PLUS: A Message-Efficient Prototype for Location-Based Applications

Yu-Ling Hsueh[†], Roger Zimmermann[‡], Wei-Shinn Ku[§], Haojun Wang[†], and Chung-Dau Wang[†]

[†]Computer Science Department, University of Southern California, Los Angeles, CA 90089
[‡]Computer Science Department, National University of Singapore, Singapore 117543
[§]Department of Computer Science and Software Engineering, Auburn University, Auburn, AL 36849
{hsueh@usc.edu, rogerz@comp.nus.edu.sg, weishinn@auburn.edu, haojunwa@usc.edu, chungdaw@usc.edu}

Abstract— **The PLUS system is designed to efficiently track moving object locations on a road network and execute continuous spatial queries in support of location-based services. PLUS implements a novel lazy position update mechanism that significantly reduces the communication overhead and server indexing load related to frequent location updates in moving object and moving query scenarios. The contribution of this demo is to present how the lazy position update scheme can achieve message-efficiency under various conditions which can be interactively set via user-selectable parameters in a graphical user interface.**

I. Introduction

As a result of recent technological advances, GPS-equipped mobile devices with significant computational abilities, gigabytes of storage, and wireless communication capabilities provide a compelling environment to support location-based services to mobile users. The efficient evaluation of continuous spatial queries is a fundamental feature needed in many practical applications. An example query launched from a fire engine while battling flames might be to "continuously locate other fire engines within two miles of my current location." Since all units (i.e., users) are constantly moving, frequent location updates often result in high server re-indexing costs and immense communication overhead. We have recently designed the *Partition-Based Lazy Update* algorithm [1] to evaluate continuous queries by maintaining a *Location Information Table* (LIT) on each mobile device. A LIT is a grid data structure where each cell stores a value that represents its distance to the closest query boundary. Figure 1 shows an example of a mobile-side LIT cached in the local memory of a mobile client, where the cell elements with zero value represent the area overlapping with the three example query boundaries. If the value of a cell element is greater than zero, the cell is completely outside of any query area. On the other hand, if the value of a cell element is less than zero, the cell is inside of a query boundary. Hence all the data points in such a cell are answer points. PLUS avoids transmitting location updates that do not affect any query results through the use of LITs which (a) allow each moving object to estimate possible query movements and issue a location update only when it may affect any query results and (b) enable smart server probing that results in fewer message exchanges. When an event (a voluntary mobile-side location update or a server-side location probe) occurs, then a mobile unit receives an up-to-date LIT from the server. In between events, the LIT is locally maintained by the moving object based on the worst-case estimated movements of nearby query boundaries for handling dynamic queries. A number of previously proposed techniques have provided significant insight into this issue. The mobile client may be equipped with computation capabilities to maintain a *safe region* [3] with the purpose that movements within the safe region will not affect any query results. Safe regions are bounded by the nearest query rectangles around a mobile client and must be recomputed when certain events take place such as a new query is inserted or a moving object moves beyond its safe region boundary. However, because of the usually simple shape of safe regions (e.g., rectangles or spheres) they can only help to avoid a fraction of unnecessary location updates. In the example shown in Figure 1, rectangles are a set of moving queries and the gray area represents the traditional safe region of moving object p_1. As p_1 moves out of its safe region (in the direction of the arrow) to a new location that is irrelevant to the query results, it issues an unnecessary update because of the limited safe region information. Furthermore, the safe region of a moving object is determined based on its current location. In some cases the server must probe a client object about its current location. In response to such a probe (which consists of one downstream message) the client replies with its current position (one upstream message) and the server determines a new safe region which it sends back to the client (one downstream message). Hence a total of three network messages are sent back and forth between the server and each mobile client.

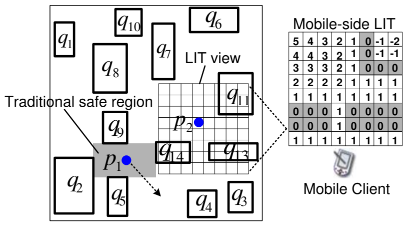


Fig. 1. Traditional Safe regions and LITs.

In contrast, we define a grid-like LIT which provides a moving object with a fine-grained view of the surrounding query locations across the terrain to reduce unnecessary location updates. As an additional advantage, an LIT is determined without referring to the locations of moving objects. Therefore, if a query is inserted, the server can send the new LIT with the added query information to the affected moving objects directly, and only a fraction of the mobile clients that receive the updated LIT must issue location updates back to the server (– namely if they are part of a new query result). Therefore, the number of network messages is reduced to at most two per server location probe. In this demonstration we present a PLUS prototype which includes the tasks that execute on both the server and the mobile units in a road network environment. In particular, the PLUS system possesses the following distinguishing characteristics:

- **Partition-based lazy update continuous query processing.** PLUS performs and visualizes a novel partition-based lazy update continuous query algorithm with a large number of mobile users.
- **Scalability.** PLUS mobile users utilize *Location Information Tables* to index queries to reduce update messages and hence improve system scalability. The PLUS demonstration system computes and displays comparative results of the LIT-based approach and traditional safe region techniques [2], [3] to illustrate performance benefits.
- **Realistic movement on road networks.** The movement of mobile users in PLUS is based on underlying real-world road networks from the TIGER/Line data set. Mobile users in PLUS automatically travel on road segments and the velocity of the movement is determined by the speed limit of each road segment.

II. System Architecture

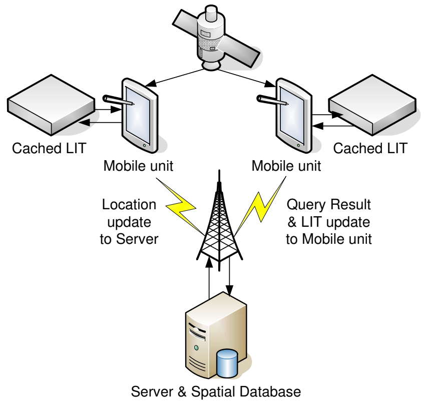

Cached LIT
Mobile unit
Mobile unit
Cached LIT
Location update to Server
Query Result & LIT update to Mobile unit
Server & Spatial Database
Fig. 2. System infrastructure.

Figure 2 illustrates the infrastructure of the PLUS system. We assume that the communication between the centralized server and the mobile units are through cellular or WiMAX networks. The mobile units such as vehicles or hand-held devices (e.g., cell phones and PDAs) are able to provide the server with their positions from a built-in GPS locator. PLUS consists of two major components: the *Server Task Module* and the *Mobile Task Module*. The server module supports an event-driven mechanism to handle all the object requests such as object location updates or new query insertions. Furthermore, the module reduces the data size of a LIT before transmitting it to a moving object in order to fit the information into a single network packet. For the mobile units, we assume that each device has enough computational capabilities and memory space to carry out the required tasks. On top of an underlying road network, a mobile unit can move arbitrarily without exceeding a predetermined maximum speed limit. For the demonstration purpose, a mobile-side application is provided to emulate a real mobile client. To observe the scalability of the system, a set of virtual mobile units is generated. Therefore, the query evaluation is performed on both the virtual and real mobile clients. Figure 3 illustrates the detailed PLUS flow chart. Any updates from query or data objects **(A)** are handled differently by the **ObjUpdate Operation** based on the event types. If the request is a query update **(B)** or a new query registration **(C)**, an **ObjProbe Operation** or a **QurInsert Operation** is performed, respectively, to determine a set of candidate data objects that may become new query answer points. After the data objects are re-indexed **(D)**, the system proceeds to **(E)** which triggers the **LIT Generation and Update** operation. A new mobile-side LIT with an update flag is extracted, compressed and sent to each candidate object **(F)**. The system then performs a **Query Evaluation (G)** to compute new query results after receiving the location updates from the probing objects. In case that the request is a data object update, a new, compressed LIT is sent directly to the object after (object) indexing. On the mobile side, first, a mobile unit retrieves its current location from its GPS tracker **(H)**. Together with the **Query Object Movement Prediction module** to estimate possible surrounding query movements **(I)**, the mobile unit determines whether a location update is necessary through the **Location Update Check** procedure. In the following sections, we describe the details of LITs and the aspects of the server and mobile task modules in PLUS.

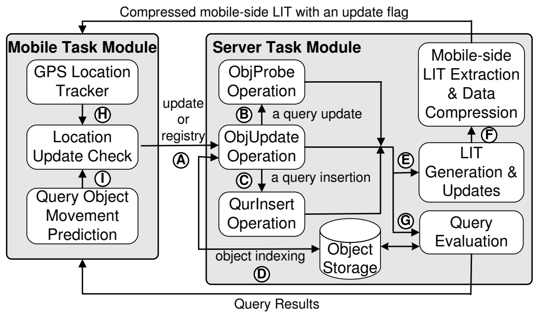


Fig. 3. PLUS system flow chart.

A. Location Information Tables

In PLUS, we use a server-side LIT (LIT_{serv}) that covers the entire service space and a mobile-side LIT (LIT_{mov}) for each moving object. The LIT_{serv} can be built on top of any existing data indexing structures (e.g., grids or R-trees) and it is maintained at the server and updated when one of the following two events happen: (1) an existing query changes its location or (2) a new query is registered with the system.

A mobile-side LIT is a subset table extracted from the latest LIT_{serv}. Furthermore, each moving object maintains (i.e., updates) the mobile-side LIT locally after receiving it from the server. The LIT_{mov} is synchronized with the LIT_{serv} again when a specific event occurs. To generate a LIT, we first partition the space and mark the cells zeros that are covered by query boundaries as shown in the example of Figure 1. We call these areas *border zones*. Each of the rest of the cells is assigned an integer number (a LIT value), which represents the minimal linear distance in cells from itself to its nearest query boundary. For handling dynamic queries, each mobile unit periodically performs a *mobile-side LIT revision* to capture possible query movements. The queries that define the border zones might move to their surrounding cells in any direction as time proceeds. Therefore, for simplicity, a mobile unit can predict the possible query movements by expanding the border zone outwards by the length of the maximum moving distance for every time instance. The area covered by the expanding border zone is called a *prediction zone*, serving the same role as a border zone that triggers a location update when a moving object enters it. Any object moving into a prediction zone might become part of the query results.

B. Server Task Module

The **Server Task Module** supports an event-driven mechanism to handle a request from a mobile unit which it dispatches to perform a specific operation based on the event type. There are four operations implemented in the system: *Object Update* (object location update event), *Object Position Probing* (query update event), *Query Insertion* (query insertion event), and *Data Compression*.

• An ***Object Update*** **(ObjUpdate) Operation** re-indexes the location of a data or query object on the server based on its transmitted coordinates.
• An ***Object Position Probing*** **(ObjProbe) Operation** triggers smart on-demand location probes to reduce the number of location update messages when the request is a query update. The operation determines a set of objects and sends them the latest LIT encoded with an *update flag* to notify the mobile objects to respond with their current locations.
• A ***Query Insertion*** **(QurInsert) Operation** determines a set of affected objects without causing any missed location updates when a new query is inserted. A new LIT with a "delay" update flag is then sent to each affected object, which replies to the server with its current position only when objects fall into the query boundary. As a result, only a fraction of the affected objects must issue location updates.
• A ***Data Compression*** **Operation** compresses a mobile-side LIT to reduce the data stream size. While a mobile-side LIT provides more detailed query boundary information than a safe region, the data transmission of a potentially large LIT needs to be broken into more packets which may adversely affect performance. In the PLUS demonstration system, we use the Internet standard for the largest data packet payload size (MTU) equal to 1500 bytes. We apply three consecutive lossless data compression methods: delta encoding, run-length encoding (RLE) and Huffman encoding. First, we de-correlate the LIT values by subtracting pairs of adjacent LIT numbers. Second, RLE is utilized to take advantage of the large amount of spatial redundancy in a LIT and we use a Hilbert curve as the data scanning path along which we count repeated numbers. Finally, we performed Huffman encoding which is based on the frequency of occurrence of a data item and uses a lower number of bits to encode the data that occur more frequently.

To complete, the process invokes the mobile-side LIT extraction. The compressed LIT, combined with an update flag is sent to the set of moving objects determined by the previous operation. Finally, the query evaluation procedure is executed to retrieve the query answers, and the result is sent back to the query objects.

C. Mobile Task Module

The **Mobile Task Module** includes two major functions: (1) *query object movement prediction*, and (2) *location update check* to determine whether a location update is necessary. The first function predicts the possible movements of queries by updating the LIT cells to prediction zones which might be covered by nearby moving queries as time proceeds. Next, each mobile unit determines a location update by referring to its current location detected by the GPS location tracker and the revised mobile-side LIT. If an object steps into a query boundary zone or a prediction zone indexed in its LIT, it issues a location update.

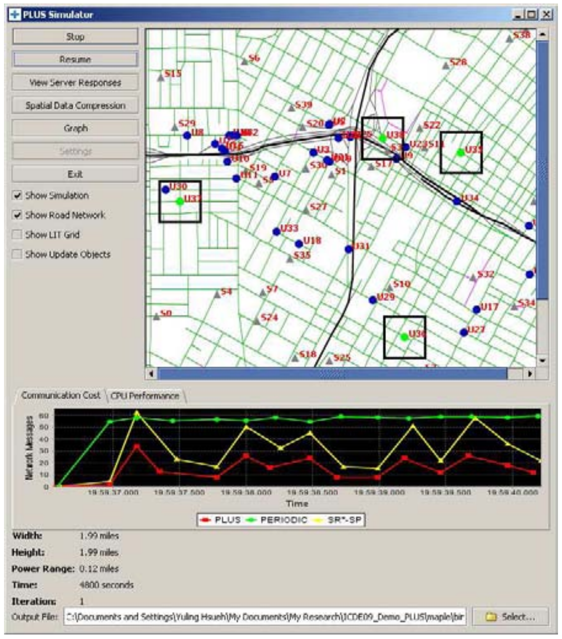
III. Simulation Data Sets

To produce realistic mobile movements, PLUS imports the road network from the TIGER/Line [4] street vector data set available from the U.S. Census Bureau and then integrates the road segments (e.g., freeways, primary highways, secondary and connecting roads, and rural roads) into a complete road network. Mobile users in PLUS automatically travel on road segments and the velocity of the movement is determined by the speed limit of each road segment. The movements of the mobile objects are generated continuously on top of the road network within a given time period and the speed limit for any moving object is in the range of 35 to 90 miles per hour (MPH). The mobility rate (the percentage of objects that move within a time step) is selectable in the range from 0% to 100%. The query objects are randomly chosen from the set of moving objects and the number of queries can be specified as being launched from 0% to 100% of the moving objects.

IV. System Demonstration

A. PLUS Server-side System Interface

Figure 4 shows the PLUS server-side interface. We also implemented a well-known safe region update scheme [3] using a sphere shape (SR*-SP for short) and a periodic scheme (PERIODIC). The top panel on the interface visualizes the server view of the object movements on the real-world road

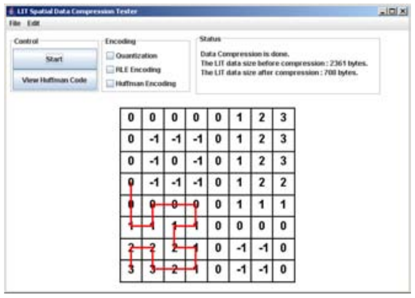


[✓ Show Simulation] [✓ Show Road Network] Show LIT Grid, Show Update Objects

Width: 1.99 miles
Height: 1.99 miles
Power Range: 0.12 miles
Time: 4800 seconds
Iteration: 1
Output File: C:\Documents and Settings\Yuling Hsueh\My Documents\My Research\ICDE09_Demo_PLUS\maple\bir

Fig. 4. PLUS main interface.

segments in the Los Angeles area. In the first tabbed panel (the lower part of the frame), the user can compare the location update frequency of PLUS, SR*-SP, and PERIODIC by observing a dynamic line chart which shows the number of location updates for the three approaches over time. The CPU performance corresponding to the three approaches is shown in the second tabbed panel. Mobile units are distinguished based on their types (a query or moving object) and status (e.g., issuing location updates) by marking them in different colors. In addition, a query is shown with a rectangle as the query boundary. For example, query no. 37 contains one answer point (no. 30). PLUS is controlled via user-selectable parameters (e.g., maximum speed, number of query and data objects). In particular, a user can specify the number of mobile-side and server-side LITs to explore the impact of the LIT size. Figure 5 illustrates the interface of the spatial data compression module (using a Hilbert curve as the data scanning path) for a LIT. The user can upload an existing LIT and execute the data compression methods selected from the provided options. The resulting compression reduction is shown in the status area.



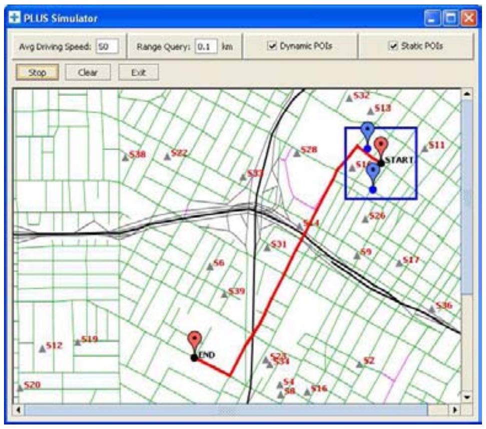
Data Compression is done.
The LIT data size before compression : 2361 bytes.
The LIT data size after compression : 708 bytes.

0	0	0	0	0	1	2	3
0	-1	-1	-1	0	1	2	3
0	-1	0	-1	0	1	2	3
0	-1	-1	-1	0	1	2	2
0	0	0	0	0	1	1	1
1	1	1	1	0	0	0	0
2	2	2	1	0	-1	-1	0
3	3	2	1	0	-1	-1	0

Fig. 5. The data compression module.

B. PLUS-client: Client-side System Interface

We also provide a mobile-side interface (PLUS-client) shown in Figure 6 for a mobile user to interact with PLUS. The PLUS-client is a web-based application where the geographic region generated from the TIGER/Line is the same as the road network imported to the server-side PLUS. The mobile client can specify the starting and ending points of a path and then we adopt the A^* search algorithm to find the travel routes for the client. The average moving speed, query range and the trajectory information are sent to PLUS. The rectangle in the figure represents the query boundary and the points of interests shown in the rectangle are the query answer points sent back from PLUS.



Avg Driving Speed: 50
Range Query: 0.1 km
[✓ Dynamic POIs] [✓ Static POIs]

Fig. 6. PLUS-client interface.

V. Conclusions

The PLUS demonstration system implements both server-side and mobile-side aspects of a location-based service that illustrates the partition-based lazy updates approach for continuous queries. PLUS shows the server view with the latest updated mobile-unit locations integrated on a road network with realistic speed limits. The PLUS system continuously compares the performance with the traditional periodic approach and the safe region update approach in terms of the number of network messages which are simultaneously shown in a curve chart on the system's main interface. The visualization of the data compression methods adopted by PLUS is implemented with an interface to test the data size reduction by specifying three optional data compression methods. A mobile-side application is also implemented to interactively observe the system processes.

VI. Acknowledgments

This research has been funded in part by NSF grants CNS-0831502 (CT), IIS-0534761, NUS AcRF grant WBS R-252-050-280-101/133.

References

[1] Y.-L. Hsueh, R. Zimmermann, H. Wang, and W.-S. Ku. Partition-based lazy updates for continuous queries over moving objects. In *GIS '07*. ACM, 2007.
[2] H. Hu, J. Xu, and D. L. Lee. A generic framework for monitoring continuous spatial queries over moving objects. In *SIGMOD Conference*, pages 479–490, 2005.
[3] S. Prabhakar, Y. Xia, D. V. Kalashnikov, W. G. Aref, and S. E. Hambrusch. Query Indexing and Velocity Constrained Indexing: Scalable Techniques for Continuous Queries on Moving Objects. *IEEE Trans. Computers*, pages 1124–1140, 2002.
[4] TIGER/Line. http://www.census.gov/geo/www/tiger/.