Elaps: An Efficient Location-Aware Pub/Sub System

Long Guo [#1], Lu Chen [#2], Dongxiang Zhang [#3], Guoliang Li [†4], Kian-Lee Tan [#5], Zhifeng Bao [*6]

[#] *National University of Singapore* [†]*Tsinghua University* [*]*University of Tasmania*

{[1]guolong,[2]chenlu,[3]zhangdo,[5]tankl}@comp.nus.edu.sg, [4]liguoliang@tsinghua.edu.cn, [6]baoz@utas.edu.au

***Abstract*—The prevalence of social networks and mobile devices has facilitated the real-time dissemination of local events such as sales, shows and exhibitions. To explore nearby events, mobile users can query a location based search engine for the desired data. However, operating under such a pull based model means that users may miss interesting events (because no explicit queries are issued) or processing/communication overheads may be high (because users have to continuously issue queries).**

In this demo, we present Elaps, an efficient location-aware publish/subscribe system that can effectively disseminate interesting events to moving users. Elaps is based on the push model and notifies mobile users instantly whenever there is a matching event around their locations. Through the demo, we will demonstrate that Elaps is scalable to a large number of subscriptions and events. Moreover, Elaps can effectively monitor the subscribers without missing any event matching, and incur low communication overhead.

I. Introduction

With the prevalence of social networks and map applications, more and more local events are published and disseminated online. To explore nearby events, users tend to pull information they are interested from news feed or locational search engines. However, many local events, such as sales, shows and exhibitions, have a short life time and the pull model poses a high chance of missing interesting events. Another method for users to explore nearby events is to query locational search engines which support continuously moving queries [1], [2], [3] until they get some results they want. However, [1], [2], [3] only deal with existing events and ignore the arrival of new events. Thus, users may still miss newly arrived interesting events around them. To handle the case where new events are published continually, a push model should be used. Users submit subscriptions representing their interest and are notified instantly whenever there is an event matching their interest near them. However, existing push-based location-aware pub/sub systems [4], [5], [6], [7] cannot handle the scenario where users are moving all the time, since they assume users specify a static location and send matching events that are near to the location to the users.

To improve the effectiveness of interesting event dissemination to moving users, we build an efficient location-aware publish/subscribe system, named Elaps. Our system is based on the push model: mobile users are instantly notified whenever there is a matching event near them. Compared to [4], [5], [6], [7], Elaps has two distinguished features. First, it allows users to specify their interests with Boolean expressions, which provides better flexibility and expressiveness in shaping an interest than keyword subscription [5], [6], [7]. Second, it continuously monitors users' locations and sends nearby notifications in real time. To the best of our knowledge, Elaps is the first location-aware pub/sub system that takes into account moving subscribers as well as dynamic subscriptions and events.

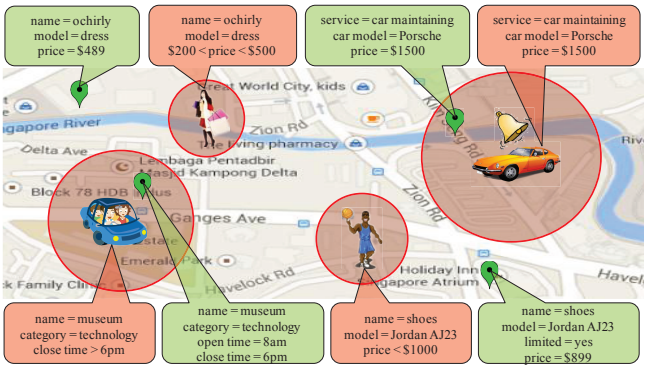

Fig. 1. A working senario of Elaps

Fig. 1 illustrates a working scenario of Elaps. In the application, the subscribers with mobile devices are moving objects. A subscription is represented in the form of Boolean expression. For example, if a user is interested in Jordan basketball shoes, he can use the following Boolean expression to model his interest:

$$(\text{name=shoes} \wedge \text{model=Jordan AJ23} \wedge \text{price} < \$1000)$$

To specify the locational constraint, a subscriber can set a notification range[1] so that events lying inside the circle are considered as candidates. For example, the circles in Fig. 1 represent the notification regions of different users. When a user moves, the notification region moves along. On the publisher side, an event is published at a location. If a shoe shop is on sale, then the location of the shop is the event location. Our system allows any locational event to be published, including sales, promotions, parties, exhibitions and other social events. As long as an event matches a subscriber's interest and within his notification region, the user will be notified instantly. In Fig. 1, the user looking for car maintenance receives a notification because there is a matching car service centered within his notification region. However, the family interested in museum will not be notified although there is a museum within his notification region - the closing time does not match.

In this demo, we showcase Elaps' scalability with respect to the number of subscriptions and events. We also demonstrate

[1]The range can be distance-based (i.e., 1 km) or time-based (i.e., 5 minutes, which can be translated to distance-based according to the speed). In this demo, we use distanced-based range for simplicity.

how Elaps can effectively monitor the subscribers without missing any matching notification and yet incurring low communication overhead. To improve the matching performance, Elaps employs novel indexes for locational subscription matching and event matching. To reduce the commmunication overhead, Elaps utilizes a safe region and impact region for each subscriber. In addition, a cost-based approach is adopted to minimize communication cost. Based on the novel cost model, Elaps supports two effective strategies iGM and idGM for the safe region and impact region construction.

The rest of the paper is organized as follows. Section II introduces the spatial subscription and event model as well as spatial subscription/event matching. Section III introduces the architecture and design of Elaps. Finally, Section IV presents the demonstration details.

II. Problem Definition

Elaps has two types of roles in the system. A subscriber represents his interest in the form of spatial subscription and is modeled as a moving object. A publisher is associated with a geo-location and publishes spatial events. In the following, we introduce our spatial subscription and event model as well as spatial subscription/event matching.

Spatial Subscription. A spatial subscription extends a boolean expression with a notification region $\mathcal{O}$. In this paper, we model a boolean expression as a conjunction of predicates. Each predicate is determined by three elements: an attribute A, an operator f_{op} and an operand $\overline{o}$. It accepts an input value x and the output is a boolean value indicating whether the operator constraint is satisfied or not: $P^{(A,f_{op},\bar{o})}(x) \rightarrow \{0,1\}$. SystemPS can support relational operators $<, \leq, =, >, \geq, [\ ]$ [2] and set operators $\in, \notin$. As mentioned, the notification region $\mathcal{O}$ is a circle centered at user's current location with radius r. Formally, a spatial subscription s is defined over $|s|$ predicates and an notification region $\mathcal{O}$:

$$s : P_1^{A,f_{op},\overline{o}}(x) \wedge P_2^{A,f_{op},\overline{o}}(x) \wedge ... \wedge P_{|s|}^{A,f_{op},\overline{o}}(x) \wedge \mathcal{O}$$

For example, a user interested in Samsung TQ can submit a subscription like *(brand=samsung $\wedge$ size>50)$\wedge$ r=1km $\wedge$ lat=1.28 $\wedge$ lng=103.8)*. Note that the location is detected automatically. Hereafter, we use spatial subscription and subscription interchangeably.

Spatial Event. A spatial event e contains $|e|$ tuples and a location loc: $e : (A_1 = \overline{o}_1) \wedge (A_2 = \overline{o}_2) \wedge ... \wedge (A_{|e|} = \overline{o_{|e|}}) \wedge loc$, where A_i is the attribute and $\bar{o}_i$ is the associated value or operand. For example, a Samsung TQ promotion event can be represented by *(brand=samsung $\wedge$ size=55 $\wedge$ 3D=yes $\wedge$ lat=1.28 $\wedge$ lng=103.8)*.

Spatial Subscription Match. The match between a spatial boolean expression s and an event e consists of two aspects: boolean expression match and spatial match, as defined below.

Definition 1 (Boolean Expression Match): A boolean expression match is satisfied if for each predicate P in s, P is satisfied by a tuple $A_i = \overline{o}_i$ in e. We use $s \sim_b e$ to denote a boolean expression match and say s *be-matches* e.

[2]In this case, the operand $\overline{o}$ contains two values: $\overline{o}.l$ and $\overline{o}.r$

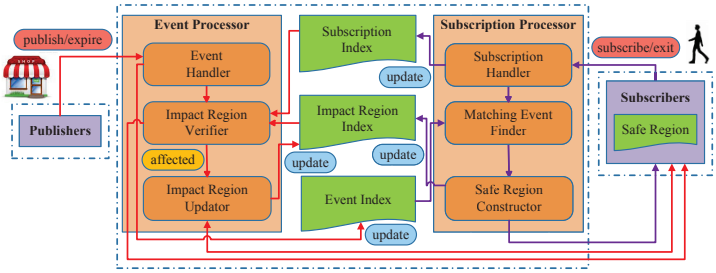

Fig. 2. System architecture of Elaps

Definition 2 (Spatial Match): We say there is a spatial match between s and e, denoted by $s \sim_s e$, if the location loc of e is inside the notification region $\mathcal{O}$ of s.

Definition 3 (Match): We say a subscription s matches an event e, denoted by $s \sim e$, if $s \sim_b e$ and $s \sim_s e$.

III. System Architecture

The system architecture of Elaps is shown in Fig. 2. It is designed to efficiently handle the following messages.

A. Subscription arrival/expiration

In Elaps, subscriptions are continually submitted by the subscribers. A subscription is associated with a valid life span and expires when the user is no longer interested in receiving matching events. Such kind of messages are handled by the *Subscription Processor*. When a new subscription arrives, the *Subscription Handler* inserts the subscription to the *Subscription Index*. After that, the *Matching Event Finder* searches the *Event Index* for events matching this subscription. The user will be notified if there is a match within his notification region.

To guarantee the user be notified in real time, Elaps is equipped with a novel index named **BEQ-Tree** used by the *Subscription Index*. BEQ-Tree adopts a two-layer partitioning scheme for efficient Boolean expression matching and spatial matching. In the first layer, the events are partitioned based on the spatial attribute. We use a quadtree-like data structure to organize the locations of events. In the second layer, the events are partitioned based on the attributes. We use the inverted list to group the events and each inverted list is sorted by the attribute value of the event.

B. User location monitoring

In our data model, the subscribers are moving objects and we need to monitor their locations so as to support instant notification. To save the communication cost, we construct a safe region for each subscriber and derive an impact region from the safe region using the *Safe Region Constructor*. The impact region serves as a filtering mechanism so that only newly published events located within the impact region has a probability to change the safe region. Then we insert the impact region into the *Impact Region Index* and send the safe region to the user. As long as the subscribers stay within his safe region and no event appears in the impact region, the subscribers are safe to disconnect from the server without

missing any matching event. The safe region will be updated as the user moves out of it or a new event arrives at the impact region.

The dynamic environment in Elaps where events are published rapidly poses a new challenge to the construction of safe region. Thus, Elaps deploys two effective strategies iGM and idGM to construct the safe region and impact region incrementally based on a novel cost model.

Cost Model. Although safe region has been shown to be effective in reducing communication overhead for location update, we argue that maximizing the safe region could not optimize the system performance because *a larger safe region can reduce the communication cost for location update of subscribers, but it increases the size of impact region and there is a higher probability that a new event occurs in the impact region and triggers an update request for a new safe region*. Traditional safe region construction methods do not work well in the dynamic environment where events are published rapidly, since they ignore safe region update notification from server to subscriber. Thus, Elaps employs a new cost model for safe region construction which can capture the tradeoff between safe region size and communication overhead and minimize the communication overhead:

$$b_m(\mathcal{R}) = \frac{t_s(\mathcal{R})}{t_i(\mathcal{R})}$$

where t_s measures the elapsed time before a subscriber exits his safe region and t_i measures the elapsed time before a new matching event arrives at his impact region. Our goal is to minimize the communication overhead which can be considered as maximizing the expected elapsed time before next communication occurs. The expected elapsed time is denoted by $f_{obj}(\mathcal{R})$ and used as the objective function to maximize.

iGM algorithm. Based on the relation between $f_{obj}(\mathcal{R})$ and $b_m(\mathcal{R})$, to maximize $f_{obj}(\mathcal{R})$, we should build a safe region that satisfies $b_m \leq 1$ and has an area as large as possible. Our incremental grid-based method, or iGM for brevity, partitions the space into grid cells. To calculate the safe region for a moving subscriber, the method starts from the cell containing the subscriber and expands to neighboring cell with the minimum distance to the subscriber at each iteration. The expansion terminates when there is no more cell satisfying $b_m \leq 1$ when they are included in the safe region.

idGM algorithm. Since most smartphones are equipped with sensors for direction detection, Elaps also incorporated a direction-aware iGM, named idGM, to take advantage of such information and further reduce the communication overhead. The main idea of idGM is to introduce a preference score τ composed of the distance preference and direction preference. Then idGM expands the safe region incrementally based on the preference score τ rather than the distance only. Thus, idGM constructs a direction-aware safe region which takes a longer period before the subscriber exits his safe region.

C. Event arrival/expiration

In Elaps, events are continually published by the publishers which are also associated with a life span. Such kind of messages are handled by the *Event Processor*. When an event e arrives, the *Event Handler* first inserts it into the *Event Index*. Then, we need to detect which subscriptions are affected by e using the *Impact Region Verifier*. The *Impact Region Verifier* first uses the *subscription index* to find the subscriptions that be-match e, and then uses the *impact region index*, which is maintained as a hash table, to find those be-matching subscriptions whose impact region contains e. Finally, e is sent to the corresponding affected subscribers. There are two cases that require actions: (i) e matches a subscription s ($e \sim s$). Then, the subscriber will be notified of e and its safe region remains the same. (ii) e be-matches a subscription s ($e \sim_b s$). Then the subscriber updates the safe region himself. And the server updates the impact region accordingly using the *Impact Region Updater*.

To support efficient event matching over a large quantity of subscriptions, we adopt our **OpIndex** [8] as the *Event Index*. OpIndex adopts a three-level partition mechanism. We first select a pivot attribute for each subscription. The subscriptions with the same pivot attribute are grouped. The second and third level partitions are based on operator and attribute name respectively. The predicates with the same operator are clustered so that specifically designed index can be applied to support various operators and enhance the expressiveness. The operands in the inverted lists are sorted and the sequential memory access pattern can facilitate the cache-concious query processing. More details of OpIndex can be found in [8].

IV. Demonstration

We demonstrate our Elaps system in four scenarios. We have implemented a desktop prototype to illustrate the functionality and performance of Elaps, shown in Fig. 3. We also show the mobile device interface in the prototype.

Scenario 1 - Subscriber Side

This scenario shows how a subscriber submits a subscription. In the mobile device interface for the subscriber, the subscriber specifies the predicate of the Boolean expression one by one in the top text box and submits the predicate to the bottom text area by clicking the "Add" button. The subscriber can also specify the radius of his notification region. Finally, the subscriber submits his subscription by clicking the "Submit" button. The subscriber will be notified instantly whenever there is a matching event located within his notification region.

In order to illustrate the functionality of Elaps, the prototype can simulate the moving behavior of the subscribers. For each subscriber, we can choose different moving patterns for this subscriber, such as walking or driving. Each moving pattern corresponds to a list of speeds to choose from. Then we can specify the starting location and destination location for this subscriber by using the text box or clicking on the map. Finally, the subscriber is simulated on the map by clicking on the "Submit" button. For example, as shown in Fig. 4,

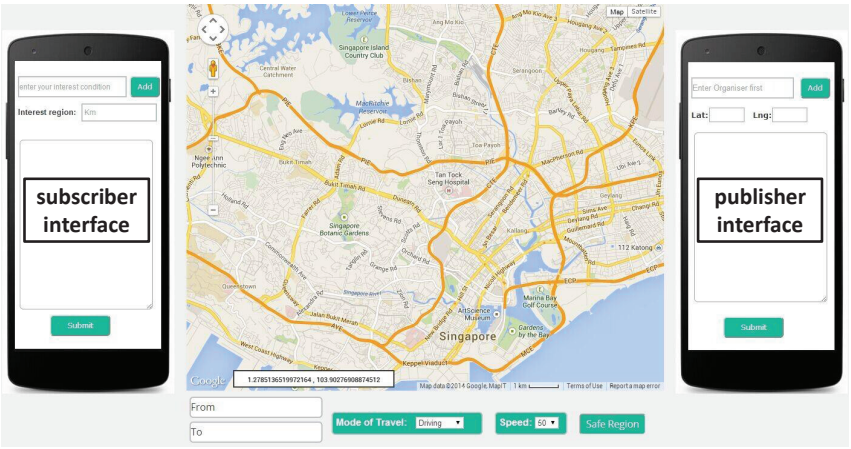

Fig. 3. Prototype of Elaps

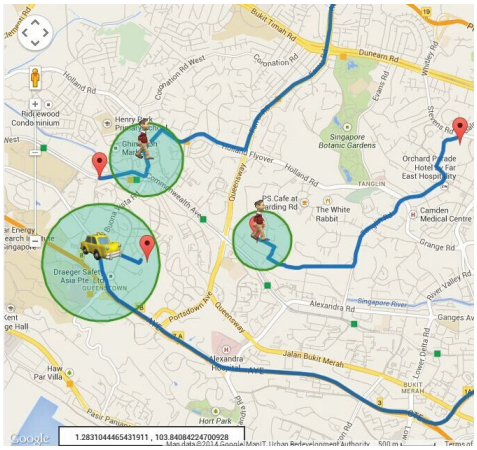

Fig. 4. Simulating moving subscribers in Elaps

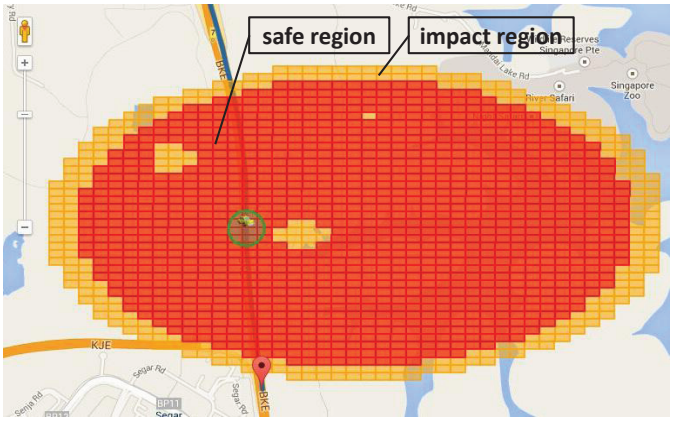

Fig. 5. Safe Region and Impact Region Demonstration

we generate three moving subscribers with different moving speeds and notification regions.

Scenario 2 - Publisher Side

This scenario shows how a publisher publishes an event. In the mobile device interface for the publisher, the publisher specifies the event in the top text box and provides the address of the event. Besides that, we can also add a new event to Elaps by clicking on the map. With this functionality, we want to show the scenario when a new matching event arrives at the notification region of a subscriber and the subscriber is notified.

Scenario 3 - Safe Region and Impact Region

This scenario illustrates the underlying safe region construction method of Elaps. In Elaps, subscribers are moving all the time and can be notified instantly whenever there is a matching event located within their notification regions. With the help of the safe region and impact region, the computation and communication cost can be reduced significantly. For each subscriber, we can show his safe region and impact region by clicking on the "Safe Region" button, as shown in Fig. 5. In addition, before we click on the "Safe Region" button, we can choose the construction method (i.e., iGM or idGM). As a result, for each subscriber, we can see different safe regions and impact regions constructed by different methods. Besides that, we will show the functionality of impact region by adding an event into the impact region. In this case, the safe region and impact region will be reconstructed. By showing the safe region and impact region, audiences can better understand the roles of these two kinds of regions.

Scenario 4 - Performance Demonstration

We use two real-world data sets for the demonstration. The first data set is crawled from Foursquare and the second data set is collected from Twitter. We consider event locations within Singapore. The event locations follow the Gaussian distribution. These two datasets are used as the existing events in Elaps. The demonstration will show that Elaps is able to disseminate matching events to users in real-time.

ACKNOWLEDGMENT

This work is funded by the NExT Search Centre (grant R-252-300- 001-490), which is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Program Office.

REFERENCES

[1] D. Wu, M. L. Yiu, C. S. Jensen, and G. Cong, "Efficient continuously moving top-k spatial keyword query processing," in *ICDE*, 2011, pp. 541–552.

[2] W. Huang, G. Li, K.-L. Tan, and J. Feng, "Efficient safe-region construction for moving top-k spatial keyword queries," in *CIKM*, 2012, pp. 932–941.

[3] L. Guo, J. Shao, H. Aung, and K.-L. Tan, "Efficient continuous top-k spatial keyword queries on road networks," *GeoInformatica*, pp. 1–32, 2014.

[4] J. Bao, M. Mokbel, and C.-Y. Chow, "Geofeed: A location aware news feed system," in *ICDE*, 2012, pp. 54–65.

[5] G. Li, Y. Wang, T. Wang, and J. Feng, "Location-aware publish/subscribe," in *KDD*, 2013, pp. 802–810.

[6] L. Chen, G. Cong, and X. Cao, "An efficient query indexing mechanism for filtering geo-textual data," in *SIGMOD*, 2013, pp. 749–760.

[7] L. Chen, Y. Cui, G. Cong, and X. Cao, "Sops: A system for efficient processing of spatial-keyword publish/subscribe." 2014, pp. 1601–1604.

[8] D. Zhang, C.-Y. Chan, and K.-L. Tan, "An efficient publish/subscribe index for ecommerce databases," *PVLDB*, vol. 7, no. 8, pp. 613–624, 2014.