SENSTRAC: Scalable Querying of SENSor Networks from Mobile Platforms Using TRACking-Style Queries

Stefan Pleisch Ken Birman
Department of Computer Science, Cornell University, Ithaca, NY 14853
{pleisch | ken}@cs.cornell.edu

***Abstract*— Future applications running on mobile platforms will sometimes need to query sensors and track sensor data over time. This paper uses the publish-subscribe paradigm as a natural solution to querying sensors from mobile platforms, and proposes a scalable approach to implement publish-subcribe, driven by the querying application. Our approach is evaluated by simulation, focusing on scalability.**

I. Introduction

With the widespread availability of wireless technology and the deployment of an increasing variety of sensors, information generated by sensors is becoming available to applications running on mobile nodes. Retrieving this information in a reliable, efficient manner will be an important building block for many applications. However, unreliable, low bandwidth communication links and node mobility make efficient, reliable, and scalable information retrieval a challenge.

Consider the example of an amusement park or urban environment, in which a large number of visitors moves around. Every visitor carries a PDA or cell phone running the relevant monitoring application (called query node). This application allows the user to query the environment in order to ask about queue length, waiting time, ride status, directions (e.g., shortest path), etc. On his PDA, the monitoring application constantly updates the results of his query. The user is generally interested in the sensors within a certain area, and this area moves with him.

In this scenario, the query nodes want to track the data of particular sensors over time. For instance, the length of a waiting queue may be interesting for the query node when it drops below a certain threshold. Hence, the query node needs somehow to be notified when this is the case. Note that queries can be much more complex, involving multiple sensors or types of sensors. While traditional approaches typically assume power-constrained sensor nodes and thus try to minimize the number of sent packets, the applications and sensors we have in mind will not be that power-constrained, although they will need to cope with wireless communication issues. For instance, sensors can be integrated into devices that are connected to electrical power, such as light bulbs. Moreover, critical sensors such as the ones used for disaster response will probably have sufficient power during periods of activity.

Traditional approaches (e.g., [1]) generally use so-called *in-network processing and aggregation*, by which the query is flooded within a certain area and a routing tree rooted at the query node is used to return the result to the query node. If the query permits, the results are aggregated on their way to the query node. This type of aggregation, which we call *intra-query aggregation*, reduces the size of the query reply and thus scales well in the number of considered sensors. However, these approaches generally only consider stationary query nodes.

Inspired by the work in [2] and [3], we propose to use the publish-subscribe paradigm to approach the problem of querying sensors from mobile nodes. In our approach, a query node periodically runs an algorithm to identify the sensors it wishes to track. It then ”subscribes” to updates, which these sensors periodically “publish”. The query node is now able to repeatedly evaluate the query, presumably updating a map or other application-specific user interface. At some second frequency, the query node recomputes the sensors of interest. Thus perhaps the query node decides which sensors to monitor every minute, but the sensor nodes send updates every few seconds. In a sense, instead of intra-query aggregation we transform the query into subscriptions to sensor updates and aggregate at the level of these subscriptions. This allows us to scale up to a large number of query nodes, as query nodes take advantage of and share preexisting subscriptions.

The paper's main contribution is the novel, highly scalable mapping from queries to topics and the corresponding underlying sensor network structure. More specifically, we show that structuring the sensor network into a regular grid provides a convenient underlying network structure for this class of applications. The mapping is entirely driven by the querying application. Our performance simulation measures the impact of query node mobility and the number of considered sensors on the quality of the query result and the message overhead. It shows that our approach scales well even for large numbers of query nodes.

The remainder of the paper is structured as follows: In Section II we define a query model. Section III discusses how to generate the result of the query, in particular the mapping from query to topic and from topic to the underlying sensor network. In Section IV, we present SENSTRAC and we show the corresponding simulation results in Section V. Section VI puts our work in larger context with existing work and Section VII concludes the paper.

II. Query Model

We consider queries that track sensor data over time; as updates are received the result of the query continuously evolves.

Typically, queries are constrained by geographical boundaries - generally somewhere in the proximity of the query node. The geographical boundaries are defined by the query node's *area of interest (AoI)*. For simplicity, we consider a two-dimensional area of interest, which is represented by the smallest square[1] that includes all affected sensors; the generalization to three dimensions is straightforward. The AoI generally moves with the query node.

We also assume that the sets of sensors upon which a query depends overlap considerably between two instances of a query from the same query node, unless the AoI is explicitly reassigned by the query node. In intuitive terms, we assume that the query nodes move at a moderate speed (e.g., walking or running speed), although our approach could also support faster-moving query nodes. Generally, the query node speed that can be supported also depends on the transmission range. The larger the transmission range, the longer two nodes are likely to be within each other's transmission range.

We support any query that depends on current and future values of a set of sensors; queries that ask for sensed values in the past are not supported. Once a query is running, sensor data is tracked as it evolves, however, hence queries can depend on a sequence of values from a sensor. For instance, the query node can ask to be notified when any temperature sensor detects a drop exceeding 60 degrees F.

Clearly, not all queries can be answered with the same efficiency and accuracy. In our model, accuracy is a complicated function of update rate, mobility rate, transmission ranges, and network load; here, we explore the question using simulations.

Queries are expressed in traditional SQL syntax [4]. They depend on a set of tables defined across the sensor network that conceptually represent the available data. The core table, called SensorTypes, indicates in its columns the sensor ID, the x and y coordinates of its position, and its type (e.g., temperature, queueLength). This table (and other tables defined below) are not "stored" at any single location. Instead, they can be considered as "virtual" tables. Indeed, the way the information in these tables is collected and stored is the main focus of this paper.

For instance, the following query returns the temperature, ID, and location (in x and y coordinates) of all sensors whose x and y coordinates lie in the interval $[xcoord1 \ldots xcoord2]$ and $[ycoord1 \ldots ycoord2]$, respectively.

```
SELECT Temperature.VALUE, Temperature.SID,
       SensorTypes.LOCX, SensorTypes.LOCY
FROM Temperature, SensorTypes
WHERE Temperature.SID = SensorTypes.SID
 AND SensorTypes.LOCX BETWEEN xcoord1 AND xcoord2
 AND SensorTypes.LOCY BETWEEN ycoord1 AND ycoord2
 AND VALUE > 100F
```

III. Querying the Sensors

Traditionally, a tree-based in-network aggregation protocol queries sensors by flooding the entire query within the AoI. While this works well for single or small number of subscribers, it does not leverage the fact that multiple subscribers may require a reply from the same sensor. In other words, it is generally very hard to combine the queries from two different subscribers. Hence, such an approach does not scale well in the number of subscribers.

Moreover, tree-based in-network aggregation works best if the AoI is centered around the query node. However, we can easily imagine cases in which the AoI may not include the location of the query node and may not be directly accessible from the query node, e.g., if it is located in the next valley in a mountainous region.

[1]The reason for using a square becomes apparent later in the paper. It is related to the fact that cells are represented as rectangles.

In this case, flooding is no longer the most suitable and efficient query distribution mechanism.

The alternative explored here uses the publish-subscribe (pub/sub) communication paradigm to collect query results. Pub/sub is a widely used communication paradigm and a variety of specifications have appeared over time (e.g., [5]). Its most important feature is the decoupling of the message sender from the receivers, and the asynchronous nature of the communication. In its simplest form, publishers (the sensors in our case) publish messages to a particular *topic*, while subscribers (query nodes) subscribe to all the topics that match their interests. The pub/sub system acts as an intermediary, hence the publisher does not need to know the subscribers. If the subscriber looses interest in the messages published on a particular topic, it unsubscribes from the topic. Many pub/sub systems provide message filtering on topics, which allows a subscriber to specify his interest in more detail. Only messages that match the topic and also pass the filter are delivered to the subscribers.

In contrast to the traditional pub/sub systems, our subscriptions are lease based, i.e., they time out after a certain time. As a consequence, we do not explicitly support the unsubscribe method traditionally present in pub/sub systems. Indeed, unsubscribe only makes sense if it is guaranteed that the unsubscribe eventually removes all subscriptions. In a mobile system, this cannot be ensured and thus seems too much of a constraint to be supported.

Querying sensors using the pub/sub paradigm requires that we implement two different mappings: (1) the mapping from the query to topic subscriptions, and (2) the mapping from topics to the actual sensors. These two mappings are not entirely independent; a particular choice in one mapping may influence the choices in the other. While (1) is a more abstract, high-level mapping (Section III-A), (2) is concerned with implementing pub/sub and thus structuring the sensor network (Section III-B) to provide efficient pub/sub.

A. Mapping Queries to Topics

The mapping of queries onto topics involves a trade-off between the number of subscribed topics and the number of messages unnecessarily delivered to the subscriber. Every topic relevant to a query incurs the cost of subscribing to it. In contrast, subscribing to a topic to which sensors not included by the query will publish values can result in situations in which undesired messages are delivered to a query node, forcing it to filter and discard them.

Since for any query the primary selection criteria on the sensors is the area of interest, it makes sense to first group the sensors into geographic areas and then to assign topics within each area. This limits the geographic span of the sensors that publish to a topic, improving the scalability in the size of the sensor field. The difficulty here is to find a geographical grouping that maps closest most query nodes' typical area of interest. The simple grouping considered here is that of a grid of cells. Within a cell, sensors can then be organized into groups according to their types. Using this grouping, which is apriori known to all nodes, queries can be quite easily mapped onto topics. To simplify the identification of topics and to avoid having to explicitly send topic names around in the network, we define a one-to-one mapping between topic names, types, IDs, and geographical regions. Assume, for instance, that a topic is defined for every cell in a grid overlaying the sensors. Such a topic could have the same name as the corresponding cell, e.g., *B-4*, while temperature sensors in this cell publish to topic */B-4/Temperature*. Revisiting the SQL query in Section II results in a subscription to topics */B-4/Temperature, /B-5/Temperature, /C-4/Temperature* and */C-5/Temperature*, assuming that the AoI is covered by cells *B-4,B-5,C-4* and *C-5*.

B. Structuring the Sensor Network

In this section, we start by briefly exploring two routing tree-based pub/sub architectures and giving reasons that make them not practical in our setting. We then present our broker-based pub/sub architecture for query applications in sensor networks.

Sensors and query nodes communicate by establishing (mobile) wireless ad hoc networks. Nodes within transmission range of each other can communicate directly. More distant nodes rely on other nodes to forward messages. A routing protocol sets up a route between a sender and a distant destination.

a) Tree-based architectures: In such settings, pub/sub can be built over overlay routing trees among subscribers and publishers. These routing trees are either (1) rooted at the subscribers, or (2) rooted at every publisher.

Approach (1), inspired by prior research on in-network aggregation [6], [1], constructs a tree for every subscriber (see Fig. 1(i)). The subscriber floods its subscriptions into the sensor network, with the relevant publishers responding along a tree structure implicitly built during the flooding phase. This scales well in the number of

included publishers, but not in the number of subscribers, due to the need for one flooding broadcast per issued subscription. It is suitable for stationary subscribers, but lacks appropriate support for their mobility, making tracking sensors over time costly. To adjust to subscriber mobility, one could imagine a so-called proxy root that forwards the results to the subscriber. This requires that a path exists from the proxy to the subscriber. Moreover, every subscriber potentially uses another proxy, which does not scale well, raising again route discovery and maintenance issues.

Approach (2), used in [2], [3], constructs one routing tree per (group of) publisher (see Fig. 1(ii)), in which the subscribers are generally located at the leaves of the tree (some subscribers may act as intermediate nodes). This scales well in the number of subscribers, but not in the number of publishers. If a subscriber moves, then the routing tree of every included publisher needs to be adjusted, which leads to a high route maintenance overhead. Moreover, route discovery requires that the publishers find at least one node that knows a route to the tree. If the publisher is very far from the subscriber, route discovery can become expensive. Both Approach (1) and (2) are not explored any further. A representative finding, comparing Approach (1) with the algorithm we favored, appears in Fig. 2(e).

b) Broker-based architecture: Having ruled out the use of these architectures, we now present the architecture that we will explore further. To address the route discovery and maintenance issues we use a broker-based architecture (see Fig. 1(iii)), in which some sensors are designated as brokers that take over the role of routers. The first issue that arises is how to position the brokers in the network. As queries use the AoI as primary criterion (see Section II), we cluster the network into geographic regions, corresponding to the static grid cells defined in Section III-A. We thereby assume that the sensors know their approximate coordinates, either by using on-board GPS or by inferring their location from their neighbors.

The important advantage of static in contrast to dynamic clustering is that every node a priori knows the clustering and can compute another node's cell ID solely based on this node's coordinates.

Each cell/cluster contains one or, in the case of partitions within the cell, multiple brokers. Publishers send updates to the broker in their cell (or in their partition within their cell). Subscribers send their subscriptions to any close broker (see Fig. 1(iii)). The brokers communicate updates and subscriptions among them, along the "logical" communication links represented by dashed lines in Fig. 1(iii). Hence, the communication between cells is routed through the brokers. This has the advantage that updates are sent as a single copy between brokers and only need to be duplicated for the last part of the routing path, namely from the broker to the subscribers. Thus, the route maintenance cost between brokers is distributed over multiple subscriptions shared by many subscribers. Moreover, the broker does not need to forward a new subscription if it already has existing subscriptions that consider all topics that are in the new one. Using an overlay network that primarily routes messages between brokers keeps a large part of the routing infrastructure stable and only modifies the relatively short routes that suffer from mobility, i.e., the routes between subscribers and brokers.

Moreover, the brokers will eventually be well-known in the sensor network and many sensors will have routing information to the brokers in their routing table, making route discovery very efficient.

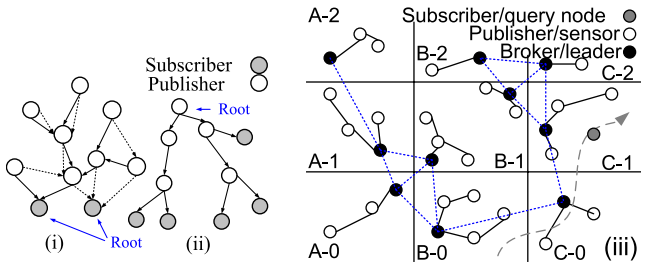


Fig. 1. Options for structuring the network.

IV. SENSTRAC

We distinguish between *intra-cell* and *inter-cell* routing, using a variant of the landmark hierarchy [7]. Intra-cell routing addresses the routing infrastructure among the (sensor) nodes within a particular cell. Inter-cell routing, in contrast, governs the routing among the cells.

c) Intra-Cell Routing.: The scope of messages sent in the context of intra-cell routing is limited to that particular cell. In the intra-cell routing scheme, we rely on a *leader* node. Leaders handle routing, while brokers are query processing intermediaries. Although we will use the leaders as brokers in our implementation, we use different terms because one could imagine using different nodes for these different roles. The leader is selected dynamically, according to a certain deterministic criteria such as lowest node ID. The sensors in the same cell build a shortest path tree rooted at the cell leader (see Fig. 1(iii)). Leaders can change over time to handle failures and load balancing [8].

Partitions within a single cell in the sensor network may lead to multiple leaders (e.g., Cell B-1 in Fig. 1(iii)).

These partitions may have been artificially introduced by subdividing the grid into cells.

d) Inter-Cell Routing.: Inter-cell routing governs communication among the leaders of different cells. All messages are sent from and directed to a leader. We use two mechanisms by which leaders learn of the existence of other leaders. The first periodically forwards leader hello messages overheard from neighboring cells to the leader of this cell. We call the leaders of neighboring cells, of which the leader learns, *neighbor leaders*.

The second is based on gossiping leader information among neighbor leaders. Periodically, a leader chooses a random subset of all leaders it is aware of within a certain range (measured in leader hops, i.e., the number of intermediate leaders before getting to the destination leader) and sends it to a random number of its neighbor leaders. Gossiping leader information results in an overlay mesh among the leaders (see Fig. 1(iii)).

We use the Ad hoc On-demand Distance Vector (AODV) [9] routing protocol for gossiping and, in general, for all messages (e.g., also updates) sent between two leaders. Clearly, other routing protocols are also possible (e.g., [10]).

A. Implementing Pub/Sub

A broker channels subscriptions and updates between interested parties. As noted, our implementation reuses the leaders for this second role. Sensors publish their updates by sending them to the cell leader (broker). Query nodes send their subscriptions to the closest broker, usually the one within the same cell.

Upon reception of a subscription, the broker adds the subscription to its subscription list. Then, it maps the subscription to the cells and subscribes to the broker that is on the shortest path to the broker(s) in the corresponding cells, using inter-cell communication. The brokers remember active subscriptions to avoid loops.

Although loops are prevented in the subscription mechanism, update messages may still loop. We cache update messages for some time to prevent any broker from forwarding an update message more than once.

Having forwarded a subscription for a topic to another broker, a broker will not forward any new subscriptions on the same topic for some time t. This allows the broker to take advantage of already existing subscriptions and reduces the number of subscriptions sent between brokers. However, if the first subscription is lost, then no updates published to this topic will arrive during time t and the time until a new subscription is received by the broker. Hence, the value chosen for t balances a tradeoff between reducing the number of sent subscriptions and the consequences of subscription message loss.

B. Routing Between Query Node and Broker

The query node periodically queries its neighboring sensors for their leader/broker. The routing between query node and broker takes advantage of the fact that each sensor node knows a route to the cell leader (broker) and thus easily can route the message from the query node to its leader. While the routing path from the query node to the leader is given, the inverse is not true. To forward the updates to the query node, we use the following approach: When intermediate sensor nodes forward subscriptions from the query node to the broker, they store a copy locally.[2] They then use the stored subscriptions to determine whether or not to forward an update from the broker to the query node. The advantage of using broadcasts rather than point-to-point communication is that a parent node only broadcasts an update once, instead of sending it to all its children sensor nodes one after the other. Thus, the parent sensor node in the routing tree does not explicitly know its children nodes. Rather, it forwards the update based on the currently active subscriptions and the corresponding filters. This routing scheme is different from the approaches traditionally used in multicast trees.

When the query node moves, the routing path to the leader may be broken. To detect a link breakage, the first sensor in the routing path from the query node to the leader periodically broadcasts a hello message during periods of broadcast inactivity. If the query node detects a link breakage, it tries to establish a new route to the same broker, resending a previous subscription, and to a new broker if this is not possible.

V. Simulation

For our simulation we used JiST/SWANS v1.0.4 [11], [12], a simulation environment for ad hoc networks. Java applications written for a real deployment can be ported to the simulation environment and then placed under a variety of simulated scenarios and loads. Jist/Swans intercepts the calls to the communication layer and dynamically transforms them into calls to the simulator's communication package.

A. Setup

We consider a set of sensor and query nodes. While sensor nodes are stationary or relatively immobile, query

[2]To save memory, intermediate sensor nodes can also store an aggregate of the subscriptions and filters.

nodes are mobile. Communication between two nodes m and n occurs in an ad hoc manner with a transmission range of 88m. We use a CSMA MAC protocol as defined in the 802.11 standards [13], but without the RTS/CTS and ACK mechanism. Communication can be subject to interference, in which case the message cannot be received. Interference can occur without the sender being able to detect it (this is called the *hidden terminal problem* [14]).

Our work does not inject artificial packet loss, although we do model disconnections due to mobility, transmission range limits, and the hidden terminal problem just mentioned (using JiST/SWANS' *RadioNoiseIndep* package, which uses a radio model identical to ns-2 [15]). Unless otherwise mentioned, we use the default values defined in JiST/SWANS, such as a bandwidth of 1Mbs.

A total of 600 sensor nodes are uniformly distributed within the sensor field, which ensures, with high probability, that at least one route exists between any two sensor nodes. In the case of single query node, the query node moves along a straight line $[200, 200]$ to $[900, 900]$ in the field of 1200x1200m with origin $[0, 0]$. For multiple query nodes, we use the random waypoint model [16] with a fixed speed and zero pause time, thereby removing the randomness caused by varying speeds and pause times.[3] We limit the scope of the query node's movement such that its AoI does not extend beyond the field boundaries. This prevents unrealistic results due to the measurement setup.

The sensor and query nodes start up at random times, uniformly distributed between 0 and 100s, and positions. When they are all up and running, we start our measurements (also called steady-state simulation).

B. Results

In this section, we present the results of our simulation. The query node periodically, every 60s, sends its query, i.e., subscribes to the relevant sensors. The sensors publish their current value every 50s (we vary this value in the simulation displayed in Fig. 2(b)). To enable the evaluation of our approach, every sensor reports the current time as its value.

This value is then used to measure three types of sensor coverage, i.e., the sensors from which the query node received an update of the corresponding type over all sensors currently in the AoI: *recent* denotes the ratio of sensors for which the query node has received an update within the latest subscription period (known value is less than 60s old); *1-stale* the ratio from which the query node has missed the update in the latest subscription period (60 to 120s old); and *n-stale* if the latest update is older (more than 120s old). N-stale is generally omitted for space reasons. The sum of recent, 1-stale, and n-stale sensor coverage is 1, meaning that 100% of the sensors in the AoI are covered.

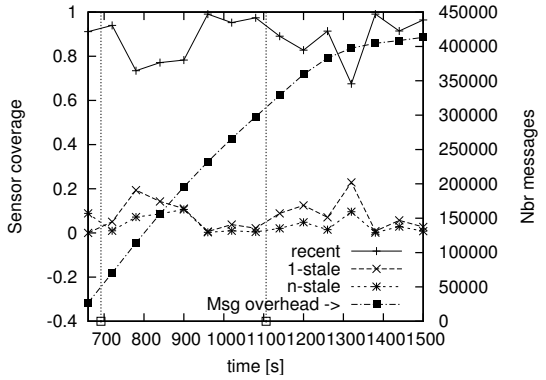
The cell size is set to 300m. The default AoI contains all sensors with coordinates $600 \leq x, y \leq 900$ (corresponding to a single cell), amounting to roughly 40 to 50 sensors with a total number of 600 sensors, or with distance smaller than 200m from the query node's position. All results give the average over 20 runs in different uniform sensor distributions. The approximate 95%-confidence intervals (CI) are: ∓ 0.05 for sensor coverage in the case of single query nodes; ∓ 0.1 for single query node with moving AoI; ∓ 0.001 for multiple query nodes; and $\mp 5\%$ for message overhead measurements.

Unless explicitly stated otherwise, we use the above default values in our measurements.

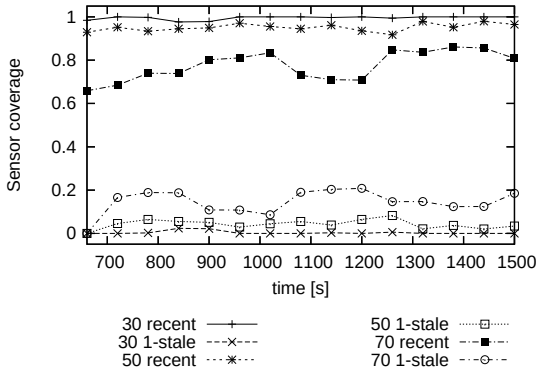
1) Mobility: We first look at the effects of the query node mobility on sensor coverage. The graph in Fig. 2(a) shows the case of a single query node travelling through the sensor field along a straight line. In this simulation run, the AoI travels with the query node. More specifically, the query node is interested in all the sensors that are within 200m of its current position. As the moving AoI may involve at times more than one cell it requires to contact more than one broker. As a consequence, the sensor coverage fluctuates to some degree. In the same graph, the forth curve (y2-axis) illustrates the lease-based nature of SENSTRAC. It displays the message overhead over time with 100 query nodes and shows that the increase in the message overhead eventually diminishes once all the subscriptions time out and update messages are no longer sent to the query nodes (around 1200s).

The next two graphs show the impact of the update rate and the speed on sensor coverage. In Fig. 2(b) we use the same setup as in Fig. 2(a), but with stationary AoI. The sensors send an update every 30, 50, and 70 seconds. It comes as no surprise that the recent sensor coverage decreases with decreasing update rate. Similarly, the recent sensor coverage generally decreases with increasing speed of the query node. The results of this experiment, with the query node moving along a straight line but with a stationary AoI, are shown in Fig. 2(c). Notice that query nodes with higher speeds reach the end point of the line (at location [900,900])

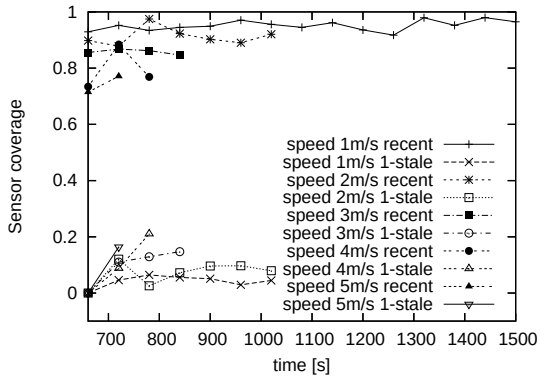
[3]Note that in [17] it has been shown that the random waypoint model is not entirely appropriate. However, for our measurements, this has no immediate impact.



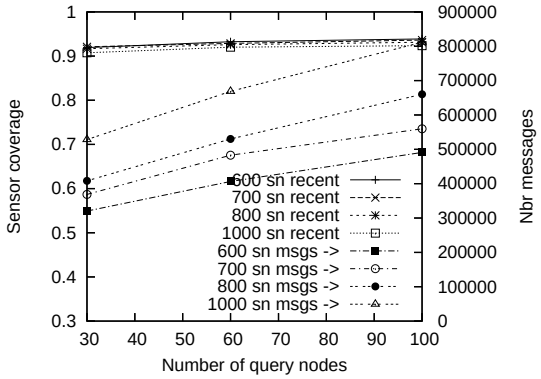
(a) Single query node and AoI move along same line.



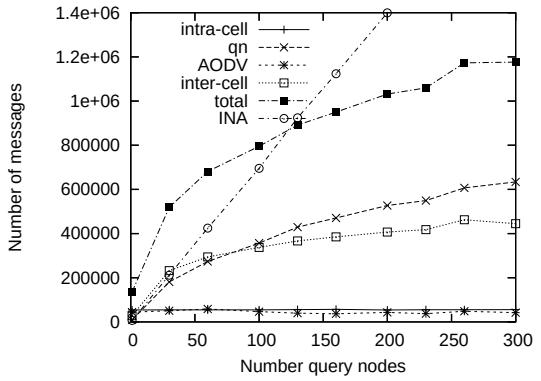
(b) Single query node, varying update rate (in seconds).



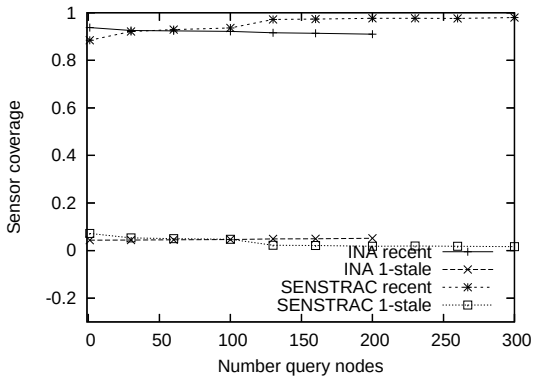
(c) Single query node, varying speed.



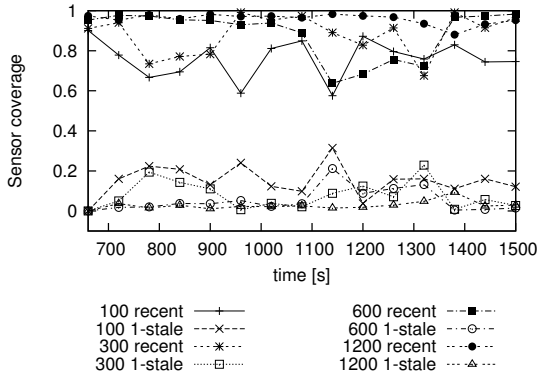
(d) Mobile AoI.



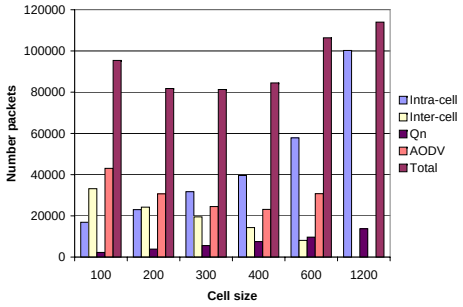
(e) 700 sn, mobile AoI, comparison with INA.



(f) 700sn, mobile AoI, comparison with INA



(g) AoI moves with qn, varying cell sizes.



(h) Message overhead with varying cell sizes.

Fig. 2. (a)-(c),(g),(h): Single query node with speed 1m/s. (d),(e),(f): Multiple query nodes sending queries for stationary AoI and a speed 1m/s.

faster and thus the corresponding simulation run stops earlier.

2) Scalability: We now evaluate the scalability of SENSTRAC with respect to the number of sensor and query nodes. Fig. 2(d) shows the sensor coverage (y1-axis) and the message overhead (y2-axis) averaged over time (1200 seconds) and all the query nodes. It shows that the message overhead increases with increasing number of sensor nodes. Clearly, the more sensor nodes are within the AoI, the more updates are routed to the query node. However, the sensor coverage is nearly the same for all the considered numbers of sensor nodes. Moreover, the message overhead increases with increasing number of query nodes.

In Fig. 2(e), we compare the message overhead of SENSTRAC with traditional tree-based in-network aggregation (INA). We consider 700 sensors and use the same AoI with radius 200m and centered at the query node's position for both, and we do not aggregate query replies. Notice that the choice of this particular AoI is biased towards the INA approach. Since the INA approach uses polling, we flood the query at double the frequency than in SENSTRAC. Still, an updated sensor value may not be discovered in the INA approach for 30s in the worst case. We can see that the message overhead for SENSTRAC increases fast for low numbers of query nodes, but then the increase diminishes with increasing number of query nodes (see curve labelled "total"). This is an indication for the scalability of our approach. Indeed, with a high number of query nodes it becomes more likely that a broker to which a query node sends a subscription already has an active subscription. This allows newly arriving query nodes to take advantage of existing subscriptions. In contrast, INA message overhead costs increase linearly with the number of query nodes. The two message types in SENSTRAC that add the most to the message overhead are the inter-cell messages and the query node messages (qn). The former result from sending updates between the brokers and include all the messages sent by intermediate hops, while the latter count the number of messages exchanged between query node and the broker, including intermediate nodes. Moreover, intra-cell packets are the messages needed to maintain the routing structure within a cell and the updates sent to the cell leader. Finally, the AODV packets indicate the number of packets sent to maintain/discover the routes between the cell brokers. Total packets gives the sum of these packets. Fig. 2(f) shows the corresponding sensor coverage. Notice that in the case of a single query node the sensor coverage for SENSTRAC considerably depends on the path chosen by the single query node (using the random waypoint model). Thus, the confidence interval is also greater (∓ 0.02) in this case.

3) Mapping of Query to Topics: In this section, we evaluate SENSTRAC with varying queries and cell sizes.

Varying cell size. Fig. 2(g) and Fig. 2(h) highlight the impact of the cell size on sensor coverage and message overhead. Consider the curve for cell size 600 in Fig. 2(g). At time 1106 the query node changes from one cell into another, right at the intersection of four cells. As a consequence, the resubscription process leads to a sharp decrease in sensor coverage. Before this, no cell changes occurred and the sensor coverage was very high in comparison with the graphs for smaller cell sizes. Fig. 2(h) measures the network load with respect to varying cell sizes. The considered packet types are the same as in Fig. 2(e). With increasing cell size, the number of intra-cell and qn packets increases and the number of inter-cell packets decreases. In the case of a single cell (of length 1200m), no inter-cell packets are sent. The AODV messages decrease as well, except in the case of four cells of length 600m. Here, the cell leaders are generally many hops away so that the route discovery becomes quite expensive. It may even involve sending multiple discovery messages with increasing hop count. Our chosen cell size is 300m, which has the lowest total message overhead. In the graph we do not consider the messages sent by the query node. Since we only consider a single query node, this number is very low and thus not significant.

In Fig. 3 we show the impact of the AoI on the quality of the query reply for 30 and 60 query nodes. As before, we select a stationary AoI that matches a cell. We then move the AoI – keeping its size unchanged – to cross one or four cell boundaries (labelled "1 c bdry" and "4 c bdries"). The results show virtually no increase in the message overhead, nor a decrease in the recent sensor coverage. Now, we increase the size of the AoI to four times the size of the original, and to the entire sensor field. Not surprisingly, the quality of the query reply decreases, while the message overhead increases.

Varying AoI size. Clearly, the size of the AoI has an impact on sensor coverage. Common sense dictates that sensor coverage decreases with increasing AoI size. Fig. 4 shows the impact of the AoI size on sensor coverage. Here, we consider a single query node moving in a straight line together with the AoI. The AoI radius varies between 200m and 600m. In general, no particular AoI size performs significantly better than the others.

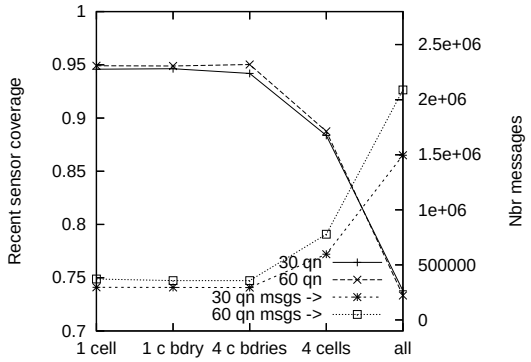


Fig. 3. Stationary AoI with varying size and position.

The reason for that is that even smaller AoIs may require the query node to subscribe to multiple brokers, and a larger AoI does not necessarily trigger subscriptions to more brokers. Thus, increasing the AoI has only minor impact on sensor coverage.

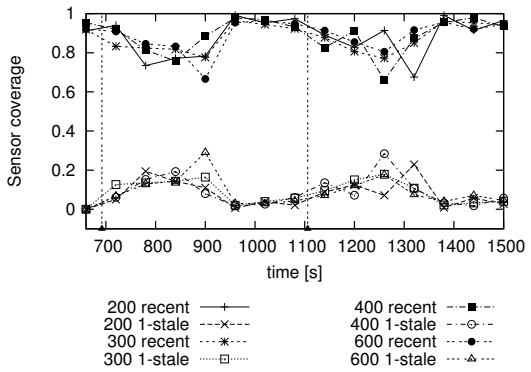


Fig. 4. AoI moves with the qn, varying sizes.

VI. Related Work

Our work draws from a large body of earlier work:

In the context of sensor networks, many approaches exist. We have already presented approaches based on in-network aggregation, e.g., [4], [6], [1].

Directed Diffusion [2] can be seen as a publish-subscribe mechanism, which is implemented using the tree-based architecture rooted at the publisher. We have discussed tree-based architectures in Section III-B and explained why we did not consider them in our work. Moreover, our simulations run with a much higher number of query nodes.

The ACQUIRE mechanism [18] uses the concept of a mobile agent and sends the query through the sensor network, while acquiring more and more results on the way. This may result in high latencies and large query result messages.

While these approaches all consider stationary query nodes, the following approaches address the issue of mobile query nodes.

In [3], Huang and Garcia-Molina explore different algorithms to construct routing trees rooted at the publishers. Again, the reader is referred to Section III-B for a discussion of tree-based architectures.

Mobile database access has been studied in [19]. However, Imielinski and Badrinath consider mobile users that access databases which are interconnected by a fixed network. Hence, their model is very different from the model considered in our work.

Pub/sub has become a mature technology in fixed networks. Research on pub/sub in mobile ad hoc networks is more recent and has mainly focused on mobile subscribers and publishers relying on a fixed broker infrastructure to support them. The focus of our work, in contrast, is not to come up with a new pub/sub implementation for ad-hoc networks. Rather, we show that pub/sub can effectively support querying sensor networks. Our contribution is thus the mapping and the application of pub/sub to querying applications. Most existing pub/sub systems provide generic pub/sub solutions and they do not consider the particular mapping to query applications, such as for instance the tight dependence on geographic regions. With this orientation, they naturally focus on how to forward notifications to the subscriber once the subscriber has moved (as suggested for instance in [20]). In our setting, these approaches are too costly given that the usefulness of an update expires after some time and leader-to-leader communication involves many intermediate nodes.

Yoneki and Bacon [21] have implemented pub/sub over mobile ad hoc networks. They assume that also the brokers are mobile and thus use ODMRP [22] to distribute the subscriptions to the brokers. This is costly, as it involves flooding. Moreover, it suffers from the other disadvantages discussed in Section III-B. In our approach, sensors are mostly stationary and thus we can avoid flooding huge areas of the network.

In [23], Kim et al. propose an approach to route messages to mobile subscribers (called sink in [23]), based on intermediary accessor nodes. These accessor nodes are defined by the sink. The protocol corresponds to the approach presented in Fig. 1(ii). Moreover, [23] does not consider the actual application, but just looks at a way to route the messages from a source to the mobile sink.

In recent years, many routing protocols based on hierarchical routing have been proposed, e.g., [8], [24]. While these routing protocols generally rely on clusters that are set up dynamically, our routing depends on a well-defined grid to form clusters. This clustering is

driven by the query applications we are considering.

VII. Conclusion

The paper has presented a natural and scalable way to query sensor networks from mobile platforms. We propose a layered approach in which the query application is mapped onto a pub/sub system. In a first step, we map queries to topics, and then show an architecture for pub/sub that is efficient in the setting we consider.

We have implemented the proposed architecture on the JiST/SWANS network simulator and have measured various aspects of our simulation. Our measurement results show the scalability of SENSTRAC with respect to the number of query nodes and its flexibility with respect to the AoI's position.

Although this paper limited itself to simulation, a merit of the JiST/SWANS technology is that the simulated code is executable on real platforms with only minor modifications. Accordingly, in our future work we hope to begin real experiments using actual sensors and query nodes.

Acknowledgments

The authors are very grateful to Rimon Barr for his help on JiST/SWANS, Robbert van Renesse for many discussions on routing in sensor networks and on pub/sub, and Rohan Murty for his implementation of MAODV. Our effort is supported by the Swiss National Science Foundation (SNF), NSF Trust STC, the NSP NetNOSS program, and the DARPA ACERT program.

References

[1] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "TAG: a tiny aggregation service for ad-hoc sensor networks," in *Proc. of 5th Symp. on Operating Systems Design and Implementation (OSDI'02)*, Boston, MA, Dec. 2002.

[2] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proc. of the 6th Int. Conf. on Mobile Computing and Networking (MobiCOM'00)*, Boston, MA, Aug. 2000.

[3] Y. Huang and H. Garcia-Molina, "Publish/subscribe tree construction in wireless ad-hoc networks," in *Proc. of the 4th Int. Conference on Mobile Data Management (MDM'03)*. London, UK: Springer-Verlag, 2003, pp. 122–140.

[4] P. Bonnet, J. Gehrke, and P. Seshadi, "Towards sensor database systems," in *Proc. of the Conference on Mobile Data Management*, Jan. 2001.

[5] B. Oki, M. Pflügl, A. Siegel, and D. Skeen, "The information bus - an architecture for extensible distributed systems," in *SOSP'93*, Asheville, NC, 1993, pp. 58–68.

[6] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan, "Building efficient wireless sensor networks with low-level naming," in *SOSP'01*, Banff, Alberta, CN, 2001, pp. 146–159. [Online]. Available: citeseer.ist.psu.edu/heidemann01building.html

[7] P. Tsuchiya, "The landmark hierarchy: a new hierarchy for routing in very large networks," in *Proc. of the Symp. on Communications Architectures and Protocols*. Stanford, CA, United States: ACM Press, 1988, pp. 35–42.

[8] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *HICSS'00*, Jan. 2000.

[9] C. Perkins and E. Royer, "Ad-Hoc On Demand Distance Vector Routing," in *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, Feb. 1999.

[10] J. Li, J. Jannotti, D. D. Couto, D. Karger, and R. Morris, "A scalable location service for geographic ad-hoc routing," in *Proc. of the 6th Int. Conference on Mobile Computing and Networking*, Aug. 2000, pp. 120–130.

[11] "JiST/SWANS," http://jist.ece.cs.cornell.edu.

[12] R. Barr, "An efficient, unifying approach to simulation using virtual machines," Ph.D. dissertation, Cornell University, Ithaca, NY, 14853, May 2004.

[13] IEEE, "802.11 specification (part 11): Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," June 1997.

[14] D. Allen, "Hidden terminal problems in wireless LAN's," in *IEEE 802.11 Working Group Papers*, 1993.

[15] "The network simulator - ns-2," http://www.isi.edu/nsnam/ns.

[16] D. Johnson and D. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, Imielinski and Korth, Eds. Kluwer Academic Publishers, 1996, vol. 353. [Online]. Available: citeseer.ist.psu.edu/johnson96dynamic.html

[17] J. Yoon, M. Liu, and B. Noble, "Random waypoint considered harmful," in *INFOCOM 2003*, Apr. 2003.

[18] N. Sadagopan, B. Krishnamachari, and A. Helmy, "The ACQUIRE mechanism for efficient querying in sensor networks," in *Proc. of the IEEE Int. Workshop on Sensor Network Protocols and Applications (SNPA'03)*, Anchorage, Alaska, May 2003.

[19] T. Imielinski and B. Badrinath, "Querying in highly mobile distributed environments," in *Proc. of the 18th Int. Conference on Very Large Data Bases*. Vancouver, Canada: Morgan Kaufmann Publishers Inc., 1992, pp. 41–52.

[20] A. W. M. Caporuscio, A. Carzaniga, "Design and evaluation of a support service for mobile, wireless publish/subscribe applications," *IEEE Transactions on Software Engineering*, vol. 29, no. 12, pp. 1059–1071, Dec. 2003.

[21] E. Yoneki and J. Bacon, "An adaptive approach to content-based subscription in mobile ad hoc networks," in *Proc. of the 2nd Conference on Pervasive Computing and Communications Workshops*, 2004, pp. 92–97.

[22] S. Lee, M. Gerla, and C. Chiang, "On-demand multicast routing protocol," in *Proc. IEEE Wireless Communications and Networking Conference, 1999 (WCNC. 1999)*, vol. 3, 1999, pp. 1298 –1302.

[23] H. Kim, T. Abdelzaher, and W. Kwon, "Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks," in *Proc. of 1st Int. Conf. on Embedded Networked Sensor Systems (SenSys'03)*. New York, NY, USA: ACM Press, 2003, pp. 193–204.

[24] X. Hong, L. Ma, and M. Gerla, "Multiple-landmark routing for large groups in ad hoc networks," in *Proc. of IEEE Military Communications Conferences (MILCOM 2002)*, Anaheim, CA, Oct. 2002.