducted a top-2 query for one of the five (5) trajectories (See Figure 1). The trajectories T2 and T3 were correctly identified as the top-2 answers to the above query.

IV. Android Implementation

We have developed a prototype system that realizes the SmartTrace framework (see Figure 2). Our client-side software is developed around the Google Map API and its installation package (i.e., APK) has a size of 510KB. Our code is written in JAVA and consists of approximately 4,500 lines-of-code (LOC). In particular, our server-code uses ∼1,500 LOC and runs over JDK 6 and Ubuntu Linux, our smartphone-code uses ∼2,500 LOC plus ∼250 lines of XML elements. The client-code runs over the Dalvik VM. In the future we plan to take the computationally-intensive and IO-intensive tasks outside the VM by implementing them in native (C) code using the Android NDK. The SmartTrace GUI allows a user to query other devices by example, plot and iterate through the responses using a variety of presentation functions, as well

as to configure a wide range of parameters, such as K. Our prototype finally enables a user to switch between Offline and Online Mode, offering in that way the possibility to simulate movement (i.e., the trajectory file can be stored on the flash media as opposed to be collected in real time). The given setting helps with playing back recorded scenarios.

V. Demonstration Settings

A. Equipment

For the actual demo at the conference we will use 5 HTC Desire smartphone devices running Android 2.1 (Eclair). These devices are equipped with a Qualcomm Snapdragon QSD 8250 1 GHz processor and provide 512 MB of Flash ROM as well as 512 MB of RAM. Although our Desire devices feature multi-homing capabilities and support Quad-band GSM/GPRS/EDGE as well as Bluetooth 2.0, we will utilize its built-in 802.11 b/g Wi-Fi connection to connect to the SmartTrace server. In the trace-driven demonstration mode, all trajectories will be logged on the 512MB on-board flash ROM for performance reasons. Alternatively, we can also utilize the 4GB external microSD memory card but at a performance penalty.

Our base station is a standard Macbook Pro running Mac OS X 10.6. The given laptop will connect to the HTC Desire devices through the available Wi-Fi hotspot. We will use a projector along with an a smartphone display export utility to present the interactions on a smartphone directly on a wall (so that attendees will be able to follow the interactions).

B. Demo Plan

Interactive Mode: At the conference site, and prior to our demonstration, we will hand out 10 smartphone devices to selected participants (after providing identification details). These participants will then be asked to start moving around for a few minutes, indoors and/or outdoors, by enabling the "Online"-mode, supported from within the user interface of SmartTrace (see 2, left). This mode will log the user's spatio-temporal trajectory to the flash card. Two arbitrary users X and Y will be asked to walk within proximity during the largest interval of their walk. At the end of this interval we will ask X to connect its device, through USB, to a laptop that will project X's screen on a projector. We will then conduct a top-1 query through X. This execution should reveal the trajectory of the Y user. The result will show up on an interactive Google-maps interface, so that user X can compare the returned subsequence to its query trajectory (see Figure 2, right). Our tests have shown that such an action is taking only a few seconds, even for large trajectories, so we expect this feature to be of particular interest.

Trace-driven Mode, In this mode, we will allow attendees to select among a number of available traces (e.g., GeoLife [12] and Oldenburg [2]). Based on these datasets, we will sample out one query trajectory Q, and add interpolated peaks of Gaussian noise, in order to create variations in the pattern of Q. We will then show how the SmartTrace framework is able to identify the K most similar trajectories to Q quickly and efficiently.

Fig. 2. **Screenshots of the SmartTrace demonstration system** Left: The configuration panel on the Android-based smartphone client; Middle: A Query Response Message; Right: The *matched trajectory* displayed using Google Maps.

Through our demo, the conference attendees will be able to appreciate how interesting spatio-temporal search applications can be implemented efficiently (for performance reasons) and without disclosing the user traces to the query processor (for privacy reasons). For instance, an attendee might be able to determine other attendees that have participated in common sessions, in order to initiate new discussions and collaborations.

Acknowledgments: This work was supported in part the third author's Startup Grant, funded by the University of Cyprus between 2010-2011, EU's FP6 Marie Curie TOK "SEARCHiN" project, EU's FP7 "CONET" project and EU's FP7 "SemSorGrid4Env" and "MODAP" projects.

References

[1] Azizyan M., Constandache I., Choudhury R.-R., "SurroundSense: mobile phone localization via ambience fingerprinting," In *MobiCom*, 2009.
[2] Brinkhoff T. : "A Framework for Generating Network-Based Moving Objects", GeoInformatica 6(2): 153-180 (2002)
[3] Campbell A., Eisenman S., Lane N., Miluzzo E., and Peterson R., "People-centric urban sensing," In *WICON*, 2006.
[4] Chen Z., Shen H-T., Zhou X., Zheng Y., Xie X. "Searching trajectories by locations: an efficiency study," In *SIGMOD*, 2010.
[5] Chow C-Y., Mokbel M.F., Aref W.G., "Casper*: Query Processing for Location Services without Compromising Privacy" In ACM TODS 34(4), pp. 1-48, 2009.
[6] Das T., Mohan P., Padmanabhan V.N., Ramjee R., Sharma A., "PRISM: platform for remote sensing using smartphones," In *MobiSys*, 2010.
[7] Google Latitude, 11/2010, http://www.google.com/latitude
[8] Liu T., Sadler C.M., Zhang P., Martonosi M., "Implementing Software on Resource-Constrained Mobile Sensors: Experiences with Impala and ZebraNet," In *MobiSys*, 2004.
[9] Thiagarajan A., Ravindranath L., LaCurts K., Madden S., Balakrishnan H., Toledo S., Eriksson J., "VTrack: Accurate, Energy-aware Road Traffic Delay Estimation using Mobile Phones," In *SenSys*, 2009.
[10] Vlachos M., Hadjieleftheriou M., Gunopulos D., Keogh E., "Indexing multi-dimensional time-series with support for multiple distance measures" In *SIGKDD*, 2003.
[11] Zeinalipour-Yazti D., Lin S., Gunopulos D., "Distributed Spatio-Temporal Similarity Search" In *CIKM*, 2006.
[12] Zheng Y., Liu L., Wang L., Xie X., "Learning transportation mode from raw gps data for geographic applications on the web," In *WWW*, 2008.