Location-Aware Pub/Sub System: When Continuous Moving Queries Meet Dynamic Event Streams

Long Guo [#1], Dongxiang Zhang [#2], Guoliang Li [†3], Kian-Lee Tan [#4], Zhifeng Bao [*5]
[#]School of Computing, National University of Singapore, Singapore
[†]Department of Computer Science, Tsinghua University, Beijing, China
[*]School of Computer Science & Information Technology, RMIT University, Melbourne, Australia
{[1]guolong,[2]zhangdo,[4]tankl}@comp.nus.edu.sg, [3]liguoliang@tsinghua.edu.cn, [5]zhifeng.bao@rmit.edu.au

ABSTRACT

In this paper, we propose a new location-aware pub/sub system, Elaps, that continuously monitors moving users subscribing to dynamic event streams from social media and E-commerce applications. Users are notified instantly when there is a matching event nearby. To the best of our knowledge, Elaps is the first to take into account continuous moving queries against dynamic event streams. Like existing works on continuous moving query processing, Elaps employs the concept of safe region to reduce communication overhead. However, unlike existing works which assume data from publishers are static, updates to safe regions may be triggered by newly arrived events. In Elaps, we develop a concept called *impact region* that allows us to identify whether a safe region is affected by newly arrived events. Moreover, we propose a novel cost model to optimize the safe region size to keep the communication overhead low. Based on the cost model, we design two incremental methods, iGM and idGM, for safe region construction. In addition, Elaps uses boolean expression, which is more expressive than keywords, to model user intent and we propose a novel index, BEQ-Tree, to handle spatial boolean expression matching. In our experiments, we use geo-tweets from Twitter and venues from Foursquare to simulate publishers and boolean expressions generated from AOL search log to represent users intentions. We test user movement in both synthetic trajectories and real taxi trajectories. The results show that Elaps can significantly reduce the communication overhead and disseminate events to users in real-time.

Categories and Subject Descriptors

H.2 [**Database Management**]: Database applications;
H.2.8 [**Database applications**]: Spatial database and GIS

Keywords

pub/sub; continuous moving queries; dynamic event streams

1. INTRODUCTION

The prevalence of social networks and mobile devices has facilitated the real-time dissemination of local events such as sales, shows and exhibitions. To avoid missing interesting events in the neighborhood, various location-aware pub/sub systems have been proposed. These fall into two categories: they either focused on how to handle incoming event streams efficiently by assuming users' locations are static [1, 2, 3, 4]; or they attempted to process continuous moving subscriptions against a static event dataset [5, 6, 7, 8, 9, 10]. None of them can really support subscriptions from mobile users moving all the time against spatial events that are continuously published by local businesses.

In this paper, we propose a new location-aware pub/sub system, Elaps, that continuously monitors moving users subscribing to dynamic event streams from social media and E-commerce applications. Users are notified instantly when there is a matching event nearby. Unlike existing pub/sub systems, Elaps uses boolean expressions, which are more expressive than keywords, to model user intent. This means users can subscribe to structured, semi-structured and unstructured data.

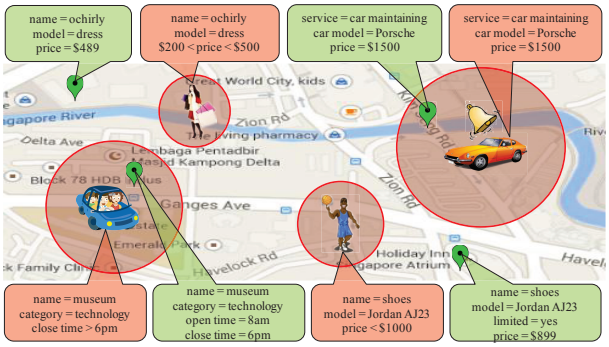


Figure 1: A working scenario of Elaps

Fig. 1 illustrates a working scenario of such a system. Here, the subscribers with mobile devices are the moving objects, and a subscription is represented in the form of a boolean expression. For example, if a user is interested in Jordan basketball shoes, he can use a boolean expression to model the interest: (name=shoes $\wedge$ model=Jordan AJ23 $\wedge$ price $<$ $1000). Note that pub/sub systems based on keyword subscription cannot support numeric attribute matching such as price $<$ $1000. To specify the locational matching constraint, a subscriber can set a notification radius 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 continuously monitors the moving subscribers and notify them once there is a matching event in their circles.

SIGMOD'15, May 31–June 4, 2015, Melbourne, Victoria, Australia.

http://dx.doi.org/10.1145/2723372.2746481.

	Location-aware news feed		Location-aware pub/sub			Continuous spatial queries		
	GeoFeed[1]	MobiFeed[11]	R^t-tree[2]	IQ-tree[3]	CLCB[4]	KNN/Range	Spatial keyword	Elaps
continuous moving queries	✗	✗	✗	✗	✗	✓	✓	✓
dynamic event streams	✓	✓	✓	✓	✓	✗	✗	✓
matching semantic	-	-	keyword	keyword	BE	-	keyword	BE

Table 1: Comparison of existing location-aware pub/sub system

To meet the above desiderata, our system is designed to tackle the two main challenges:

a) how to effectively process continuous moving subscriptions against dynamic event streams. Safe region has been widely used to reduce communication cost for continuous moving query processing [5, 6, 7, 8, 9, 10]. The intuition behind the notion of safe region is that if there are no matching events nearby, the users are safe to disconnect from the server and do not need to periodically update their locations. However, we observed that the safe region techniques used in [5, 6, 7, 8, 9, 10] fail to work effectively when dynamic events are considered. This is because newly arrived events can trigger new communication needed to update the safe regions, besides the communication incurred by location updates. Moreover, these two types of communication have conflicting requirements on the size of safe regions. Therefore, there is a need to reconsider how best to exploit safe regions.

b) how to support symmetric boolean expression matching with spatial constraints between subscribers and publishers. In other words, when a subscriber arrives, we need to search in the event database for matching events within the specified notification radius; when a new event arrives, we need to search in the subscriber database to find matching subscribers and notify them.

More specifically, our technical contributions are summarized as follows:

- We optimize the design and processing of safe regions in several ways. First, given a safe region, we derive its *impact region*. The impact region is a novel concept used to identify if its corresponding safe region is affected by newly arrived events. Second, we propose a cost-based approach to determine the optimal safe region size to keep the communication overhead low. Our cost model considers the communication cost incurred by location updates as well as that incurred by event arrival. Third, based on the cost model, we design two new schemes, iGM and idGM, to incrementally construct safe regions.

- We propose a new index named BEQ-Tree which integrates Quadtree [12] with boolean expressions seamlessly to support spatial boolean expression matching and safe region construction efficiently.

Moreover, we conduct comprehensive experiments using real datasets to evaluate the system performance. We use geo-tweets from Twitter and venues from Foursquare to simulate publishers and boolean expressions generated from AOL search log to represent users intentions. We test user movement in both synthetic trajectories and real taxi trajectories. The results show that our proposed iGM and idGM can reduce the communication overhead by 10 times. Also, our proposed index handles spatial boolean expression matching significantly faster than the competing methods.

The rest of this paper is organized as follows. We first review various location-based pub/sub systems and boolean expression matching and give a problem statement in Section 2. We then propose how to handle continuous moving queries against dynamic event streams in Section 3. We further introduce how to handle spatial boolean expression matching in Section 4. Based on the techniques proposed in Section 3 and Section 4, we present the system framework of Elaps in Section 5. Finally, we evaluate our system performance in Section 6 and conclude the paper in Section 7.

2. RELATED WORK

In Elaps, our primary goal is to build a pub/sub system that caters for both continuous moving subscribers and dynamic event streams from publishers, and our secondary goal is to support a more expressive event matching semantic than pure keywords. Thus, in this section, we first highlight the novelty of Elaps as compared with existing location-aware pub/sub systems and then present the related work about boolean expression matching. Finally we give a problem statement.

2.1 Location-aware Pub/sub

To clearly distinguish Elaps from other works in the literature, we first present a taxonomy of the existing location-aware pub/sub systems and our proposed Elaps in three aspects: support for continuous moving queries, support for dynamic event streams and event matching semantic (see Table 1).

Location-aware pub/sub system. A location-aware pub/sub system sends matching geo-tagged events to the corresponding subscribers. Compared to existing location-aware pub/sub systems [2, 3, 4], Elaps has two distinguishing features, as shown in Table 1. First, it continuously monitors users' locations and sends nearby notifications in real time, while [2, 3, 4] assume users' locations are static. Second, it allows users to specify their interests with boolean expressions, which provides better flexibility and expressiveness in shaping an interest than keyword subscription in [2, 3].

Continuous spatial queries. Another problem that is closely related to our research is the processing of continuous spatial queries such as continuous KNN/range queries [5, 6, 7] and continuous spatial keyword queries [8, 9, 10]. Given a moving query, existing works have developed techniques to continuously return a set of objects satisfying the spatial constraint [5, 6, 7] or the combined spatial and textual constraints [8, 9, 10]. To reduce the communication overhead, these methods typically require users to update their locations only when they move out of their safe regions. Moreover, users within the safe regions can safely disconnect from the server as long as there is no matching event in their neighborhood. Since we focus on range query, the safe region construction methods for spatial keyword query in [8, 9, 10] cannot be applied. Besides, those proposed for continuous knn/range queries [5, 6, 7] assume the publisher events are static, so they fail to solve our problem either (Please refer to Section 3 for a detailed justification). Lastly, these methods allow users to search for relevant events by keyword subscriptions while Elaps uses a more expressive boolean expression to model user intent.

Location-aware news feed system. We also note that location-aware news feed systems enable mobile users to share geo-tagged user-generated messages. In GeoFeed [1], users retrieve geo-related message updates from either their social friends or social media. It differs from Elaps in that: (1) GeoFeed is pull-based, which poses

a high chance of missing interesting events, (2) users are static objects, (3) users cannot customize the messages they are interested in explicitly. MobiFeed [11] extends GeoFeed to support mobile users. Instead of monitoring the location of moving users from time to time like Elaps, it predicts the potential next locations for a moving user in advance and pushes the messages around these locations to the user. Thus, it cannot support continuous moving queries; moreover, there is no guarantee that users will not miss any matching event.

Therefore, to our knowledge, Elaps is the first location-aware pub/sub system that takes into account continuous moving queries as well as dynamic event streams.

2.2 Boolean expression matching

There have been several studies on efficient *event matching* over a large quantity of subscriptions [13, 14, 15]. Whang et al. [14] proposed k-index which partitions the subscriptions into inverted lists, whose key is a triple of subscription size, attribute name and attribute value. To further improve efficiency and expressiveness, Sadoghi and Jacobsen proposed the BE-Tree [15] and developed a two-stage partition mechanism to facilitate pruning. Zhang et al. [16] proposed a scalable and extensible index named OpIndex to support high-dimensional and sparse database effectively. OpIndex adopts a two-layer partition scheme and can be extended to support more expressive subscriptions.

In Elaps, we require not only *event matching* but also *subscription matching* over a large quantity of events. Among the above methods, only k-index and OpIndex can be extended to support *subscription matching*. Both indexes adopt a two-layer partitioning scheme and use the inverted list to group the attributes in the second layer. Their difference is the way they partition the events in the first layer. k-index partitions the events based on the event size, while OpIndex partitions the events based on the pivot attribute selected for each event. However, both partitioning schemes are not efficient in supporting subscription matching, especially when the spatial matching is taken into consideration. In this paper, we propose a more efficient index BEQ-Tree to support spatial subscription matching.

2.3 Problem Statement

In this paper, we study the efficient processing of continuous moving range queries against dynamic event streams. Given a set of continuous arriving events, for a moving subscriber with a boolean expression subscription associated with a notification region, the subscriber is notified once there is a matching event located within his notification region. We aim for a solution that (i) optimizes the client/server communication cost and (ii) guarantees that the matching events are disseminated to subscribers in real-time.

3. CONTINUOUS MOVING QUERIES OVER DYNAMIC EVENT STREAMS

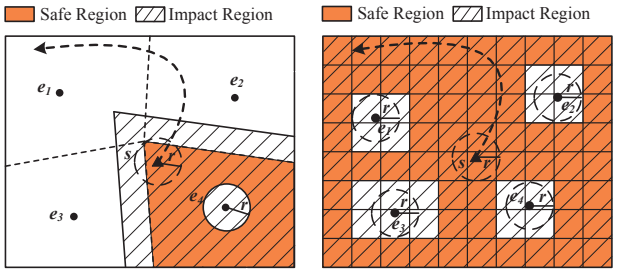
Compared to existing location-based pub/sub systems, Elaps is the first to consider continuous moving queries against dynamic event streams from publishers. The main challenge is to reduce communication overhead because users have to periodically report their current locations to the server to guarantee that no matching events in their neighborhood are missed. Existing pub/sub systems that can handle moving users mainly use safe region techniques to reduce communication cost. The intuition behind the notion of safe region is that if there are no matching events nearby, the users are safe to disconnect from the server and do not need to periodically update their locations. In this way, the communication cost can be significantly reduced and there would be no matching events missed. In this section, we first propose how to apply these techniques into our pub/sub application. Then, we explain why the safe regions constructed in these systems fail to work well when dynamic event streams are considered. Finally, we propose our solutions based on a concept named *impact region* as well as a new cost model for safe region construction.

3.1 Safe Region against Static Event Datasets

Safe region has been widely used to reduce communication cost for continuous spatial query processing [5, 6, 7]. In these work, a common assumption is that the continuous query is issued against a static dataset and the goal is to determine an area in which there is no matching events or the matching events remain the same. Since the publisher dataset is static, there would be no new event matching in the safe region. For our application, we define a region is safe if its minimum distance to any matching event is larger than the user's notification radius, denoted by r[1].

DEFINITION 1 (SAFE REGION). *The safe region $\mathcal{R}$ for a subscriber s is a region such that $s \in \mathcal{R}$ and for any matching event e, we have $d(p, e) > r$ for any $p \in \mathcal{R}$.*

In the following, we examine existing safe region techniques and show how to apply them for continuous proximity detection between subscribers and their matching events.



(a) VM (b) GM

Figure 2: Applying existing methods for safe region construction

Voronoi-based Method (VM). Voronoi diagram is often used in processing continuous kNN queries in continuous spatial query applications [5, 6]. Since the publisher dataset is static, the space is partitioned into voronoi cells based on the locations of publishers. Each voronoi cell indicates a region dominated by a publisher such that as long as a subscriber moves in that cell, the publisher is assured to be the nearest neighbor. To apply voronoi diagram in our application, we first find all the matching events outside of the notification region [2] and construct voronoi cells based on the matching events. Each cell contains one matching event. The safe region is the voronoi cell containing the user, excluding the circle centered at the matching event with notification radius.

Figure 2(a) shows an example of safe region based on voronoi diagram. There are four matching events that partition the space into four voronoi cells. Since the subscriber is located in e_4, the safe region is the voronoi cell for e_4 excluding the circle and marked in the shaded area. As long as the user moves in the safe region, we can guarantee that there is no matching event nearby and the

[1]A notation table consisting of symbols and their meanings is provided in Table 3(Appendix A)

[2]If a matching event appears in the notification region, we notify the subscriber immediately about the matching. Then, the event will not be considered again for this user in the future.

user only need to check his distance with e_4 and can temporarily disconnect from the server.

Grid-based Method (GM). In [7], safe region based on grid cells was proposed to handle spatial alarm applications. By partitioning the space into cells, a safe region is represented by a set of cells whose distance to any matching events is larger than the notification radius. An example of grid-based safe region is shown in Figure 2(b). The safe region for the subscriber contains the whole space except the cells close to the matching events. Compared to the voronoi diagram method, the grid based safe region is easier to construct and contains a larger region than VM.

When the event dataset is static, GM generates a larger safe region and achieves better performance in terms of communication I/O. However, when we consider dynamic event streams, the results can be totally different. In this case, a larger safe region does not necessarily mean a better solution. When a new matching event arrives close or inside the safe region, the region may not be "safe" any more. Then the whole safe region has to be re-constructed, leading to additional communication I/O. This observation motivates us to first develop a new concept named impact region to identify whether a safe region is "safe" or not when a new event arrives, and then propose a novel cost model based on the safe region and impact region to guide the construction of the safe region .

3.2 Impact Region

We first define the *impact region* for a certain safe region $\mathcal{R}$, and use it to determine whether $\mathcal{R}$ will be affected by the arrival of a new event. Intuitively, if a new matching event arrives in the impact region, the safe region $\mathcal{R}$ is not "safe" any more and needs to be updated. Otherwise, the safe region remains the same. With the help of the impact region, we do not need to update the safe region each time a new matching event arrives. Thus, the impact region can help reduce the communication cost incurred by event arrival. The concept of impact region will also be needed in our proposed cost model to estimate the expected time before the next matching event affects its corresponding safe region.

DEFINITION 2 (IMPACT REGION). *Given a safe region $\mathcal{R}$, the impact region $\mathcal{I}$ for $\mathcal{R}$ is defined as*

$$\{p \mid p \in U \text{ and } \exists\, p' \in \mathcal{R}\,(d(p,p') < r)\}$$

where U is the whole space and r is the notification radius.

Based on the definition, we know that if a point is located outside the impact region, its minimum distance to safe region $\mathcal{R}$ must be larger than r. Hence, we can consider impact region as an expansion of the safe region by the length of notification radius. Note that the impact region is uniquely determined by the safe region and should always be used together with the safe region. In the following, we briefly introduce how to construct impact regions from the safe regions in Figure 2.

EXAMPLE 1. *In VM, the impact region is constructed by expanding the Voroni cell of the nearest event with distance r. As shown in Figure 2, the impact region covers the notification region of the subscriber after the expansion. We can guarantee that when a new event arrives outside the impact region, its minimum distance to the impact region is larger than the notification radius. This new event will not fall inside the notification region as long as the user moves within the safe region. Hence, we only need to update the safe regions whose impact region contains this new event.*

In GM, the impact region is constructed by expanding the safe region with r to see which cells are contained in or intersected with the expanded region, which is the whole space in this example. When a new matching event arrives in the impact region, we need users to report their location to guarantee that the current safe region is "safe" and no matching notification is missed. Since the impact region contains the whole space, new communication overhead is incurred each time when a new matching event arrives.

Up till now, we have introduced the three types of regions maintained for each moving user: notification region $\mathcal{O}$, safe region $\mathcal{R}$ and impact region $\mathcal{I}$. The notification region is set by the subscribers and move with them. It is a circle centered at the subscriber's current location with radius r. For each subscriber, the system constructs a safe region and an impact region for him. The safe region is sent to the subscriber to monitor the location update and the impact region is stored at the server side to monitor the event arrival. Their relationships are summarized as follows:

LEMMA 1. *$\mathcal{O}$ is contained in $\mathcal{I}$, denoted by $\mathcal{O} \subseteq \mathcal{I}$.*

PROOF. Suppose we can find a point $p \in \mathcal{O}$ but $p \notin \mathcal{I}$. Since $p \notin \mathcal{I}$, based on the definition of the impact region, for any point p' in the safe region $\mathcal{R}$, we have $d(p,p') > r$. Since the safe region is required to cover the user's current location s, i.e, $s \in \mathcal{R}$, we have $d(p,s) > r$. However, since p is located in the notification region, for any points $p \in \mathcal{O}$, we have $d(p,s) < r$. This leads to a contradiction. □

LEMMA 2. *$\mathcal{R}$ is contained in $\mathcal{I}$, denoted by $\mathcal{R} \subseteq \mathcal{I}$.*

PROOF. Based on the definition of impact region, for any point $p \in \mathcal{R}$, if we can find a point $p' \in \mathcal{R}$ such that $d(p,p') < r$, we know p is also a point in the impact region. Let $p = p'$ and we finish the proof. □

Note that the safe region $\mathcal{R}$ does not necessarily contain the notification region $\mathcal{O}$. In fact, when a user increases his notification radius r, the safe region shrinks such that the minimum distance of the new safe region to any matching event is guaranteed to be larger than r. In addition, we can prove that the area of impact region grows when the safe region expands.

LEMMA 3. *Given $\mathcal{I}_1$ derived from $\mathcal{R}_1$ and $\mathcal{I}_2$ derived from $\mathcal{R}_2$, if $\mathcal{R}_1 \subseteq \mathcal{R}_2$, we have $\mathcal{I}_1 \subseteq \mathcal{I}_2$.*

PROOF. Suppose we can find a point $p \in \mathcal{I}_1$ but $p \notin \mathcal{I}_2$. Since $p \notin \mathcal{I}_2$, based on the definition of the impact region, for any point $p' \in \mathcal{R}_2$, we have $d(p,p') > r$. Since $\mathcal{R}_1 \subseteq \mathcal{R}_2$, this conclusion also applied in the sub-region, i.e., for any point $p' \in \mathcal{R}_1$, we have $d(p,p') > r$. This leads to a contradiction with $p \in \mathcal{I}_1$. Hence, if $p \in \mathcal{I}_1$, we have $p \in \mathcal{I}_2$. a new matching event e with p as its location will influence $\mathcal{R}_1$ because e is located within $\mathcal{I}_1$. Since $\mathcal{R}_1 \subseteq \mathcal{R}_2$, e will also influence $\mathcal{R}_2$. However, since e is located outside $\mathcal{I}_2$, this contradicts with the definition of impact region that any matching event located outside the impact region will not influence the safe region. Hence, if $p \in \mathcal{I}_1$, we have $p \in \mathcal{I}_2$. □

In the following, we show that there is no matching event in the impact region.

LEMMA 4. *Suppose $\mathcal{I}$ is an impact region constructed for subscriber s, for any event e matching s, e is located outside $\mathcal{I}$.*

PROOF. If there is a matching event $e \in \mathcal{I}$, based on the definition of the impact region, we can find a point p in the safe region such that $d(p,e) < r$. Then, by following the definition of safe region, since $p \in \mathcal{R}$, we have $d(p,e) > r$. This leads to a contradiction. Hence, e must be located outside $\mathcal{I}$. □

From Lemma 4, we can ensure that if a matching event expires, it is located outside the impact region and has no effect on the current safe region.

3.3 Cost Model for Safe Region Construction

Since we consider continuous query processing against dynamic event streams, we first identify all the possible cases in which new communication can be triggered for an existing subscriber.

1. The subscriber moves out of his current safe region. He needs to report his new precise location to the server. At the server side, a new safe region is calculated and sent back to the user.
2. A new matching event arrives in the system. Whether a communication is triggered depends on the location of the new event. If the event is located outside the impact region, the safe region is not affected based on the definition. Otherwise, the safe region has to be updated. This triggers a new communication. First, the server notifies the subscriber to update the location. Then, the precise location is reported by the client. When the server receives the accurate location, it calculates the distance from the event to the subscriber. If the distance is smaller than the user's notification radius, a matching notification is sent to the user. Otherwise, the server needs to calculate and sends a new safe region to the client. In the meanwhile, the impact region is updated, but stored at the server side.
3. An existing matching event is expired and removed from the system. Based on Lemma 4, the matching event is located outside the impact region. We can guarantee that the current safe region is still "safe". As long as the subscriber is in the current safe region, he can disconnect from the server and no matching event will be missed.

Therefore, there are two types of communication incurred when handling a continuous query over dynamic event streams: I) The subscriber moves out of the safe region. II) A new matching event arrives in the impact region. These two types of communication have conflicting requirements on the size of safe region. The first type prefers larger safe region so that it takes longer time for the subscriber to move out of the safe region. However, the second type prefers smaller safe region. This is because a smaller safe region results in a smaller impact region according to Lemma 3 and it becomes less likely for a matching event to occur in the impact region. Existing safe region techniques do not work well for dynamic event streams because they neglect the second type of communication. Therefore, we propose a new cost model for safe region construction taking into account all these two types of communication.

The goal of our cost model is to minimize the communication overhead which is measured by the number of the two types of communication I/O for a subscriber s. Hence, we construct a safe region $\mathcal{R}$ for s such that the expected elapsed time before the next communication I/O is maximized. The expected elapsed time is denoted by $f_{obj}(\mathcal{R}, \mathcal{I})$ and used as the objective function to maximize. Let $t_s(\mathcal{R})$ denote the expected time to move out of the current safe region $\mathcal{R}$ and $t_i(\mathcal{I})$ denote the expected time before the next matching event occurs in the impact region $\mathcal{I}$ constructed from $\mathcal{R}$. Since there are only two circumstances in which communication I/O is triggered, we have

$$f_{obj}(\mathcal{R}, \mathcal{I}) = \min(t_s(\mathcal{R}), t_i(\mathcal{I})) \quad (1)$$

Next, we define b_m to measure the tradeoff between the two types of communication.

$$b_m(\mathcal{R}, \mathcal{I}) = \frac{t_s(\mathcal{R})}{t_i(\mathcal{I})} \quad (2)$$

We can prove that $b_m(\mathcal{R}, \mathcal{I})$ has a positive correlation with the area of $\mathcal{R}$:

LEMMA 5. *Given two safe regions $\mathcal{R}$ and $\mathcal{R}'$ with $\mathcal{R} \subseteq \mathcal{R}'$, we have $b_m(\mathcal{R}, \mathcal{I}) \leq b_m(\mathcal{R}', \mathcal{I}')$.*

PROOF. Since $\mathcal{R} \subseteq \mathcal{R}'$, we know that $t_s(\mathcal{R}) \leq t_s(\mathcal{R}')$ and $t_i(\mathcal{I}) \geq t_i(\mathcal{I}')$ based on Lemma 3 and the definition of t_s and t_i. Hence, $b_m(\mathcal{R}, \mathcal{I}) = \frac{t_s(\mathcal{R})}{t_i(\mathcal{I})} \leq \frac{t_s(\mathcal{R}')}{t_i(\mathcal{I}')} = b_m(\mathcal{R}', \mathcal{I}')$. □

If $b_m(\mathcal{R}, \mathcal{I}) \leq 1$, we have $t_s(\mathcal{R}) \leq t_i(\mathcal{I})$ and $f_{obj} = t_s(\mathcal{R})$. In this case, we need to maximize $t_s(\mathcal{R})$ and we prefer a larger safe region for $\mathcal{R}$. If $b_m(\mathcal{R}, \mathcal{I}) > 1$, we have $f_{obj} = t_i(\mathcal{I})$ and we prefer a smaller safe region for $\mathcal{R}$. The relationship between $f_{obj}(\mathcal{R}, \mathcal{I})$ and $b_m(\mathcal{R}, \mathcal{I})$ are stated in the following two lemmas:

LEMMA 6. *Given two safe regions $\mathcal{R}$ and $\mathcal{R}'$ with $\mathcal{R} \subseteq \mathcal{R}'$, suppose $b_m(\mathcal{R}', \mathcal{I}') \leq 1$, we have $f_{obj}(\mathcal{R}, \mathcal{I}) \leq f_{obj}(\mathcal{R}', \mathcal{I}')$.*

PROOF. Based on Lemma 5, $b_m(\mathcal{R}, \mathcal{I}) \leq b_m(\mathcal{R}', \mathcal{I}') \leq 1$ because $\mathcal{R} \subseteq \mathcal{R}'$. For $\mathcal{R}$, since $b_m(\mathcal{R}, \mathcal{I}) \leq 1$, we have $t_s(\mathcal{R}) \leq t_i(\mathcal{I})$ and thus $f_{obj}(\mathcal{R}, \mathcal{I}) = t_s(\mathcal{R})$. Similarly, $f_{obj}(\mathcal{R}', \mathcal{I}') = t_s(\mathcal{R}')$. Since $\mathcal{R}$ is contained in $\mathcal{R}'$, we have $t_s(\mathcal{R}) \leq t_s(\mathcal{R}')$. Therefore, $f_{obj}(\mathcal{R}, \mathcal{I}) = t_s(\mathcal{R}) \leq t_s(\mathcal{R}') = f_{obj}(\mathcal{R}', \mathcal{I}')$. □

LEMMA 7. *Given two safe regions $\mathcal{R}$ and $\mathcal{R}'$ with $\mathcal{R} \subseteq \mathcal{R}'$, suppose $b_m(\mathcal{R}, \mathcal{I}) \geq 1$, we have $f_{obj}(\mathcal{R}, \mathcal{I}) \geq f_{obj}(\mathcal{R}', \mathcal{I}')$.*

PROOF. Based on Lemma 5, $b_m(\mathcal{R}', \mathcal{I}') \geq b_m(\mathcal{R}, \mathcal{I}) \geq 1$ because $\mathcal{R} \subseteq \mathcal{R}'$. For $\mathcal{R}$, since $b_m(\mathcal{R}, \mathcal{I}) \geq 1$, we have $t_s(\mathcal{R}) \geq t_i(\mathcal{I})$ and thus $f_{obj}(\mathcal{R}, \mathcal{I}) = t_i(\mathcal{I})$. Similarly, $f_{obj}(\mathcal{R}', \mathcal{I}') = t_i(\mathcal{I}')$. Since $\mathcal{R}'$ is contained in $\mathcal{R}$, we have $t_i(\mathcal{I}) \geq t_i(\mathcal{I}')$. Therefore, $f_{obj}(\mathcal{R}, \mathcal{I}) = t_i(\mathcal{I}) \geq t_i(\mathcal{I}') = f_{obj}(\mathcal{R}', \mathcal{I}')$. □

Our safe region construction method relies on the above lemmas. The idea is to start from the user's current location and incrementally expand the area towards an "optimal" safe region (i.e., f_{obj} is maximized). Initially, $\mathcal{R}$ contains only a point. Thus, t_s can be seen as 0 and $b_m(\mathcal{R}, \mathcal{I}) \leq 1$. When we expand $\mathcal{R}$, $b_m(\mathcal{R}, \mathcal{I})$ and $f_{obj}(\mathcal{R}, \mathcal{I})$ also increase (Lemma 5 and 6). In this case, it encourages us to expand the safe region until any further expansion would cause $b_m(\mathcal{R}, \mathcal{I}) \geq 1$ or the region not "safe". This is because $f_{obj}(\mathcal{R}, \mathcal{I})$ decreases when $b_m(\mathcal{R}, \mathcal{I}) \geq 1$ if we continue to expand $\mathcal{R}$ based on Lemma 7.

Note that there are many possible ways to expand a safe region, resulting in lots of candidate safe regions. For each of these candidate safe regions, we have $b_m \leq 1$ and thus $f_{obj} = t_s$. Hence, the "optimal" safe region is the candidate safe region with the largest t_s. Based on this observation, we should expand the safe region in such a way that the corresponding t_s can be maximized.

3.4 Incremental Grid-based Method

We are now ready to present our incremental method, named iGM (incremental Grid-based Method), towards optimal safe region construction. Since we need a flexible way to represent safe region in arbitrary shape, we partition the space into $N \times N$ cells and a safe region is represented by the set of cells that it covers. Our algorithm starts from the cell containing the user's current location and iteratively expands it to cover nearby cells. In each expansion, a "good" cell based on certain criteria is added to the current safe region. When a cell is added into the current safe region, we also need to expand the corresponding impact region to calculate $b_m(\mathcal{R}, \mathcal{I})$. The algorithm terminates when any further expansion would cause $b_m(\mathcal{R}, \mathcal{I}) > 1$ or there is no more valid adjacent cells to expand. In the following, we introduce the estimation of $b_m(\mathcal{R}, \mathcal{I})$ for a candidate safe region $\mathcal{R}$ as well as the selection criteria for the next cell to expand.

Algorithm 1: ConstructSafeRegion

input: Subscription s
output: Safe region $\mathcal{R}$ and impact region $\mathcal{I}$

1 $b_m \leftarrow 0$
2 $n_e \leftarrow 0$
3 $\mathtt{d}(s, \mathcal{R}) \leftarrow 0$
4 $\mathcal{H} \leftarrow \emptyset$
5 $c \leftarrow$ the cell that contains s
6 HEntry $h \leftarrow$ tuple$(c, d(s, c))$
7 insert h into $\mathcal{H}$, mark c as visited
8 **while** $\mathcal{H} \neq \emptyset$ **do**
9 HEntry $(c', d(s, c')) \leftarrow \mathcal{H}.pop()$
10 **if** $\beta[c'] \neq false$ **then**
11 $\mathtt{d}(s, \mathcal{R}) \leftarrow min\{\mathcal{H}.top().dist, min\{d(s, c'')\}\}$
12 $\mathcal{I}_c \leftarrow getImpactCells(c')$
13 **foreach** $c_n \in \mathcal{I}_c$ **do**
14 $n_e = n_e + \phi[c_n]$
15 $b_m = \frac{f \cdot n_e \cdot \mathtt{d}(s,\mathcal{R})}{n \cdot v_s}$
16 **if** $b_m \leq 1$ **then**
17 $\mathcal{R} \leftarrow \mathcal{R} \cup c'$
18 $\mathcal{I} \leftarrow \mathcal{I} \cup \mathcal{I}_c$
19 **foreach** non-visited adjacent unit cell c'' **do**
20 HEntry $h \leftarrow$ tuple $(c'', \mathtt{d}(s, c''))$
21 insert h into $\mathcal{H}$, mark c'' as visited
22 return $\mathcal{R}$ and $\mathcal{I}$

Assume that subscribers are moving with linear motion functions. The expected time $t_s(\mathcal{R})$ to move out of $\mathcal{R}$ can be calculated by

$$t_s(\mathcal{R}) = \frac{\mathtt{d}(s, \mathcal{R})}{v_s} \tag{3}$$

where $\mathtt{d}(s, \mathcal{R})$ is the minimum distance from a subscriber's location to the boundary of a candidate safe region $\mathcal{R}$ and v_s is the current moving speed.

The expected time $t_i(\mathcal{I})$ before the next matching event occurs in the impact region $\mathcal{I}$ for $\mathcal{R}$, is estimated by

$$t_i(\mathcal{I}) = \frac{t_e}{p(e, \mathcal{I})} \tag{4}$$

where t_e is the average time interval between two new events arriving at the system and $p(e, \mathcal{I})$ is the probability for a new event to match subscriber s and occur in his impact region $\mathcal{I}$.

We can estimate t_e from the average arrival speed in the event streams. We denote the average arriving rate of the new events by f and we have $t_e = \frac{1}{f}$. The probability $p(e, \mathcal{I})$ can also be estimated from the distributions of the existing events in the system. Let n_e be the number of matching events located in the impact region $\mathcal{I}$. n_e can be estimated by $n_e = \sum_{c \in \mathcal{I}} n_c$ where n_c is the number of existing matching events located within cell c. Let n be the total number of events in the system. We can estimate $p(e, \mathcal{I}) = \frac{n_e}{n}$. Then t_i can be calculated as follows.

$$t_i(\mathcal{I}) = \frac{n}{f \cdot n_e} \tag{5}$$

Based on Equation 3 and Equation 5, we have

$$b_m(\mathcal{R}, \mathcal{I}) = \frac{t_s(\mathcal{R})}{t_i(\mathcal{I})} = \frac{f \cdot n_e \cdot \mathtt{d}(s, \mathcal{R})}{n \cdot v_s} \tag{6}$$

Note that among the parameters, v_s, f and n are system statistics and are independent of the safe region and impact region. $\mathtt{d}(s, \mathcal{R})$ and n_e are dependent on the areas of the candidate safe region and impact region respectively.

Next, we introduce our expansion criteria. Suppose the current safe region is $\mathcal{R}$. We mark all the cells covered by or intersected with $\mathcal{R}$ as visited. The candidate cells to expand include all the adjacent cells of $\mathcal{R}$ that are unvisited. Our expansion criteria is to pick the cell c with the minimum distance to the subscriber. In other words, we expand the area circularly. This is because we expand the safe region until any expansion would cause $b_m(\mathcal{R}, \mathcal{I}) \geq 1$. In this process, we have $b_m(\mathcal{R}, \mathcal{I})$ always smaller than 1 and our goal is to maximize t_s. Since the moving pattern of the subscriber is not available, we expand the cells in every direction uniformly so as to improve the performance in the worst case.

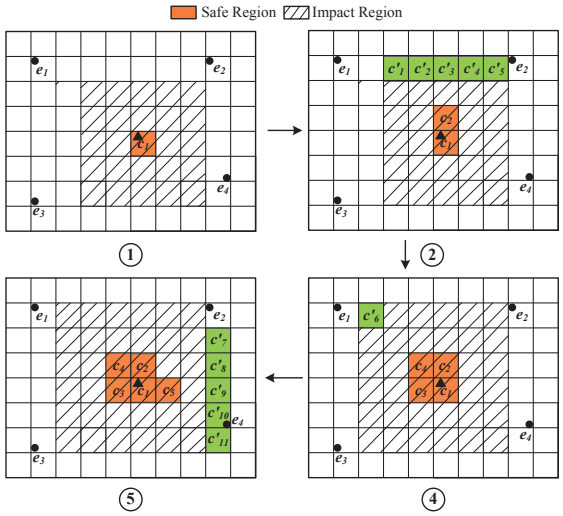


Figure 3: Safe region and impact region expansion

Our iGM algorithm for safe region construction is shown in Algorithm 1. The input is a subscription s with notification radius r and speed v_s. We first initialize the parameters b_m, n_e and $\mathtt{d}(s, \mathcal{R})$ (lines 1-3) and build a min-heap $\mathcal{H}$ whose root stores the cell candidate with the minimum distance to s. The heap is initialized to contain the cell c where the subscriber is located (lines 4-7). After the initialization steps, we expand the safe region in a breadth-first fashion (lines 8-21). The candidate cell c' with the minimum distance to s is popped in each iteration (line 9). A boolean array $\mathcal{B}$ is used to indicate whether a grid cell is *safe* or not. A cell in the array is set to *safe* if its distance to the nearest matching event is larger than the notification radius of s. If c' is not *safe*, we simply ignore it and continue to examine the next cell in the heap. Otherwise, we calculate $\mathtt{d}(s, \mathcal{R})$ and n_e for the enlarged safe region $R \cup c'$ and impact region $\mathcal{I} \cup \mathcal{I}_c$ to evaluate b_m (lines 10-18), where $\mathcal{I}_c$ is the set of candidate cells that need to be added to $\mathcal{I}$ if c' is added to $\mathcal{R}$. The calculation of $\mathtt{d}(s, \mathcal{R})$ follows the equation:

$$\mathtt{d}(s, \mathcal{R} \cup c') = min\{\mathcal{H}.top().dist, min\{d(s, c'')\}\} \tag{7}$$

where c'' is an unvisited adjacent cell of c'(line 11). The procedure $getImpactCells$ is used to get $\mathcal{I}_c$(line 12). Then we update $n_e(\mathcal{I} \cup \mathcal{I}_c)$ (lines 13-14) and calculate $b_m(\mathcal{R} \cup c', \mathcal{I} \cup \mathcal{I}_c)$ (line 15). If $b_m(\mathcal{R} \cup c', \mathcal{I} \cup \mathcal{I}_c) \leq 1$, we enlarge $\mathcal{R}$ to contain cell c' and $\mathcal{I}$ to contain the cells in $\mathcal{I}_c$ (lines 16-18). We then insert the adjacent cells of c' which are not visited to $\mathcal{H}$ (lines 19-21). The whole algorithm terminates when $\mathcal{H}$ becomes empty and we return the safe region and impact region. Note that the two types of regions are constructed together in the expansion.

When a cell is added into the current safe region, we expand the impact region in the meanwhile. A naive solution is to scan all the cells within the notification radius and add them into current impact region. However, this incurs many redundant operations because a candidate cell will be added into impact region multiple times. To

avoid scanning so many cells in each expansion, we propose an incremental solution that is able to add only the unvisited cells into the impact region. We use an example to illustrate our idea.

EXAMPLE 2. *Illustrating examples of safe region expansion in multiple steps are shown in Figure 3. In the first step, the safe region $\mathcal{R}$ is initialized to the cell containing s, i.e., $\mathcal{R} = c_1$ and the impact region is an enlarged area of $\mathcal{R}$ by notification radius r. In the second step, a neighboring cell c_2 with the minimum distance is selected. Since c_2 has a neighboring cell c_1 included in the safe region, we know that the cells within distance to c_1 has been added in the impact region. Thus, we can directly add cells c'_1, c'_2, c'_3, c'_4 and c'_5 into the impact region without scanning all the cells. In the third step, c_3 is selected into the safe region. The expansion of impact region is similar to the second step and we do not show this step in the figure. In the fourth step, the nearest unvisited cell c_4 is selected. This time, the cell has two adjacent cells included in the safe region. We only need to add one cell c'_6 to the impact region. As the expansion process continues, the safe region and impact region become larger, resulting in a larger b_m. Suppose c_5 is the last cell to cause $b_m \leq 1$ during the expansion, we can terminate the algorithm after we expand the safe region to include c_5 and update the impact region accordingly.*

Next, we propose a direction-aware version of iGM when the direction information of the moving objects is available.

3.5 Incremental Direction-aware GM

Since most smartphones are equipped with sensors for direction detection, we propose a direction-aware iGM, named idGM, that takes into account user moving direction to better maximize $t_s(\mathcal{R})$, the expected time to leave a safe region.

To make iGM direction-aware, we should take into account the direction information when expanding the safe region. In other words, the original expansion mechanism relies on the distance between $d(s, \mathcal{R} \cup c)$ and each time the cell with the minimum distance is selected. We extend the idea to propose a more general scoring function in determining the next cell to expand. The ranking function takes into account of the user moving direction as well as the cell distance and is defined as follows:

$$\tau(s, c) = \alpha \cdot \mathcal{A}(s, c) + (1 - \alpha) \cdot \mathcal{D}(s, c) \tag{8}$$

The direction preference $\mathcal{A}(s, c)$ is the cosine value of the angle θ between the moving direction $\vec{v_s}$ and the vector $\vec{sc}$ from s to c.

$$\mathcal{A}(s, c) = cos\theta = \frac{\vec{v_s} \cdot \vec{sc}}{\|\vec{v_s}\| \|\vec{sc}\|} \tag{9}$$

The distance preference $\mathcal{D}(s, c)$ is the normalized distance from c to s, which is defined as

$$\mathcal{D}(s, c) = \frac{d(s, c)}{d_{max}} \tag{10}$$

where d_{max} is a normalization parameter which can be set to the maximum distance between any two points in the space.

Note that our goal is to maximize t_s in the safe region construction until the condition $b_m \leq 1$ is not satisfied. If the user moving pattern is predictable and we have high confidence that the user will continue to move along the direction, we can set α to a value close to 1. Then, the cells within the user's moving direction have higher priority to be added to the safe region. As long as the direction does not change, the user can stay in the safe region for a long time. If the user moving pattern is not clear and there are many uncertainties, we can set α to be a small value and uniformly expand the safe region in all directions.

With the new scoring function τ, we can modify Algorithm 1 to be direction-aware. The entry of $\mathcal{H}$ is modified as a tuple $(c, d(s, c), \tau)$ (line 7). And the entries in $\mathcal{H}$ are sorted in increasing order of τ. Now the algorithm expands the safe region by taking both the distance preference and direction preference into consideration. The grid cell with the minimum τ will be accessed firstly. Compared to iGM, idGM constructs a direction-aware safe region which takes a longer period before the next communication occurs. In Appendix B, we present how to reduce the bytes transferred in the communication between the server and the subscribers.

4. SPATIAL BE-MATCHING

In Section 3, we have introduced how to optimize the communication cost between the server and the client. In this section, we present how to disseminate the matching events to the subscribers in real-time. We observe that existing pub/sub systems using boolean expression matching [13, 14, 15, 16] rarely pay attention to index construction for the event stream. However, in the location-based service scenario, a subscriber wants to be notified when there is a matching event near him, even though this event has already been published before his subscription. This motivates us to build an index to cater for continuously arriving subscriptions' matching, namely BEQ-Tree (Boolean Expression Quad-Tree). In addition, we can utilize BEQ-Tree to improve the efficiency of construcing the safe region in iGM and idGM.

First of all, we describe the problem setting. In Elaps, a subscriber expresses his interest in the form of spatial subscription and is modeled as a moving object, while a publisher is associated with a geo-location and publishes spatial events.

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: $P^{(A, f_{op}, \bar{o})}(x) \rightarrow \{0, 1\}$. Elaps can support relational operators $<, \leq, =, >, \geq, [\,]$ [3] 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 3 (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.*

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

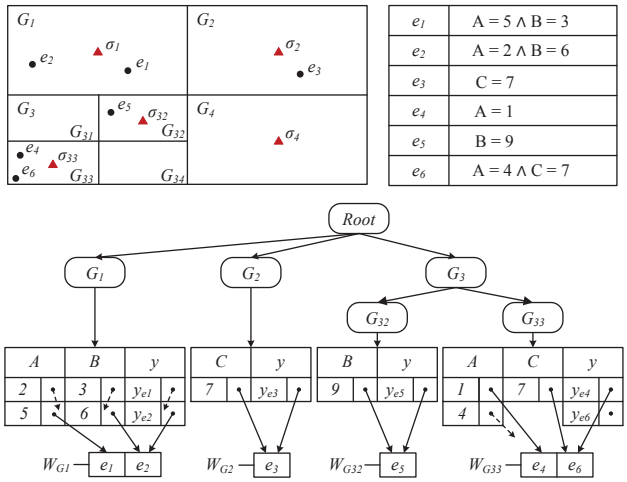


Figure 4: An example for BEQ-Tree

DEFINITION 4 (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 5 (MATCH). *We say a subscription s matches an event e, denoted by $s \sim e$, if $s \sim_b e$ and $s \sim_s e$.*

4.1 Spatial Event Index

BEQ-Tree is designed to be efficient in both query processing and event update. It adopts a two-layer partitioning mechanism, one for the spatial attribute and the other for the predicates in the boolean expression.

In the first layer, we partition the events based on the spatial attribute. Since Quadtree [12] can answer spatial range query quickly and support efficient update operations, we adopt it to partition the space such that each cell in the leaf level contains at most E_{max} events, where E_{max} is a moderately large number. In each cell, we select a reference point and adopt the idea of *iDistance* [17] to calculate and index the distance from each event to the reference point. The reference point is denoted as σ and selected as the center of a tree cell in this paper.

In the second layer, we further partition the events in each cell based on the attributes in the predicates. The predicates with the same attribute and appear in the same tree cell are organized in the same inverted list, denoted by $L_{\langle G_i, A\rangle}$ where G_i is the tree cell and A is the attribute. The location of an event e in G_i is converted to a single dimensional value $y = dist(e, \sigma_i)$, where $dist(e, \sigma_i)$ represents the Euclidean distance between e and σ_i. We then build an inverted list for attribute y for each cell. Each inverted list is sorted by the operand value. For each cell, we also maintain a counter array for all the events in this cell.

EXAMPLE 3. *Fig. 4 shows an example for the BEQ-Tree, where we set $E_{max} = 2$ for a Quadtree cell. The space is first hierarchically partitioned into cells. Each cell G_i is associated with a set of inverted lists to store predicates with the same attribute. All tuple lists are sorted in ascending order of their tuple values. There is an extended attribute y to store the distance from the event to the reference point in a cell. Each predicate has a pointer to a counting array for the corresponding Quadtree cell, which will be used in subscription matching.*

Algorithm 2: BESpatialMatch(Subscription s, Cell partition G)

```
1  R_e ← ∅
2  for each predicate (A f_op ō) ∈ s do
3  |  if G does not contain A then
4  |  |  return R_e
5  W_G ← counter array associated with G
6  for each predicate (A f_op ō) ∈ s do
7  |  for each operator f_op ∈ {=, ≠, ≤, ≥, [ ]} do
8  |  |  determine the range R_a in L_⟨G,A⟩
9  |  |  for each matching entry t ∈ R_a do
10 |  |  |  ++W_G[t.e]
11 y ← dist(s, σ)
12 if s is located within G then
13 |  d_min ← y − r, d_max ← y + r
14 else
15 |  d_min ← y − r
16 |  determine d_max accordingly
17 for each t ∈ L_⟨G,y⟩ and t.ō ∈ [d_min, d_max] do
18 |  if W_G[t.e] == |s| then
19 |  |  if dist(s.l, t.l) ≤ r then
20 |  |  |  add the corresponding event into R_e
21 return R_e
```

4.2 Subscription Matching

In the following, we show how to find the matching events given a subscription s with an notification region $\mathcal{O}$. We first find all the leaf cells that intersect with $\mathcal{O}$ using the Quadtree, and then examine the events in each candidate leaf cell G by calling Algorithm 2.

In Algorithm 2, we first check whether a cell partition G in the BEQ-Tree contains all the attributes that appear in s. If we find an attribute of s not appearing in G, we can prune the search space and examine other cells (lines 2-4). Otherwise, we copy the counter array associated with G and set the initial values of the entries to be 0 (line 5). Next, we introduce how to find the matching events by performing boolean expression match (lines 6-10) and spatial range match (lines 11-20).

The boolean expression match algorithm adopts the idea of the classic counting algorithm in [18, 13, 14]. Given a predicate A f_{op} $\bar{o}$, we use different accessing strategy for different operators in s (lines 6-8). If f_{op} is '=', we can check whether $\bar{o}$ appears in the sorted tuple list $L_{\langle G,A\rangle}$ using binary search. If f_{op} is '$\neq$', all the values in the tuple list except $\bar{o}$ are visited. If f_{op} is '$\leq$' ('$<$'), all the tuple entries whose value is no larger (smaller) than $\bar{o}$ match A f_{op} $\bar{o}$. The case is similar for '$\geq$' (or '$>$') and '[ ]'. For each list entry visited, we increase the corresponding counter value by 1 (lines 9-10). If the value increases to the size of s, the corresponding event be-matches s (see Definition 3).

We use Fig. 5 to explain how to perform spatial range match. Similar to boolean expression match, we need to locate the interval in the spatial list within which the corresponding event may fit in the notification region $\mathcal{O}$. There are two cases to consider. First, s is located within G (lines 11-13). In this case, the interval is $[y - r, y + r]$, where y is the one-dimensional distance value of s. Second, s lies out of G (lines 14-16). In this case, the lower bound of the interval is $y - r$. If the notification range contains a vertex of G, we have $d_{max} = \infty$, which means that we start from d_{min} and traverse to the end of the spatial list. Otherwise, $d_{max} = max\{dist(\sigma, p_i)\}$, where p_i is one of the intersection points between G and $\mathcal{O}$. As shown in Fig. 5, the notification region crosses two grid cells and we need to conduct spatial range search in these two partitions. The shaded areas indicate the search interval using the indexed distance. The interval for G_1 is $[y-r, y+r]$, because s is located within G_1. The interval for G_2 is $[y-r, d_{max}]$,

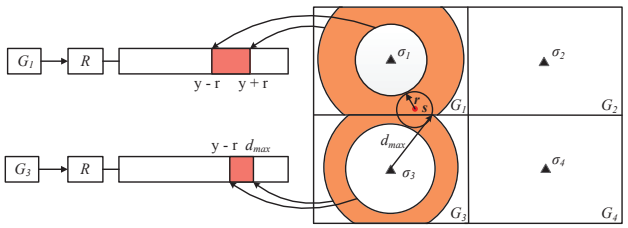


Figure 5: Spatial range match

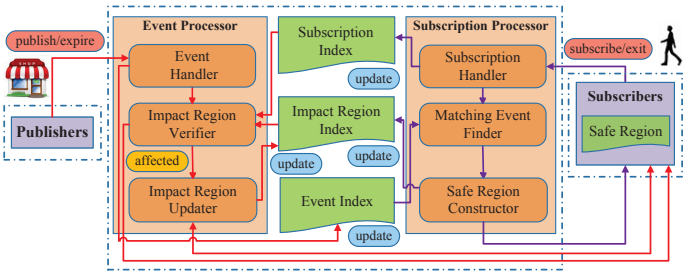


Figure 6: The workflow in Elaps framework

because s is located outside G_2 and $\mathcal{O}$ does not contain the vertex. For each tuple in the interval, if its corresponding event be-matches s, we check whether the event is located within $\mathcal{O}$. If yes, we find a match (lines 17-20). Since subscribers tend to specify a small notification range, the spatial range match would be very efficient, because only a few entries in the spatial list are traversed.

By performing the boolean expression match and spatial match, we only need to traverse a small part of entries in a list, which can further improve the matching performance besides the spatial pruning of the first layer.

BEQ-Tree used in iGM and idGM. To construct the safe region in iGM and idGM, the set of be-matching events should be found first. A naive method is to find all the be-matching events in the whole space. However, one property of iGM and idGM is that these two algorithms usually do not expand to the whole space with the constraint of $b_m(\mathcal{R}, \mathcal{I}) \leq 1$. Based on this property, we use BEQ-Tree in an incremental manner to get the be-matching events on demand. To construct the safe region for a subscriber s, we start from the cell c containing s and search the BEQ-Tree to get the set of be-matching events in c. In the meantime, if the leaf Qudatree cells intersected with c contains other cells around c, we also get the be-matching events in these cells. When iGM or idGM expands to a cell whose be-matching events have not been found, we check the surrounding Quadtree cells to get the be-matching events. In this way, we only need to find the set of be-matching events on demand and traverse only a part of the space.

5. SYSTEM FRAMEWORK

So far, we have introduced the techniques of safe region and impact region to reduce communication I/O and the BEQ-Tree to reduce the response time. In this section, we present the system framework in Fig. 6 as a whole picture to see how different function components are connected. In particular, we introduce how to process subscription arrival/expiration, event arrival/expiration and user location update.

Subscription arrival/expiration. In Elaps, users can submit new subscriptions and an existing subscription expires if 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, we need to find if there exist any matching events in the event database. Since this is an extension of boolean expression matching with spatial constraints, we propose a new index named BEQ-Tree to solve the problem in Section 4. For each new subscriber, we also call Algorithm 1 in Section 3 to construct a new safe region and impact region for the user. The safe region is sent to the user and the impact region is maintained at the server side. The impact region is inserted into an inverted index with cell id as the key and the elements are impact regions covering the cell. When a subscription expires, we remove it from the subscription index and impact region index.

Event arrival and expiration. When a new event e arrives, we need to identify from the subscriber database the users whose interests match the new event (e is contained in the notification region) and those users whose safe regions are affected (e is contained in the impact region). Since we use safe region to reduce the communication overhead, the subscriber's precise location is not maintained at the server side. We only know the user is currently in the safe region but the precise location is not available. Thus, we build separate indexes to handle boolean expression matching and spatial constraint verification. For the boolean expression matching, we can simply adopt existing subscription index such as OpIndex [16] and BE-Tree [15]. For the spatial attribute, we store the impact region for subscribers in the impact region index maintained as a hash table. When e arrives, we find the matching users U from the subscription index. For each subscriber $u \in U$, we check whether e locates within his impact region. If yes, the server send e to u. If the distance from u to e is smaller than the notification radius, the user is notified. Otherwise, we calculate a new safe region and impact region. The new safe region is sent to the subscriber and the impact region index is also updated.

User location update. Each time a subscriber moves out of the safe region, he reports the new location to the server. Once the server receives the new location, it calls Algorithm 1 to construct a new safe region and impact region for the user. Again, the safe region is sent to the subscriber and the inverted index for impact regions is updated.

6. EXPERIMENTAL STUDY

In this section, we present a comprehensive evaluation of the performance for Elaps. In particular, we are interested in evaluating (1) the communication overhead caused by continuous moving query processing against dynamic event streams and (2) the matching efficiency of our proposed BEQ-Tree. For safe region techniques, we compare our proposed iGM and idGM based on the new cost model against existing methods VM and GM. In all these methods, we calculate and store the impact regions at the server side for performance evaluation. For spatial boolean expression matching, we compare BEQ-Tree with three baseline algorithms: (i) Quad-tree, which first filters events outside the notification region using the spatial index Quad-tree [19] and then verifies the boolean expression matching; (ii) extension of k-index [14], which finds matching events for a given subscription. The be-matching events are further verified for spatial matching; (iii) OpIndex, which first filters the events not matching the boolean expressions using a variant of OpIndex [16] and then verifies the spatial matching. Note that all the indexes are memory resident and all the approaches produce the same and complete results. We implemented all the methods in C++ and conducted the experiments on a server with 48GB memory running Centos 5.6.

6.1 Experimental Setup

Event and Subscription Datasets. We use geo-tweets from Twitter and venues from Foursquare to simulate the event streams. In Twitter, each geo-tweet is considered as a spatial event. We treat each keyword as an attribute and the value is the frequency of the keyword in the tweet. This converts a geo-tweet into a list of attribute-value pairs. In the following experiments, we use 50 million geo-tweets as the event database and another 10 million to simulate a dynamic event stream. To generate subscriptions on the Twitter dataset, we adopt the same technique in [16] to convert a keyword query in AOL search log [4] into a boolean expression. For example, a query "SIGMOD Melbourne" can be transformed to (SIGMOD=1 $\wedge$ Melbourne=1) to support equal operator or (SIGMOD$\in$[5,20] $\wedge$ MelBourne$\in$[2,8]) to support interval operator. The dataset description and experimental results of Foursquare is presented in Appendix D.2 due to space limitations.

User Trajectory Datasets. We use both synthetic trajectories and real trajectories to simulate user moving patterns. For the synthetic trajectories, we generate $10,000$ trajectories using Brinkoff's generator [20]. The average travel period of each trajectory is 1000 timestamps. In our experiments, timestamp is used to capture the periodicity of location update. We set each timestamp to be 5 seconds in the following experiments, which means the GPS location of a user is sampled every 5 seconds. For the real trajectories, we use the GPS probing records of taxies in Singapore as the real trajectories [21]. We extract 10,000 trajectories from different taxies and each trajectory contains 1000 sequential points.

Communication Overhead	
Event arrival rate f $(/tm)$	10, 50, **100**, 500
Moving speed v_s (m/tm)	20, 40, **60**, 80, 100
Notification radius r (km)	1, **2**, 3, 4, 5
Number of events $\mathbb{E}$	10M, **20M**, 30M, 40M, 50M
Matching Performance	
Number of events $\mathbb{E}$	10M, **20M**, 30M, 40M, 50M
Avg. sub size δ	1, 2, 3, **4**, 5
Notification radius r (km)	1, **2**, 3, 4, 5

Table 2: Parameters evaluated in the experiments

Evaluation Parameters. Table 2 shows the main parameters and values used throughout the experiments (default values are in bold). To evaluate continuous moving query processing, we examine the scalability with respect to increasing event arrival rate f, user moving speed v_s, notification radius r and event corpus size $\mathbb{E}$. To evaluate the matching performance of spatial boolean expressions, we examine increasing event corpus size $\mathbb{E}$, average subscription size δ and notification radius r. The system performance can be measured by the average communication I/O and event matching time for a subscriber. Note that we do not need to examine the performance w.r.t. increasing number of subscribers. This is because the subscribers will not affect each other in terms of communication overhead and event matching efficiency. In our system, the number of subscribers directly influences the scalability of the subscription index (see Fig. 6). However, in this work, we adopt an existing index (OpIndex [16]) which has been shown to be scalable even for a large number of subscribers. Moreover, the focus of this work is not on the subscription index. Thus, we did not look into this any further. Pease refer to [15] for details.

Parameter Tuning. There are two parameters that require tuning for iGM and idGM: N (the whole space is partitioned into $N \times N$ cells) and α (the tuining parameter of the preference score τ), and one parameter that requires tuning for BEQ-Tree: E_{max} (the maximum number of events in a Qudatree cell). We conduct several experiments to tune the parameters and the results are shown in Appendix D.1. Based on the results, we set N to be 600, α to be 0.5 and E_{max} to be 60K.

[4] http://www.gregsadetsky.com/aol-data/

6.2 Continuous Moving Query Processing

In this part, we evaluate the communication overhead caused by continuous moving query processing against dynamic event streams.

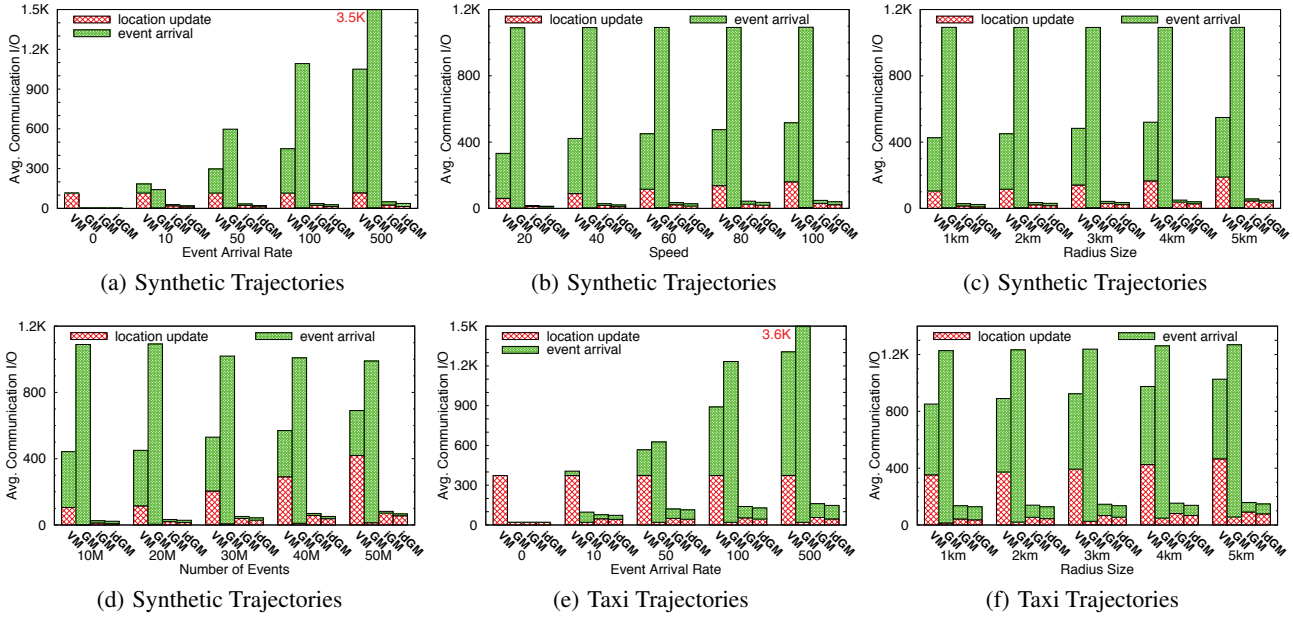
6.2.1 Synthetic Trajectories

The first set of experiments on continuous moving query processing are conducted on the synthetic trajectories using the Twitter and Foursquare events.

Effect of the event arrival rate f. Fig. 7(a) presents the impact of the event arrival rate f on the performance. We report the communication overhead incurred when users move out of the safe region (location update) and when a new matching event occurs in the impact region (event arrival) respectively. We have the following observations. (1) As f increases, all the methods scale in varying degrees. Because there would be more matching events that fall within the impact region, which increases the cost incurred by event arrival. (2) in terms of the total communication cost, iGM and idGM can outperform the other baseline methods by one order of magnitude. Such performance gain increases when f increases, especially on the cost incurred by event arrival. This is because iGM and idGM can adjust the size of the safe region dynamically according to several parameters of the system to balance the cost incurred by location update and the cost incurred by event arrival. (3) Regarding the cost incurred by location update alone, GM has the smallest one, because it constructs the largest safe region. VM performs much worse than the other methods, because it constructs a safe region around the nearest matching event regardless of the user's location. idGM performs better than iGM, because idGM can construct a safe region with a larger t_s which is the expected time before the subscriber leaves the safe region by considering the user's moving direction. (4) Regarding the cost incurred by event arrival alone, GM is the highest, because it constructs a rather large impact region even though f is very high. Although VM can construct a smaller impact region compared to GM, the impact region of VM always contains some highly skewed area (around the nearest matching event). Therefore, the cost incurred by event arrival of VM is also high. When f is larger, iGM and idGM would construct a smaller safe region and impact region based on the cost model. Thus, the cost incurred by event arrival can be controlled very well.

Effect of the speed v_s. Fig. 7(b) depicts the effect of another key factor, the moving speed v, on the communication overhead. When v increases, the subscriber would leave the safe region more frequently, resulting in an increased cost incurred by location update. Therefore, the total communication cost increases for all methods except GM. GM is not sensitive to v for Foursquare and Twitter, because it constructs a rather large safe region. As shown, iGM and idGM still outperform the rest, since they can dynamically increase the size of the safe region as v increases. We can also observe that the superiority of idGM compared with iGM is more significant with a larger v.

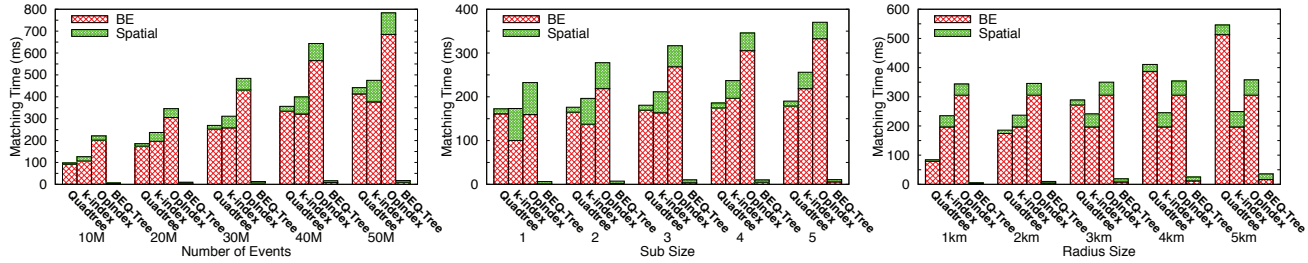
Effect of the notification radius r. Next, we evaluate the effect of the notification radius r. The results are shown in Fig. 7(c). A larger r results in a smaller safe region, which increases the cost incurred by location update for all the methods. For VM, the cost incurred by event arrival increases. This is because we expand the safe region for VM by the length of r and thus a larger r results



(a) Synthetic Trajectories (b) Synthetic Trajectories (c) Synthetic Trajectories

(d) Synthetic Trajectories (e) Taxi Trajectories (f) Taxi Trajectories

Figure 7: Communication I/O on Twitter.



(a) Varing number of events (b) Varing sub size (c) Varing radius

Figure 8: Experiments on the event index.

in a larger impact region. In contrast, iGM and idGM can adjust the size of the impact region dynamically even though r increases. Thus, iGM and idGM show 10X better performance than the other baseline methods.

Effect of the number of events $\mathbb{E}$. Lastly, we study the communication overhead when increasing the size of the Twitter events from 10M to 50M. The results are shown in Fig. 7(d). More events (i.e., more matching events) in the system result in a smaller safe region and impact region. A smaller safe region leads to more cost incurred by location update, while a smaller impact region leads to less cost incurred by event arrival. For VM, the overall communication overhead increases because the increase in the cost incurred by location update surpasses that in the cost incurred by event arrival. On the other hand, for GM, the overall communication overhead decreases, because the performance of GM relies more on the impact region. Compared with VM and GM, our two methods can adjust the area of the safe region and impact region better when the system environment changes and thus scale very well when the event size increases.

6.2.2 *Taxi Trajectories*

The second set of experiments are conducted on the real taxi trajectories in Singapore. Since the trajectories are only located within Singapore, we extract those geo-tweets located within Singapore from the Twitter dataset. In total, we extract 906,977 geo-tweets. We use 0.5 million of them as the event database and the remaining to simulate the event stream. We evaluate the communication overhead with respect to the event arrival rate f and the notification radius r. The results are shown in Fig. 7(e) and Fig. 7(f). Compared with the synthetic trajectories where the speed is constant, the taxi trajectories contain all kinds of taxies with different moving status. Besides, the moving speed of a taxi is influenced by the road traffic greatly. Therefore, it is more difficult to predict the moving behavior for the taxi trajectories. However, as can be seen, our two methods can still achieve a much better performance compared with the other baseline methods. As compared to the counterpart GM, our iGM and idGM have a comparable performance in terms of the cost incurred by location update and reduce the cost incurred by event arrival significantly by more than 1 order of magnitude.

6.2.3 *Cost Model Evaluation*

In this part, we study the optimality and robustness of our cost model.

Optimality Evaluation. When constructing a safe region for a subscriber, we start from his location and incrementally expand. In this process, the value of $b_m(\mathcal{R}, \mathcal{I})$ increases and our cost model indicates that the best safe region occurs when $b_m(\mathcal{R}, \mathcal{I}) = 1$. To test the optimality, we terminate the expansion at different values

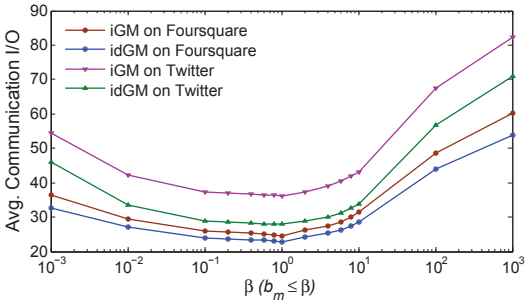
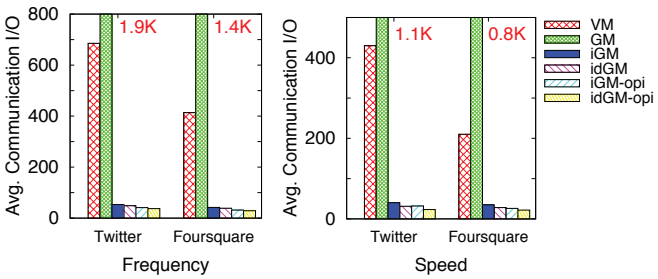


Figure 9: Optimality evaluation.



(a) Dynamic arrival rate (b) Dynamic speed

Figure 10: Adaptability evaluation.

of $b_m(\mathcal{R}, \mathcal{I})$ and this value is denoted by β. As shown in Fig. 9, we can see that when terminating too early ($\beta < 1$) or too late ($\beta > 1$), the performance is inferior to the case when $\beta = 1$.

Adaptability Evaluation. To evaluate the robustness of our cost model, we look into two parameters that may change frequently in reality: the event arrival rate f and the speed v_s (The other parameters are dependent on the area of safe region and impact region). We set up a dynamic environment in which f or v_s varies all the time. For comparison, we designed two optimal methods, iGM-opi and idGM-opi with the knowledge of future pattern update in advance. When f or v varies, the optimal methods would construct a new safe region accordingly using the new parameters and such safe region update is not counted as a new communication I/O in the experiments. Fig. 10(a) and Fig. 10(b) show the experimental results with dynamic f and v_s respectively. We gradually increase the event arrival rate f from $0/tm$ to $500/tm$ and then reduce it back to $0/tm$. This process is repeated 10 times. The dynamic moving speed v_s is set in a similar way ($0 \rightarrow 100m/tm \rightarrow 0m/tm$). As shown in Figure 15, iGM and idGM can achieve a comparable performance as iGM-opt and idGM-opt due to the good adaptability, because they can adapt to the update of f and v_s when the update causes a communication. This shows that our methods are robust to different user motion and event arrival patterns.

6.3 Spatial Boolean Expression Matching

In this section, we study the efficiency in spatial boolean expression matching, which can be further categorized into the elapsed time on the boolean expression match and that on the spatial match, namely BE and Spatial respectively. In particular, we use the Twitter dataset to compare our proposed BEQ-Tree with several baseline indexes as described at the beginning of Section 6.

Effect of the number of events $\mathbb{E}$. We evaluate the average matching time when increasing the number of events. From the results presented in Fig. 8(a), we have the following findings: (1) Quadtree can do the spatial match quickly, but it needs much time to filter candidate events. (2) k-index and OpIndex need more time for spatial match compared to Quadtree. (3) BEQ-Tree outperforms the other methods, and exhibits a 97% better matching time as compared to the next best algorithm. This is because it utilizes a Quadtree-like structure in the first layer that exhibits good pruning power, and maintains a sorted inverted list in the second layer, with which fewer events are examined. As shown, BEQ-Tree can answer the subscription matching within a few microseconds for a 20 million dataset, which is highly favored in real applications.

Effect of the subscription size δ. Fig. 8(b) presents the elapsed time w.r.t the varying subscription size δ. We make three observations. (1) Overall, BEQ-Tree achieves a speedup of 20 times w.r.t. Quadtree, k-index and OpIndex respectively. (2) Regarding the boolean expression match (BE) time, more attributes are involved in the match as δ increases, resulting in a higher computation time. (3) Regarding the spatial match time, k-index and OpIndex are sensitive to the subscription size, because they generate fewer candidate events as δ increases.

Effect of the notification radius r. The matching performance w.r.t. varying notification radius r is presented in Fig. 8(c). Only Quadtree is sensitive to r, because more candidate events are generated in the first step. In contrast, the performance of BEQ-Tree is stable and scales very well with r.

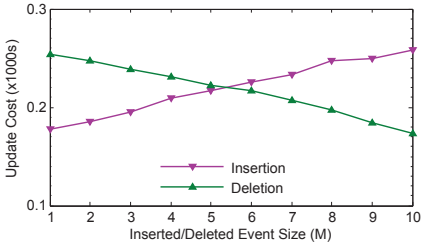


Figure 11: Update cost for BEQ-Tree.

BEQ-Tree Update Cost. At last, Fig. 11 shows the update cost for BEQ-Tree. To test the insertion cost, we start from a BEQ-Tree with 20 million events and incrementally insert 10 million events. The total time of inserting every 1 million tuples is plotted in Figure 11. We can see that as the size BEQ-Tree increases, it takes more time to insert the same number of tuples because the tree height also increases. The deletion process starts from a BEQ-Tree with 30 million events and the cost for deleting every 1 million records is reported in Figure 11 as well. We can see that as more records are deleted, the tree becomes smaller and the deletion operation becomes faster. In both cases, it takes less than 300 seconds to delete or insert 1 million events, which means our BEQ-Tree is efficient in update.

7. CONCLUSION

In this paper, we build a novel location-aware pub/sub system, Elaps, which takes into account continuous moving queries over dynamic event streams. To reduce communication overhead, we propose a concept named impact region and a novel cost model. Based on the cost model, we propose two incremental methods to construct the safe region and impact region. To reduce the response time of Elaps, we propose a novel index BEQ-Tree which can support efficient spatial subscription matching over a collection of events in the dynamic event environment. Experimental results on real datasets show that Elaps can greatly reduce the communication overhead and disseminate events to users in real-time.

8. 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. Guoliang Li is partly supported by the Chinese Special Project of Science and Technology (2013zx01039-002-002), the NSFC project (61422205, 61472198) and the 973 Program of China (2015CB358700).

9. REFERENCES

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

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

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

[4] G. Cugola and A. Margara, "High-performance location-aware publish-subscribe on gpus," in *Middleware*, 2012, pp. 312–331.

[5] J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D. L. Lee, "Location-based spatial queries," in *SIGMOD*, 2003, pp. 443–454.

[6] M. Hasan, M. A. Cheema, X. Lin, and Y. Zhang, "Efficient construction of safe regions for moving knn queries over dynamic datasets," in *SSTD*, 2009, pp. 373–379.

[7] B. Bamba, L. Liu, A. Iyengar, and P. S. Yu, "Safe region techniques for fast spatial alarm evaluation," *Georgia Institute of Technology*, 2008.

[8] 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.

[9] 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.

[10] 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.

[11] W. Xu, C.-Y. Chow, M. L. Yiu, Q. Li, and C. K. Poon, "Mobifeed: A location-aware news feed system for mobile users," in *SIGSPATIAL '12*, 2012, pp. 538–541.

[12] R. Finkel and J. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta Informatica*, vol. 4, no. 1, pp. 1–9, 1974.

[13] F. Fabret, H. A. Jacobsen, F. Llirbat, J. Pereira, K. A. Ross, and D. Shasha, "Filtering algorithms and implementation for very fast publish/subscribe systems," in *SIGMOD*, 2001, pp. 115–126.

[14] S. E. Whang, H. Garcia-Molina, C. Brower, J. Shanmugasundaram, S. Vassilvitskii, E. Vee, and R. Yerneni, "Indexing boolean expressions," *PVLDB*, vol. 2, no. 1, pp. 37–48, 2009.

[15] M. Sadoghi and H.-A. Jacobsen, "Be-tree: An index structure to efficiently match boolean expressions over high-dimensional discrete space," in *SIGMOD*, 2011, pp. 637–648.

[16] 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.

[17] H. V. Jagadish, B. C. Ooi, K.-L. Tan, C. Yu, and R. Zhang, "idistance: An adaptive b+-tree based indexing method for nearest neighbor search," *TODS*, vol. 30, no. 2, pp. 364–397, 2005.

[18] T. W. Yan and H. García-Molina, "Index structures for selective dissemination of information under the boolean model," *TODS*, vol. 19, no. 2, pp. 332–364, 1994.

[19] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *SIGMOD*, 1984, pp. 47–57.

[20] T. Brinkhoff, "A framework for generating network-based moving objects," *Geoinformatica*, vol. 6, no. 2, pp. 153–180, 2002.

[21] J. Zhou, A. K. Tung, W. Wu, and W. S. Ng, "A “semi-lazy” approach to probabilistic path prediction in dynamic environments," in *KDD '13*, 2013, pp. 748–756.

[22] G. Antoshenkov and M. Ziauddin, "Query processing and optimization in oracle rdb," *The VLDB Journal*, vol. 5, no. 4, pp. 229–237, 1996.

[23] K. Wu, E. J. Otoo, and A. Shoshani, "Optimizing bitmap indices with efficient compression," *ACM Trans. Database Syst.*, vol. 31, no. 1, pp. 1–38, Mar. 2006.

[24] F. Ramsak, V. Markl, R. Fenk, M. Zirkel, K. Elhardt, and R. Bayer, "Integrating the ub-tree into a database system kernel," in *VLDB '00*, 2000, pp. 263–272.

APPENDIX

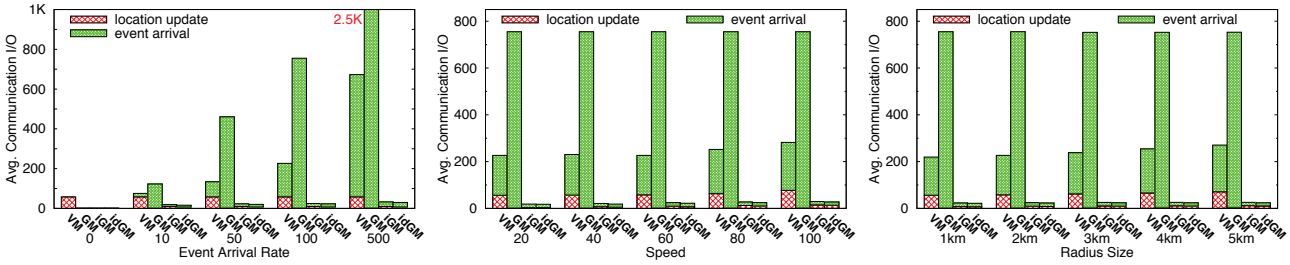
A. LIST OF SYMBOLS

Symbol	Description
s	a moving subscriber
e	a spatial event
$\mathcal{O}$	notification region specified by a subscriber
r	notification radius
$\mathcal{R}$	safe region of a subscriber
$\mathcal{I}$	impact region of a subscriber
f	event arrival rate
v_s	user moving speed
n	total number of events in the system
n_e	total number of matching events in $\mathcal{I}$
$\mathtt{d}(s, \mathcal{R})$	the minimum distance from s to the boundary of $\mathcal{R}$
c	a grid cell in iGM and idGM
G	a cell partition in BEQ-Tree
σ	a reference point in G
y	the spatial attribute for distance indexing
W	counter array in each cell partition in BEQ-Tree

Table 3: Summary of Notations

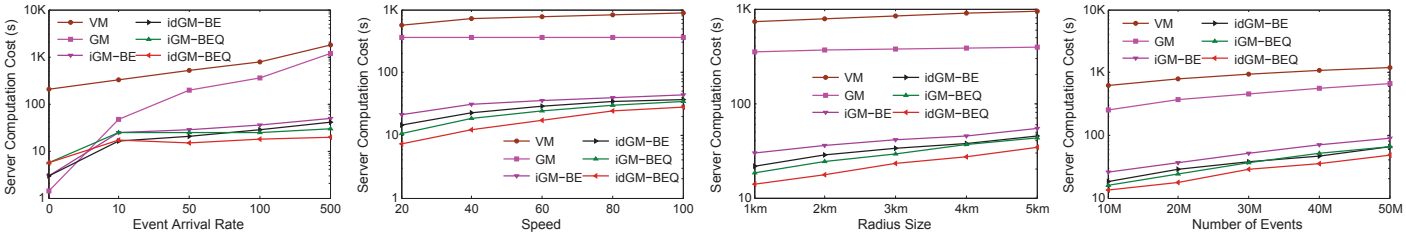
B. REPRESENTATION OF THE SAFE REGION

To reduce the bytes transferred in the communication between servers and subscribers, we use Bitmap as a more compact representation of safe region. We allocate a Bitmap whose length is the number of cells. If a cell is in the safe region, we set the corresponding element of the Bitmap to be 1. During the communication, we adopt the run-length encoding compression method (e.g., BBC [22] and WAH [23]) to further reduce the size of the Bitmap.



(a) Varing event arrival rate (b) Varing speed (c) Varing radius

Figure 12: Communication I/O on Foursquare.



(a) Varing event arrival rate (b) Varing speed (c) Varing radius (d) Varing number of events

Figure 13: Server computation cost for safe region construction.

In this way, we achieve a very compact representation of the safe region. When subscribers receive the message, they first decode it into the original Bitmap and can easily detect whether the cell they are located in is in the safe region or not.

In our implementation, we assign each cell a value derived from the z-ordering of the cells [24]. Based on the z-order, the cells close to each other will be assigned similar ids. Then, we use the classic WHC [23] to compress the list of ids. Our experimental results show that the size of compressed ids is around $5\% - 10\%$ of the original size.

C. INDEX MAINTENANCE OF BEQ-TREE

In our system, new events will be continually published by the publishers. Each event has a valid period and will expire after this period. Thus, BEQ-Tree should be efficient in terms of the maintenance cost.

We first consider how a new event is inserted into the BEQ-Tree index. Given an event E, we first find the cell with the lowest level that contains E. If the number of events within that cell is smaller than max_{event}, we append a new entry e in the associated counter array V_{G_i} and set its value to the location of E. For each non-spatial attribute, we insert its value and the key pointing to e into the corresponding tuple list. For the spatial attribute, we convert it to the one-dimensional distance y and insert y to the spatial list. If the cell is full, we need to partition it into four child cells and insert the event into the corresponding child cell.

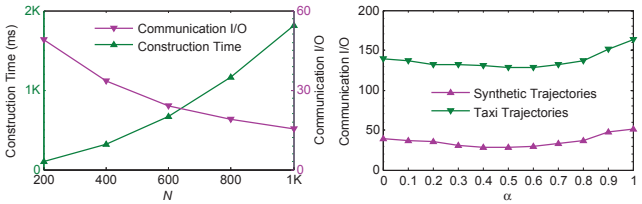
The delete operation is processed as follows. We first find the cell with the lowest level that contains E. Then, we traverse the inverted lists whose attribute is contained in E and delete the tuple. The deletion is fast because the list is sorted and we can use binary search to quickly identify the tuple to delete. If the cell becomes empty after deletion, we check whether its sibling nodes are also empty. If yes, we merge them to the parent node.

Memory Cost. Let E denote the set of spatial events published to Elaps, $\mathbb{E}$ denote the size of E, $|T|$ denote the total number of tuples in E and $|t|$ denote the memory space cost by an entry in the list. The event index contains two components: the tuple lists and the counter arrays. Each tuple corresponds to a unique entry in the tuple lists. Thus, the tuple lists occupy $O(|T||t|)$ memory space. In addition, each event corresponds to a unique entry in the counter arrays. Since the size of an entry in the counter arrays is much smaller than the size of an entry in the tuple list, the total memory cost for the index is $O(|T||t|)$. As can be seen, our index takes linear space cost. The index maintenance of BEQ-Tree is presented in Appendix C due to space limitations.

Update Complexity. Let $\mathbb{N}$ denote the maximum level of the BEQ-Tree, $|L|$ denote the maximum length of the tuple lists and $\mathbb{P}$ denote the maximum number of conjunctions in a subscription. The cost of Quadtree cell identification is $O(\mathbb{N})$. The insertion or deletion cost of a conjunction into a sorted list is $log(|L|)$. Thus, the total insertion or deletion complexity is $O(N) + O(\mathbb{P}log|L|) = O(\mathbb{P}log|L|)$, because usually $N \ll L$ in our BEQ-Tree. Note that with the help of the hierarchical structure in our index, $|L|$ will not be too large, making the index maintenance very efficient.

D. ADDITIONAL EXPERIMENTAL RESULTS

D.1 Parameter Tuning



(a) Varying N (b) Varying α

Figure 14: Parameter tuning for iGM and idGM.

In continuous moving query processing, our proposed iGM and idGM use small cells to approximate the safe region and impact region. Suppose the whole space is split into $N \times N$ cells and we increase N from 200 to 1000 to test how the performance varies when the cell size becomes smaller. As shown in Fig. 14(a), when N increases, we can represent the cell in a more precise fashion and the optimal safe region is better approximated. Thus, it can reduce communication I/O. However, it requires more construction time since we need more iterations to expand the safe region until the termination condition is satisfied. For the following experiments, we set N to be 600 as a tradeoff between communication I/O and CPU cost in safe region construction[5]

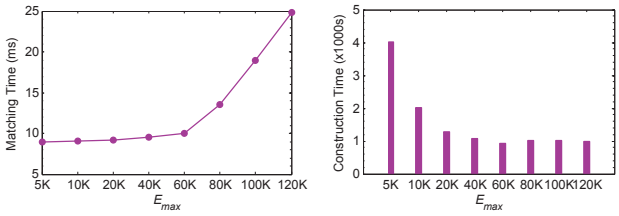
Another parameter in $idGM$ is α which reflects the confidence level of user moving pattern. In the synthetic trajectories, we increase α from 0 to 1.0. If $\alpha = 0$, idGM degrades to iGM. We can see that direction is a factor that can improve system performance. The optimal performance occurs when $\alpha = 0.5$. If α is set very high, the performance is bad because the system assumes that the subscriber always moves along the original direction and is likely to construct a safe region along the direction. Then, it is easy for the subscriber to move out of the safe region when he changes the moving direction.

For BEQ-Tree, we tune the parameter E_{max} which is the maximum number of events in a cell. Fig. 15(a) and Fig. 15(b) illustrate the matching time and construction time of BEQ-Tree when varying E_{max}. When E_{max} increases, the cell becomes larger and the effect of spatial pruning degrades. Thus, it takes more matching time (Fig. 15(a)). However, a smaller E_{max} results in more levels in the BEQ-Tree, which increases the update cost (Fig. 15(b)). Therefor, we set E_{max} to be 60K as a good tradeoff between subscription matching time and event update cost.

D.2 Experiments on Foursquare

Foursquare Dataset. In Foursquare, each venue is considered as one spatial event. We extract structured information, i.e., attribute-value pairs, from the venues. Each venue has around 50 attributes and we harvest $1,832,418$ venues. Among them, we use 1.5 million venues as the event database and the remaining to simulate the event stream. To generate the subscriptions over the Foursquare venue stream, we let the subscriptions follow the same distribution as the venues. In other words, if an attribute is frequent in the venues, it is also frequent in the subscriptions. The operators and operands in the predicates are synthetically attached.

[5]The parameter tuning in this section hold for all the data set and experiments.



(a) Matching Time (b) Index Construction Time

Figure 15: Parameter tuning for BEQ-Tree.

Fig. 12 shows the average communication I/O on Foursquare dataset. As shown, the performance is similar with that tested on the Twitter dataset. iGM and idGM can reduce the communication cost incurred by event arrival significantly, and achieve a comparable performance as GM to constrol the communication cost incurred by location update.

D.3 Server Computation Cost

In the following, we report the average server computation cost for safe region and impact region construction during the experiment period (i.e., 1000 timestamps), which have not been included into the main experimental section due to space limitations. For VM, GM, iGM-BE, idGM-BE, we use k-index to find all the matching events first and then construct the safe region and impact region. For iGM-BEQ and idGM-BEQ, we use BEQ-Tree to find the set of matching events on demand, as described in Section 4.2. The results are shown in Fig. 13. We have the following observations. (1) VM performs worse than GM in terms of computation cost although VM incurs less communication cost than GM, because VM needs more time to construct the safe region and impact region. (2) iGM and idGM can outperform VM and GM by one order of magnitude, because each communication would trigger an update for the safe region and impact region while the communication cost of iGM and idGM can be reduced significantly with the guide of the cost model. (3) iGM-BEQ and idGM-BEQ show superior performance than iGM-BE and idGM-BE, because our safe region construction methods usually do not expand to the whole space and we can use BEQ-Tree to traverse only a part of the whole space. In addition, the advantage is more obvious when the event arrival rate is higher or the number of events is larger, making our system more scalable.