Computer Communications 33 (2010) 1515–1523

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

A content-guided publish/subscribe mechanism for sensor networks without location information

Duc A. Tran *, Cuong Pham

Department of Computer Science, University of Massachusetts, Boston, MA 02125, USA

ARTICLE INFO

Article history:
Received 14 October 2009
Received in revised form 10 February 2010
Accepted 16 April 2010
Available online 22 April 2010

Keywords:
Publish/subscribe
Sensor networks
Search

ABSTRACT

The publish/subscribe paradigm represents a large class of applications in sensor networks as sensors are designed mainly to detect and notify upon events of interests. Thus, it is important to design a publish/subscribe mechanism to enable such applications. Many existing solutions require that the sensor node locations be known, which is not possible for many sensor networks. Among those that do not require so, the common approach currently is to use an expensive gossip procedure to disseminate subscription queries and published events. We propose a solution that does not require location information yet aimed at better efficiency than the gossip-based approach. Our theoretical findings are complemented by a simulation-based evaluation.

1. Introduction

Leveraged by sensor technologies which allow for inexpensive sensors with increasing sensing capability, there is a growing trend to deploy sensor networks in the physical world to observe important happenings [1–5]. Essential to many of these networks is a publish/subscribe functionality that enables the users to subscribe to events of interest and the sensor nodes to publish event data, such that the users can be notified when the matching events occur. As an example, for disaster monitoring, a user can subscribe in advance with a query specifying the early warnings of a wildfire, so that any event matching these warnings, when detected by a sensor, can be published to the network quickly to notify the subscriber. In military, we can help our soldiers navigate more safely on a battlefield by deploying a sensor network that is notified upon events implying the existence of enemy forces.

It is, however, challenging to design an efficient publish/subscribe mechanism for sensor networks. A subscriber node, that is interested in some event, does not know whether the event may occur, when, and where. On the other hand, a publisher node – a sensor node that detects some event – does not know who among the nodes is interested in this event. For these nodes to find each other, the simplest way is via broadcast, in which each subscription query is sent to all the nodes in a broadcast tree [6,7], or, alternatively, each event is sent to all possible subscriber nodes [8]. This approach offers quick event notification but the communication cost due to broadcasting can be prohibitively high for large-scale sensor networks. On the other extreme, we could employ a central node to receive all the queries and event publications [7]. Although the communication involved is less, most of the overhead is on the nodes leading to the central node, making this area a severe bottleneck, especially if the number of queries and events is high.

Our focus is to design a distributed publish/subscribe mechanism for sensor networks that enables the subscribers and publishers to find each other quickly and efficiently. Given a query or an event, it should be sent to a set of selected nodes (not to one globally-known node, not to every node) so that if an event matches a query there exists a rendezvous node for both the event and the query. A question is, how to determine the rendezvous nodes? If these nodes are identified by their physical IDs, a query or event may be mapped to an ID that does not exist (because the node that initiates the query or event possesses only a local view of the network). If the sensor node locations are known, this problem can be avoided with a location-based method [9,10]: map the query or event to a set of geographic locations and publicize it at the nodes nearest these locations. Unfortunately, due to cost or environmental factors, not every sensor network is equipped with GPS-like capability. Although we can use a localization technique to estimate the node locations, this estimation is not free from error and can be very expensive [11]. A challenge, therefore, is to devise a solution that does not require knowledge about location, and this is applicable to a wider range of sensor networks.

Among the existing publish/subscribe mechanisms that do not require location information, the common approach is based on some form of gossip: the message about each query or event is spread throughout the network by gossiping via selected neighbor links (e.g., [12,13]) or by broadcasting via all the neighbor links but within a limited broadcasting scope (e.g., [14,15]). The general idea is that if we publicize a query to a set of random nodes in the

* Corresponding author. Tel.: +1 617 287 6452.
E-mail addresses: duc@cs.umb.edu (D.A. Tran), cpham@cs.umb.edu (C. Pham).

0140-3664/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2010.04.022

network, an event to another set of random nodes, and these sets are sufficiently large, there almost certainly exists a rendezvous node for both the query and the event. A nice property of the gossip-based approach is that queries and events can be expressed in any sophisticated form.

The drawback, however, is in the large number of nodes where each query or event message must be sent to. Because of the communications required to send the message to these nodes, there may be a lot of signal collision caused by simultaneous transmissions from nodes that are close, thus reducing the effectiveness significantly. Not only so, each event must be evaluated at every node visited, exhausting the computational resource already scarce at the node.

Thus our research is to find a more efficient mechanism that does not require location information. The drawback of the gossip-based approach is because it treats a query/event message blindly, independent of its content and the guarantee that an event will meet *every* query (including those queries not matching the event) is too strong, leading to unnecessary query and event forwardings. Our hypothesis is that more efficiency may be obtained if a query/event message is routed by its content. In this paper we provide evidences to support this direction. We propose a mechanism with which we show that not only the notification delay can be kept reasonably small, but so can the computation cost because an event is evaluated at only a single node where every matching query, if any, will be found. The proposed mechanism is also efficient in terms of storage and communication costs, which are better than a representative gossip-based mechanism we compared to in our performance study.

The remainder of this paper is organized as follows. We discuss the related work in Section 2. We propose the details of our mechanism in Section 3. We present the simulation results in Section 4. We conclude the paper in Section 5.

2. Related work

Distributed publish/subscribe solutions differ fundamentally by their approach to designing the communication architecture. The communication protocols for subscription, publication, and notification form the foremost component of any publish/subscribe system as they dictate how queries and events are routed in the network to find each other. The existing efforts can be categorized into three approaches: broadcast-based, gossip-based, and location-based.

- *Broadcast/multicast-based*: Directed Diffusion [6] is one of the earliest publish/subscribe techniques for sensor networks. Given a query, the first step is for the subscriber node to broadcast the query to the network. Upon receipt of this query, each node creates a "gradient" entry in the routing table to point toward the neighboring node from which the query is received. Using a gradient path, a matching event can be sent towards the subscriber. Since there may be more than one such gradient path, the choice of which path to use is made based on some performance factors to improve energy efficiency. Instead of building a broadcast tree for each subscriber node, the PST technique in [8] builds a multicast tree rooted at each publisher node to broadcast its events. Each node in this tree stores its own subscribed queries as well as the union of those queries downstream. This knowledge is used at each node to route an event to appropriate tree branches leading to the matching subscribers. As an alternative to building multiple trees separately for different publishers, it is also proposed that a shared root is selected to receive all the events from the publishers and a single tree is built from this root to broadcast the events. An analysis of publisher-rooted multicast trees was presented in [16], in which a methodology was proposed to construct these trees taking into account inter-tree optimization through aggregation across multicast trees at intermediate nodes.
- *Gossip-based*: The broadcast-based approach is not efficient for large networks in which any node can be a publisher or subscriber, as will be prevalent in next-generation sensor and actuator networks. Instead of broadcasting, the gossip-based approach is based on the idea that if a query is replicated to a set of random nodes in the network, an event published to another set of random nodes, and these sets are sufficiently large, there almost certainly exists a rendezvous node for both the query and the event. For example, a gossip idea proposed in [15] can be used for enabling publish/subscribe services in sensor networks: each query is replicated in a random-walk [12,13] based tree of $q = O(\sqrt{n})$ nodes rooted at the subscriber node, and each event is sent along a similarly-built tree of c^2n/q nodes rooted at the publisher node where n is the number of nodes and $c \geqslant 1$ a constant. It is shown that the probability to have a rendezvous node is $r = 1 - \exp(-c^2)$. Another idea is proposed in [14], which is based on "selective" broadcasting. A query is broadcast to an extent defined by the subscription horizon ϕ which limits the number of times the query is rebroadcast. To broadcast an event, only a fraction τ of the neighbor links at each current node is used to forward the event. The overhead and effectiveness of the system can be tuned by choosing appropriate values for ϕ and τ.
- *Location-based*: The method of Geographic Hash Tables (GHT) was proposed in [9] – to hash values in a data space to geographic locations. A value is represented by a one-dimensional identifier called a "key" k, which corresponds to a geographic coordinate $h(k)$ determined based on the hash function. Thus, if each event or query can be referred to by a single key, GHT can be used. This is perfect for a type-based publish/subscribe system because the *'type'* value can be used as the key value for GHT. A query with key k (i.e., type k) will be stored at the node at location $h(k)$. When an event with the same key k (i.e., same type k) emerges, it will be routed to the node $h(k)$ and thus can find its matching queries. For routing from a sensor location to another sensor location, one can use a geographic routing protocol, e.g., GPSR [17]. An extension of the GHT technique in combination with landmark-based routing to improve routing efficiency is proposed in [10]. Another location-based technique called Double-Ruling is proposed in [18]. A nice property of Double-Ruling is its guarantee that the length of the notification path from a publisher to a matching subscriber is guaranteed to be within a small constant factor of their direct distance. The GHT method does not make this guarantee.

2.1. Need for a better approach

Location-based methods are effective for type-based and simple content-based services where events and queries can each be represented by a single key. This is not the case for most content-based services, where an event can have many attributes and a query can be more sophisticated than an exact-match query. No location-based method has been extended to address these services. Also, location information is not available for many sensor networks. On the other hand, although more efficient than the pure-broadcast methods, the gossip-based approach still incurs significant communication, storage, and computation costs because each query or event needs to be disseminated to a large portion of the network to have a good chance to meet its matches at rendezvous nodes. Further, the guarantee that a rendezvous node exists for every pair of queries and events, including those not matching each other, is unnecessarily strong and thus leads to unnecessary traffic. As details will be presented in the next section,

our proposed research takes a new approach that is fundamentally different from the existing approaches. This approach enables efficient publish/subscribe mechanisms to support a large class of content-based services running on networks of any size, with or without location information. It is noted that our work is not the first that suggests content-based routing in sensor networks. Indeed, it has been proposed in earlier work; for example, in [19] deals with tree construction algorithms for QoS-aware data reporting, in which the content of the events can be used to influence routing or forwarding decisions. Our work also takes a content-based routing approach, but is aimed at a different problem – the publish/subscribe problem.

3. The proposed solution

We assume that the sensor network is a connected graph of n stationary nodes $\{S_1, S_2, \ldots, S_n\}$ and that there are one or more nodes $\{S_1^*, S_2^*, \ldots, S_m^*\}$ ($m \geqslant 1$) that are stable and should never fail, called the "root" nodes. Two nodes are called "neighbors" if they can communicate directly with each other. No further assumption is made about the network topology. Location information is not required, nodes may have different communication ranges, and coverage obstacles may exist in the network. An example of the type of sensor networks that our solution is best fit for is the CitySense network currently deployed in Cambridge, MA. [20]. CitySense consists of more than 100 wireless sensor nodes deployed across the city, such as on light poles and private or public buildings. Each node is a powered Linux PC with 802.11a/b/g interface and connected to various sensors for monitoring air quality, weather, road traffic, contaminants, etc. When a phenomenon occurs (e.g., traffic congestion or chemical leak), the event will be notified to the appropriate subscribers such as the police and traffic control units. These subscribers can be anywhere in the city. For this example network, our proposed technique should be implemented at the layer of these PC nodes, some of which can be chosen to serve as the root nodes.

Our proposed publish/subscribe mechanism consists of two key design components: the virtualization component and the indexing component. The virtualization component maintains a naming structure by assigning to each node a unique virtual address. The indexing component determines the corresponding subscription and notification paths for given queries and publications, in which routing is based on the virtual addresses of the nodes.

3.1. Virtualization

A virtualization procedure can be initiated by any root node to result in a "virtual address tree (VA-tree), where each node is assigned a virtual address (VA) being a binary string chosen from $\{0,1\}^*$. Suppose that the initiating root node is S^*. In the corresponding VA-tree, denoted by TREE (S^*), we denote the VA of each node S_i by $VA(S_i{:}S^*)$. To start the virtualization, node S^* assigns itself $VA(S^*{:}S^*) = \varnothing$ and broadcasts a message inviting its neighbor nodes to join TREE (S^*). A neighbor S_i ignores this invitation if already part of the tree. Else, by joining, S_i is called a "child" of S^* and receives from S^* a VA that is the shortest string of the form $VA(S^*{:}S^*) + $ '0*1' unused by any other child node of S^*. Once assigned a VA, node S_i forwards the invitation to its neighbor nodes and the same VA assignment procedure repeats. In general, the rule to compute the VA for a node S_j that accepts an invitation from node S_i is: $VA(S_j{:}S^*)$ is always the shortest string of the form $VA(S_i{:}S^*) + $ '0*1' unused by any other child node currently of S_i.

Eventually, every node is assigned a VA and the VAs altogether form a prefix-tree. Fig. 1 shows the VA-tree with VAs assigned to the nodes as a result of the virtualization procedure initiated by node 1 (assuming that node 1 is a root node). The nodes' labels (1,2,...,24) represent the order they join the VA-tree. Each time a node joins, its VA is assigned by its parent according to the VA assignment rule above. Thus, node 2 is the first child of node 1 and given $VA(2{:}1) = VA(1{:}1) + $ '1' = '1', node 3 is the next child and given $VA(3{:}1) = VA(1{:}1) + $ '01'='01', and node 4 last and given $VA(4{:}1) = VA(1{:}1) + $ '001'='001'. Other nodes' VAs are assigned similarly. For example, consider node 18 which is the third child of node 8 (VA '011'). The VA of node 18 is the shortest binary string that is unused by any other child node of node 8 and of the form $VA(8{:}1) + $ '0*1'. Because the other children 16 and 17 already occupy '0111' and '01101', node 18's VA is '011001'.

It takes only a broadcast from the root of the tree to all the nodes in the network to build a VA tree. The cost therefore is the same as that used to set up a spanning tree in a sensor network. In fact, we can use an efficient tree construction technique (e.g., [21]) and after the child/parent links are known, the VAs will be assigned accordingly.

In $TREE(S^*)$, each node S_i is associated with a "zone", denoted by $ZONE(S_i{:}S^*)$, consisting of all the binary strings str such that: (i) $VA(S_i{:}S^*)$ is a prefix of str, and (ii) no child of S_i has a VA that is a prefix of str. In other words, among all the nodes in the network, node S_i is the one whose VA is the maximal prefix of str. We call S_i the "designated node" of str and use $NODE(str{:}S^*)$ to denote this node. For example, using the VA-tree $TREE(1)$ in Fig. 1, the zone of node 11 (VA '00101') is the set of binary strings '00101', '001010', and all the strings of the form '0010100...', for which node 11 is the designated node.

It is true that

- $ZONE(S_i{:}S^*) \cap ZONE(S_j{:}S^*) = \emptyset$, for every $i \neq j$ (i.e., the zone of a node does not overlap with the zone of any other node).
- $\bigcup_{i=1}^{n} ZONE(S_i : S^*) = \{0,1\}^*$ (i.e., any binary string must belong to the zone of some node).
- $\bigcup \{ZONE(S'{:}S^*) | (S'$ is S_i or a descendant of $S_i\} = \{str \in \{0,1\}^* | VA(S_i{:}S^*)$ is a prefix of $str\}$, for every i (i.e., any binary string whose prefix is $VA(S_i{:}S^*)$ must belong to the zone of S_i or that of a descendant node of S_i).

These properties are important to designing our indexing component.

3.2. Indexing

For ease of presentation, we assume for now that events are unidimensional and that there is only one VA-tree built ($m = 1$). We will discuss the case of multidimensionality and the case $m > 1$ later in Sections 3.5 and 3.4, respectively. We represent an event x as a k-bit binary string (the parameter k should be chosen

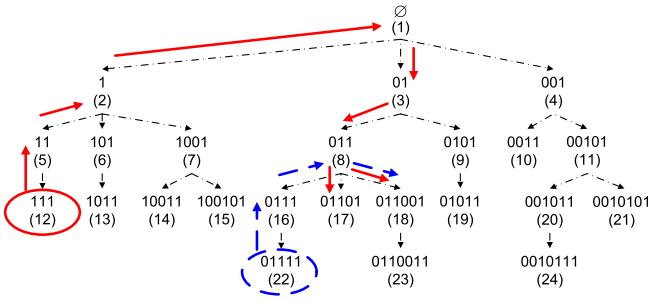


Fig. 1. Virtual address instance initiated by node 1 (the VAs of all the nodes form a prefix tree): solid-bold path represents the subscription path of query ['0110001','0110101']; dashed-bold path represents the notification path of event <'0110010'>.

to be larger than the longest VA length in the network; we use $k = 128$ bits in the evaluation). A query Q is represented as an interval $Q = [q_l, q_h]$, where $q_l, q_h \in \{0,1\}^k$, subscribing to all events x belonging to this interval (events are "ordered" lexicographically).

Our technique allows a query to be initiated by any node in the network at any time. This is desirable because as sensor technologies continue to improve, resulting in low-cost sensor platforms with increased capacities, more and more sensor networks will be able to serve highly-sophisticated purposes, with not just the sensing and collecting functions but also the capabilities to store and process the information on the fly inside the network as well as to perform other complex tasks. A node of these networks can be a resource-limited "sensor" node that gathers information about the physical world, an "actuator" node that is notified of this information and takes appropriate physical actions, or an "actor" node that plays a role not only as an actuator but also as a resource-rich network entity to perform specialized networking and data processing jobs (e.g., the CitySense network example earlier). A query can be initiated by any actuator or actor node in such a network.

Suppose that every node has been assigned a VA as a result of a virtualization procedure initiated by some root node S^*. We propose that (i) each query Q is stored at every node S_i such that $ZONE(S_i{:}S^*) \cap Q \neq \emptyset$; and (ii) each event x is sent to $NODE(x{:}S^*)$ – the designated node of string x. It is guaranteed that if x satisfies Q then Q can always be found at node $NODE(x{:}S^*)$ (because this node's zone must intersect Q). The dissemination algorithms to subscribe a query and publish an event are presented below in Algorithms 3.1 and 3.2, respectively.

Algorithm 3.1 (*Query Subscription*). Considering a query Q:

- Initially, the subscription starts at the subscriber node of Q
- At each node S_i that receives Q
  1. Quit if S_i already received this query before
  2. Let $Z = \{str \in \{0,1\}^k | VA(S_i{:}S^*)$ is a prefix of $str\}$
  3. IF (Z does not overlap Q)
  (a) Forward Q to the parent node of S_i in $TREE(S^*)$
  4. ELSE
     (a) Store Q at S_i if $ZONE(S_i{:}S^*)$ intersects Q
     (b) Forward Q to all children of S_i in $TREE(S^*)$
     (c) IF (Z does not contain Q), forward Q to parent of S_i in $TREE(S^*)$

Algorithm 3.2 (*Event Notification*). Considering an event x:

- Initially, the notification starts at the publisher node of x
- At each node S_i that receives x
  1. IF ($VA(S_i{:}S^*)$ is not a prefix of x) THEN
     (a) Forward x to the parent node of S_i in $TREE(S^*)$
  2. ELSE
     (a) Find the child node S_j such that $VA(S_j{:}S^*)$ is a prefix of x
     (b) IF (S_j exists) THEN Forward x to S_j
     (c) ELSE Search node S_i for those queries matching x

Fig. 1 shows an example with $k = 7$. Suppose that node 12 wants to subscribe a query Q=['0110001','0110101'], thus looking to be notified upon any of the following events {'0110001','0110010', '0110011','0110100','0110101'}. Therefore, this query will be stored at nodes {8,17,18}, whose zone intersects with Q. For example, node 8's zone intersects Q because they both contain '0110001'. The path to disseminate this query is $12 \rightarrow 5 \rightarrow 2 \rightarrow 1 \rightarrow 3 \rightarrow 8 \rightarrow \{17,18\}$ (represented by the solid arrow lines in Fig. 1). Now, suppose that node 22 wants to publish an event $x = <$'0110010'$>$. Firstly, this event will be routed upstream to node 8 – the *first* node that is a prefix with '0110010' (path $22 \rightarrow 16 \rightarrow 8$). Afterwards, it is routed downstream to the designated node $NODE$('0110010':1), which is node 18 (path $8 \rightarrow 18$). Node 18 searches its local queries to find the matching queries. Because query Q = ['0110001','0110101'] is stored at node 18, this query will also be found.

The storage and communication costs for a query's subscription depend on its range; the wider the range, the larger costs. For an event, the communication cost measured as the number of hops traveled to publish an event is $O(h)$ where h is the tree height ($h = O(\sqrt{n})$ in most cases). The delay to notify a matching subscriber is the time to travel to the rendezvous node and then from this node to the subscriber; hence, less than $2h$. The computation cost should be small because only one node – the designated node $NODE(x{:}S^*)$ – needs to search its stored queries to find those matching x. Our evaluation study in Section 4 indeed finds these costs reasonably small.

In our design, it is important to note that a query does not need to be explicitly unsubscribed. We propose that each query is associated with a lease that specifies when the query will expire and thus can be deleted. This strategy is also adopted in [14]. Though not perfect for every application, this strategy is reasonable because (i) to explicitly un-subscribe a query may result in significant communication overhead, and (ii) we do not need to worry about the queries that are lost due to a node failure, because they would eventually be re-subscribed if felt necessary by the corresponding subscribers.

3.3. Update methods

There may be changes in the network such as when a new node is added or an existing node fails. These changes need to be accommodated. Supposing that the network is virtualized according to the VA-tree $TREE(S^*)$, we present the update methods below.

3.3.1. Node addition

Consider a new node S_{new} that is added to the sensor network, say, to increase the coverage of some area. We need to add this node to $TREE(S^*)$. First, this node communicates with its neighbors and asks the neighbor $S_{neighbor}$ with the minimum tree depth to be its parent; tie is broken by choosing the one with fewest children. This strategy helps keep the tree as balanced as possible so its height can be short and workload fairly distributed among the nodes. The neighbor will then assign S_{new} a VA that is the shortest unused binary string of the form $VA(S_{neighbor}{:}S^*) + $ '0^*1'.

Because $ZONE(S_{neighbor}{:}S^*)$ is changed, the next task is for the parent node $S_{neighbor}$ to delete those queries that do not intersect $ZONE(S_{neighbor}{:}S^*)$ anymore. Also, this parent node needs to forward to S_{new} the queries that now intersect $ZONE(S_{new}{:}S^*)$.

3.3.2. Node removal

When a node fails to function, this failure affects the connectedness of the VA-tree in place. Because the VAs of the child nodes are computed based on that of the parent node, the child nodes of the departing node need to find a new parent so the VA-tree remains valid.

Consider such a child node S_{child}. This node selects a new parent among its neighbors. The new parent, say node S_{parent}, computes a new VA for S_{child} (similar to node addition). Then, S_{child} re-computes the VAs for its children and informs them of the changes. Each child node follows the same procedure recursively to inform all its descendant nodes downstream. The query transfer/forwarding from S_{parent} to S_{child} and, if necessary, from S_{child} to the descendant nodes of S_{child} is similar to the case of adding a new node.

In addition, because each descendant node S_i of the removed node is now assigned a new VA and thus a new zone, the queries

that are stored at S_i before the VA adjustment may no longer intersect its new zone. These queries are simply deleted. As we discussed earlier, they would eventually be re-subscribed if the corresponding subscribers wanted to. Nevertheless, it is still desirable to reduce the number of such queries. One way is to choose for node S_{child} above a "good" new parent S_{parent}. In a network that is richly connected, which is the case for many sensor networks, the former grandparent of S_{child} may already be a neighbor of S_{child} and thus can serve as the new parent for S_{child}. As a result, many queries that have been stored at S_i also intersect the new zone of S_i and thus do not need to be deleted.

The removal of a node affects the nodes on its subtree and as such if nodes fail frequently reorganizing the tree will become prohibitively expensive. This is a disadvantage of any tree-based techniques such as ours. We therefore suggest that our technique not be used for sensor networks with frequent node failures. In practice, usually a large-scale sensor network is formed by interconnecting a set of stable nodes (e.g., gateway nodes, super nodes), each being in charge of collecting the sensor readings from locally connected sensors. Although the sensors might fail often, the stable nodes would rarely. A real-world example of such a network is the CitySense sensor network we described at the beginning of this section [20]. The proposed technique should be installed at the layer of stable nodes.

3.4. Multiple VA-trees

Because query subscription and event notification procedures are based on the VA-tree, the root node and those nearby are potential hotspots. To alleviate this bottleneck problem, a solution is to build, not one, but multiple VA-trees. We can build m VA-trees, each initiated by a root node among $\{S_1^*, S_2^*, \ldots, S_m^*\}$. We should place the root nodes randomly in the network and prevent any two root nodes from being too close. After m virtualization procedures, each node S_i will have m VAs, $VA(S_i : S_1^*)$, $VA(S_i : S_2^*)$, ..., and $VA(S_i : S_m^*)$, respectively corresponding to the m VA-trees.

In presence of multiple VA-trees, each query is subscribed to a random VA-tree and each event is published to every VA-tree. A node near the root of a VA-tree may likely be deep in other VA-trees and so the workload and traffic are better shared among the nodes. Using multiple VA-trees also increases reliability. Because an event is notified to every VA-tree, the likelihood of its finding the matching queries should remain high even if a path this event is traveling is disconnected because of some failure. Indeed, suppose that the event is currently traversing a tree $TREE(S_1^*)$ when it cannot be forwarded because of a failure (link or node). In this case, the event can switch to another tree $TREE(S_2^*)$, randomly chosen, to be forwarded to a different node, say node X on the latter tree. Upon receipt of the event, node X will forward it based on the forwarding policy of the original tree, $TREE(S_1^*)$. The event will continue traversing this tree afterwards.

It is noted that if instead we send each query to all the trees and each event to a random tree, this strategy would result in significantly more communication cost and storage cost. This is because in each tree a query usually overlaps with the zones of many nodes and as such this query would have to be sent to a large number of nodes. It is thus desirable to minimize the number of trees each query should be subscribed to. Furthermore, since an event has only one destination node on each tree, which is its designated node, the costs to send an event to a tree are much less than that to send a query. Sending each query to a random tree and each event to all the trees is therefore a more efficient strategy.

Although the storage and communication costs per query should not increase, the communication and communication costs per event increase linearly with the number of VA-trees. We will discuss these effects in our evaluation study.

3.5. Multidimensionality

In practice, an event can have multiple attributes and as such it is usually represented as a numeric value in d dimensions where d is the number of attributes. To specify a subscription, a query is often specified as a d-dimensional rectangular range of values. To work with events and queries of this general form, we need a hash mechanism f that hashes a d-dimensional value x to a unidimensional k-bit binary string $x^f = f(x)$ and a d-dimensional range Q to a unidimensional interval $Q^f = f(Q)$ of k-bit strings such that if $x \in Q$ then $x^f \in Q^f$. For this purpose, we propose to use a $(k/2)$-order Hilbert Curve mapping [22]. This mapping preserves not only the containment relationship but also the locality property. Thus, small Q in the original space is mapped to small Q^f in the unidimensional space with a high probability. Then, to subscribe a query Q we follow Algorithm 3.1 using the hash interval Q^f. Similarly, to publish an event x we route it to the designated node of x^f according to Algorithm 3.2. When the event x reaches this node, locally stored queries are evaluated to find those matching x; the query evaluation with the event is based on the original values of the query and event (Q and x), not the hash values (Q^f and x^f).

It is important to note that our proposed technique returns to a subscriber all the events that match exactly *what the subscribed query describes*. However, when a query is subscribed to be notified of sophisticated events such as those involving images and sound, we can only specify in the query the values of selected representative features of these events and as such although the returned events will match these feature values, it is impossible to guarantee that they will match exactly *what the subscriber wants*. In this case, further event evaluation needs to be conducted by the subscriber to disqualify unmatching events.

4. Evaluation study

We developed an in-house simulator program to evaluate the proposed technique. The network topology was generated with WSNET – a topology generator for sensor networks (http://wsnet.gforge.inria.fr/). The simulated network, illustrated in Fig. 2, consists of $n = 1479$ sensor nodes uniformly placed in a 500 m × 200 m field, each node having a default communication radius of 20 m (the minimum radius to keep the network connected). We also investigated the effect of increasing this radius to 25 m and 30 m.

An event is represented as a *128*-bit string and a query an arbitrary interval of *128*-bit strings (as discussed in Section 3.5, other event/query models can be transformed to this model). A query or event is initiated by a random node chosen uniformly. The event load contains 10,000 events chosen uniformly among the strings $\{0,1\}^{128}$. The subscription load contains 10,000 queries, each being an interval whose length was chosen according to the following

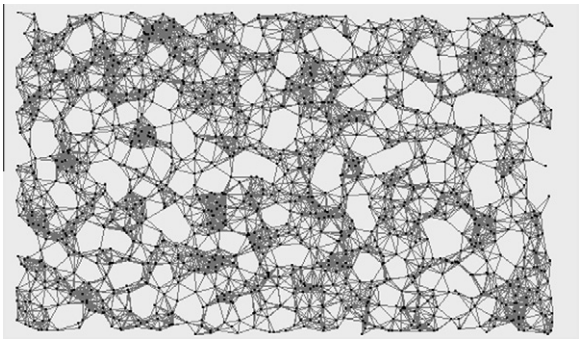


Fig. 2. Simulated network: 1479 nodes, communication radius 20 m, uniformly placed in a 500 m × 200 m area.

Zipf's law: in the set of possible length $\{2^0, 2^1, \ldots, 2^{127}\}$, length 2^i is picked with probability $\frac{1/i^{\alpha}}{\sum_{j=1}^{128}(1/j^{\alpha})}$. The number of events belonging to a query thus could be as large as half the entire event domain size (2^{127}) or just one (exact match). We considered two query range models: (1) the uniform query model: $\alpha = 0$, representing the uniform distribution, and (2) the heavy-tail query model: $\alpha = 0.8$, representing a heavy-tail distribution where a vast majority of the queries are specific. The heavy-tail model reflects many real-world applications.

Evaluation was considered in the following aspects: subscription efficiency, notification efficiency, notification delay, effect of network density, and effect of using multiple VA trees. We investigated two versions of the proposed technique, hereafter referred to as P2S (in the text and also the figures): (i) P2S-worst: the root node of a VA-tree is the most "outlier" node in the network (i.e., with maximum average-distance to other nodes); (ii) P2S-best: the root node is the "center" node in the network (with minimum average-distance to other nodes). P2S-best would result in a more balanced VA-tree, thus more efficient in distributing the queries and events. We also compared P2S to a gossip approach whose idea was adopted from BubbleStorm [15] – a recent gossip-based search technique originally designed for P2P but adaptable for sensor networks. Specifically, in our consideration of this approach (referred hereafter to as BS), we assumed that each query or event is sent to $\sqrt{n}$ nodes, resulting in a 64% query/event matching success rate (when there is no failure). The cost of BS would be higher if we wanted a higher success rate.

4.1. Subscription efficiency

To measure subscription efficiency, we compute the number of nodes that store an average query and the number of hops the query visits until reaching those nodes. Fig. 3 shows that P2S is noticeably better than BS in terms of subscription efficiency, whether the query model is uniform or heavy-tail. BS replicates an average query at more than 35 nodes. For the uniform query model, an average query in P2S-best is replicated at approximately 5 nodes and visits 14 hops. P2S-worst is worse than P2S-best but still significantly better than BS. Both P2S versions improve if the query model used is heavy-tail.

4.2. Notification efficiency

To measure notification efficiency, we compute the number of hops an average event visits before reaching the rendezvous node. This cost is independent of the query model used, and as shown in Fig. 4(a), is much lower with P2S than with BS. An average event visits fewer than 15 nodes in P2S-best, fewer than 20 nodes in P2S-worst, but more than 35 nodes in BS (more than 1.5 times higher).

This study, together with the study on subscription efficiency above, has three implications. First, P2S results in less costs (storage, communication) if the heavy-tail query model is used instead of the uniform model; this is good because the former model is more prevalent in practice. Second, the choice of the root node for the virtualization is important; we should choose the root node as close to the center of the network as possible. Third, despite the choice of the root node, P2S, by directing queries and events based on their content, is obviously more efficient (i.e., less storage, less traffic) than BS which blindly ignores the content.

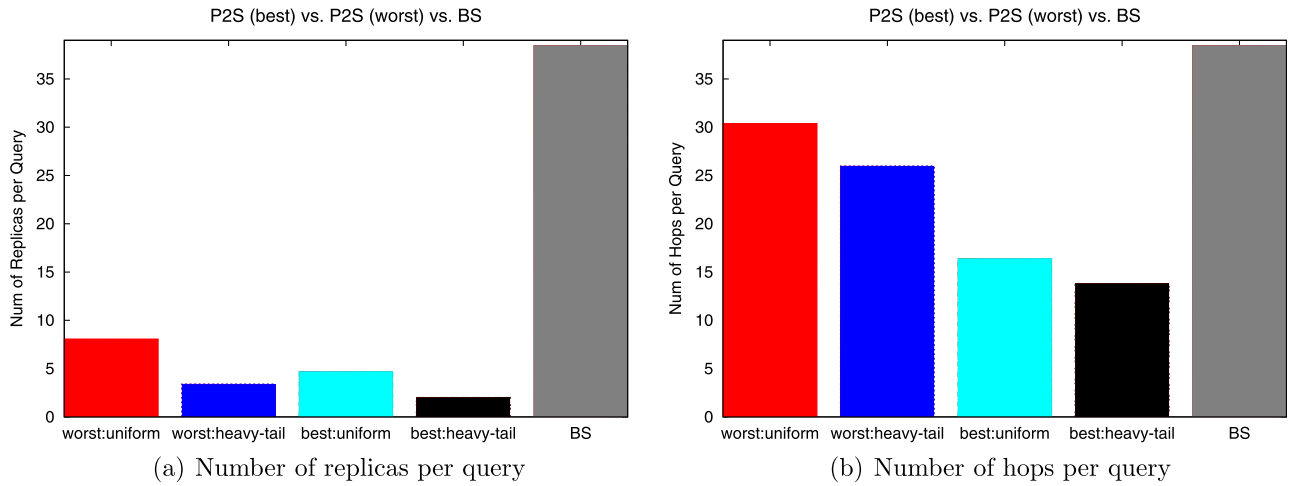
4.3. Notification delay

When an event is published, there may be more than one query subscribing to this event. To represent the notification delay for each (event, query) matching pair, we compute the ratio a/b where a is the hopcount-based distance the event has to travel from the publisher node to the subscriber node and b is the hopcount-based distance directly between these two nodes. This ratio is at least 1.0 because even if the publisher knows the subscriber, it must already take b hops to send the event to the subscriber.

Fig. 4(b) plots the notification delay computed as above incurred by P2S. Here, we select to display the 20,000 (event, query) matching pairs with the highest delay, sorted in descending order. In the case of P2S-best, more than 75% of the notifications have a delay not exceeding 2.0 (i.e., twice the perfect delay). On average, a notification takes 2.5 times as long as the perfect delay. Thus, a vast majority of events can notify their matching queries reasonably quickly. The choice of the root node affects the notification delay significantly. If P2S-worst is used, the average delay becomes 5.2, or two times higher than that in P2S-best.

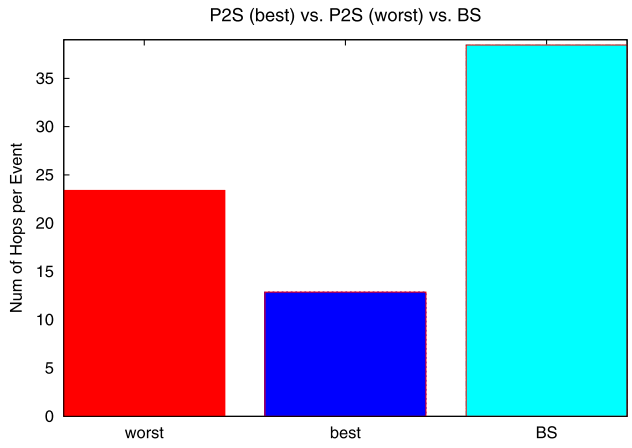
4.4. Effect of network density

In this study, we investigate three choices for the node communication radius, $r = 20$ m, $r = 25$ m, and $r = 30$ m, resulting in an average node degree of 5.9, 9.1, and 12.9, respectively. As illustrated in Fig. 5, P2S performs better if the network connectivity is richer. This is understandable. As each node has a larger number of neighbors, the VA-tree becomes shorter and thus it takes fewer hops for a query (Fig. 5(b)) or an event (Fig. 5(c)) to be routed to the corresponding rendezvous node (s). Also, because the VA-tree is

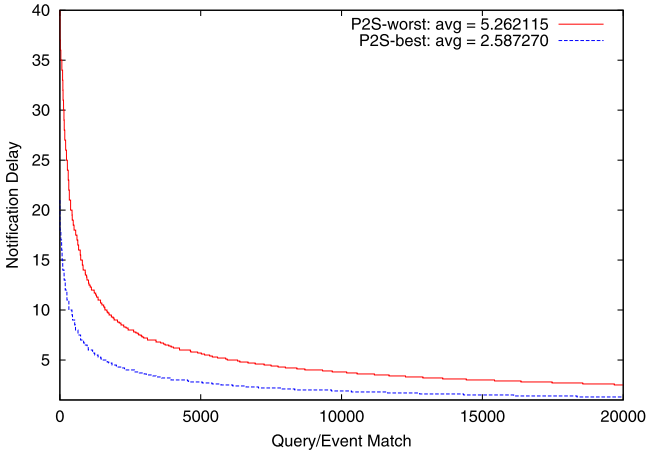


(a) Number of replicas per query (b) Number of hops per query

Fig. 3. Subscription efficiency.

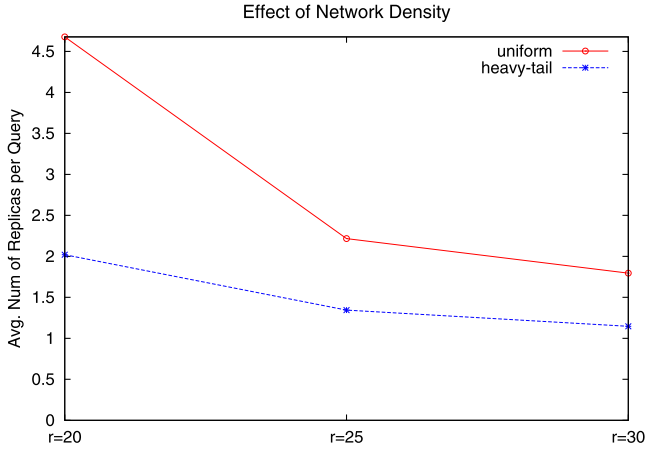


(a) Notification Efficiency

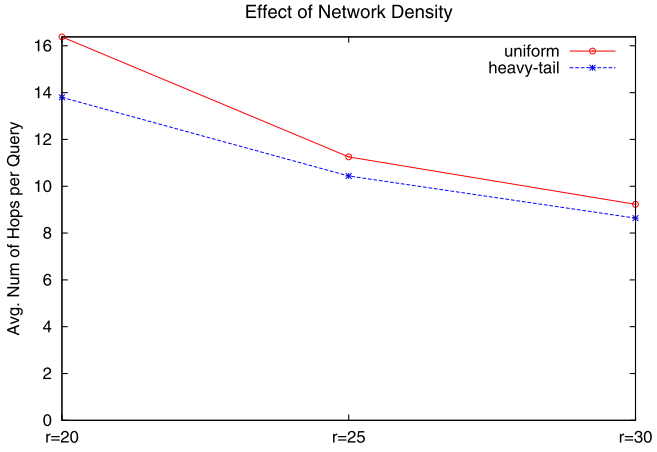


(b) Notification Delay

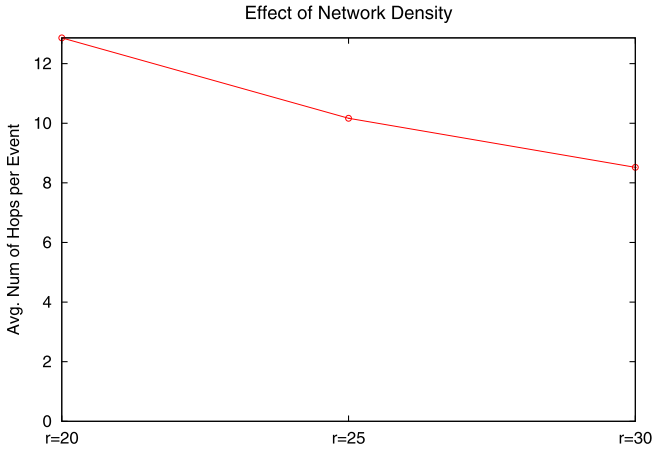
Fig. 4. Notification efficiency and delay.



(a) Number of replicas per query



(b) Number of hops per query



(c) Number of hops per event

Fig. 5. Effect of network density.

more balanced, there are fewer nodes whose zone intersects a given query, resulting in less storage (Fig. 5(a)).

4.5. Effect of multiple VA-trees

We compute for each node S_i the storage load $L_i^{store}/\sum_{j=1}^{n} L_j^{store}$, computation load $L_i^{comp}/\sum_{j=1}^{n} L_j^{comp}$, and communication load $L_i^{comm}/\sum_{j=1}^{n} L_j^{comm}$, and investigate the variation in each type of load. Here, L_i^{store} is the number of queries stored at node S_i, L_i^{comp} the number of events evaluated at node S_i, and L_i^{comm} the number of queries/events forwarded by node S_i.

One disadvantage of P2S is that because it is tree-based it gives more load burden to those nodes near the root of a VA-tree than those nodes near the leaf level. This problem can be alleviated if

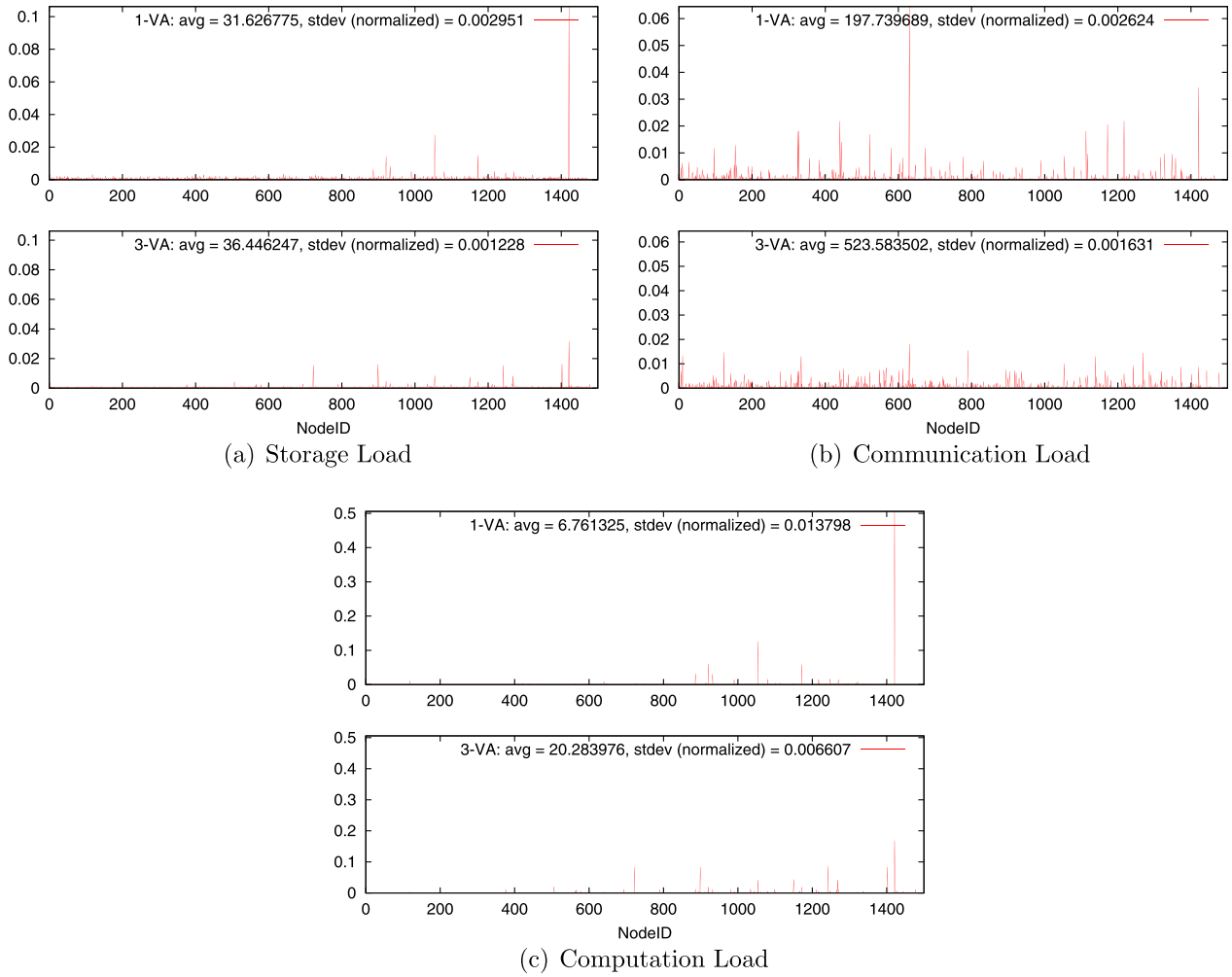


Fig. 6. Load balancing with multiple VAs.

more than one VA-tree is used. This is illustrated in Fig. 6 which shows the comparison on the load balancing between the case of using one VA-tree with the case of using three VA-trees. Here, the uniform query model is used (the heavy-tail model offers a similar result), the y-axis represents the loads computed above, the average value ("avg") shown is the absolute average (i.e., average (L^{store}), average (L^{comp}), average (L^{comm})), and the "stdev" is the standard deviation of the values plotted. In the case of one VA-tree, the heaviest-loaded node stores 10% of all the queries (Fig. 6(a)), forwards 6% of all the query replicas and events (Fig. 6(b)), and computes 50% of the events in the network (Fig. 6(c)). When three VA-trees are used, these loads for the heaviest-loaded node are 3%, 2%, and 18%, respectively. We can expect that if more VA-trees are used, better load balancing can be achieved. The computation and communication loads per node are proportionally increased if more VA instances are used. This is because each event is published to all the VA instances.

5. Conclusions

Without location information the dissemination of queries and events in publish/subscribe services in sensor networks usually relies on a gossiping protocol which ignores the message content during the dissemination. Our proposed mechanism is based on content-guided routing to offer a better efficiency (less storage and communication costs) than the traditional approach does. This is evident in an evaluation study that compares our mechanism to a representative gossip-based technique. We do not, however, suggest that the gossip approach not be used; it indeed has the advantage of being simple, fair, and suitable for many publish/subscribe applications. Instead, we advocate the direction that for better efficiency the routing of queries and events should be driven by their content to avoid unnecessary transmissions. Our work is the first attempt to implement this approach (for sensor networks without location information).

Acknowledgements

This work was supported in part by the University of Massachusetts Proposal Development Program and by the National Science Foundation (NSF) under award number CNS-0753066. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the NSF.

References

[1] D. Estrin, L. Girod, G. Pottie, M. Srivastava, Instrumenting the world with wireless sensor networks, in: International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2001), Salt Lake City, Utah, 2001, pp. 2033–2036.
[2] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, D. Culler, Design of a wireless sensor network platform for detecting rare, random, and ephemeral events, in: IPSN

'05: Proceedings of the 4th International Symposium on Information Processing in Sensor Networks, IEEE Press, Piscataway, NJ, USA, p. 70.
[3] G. Simon, M. Maróti, Ákos Lédeczi, G. Balogh, B. Kusy, A. Nádas, G. Pap, J. Sallai, K. Frampton, Sensor network-based countersniper system, in: SenSys '04: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, ACM, New York, NY, USA, 2004, pp. 1–12. doi:http://doi.acm.org/10.1145/1031495.1031497.
[4] N. Xu, S. Rangwala, K.K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, D. Estrin, A wireless sensor network for structural monitoring, in: SenSys '04: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, ACM, New York, NY, USA, 2004, pp. 13–24. doi:http://doi.acm.org/10.1145/1031495.1031498.
[5] T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J.A. Stankovic, T.F. Abdelzaher, J. Hui, B. Krogh, Vigilnet: an integrated sensor network system for energy-efficient surveillance, ACM Trans. Sen. Netw. 2 (1) (2006) 1–38. doi:http://doi.acm.org/10.1145/1138127.1138128.
[6] C. Intanagonwiwat, R. Govindan, D. Estrin, Directed diffusion: a scalable and robust communication paradigm for sensor networks, in: MobiCom '00: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, ACM Press, 2000, pp. 56–67. doi:10.1145/345910.345920.URL http://portal.acm.org/citation.cfm?id=345920.
[7] E. Souto, G. Guimaraes, G. Vasconcelos, M. Vieira, N. Rosa, C. Ferraz, J. Kelner, Mires: a publish/subscribe middleware for sensor networks, Personal Ubiquitous Comput. 10 (1) (2005) 37–44. doi:http://dx.doi.org/10.1007/s00779-005-0038-3.
[8] Y. Huang, H. Garcia-Molina, Publish/subscribe tree construction in wireless ad-hoc networks, in: MDM '03: Proceedings of the 4th International Conference on Mobile Data Management, Springer-Verlag, London, UK, 2003, pp. 122–140.
[9] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, F. Yu, Data-centric storage in sensor nets with ght, a geographic hash table, Mob. Netw. Appl. 8 (4) (2003) 427–442. doi:http://dx.doi.org/10.1023/A:1024591915518.
[10] Q. Fang, J. Gao, L.J. Guibas, Landmark-based information storage and retrieval in sensor networks, in: IEEE INFOCOM, 2006.
[11] M. Rudafshani, S. Datta, Localization in wireless sensor networks, in: IPSN '07: Proceedings of the 6th International Conference on Information Processing in Sensor Networks, 2007, pp. 51–60.
[12] C. Avin, B. Krishnamachari, The power of choice in random walks: an empirical study, Comput. Netw. 52 (1) (2008) 44–60. doi:http://dx.doi.org/10.1016/j.comnet.2007.09.012.
[13] D. Braginsky, D. Estrin, Rumor routing algorithm for sensor networks, in: WSNA '02: Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications, ACM, 2002, pp. 22–31. doi:http://doi.acm.org/10.1145/570738.570742.
[14] P. Costa, G.P. Picco, S. Rossetto, Publish-subscribe on sensor networks: a semi-probabilistic approach, in: Proceedings of the 2nd IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS05), Washington DC, USA, 2005.
[15] W.W. Terpstra, J. Kangasharju, C. Leng, A.P. Buchmann, Bubblestorm: resilient, probabilistic, and exhaustive peer-to-peer search, in: SIGCOMM '07: Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, ACM, New York, NY, USA, 2007, pp. 49–60. doi:http://doi.acm.org/10.1145/1282380.1282387.
[16] P. Jayachandran, R. Ganti, I. Gupta, T.F. Abdelzaher, Benefits of inter-tree optimizations for content-based publish/subscribe in sensor networks, Tech. rep., UIUC, 2006.
[17] B. Karp, H.T. Kung, Gpsr: greedy perimeter stateless routing for wireless networks, in: MobiCom '00: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, ACM Press, New York, NY, USA, 2000, pp. 243–254. doi:10.1145/345910.345953. URL http://portal.acm.org/citation.cfm?id=345953.
[18] R. Sarkar, X. Zhu, J. Gao, Double rulings for information brokerage in sensor networks, in: MobiCom '06: Proceedings of the 12th Annual International Conference on Mobile Computing and Networking, ACM, New York, NY, USA, 2006, pp. 286–297. doi:http://doi.acm.org/10.1145/1161089.1161122.
[19] H.J. Choe, P. Ghosh, S.K. Das, Cross-layer design for adaptive data reporting in wireless sensor networks, in: PerCom Workshops, IEEE Computer Society, 2009, pp. 1–6.
[20] R.N. Murty, G. Mainland, I. Rose, A.R. Chowdhury, A. Gosain, J. Bers, M. Welsh, Citysense: an urban-scale wireless sensor network and testbed, in: Proceedings of the 2008 IEEE International Conference on Technologies for Homeland Security, 2008.
[21] S. Upadhyayula, S.K.S. Gupta, Spanning tree based algorithms for low latency and energy efficient data aggregation enhanced convergecast (dac) in wireless sensor networks, Ad Hoc Netw. 5 (5) (2007) 626–648. doi:http://dx.doi.org/10.1016/j.adhoc.2006.04.004.
[22] J.K. Lawder, P.J.H. King, Using space-filling curves for multi-dimensional indexing, in: BNCOD 17: Proceedings of the 17th British National Conference on Databases, Springer-Verlag, London, UK, 2000, pp. 20–35.