ARTICLE IN PRESS

Ad Hoc Networks xxx (2012) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Ad Hoc Networks

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

DIBS: Efficient distributed information brokerage in large-scale sensor networks

Cuong Pham, Duc A. Tran *

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

ARTICLE INFO

Article history:
Received 17 January 2012
Received in revised form 7 August 2012
Accepted 27 August 2012
Available online xxxx

Keywords:
Sensor networks
Publish/subscribe
Information brokerage

ABSTRACT

This paper is focused on the problem of designing a distributed information brokerage infrastructure for sensor networks. We propose a solution, called DIBS, which is suitable for networks of any size with or without geographic location information, supports a wide range of applications, and allows any number of these applications to be deployed at any time without the need to reprogram the network. The design of DIBS consists of two key components: the design for a content-based routing substrate for sensor networks, and the design for a suite of subscription, publication, and notification protocols running on top of this substrate. Our theoretical findings are complemented by a simulation-based evaluation.

1. Introduction

Most sensor networks today are deployed primarily for collecting the data that have been captured by the sensor nodes to a sink for central analysis [1–3]. 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 sophisticated purposes, with not only the sensing and collecting functions but also the capability to store and process information on the fly inside the network. 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 [4,5].

A fundamental question to the principled design of such networks is how the nodes coordinate with each other so that any event when detected by a sensor node can be notified quickly to the corresponding interested nodes. Although each sensor node can be pre-programmed so that it knows the exact identities of which nodes to inform, this approach is application-specific and not flexible to future changes. Without knowing which node to notify, an alternative approach is to let each sensor node broadcast every event to the entire network. This approach, however, is not scalable with the network size. An ideal solution would be to incorporate in a sensor network a distributed information brokerage functionality that provides efficient mechanisms for a node to *subscribe* its event interests and for a node to *publish* its detected events, not requiring these nodes to know each other yet guaranteeing that every event will find the "right" nodes, quickly and correctly. The design for this functionality should work with sensor networks of any size, be supportive of a wide range of applications, and allow not just one but possibly multiple publish/subscribe services to be deployed on the same network at any time without the need for network reprogramming.

The publish/subscribe paradigm is not new and indeed it has been widely studied for the Internet environment; e.g., in the works of [6–11]. The publish/subscribe problem in sensor networks, however, is fundamentally different and much more challenging, as sensor networks are

* Corresponding author.
E-mail addresses: cpham@cs.umb.edu (C. Pham), duc@cs.umb.edu (D.A. Tran).

1570-8705/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.adhoc.2012.08.008

Please cite this article in press as: C. Pham, D.A. Tran, DIBS: Efficient distributed information brokerage in large-scale sensor networks, Ad Hoc Netw. (2012), http://dx.doi.org/10.1016/j.adhoc.2012.08.008

resource-constrained and operate based on multi-hop relay and ad hoc routing rather than an established infrastructure like the Internet. From another angle, while the publish/subscribe paradigm has been well-perceived as an important search model, most existing search techniques for sensor networks focus only on the traditional "retrieving" model [12–16]. The "retrieving" model is to inquire about the events that have already occurred and so the detected events must be stored pro-actively to be retrieved later. The publish/subscribe model presents a unique challenge as it is to inquire about events of the *future*. Hence, not the events, but the queries specifying event interests must be subscribed and stored first in the network waiting for the matching events (which may or may not occur in the future). In this paper, we propose a solution, called DIBS,[1] that works for a variety of publish/subscribe services (type-based or content-based) running on networks of any size, with or without location information. DIBS consists of two design components:

- As sensor networks are formed in an ad hoc fashion absent an infrastructure, we propose CRES,[2] a content-based routing network substrate which organizes the nodes in a structure that is convenient to run publish/subscribe mechanisms. CRES requires the existence of a few reference nodes whose location information is not necessary to be known; it is assumed only that they do not fail. CRES assigns each node in the network a virtual address based on how the node is positioned respectively with the references. These virtual addresses are used for content-based routing purposes.
- We propose a suite of efficient subscription, publication, and notification protocols, utilizing the CRES substrate. We refer to these protocols as SPN.[3] The subscription and publication protocols replicate each subscribed query and publish each detected event at appropriate nodes, respectively, so that if matching they can find each other at a rendezvous node quickly. These nodes are determined based on the matching between query/event content and node virtual addresses. Once an event reaches a rendezvous node, the notification protocol will route the event to all the interested subscribers. Forwarding decisions are based on prefix matching using message content and node virtual addresses.

The design of CRES is application-independent. It serves as the substrate allowing any number of different publish/subscribe services to run atop, at different times or simultaneously, using the proposed SPN protocols. We have conducted a simulation study to evaluate DIBS in terms of efficiency and reliability, in comparison with a representative location-based technique. We have found DIBS favorable in many aspects. In the remainder of this paper, we provide the details of DIBS in Section 3, discuss the simulation results in Section 4, and conclude the paper in Section 5 with pointers to our future work.

[1] DIBS = distributed information brokerage for sensor networks.
[2] CRES = content-based routing substrate for sensor networks.
[3] SPN = subscription, publication, and notification protocols.

2. Related work

There have been various efforts toward a distributed publish/subscribe based information brokerage infrastructure for sensor networks. Directed Diffusion [17] is one of the earliest among such efforts. 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 to the neighboring node from which the query is received. Using a gradient path, a matching event can be sent toward 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 [18] builds a broadcast 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 selectively route an event to appropriate tree branches leading to the matching subscribers. As an alternate 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.

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 if these sets are sufficiently large, there almost certainly exists a rendezvous node for both the query and the event [19,20]. Although more efficient than the pure-broadcast/multicast based 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.

If the geographic location information is known, we can use a method like GHT [13] to disseminate queries and events more efficiently. GHT is a DHT hashing the data space into geographic locations: each key k corresponds to a geographic coordinate $h(k)$. 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*, a.k.a. *subject-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 [21]. An extension of the GHT technique in combination with reference-based routing to improve routing efficiency is proposed in [16]. A location-based technique based on *double ruling* is proposed

Please cite this article in press as: C. Pham, D.A. Tran, DIBS: Efficient distributed information brokerage in large-scale sensor networks, Ad Hoc Netw. (2012), http://dx.doi.org/10.1016/j.adhoc.2012.08.008

in [22], which has a property that the length of the notification path from a publisher to a matching subscriber is bounded by a small factor of their direct distance.

Some location-less techniques such as [23,24] also attempt to imitate double ruling; they try to provide virtual coordinates to the sensor nodes first and then apply double ruling using these virtual coordinates for publish/subscribe mechanisms. Other location-less techniques [25–27] do not use double ruling but are also based on a virtual coordinate system to provide publish/subscribe services. These techniques, however, are not efficient for a large-scale network.

A comparison of different implementations of publish/subscribe techniques for sensor networks in terms of functionality and performance is discussed in [28]. These techniques run on top of a geographic-routing based data-centric architecture called DCS [29]. These implementations assume that location is available (e.g., by GPS, approximate, or virtual coordinates). A publish/subscribe technique for mobile ad hoc networks is proposed in [30], where local broadcast is used to disseminate advertise and subscribe messages, thus avoiding the expensive cost incurred by network-wide broadcast. Our work, however, is focused on sensor networks that are stationary.

3. DIBS: the proposed solution

We assume a sensor network with the existence of $m \geqslant 1$ "reference" nodes, $\{\Lambda_1, \Lambda_2, \ldots, \Lambda_m\}$, that are supposed to be stable. Geographic location information is not required for sensor nodes or reference nodes. Two nodes are called "neighbors" if they can communicate directly with each other. No further assumption is made about the network; nodes may have different communication ranges, and coverage obstacles may exist. As aforementioned, DIBS consists of two building blocks: the CRES substrate and the SPN protocols.

3.1. The CRES substrate

CRES arranges the network into a partition of m clusters, $\{CL(\Lambda_1), CL(\Lambda_2), \ldots, CL(\Lambda_m)\}$, each indexed by a reference node. Each node belongs to a cluster and is assigned a virtual address (VA) based on its relative "position" in this cluster ("position" explained later). A node X is uniquely identified by its VA, $va(X)$, and its cluster membership, $cl(X)$. In addition to knowing this information, X needs to keep track of a distance vector, $dist(X) \equiv \langle dist(X,\Lambda_1), dist(X,\Lambda_2), \ldots, dist(X,\Lambda_m)\rangle$, where $dist(X,\Lambda_i)$ is the shortest hopcount distance from reference Λ_i to X. The VA information is used to route messages inside a cluster, whereas the distance information is used to route messages from a cluster to another.

For a reference node Λ_i, we always have $va(\Lambda_i) = \varnothing$ and $cl(\Lambda_i) = CL(\Lambda_i)$). Except that, all the above information is initially unknown for every node but will eventually be filled as nodes periodically exchange information with their neighbors in a common heartbeat mechanism. Specifically, once a node X has obtained its VA and cluster membership, it periodically sends a heartbeat message advertising its information to all its neighbors. The heartbeat process should use the same period for each node. The value for this period is set at each reference node and propagated throughout the network as nodes advertise VA to their neighborhood. Upon receipt of the heartbeat message, each neighbor Y makes the following update:

1. Update distance information: set $dist(Y,\Lambda_i) \leftarrow \min(dist(Y,\Lambda_i), dist(X,\Lambda_i)+1)$, for each reference node Λ_i.
2. If $va(Y)$ has not been set, update VA and cluster information,
   (a) Set $cl(Y) \leftarrow cl(X)$ (i.e., Y will be in the same cluster as X).
   (b) Set $va(Y)$ to a binary identifier according to the *VA prefix rule*: $va(Y)$ is a string (shortest length preferred) of the form $va(X) + \text{`}\underbrace{0\ldots0}_{i\geqslant 0}1\text{'}$ unused[4] by any other neighbor node of X.

CRES is therefore constructed in a decentralized manner. Each node only needs to communicate with its neighbors, asynchronously and independently. Once a node Y hears from the first neighbor, X, having an established VA, Y obtains a VA based on the VA prefix rule and becomes a member of the same cluster that X belongs. In this case, X is called the "predecessor node" of Y and, inversely, Y is called a "successor node" of node X. We denote by $predecessor(X)$ the predecessor node of X and by $successor(X)$ the *set* of successor nodes of X. The order of assigning a VA to a node is similar to the order of a breadth-first search process; nodes near the references will have their VA assigned first before those that are far away. The partition of the nodes into clusters resembles a Voronoi diagram; nodes are assigned to a cluster represented by a reference if they are closer to this reference than they are to any other reference. Once CRES is stable, every node has an established VA and membership to a cluster, as well as knowing the minimum distance to each reference node. The VA of a node X will be as follows:

$$va(X) = \begin{cases} \varnothing & \text{if } X \text{ is a reference node} \\ va(predecessor(X)) + \text{'}\underbrace{0\ldots0}_{i\geqslant 0}1\text{'} & \text{if } X \text{ is the } (i+1)\text{th successor of } predecessor(X) \end{cases}$$

An example is illustrated in Fig. 1. The network comprises of 25 nodes {A,B,...,Y}, two of which (node D and node K) serve as the references. The figure shows the VA of each node. An arrow link, e.g., $J \rightarrow W$, represents that

[4] To avoid VA collisions, X can pre-compute such available strings and include in the advertisement message a string for each of its neighbors.

Please cite this article in press as: C. Pham, D.A. Tran, DIBS: Efficient distributed information brokerage in large-scale sensor networks, Ad Hoc Netw. (2012), http://dx.doi.org/10.1016/j.adhoc.2012.08.008

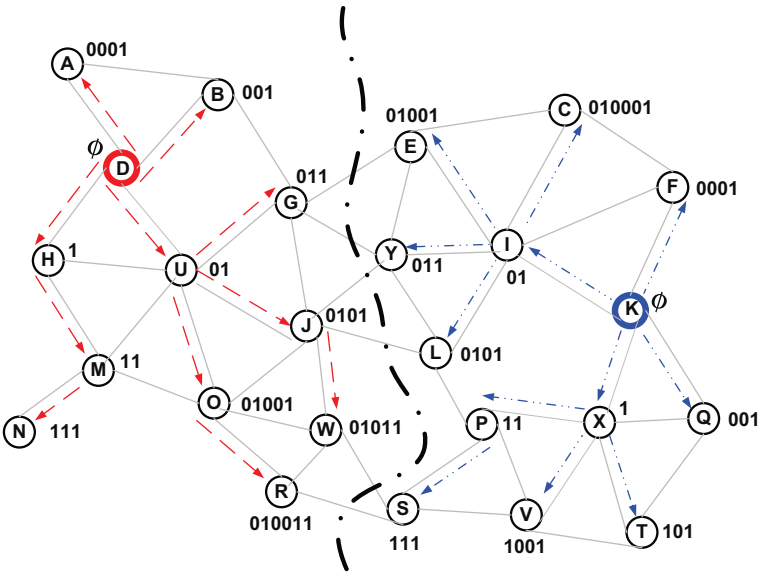


Fig. 1. The CRES substrate with two reference nodes (D, K), and their corresponding clusters ($CL(D)$ on the left side, $CL(K)$ on the right side) separated by the dotted boundary line. An arrow link represents a predecessor-successor link; e.g., $J \rightarrow W$, representing that $va(W)$ is determined based on $va(J)$. The order of filling the VA information (represented by the arrow links) resembles a breadth-first search process.

$va(W)$ is determined based on $va(J)$. Initially, only D and K have their VAs set ($va(D) = \varnothing, va(K) = \varnothing$). Each reference advertises its information to its neighborhood. Upon receipt of the advertisement from reference D, nodes A, B, U, and H obtain their corresponding VAs and thus belong to the cluster represented by D (i.e., $CL(D)$). Consequently, D is the predecessor node of these nodes and in turn each of these nodes is a successor node of D. Suppose that the order for these nodes to obtain a VA is H (first), U, B, A (last). Node H has $va(H)$ = '1' which is the shortest string of the form $va(D)$ + '$\underbrace{0\ldots0}_{i\geqslant 0}$1' = '$\underbrace{0\ldots0}_{i\geqslant 0}$1'. Node U has $va(H)$ = '01' which is the shortest string of this form available (because node H already obtains '1'). Node A (last neighbor) has $va(A)$ = '0001' because it is the shortest available after nodes H, U, and B have obtained their VA. For each remaining node with unknown VA, its VA will be determined based on the first advertisement received from a neighbor node with an established VA (if advertisements come at the same time, the advertisement with the shortest distance is chosen, the tie with same distance broken randomly). For example, node S has more than one neighbor but receives the earliest advertisement from node P; as a result, $va(S)$ = '111' which is determined based on $va(P)$ = '11' according to the VA prefix rule. Also, S will belong to the same cluster as P (which is cluster $CL(K)$). It is noted that two nodes may have the same VA, such as node G and node Y. This is because G obtains its VA from node U while Y obtains its VA from node I. The cluster information is used to distinguish two identical VAs ($G \in CL(D), Y \in CL(K)$).

It is possible to add a new reference node even after CRES has been constructed. For this purpose, the new reference node, Λ_{new}, just needs to advertise its information to its neighborhood. Because of the heartbeat mechanism, the existence of the new reference is propagated gradually throughout the network and every node's state will eventually be updated. During this process, a node that currently belongs to a cluster $CL(\Lambda_i)$ will switch to cluster $CL(\Lambda_{new})$ if it is closer to Λ_{new} than Λ_i (determined using the distance information); otherwise, this node stays in its current cluster. The node VAs are adjusted in a way similar to the updating process aforementioned.

3.1.1. Setup overhead

To setup the CRES substrate, the way that a node exchanges information with its neighbors is similar to that in classical distance-vector routing, the closest protocol being DSDV [31]. In the classical protocol, the distance vector information needs to be exchanged between neighbor nodes so that in the end routing information is available for every destination node in the network. The difference is that while "destination" in the classical protocol can be any arbitrary node, CRES considers only the reference nodes as "destination". Consequently, the overhead in terms of time and communication cost should be substantially less for CRES to converge to a stable state than for distance-vector routing.

3.1.2. Properties

The following terminology is used for the remainder of the paper:

Definition 3.1. Given a node X, we define its "prefix zone" and "key zone" as follows:

- The "prefix zone" of X, denoted by $pzone(X)$, is the set of all the binary strings, of which the VA of this node is *a* prefix.

Please cite this article in press as: C. Pham, D.A. Tran, DIBS: Efficient distributed information brokerage in large-scale sensor networks, Ad Hoc Netw. (2012), http://dx.doi.org/10.1016/j.adhoc.2012.08.008

- The "key zone" of X, denoted by $kzone(X)$, is the set of all the binary strings, of which the VA of this node is *the longest* prefix.

For example, consider Fig. 1. The prefix zone of reference D is $pzone(D) = \{0,1\}^*$ (set of all binary strings), and for a normal node $L, pzone(L) =$ '0101*' (set of all binary strings with prefix '0101'). The key zone of reference D is $kzone(D) =$ {'0', '00', '000', '0000*'}, and for a normal node $I, kzone(I) =$ {'01', '010', '0100', '01000', '010000*'. The CRES substrate has the following properties regarding key zones and prefix zones:

Property 3.2. *Given any node X, we have*

$$pzone(X) = kzone(X) + \sum_{Y \in successor(X)} pzone(Y)$$

Proof. Consider an arbitrary string $str \in pzone(X)$. Since $va(X)$ is a prefix of str, the longest prefix of str must be found in a node in the successor tree of X. This node is either node X (i.e., $str \in kzone(X)$) or a non-root node Y in this tree (i.e., $str \in kzone(Y)$). In the latter case, since $kzone(Y) \subseteq pzone(Y) \subseteq \sum_{Y \in successor(X)} pzone(Y)$, we have $str \in \sum_{Y \in successor(X)} pzone(Y)$. Consequently, $str \in kzone(X) + \sum_{Y \in successor(X)} pzone(Y)$.

Now, consider an arbitrary string $str \in kzone(X) + \sum_{Y \in successor(X)} pzone(Y)$. This implies either $str \in kzone(X)$ or $str \in \sum_{Y \in successor(X)} pzone(Y)$. In the former case, since $kzone(X) \subseteq pzone(X)$, we have $str \in pzone(X)$. In the latter case, since $va(X)$ is a prefix of the VA of any successor node X, $va(X)$ must be a prefix of str; hence, $str \in pzone(X)$. This completes the proof. □

Property 3.3. *Given any cluster, the key zones of all the nodes in this cluster form a partition of the binary universe; each node's key zone is a "part" of this partition. In other words, for any cluster $\mathbb{C}$, consisting of the nodes $\{X_1, X_2, \ldots, X_{|\mathbb{C}|}\}$, we have*

1. $kzone(X_p) \neq \varnothing \ \forall p \in [1, |\mathbb{C}|]$
2. $kzone(X_p) \cap kzone(X_q) = \varnothing \ \forall p \neq q \in [1, |\mathbb{C}|]$
3. $\sum_{p=1}^{\mathbb{C}} kzone(X_p) = \{0,1\}^*$

Proof. Property (1) holds true because $kzone(X_p)$ must contain the string $str = va(X_p) + '0\ldots'$. This string has $va(X_p)$ as a prefix and no successor node of X_p has VA being a prefix of str. Therefore, $va(X_p)$ is the longest prefix of str among all the nodes.

Property (2) holds true because otherwise $va(X_p)$ and $va(X_q)$ must be identical (so they can both be the longest prefix for some string), which is impossible.

Property (3) holds true because given any binary string str, there must be a node whose VA is the longest prefix of str. In the most special case, this node is the reference node of cluster ($\mathbb{C}$) ($\varnothing$ is a prefix of any string). Thus, $\{0,1\}^* \subseteq \sum_{p=1}^{\mathbb{C}} kzone(X_p)$. Conversely, it is trivial that $\sum_{p=1}^{\mathbb{C}} kzone(X_p) \subseteq \{0,1\}^*$. □

Property 3.2 is about the relation between key zones and prefix zones. Property 3.3 implies that, given any binary string str, there exists one and only node X in each cluster $CL(\Lambda_i)$ such that $str \in kzone(X)$. We call X the "designated node" of str in cluster $CL(\Lambda_i)$ and denote it by $node(str : \Lambda_i)$; hence, there are m designated nodes for this string, each in a cluster. For example, consider Fig. 1; given string '0100010', there are two designated nodes: $node$('0100010': D) = U and $node$('0100010': K) = C, because $va(U)$ and $va(C)$ are the longest prefixes of '0100010' in clusters $CL(D)$ and $CL(K)$, respectively.

CRES provides a convenient structure for message routing, using the VA and distance information, as described below.

3.1.3. Routing

To route a message from a source node X to a destination node with VA dst in the same cluster, we can apply *prefix routing*. A node that receives the message always forwards it to a neighbor in the same cluster with X, whose VA shares a longer prefix with dst; if no such neighbor exists and the current node is not a prefix of dst, the message is forwarded to the predecessor node. It is guaranteed that the message will eventually reach the destination. To route a message from a source node X to a destination with VA dst in a different cluster $CL(\Lambda_i) \neq cl(X)$, we apply a two-phase routing protocol as follows: (1) shortest-path phase: X forwards the message to the neighbor Y with the minimum $dist(Y, \Lambda_i)$ (this information is known from the neighbors' state information). Node Y and the subsequent intermediate nodes that receive the message apply the same greedy strategy. The routing switches to the prefix-routing phase when the message visits a node $Z \in CL(\Lambda_i)$; (2) prefix-routing phase: starting from node Z, the message is routed towards the destination node based on prefix routing as aforementioned.

Property 3.2 and Property 3.3, and the routing structure provided by CRES are useful for us to design efficient publish/subscribe mechanisms on top of CRES. In the next section, Section 3.2, we assume that CRES is stable and discuss the details of these mechanisms. In reality, this substrate needs to be maintained over time due to network dynamics such as node additions and removals. The maintenance mechanisms are described in Section 3.3.

3.2. The SPN protocol suite

For the sake of simplicity, it is assumed that the publish/subscribe data domain being indexed is $\{0,1\}^k$, i.e., the set of k-bit identifiers. The parameter k should be chosen to be larger than the longest VA length in the network. An event ρ is represented as a k-bit key in this domain. A query $\hat{\rho}$ is represented as a key interval, $\hat{\rho} = [\rho_l, \rho_h]$, where $\rho_l, \rho_h \in \{0,1\}^k$. If $\rho_l = \rho_h$, the query becomes an exact match query; this is the case for type-based publish/subscribe services where events and queries are matched based on their type. In general content-based publish/subscribe services, $\rho_l < \rho_h$ and the query is to subscribe to all the events whose key falls between these bounds (events are "ordered" lexicographically). It is noted that most applications in practice use a numeric value or range of

Please cite this article in press as: C. Pham, D.A. Tran, DIBS: Efficient distributed information brokerage in large-scale sensor networks, Ad Hoc Netw. (2012), http://dx.doi.org/10.1016/j.adhoc.2012.08.008

values to represent an event or query, respectively. Further, a value can be one-dimensional or multi-dimensional. For DIBS to work with these applications, there must be an intermediate step to transform their format to our convention; for example, we can use a $(k/2)$-order Hilbert Curve mapping for this purpose [32]. This is a requirement necessary also for publish/subscribe techniques using DHT.

According to Property 3.3, any event must belong to the key zone of one and only node in each cluster; there are m such nodes. We propose that these nodes are the destinations for the event; given an event ρ, it will be sent to every $node(\rho : \Lambda_i)$, the corresponding designated node of ρ in each $CL(\Lambda_i)$ $(1 \leqslant i \leqslant m)$. To subscribe a query, $\hat{\rho}$, we propose that it is sent to every node X in the local cluster such that the key zone of X intersects $\hat{\rho}$; i.e., $\hat{\rho} \cap kzone(X) \neq \varnothing$. It is easy to see that if ρ satisfies $\hat{\rho}$ then ρ can always be found at its designated node in the local cluster of $\hat{\rho}$. Based on CRES routing, the protocols for query subscription and event publication work in detail as follows:

- *Query subscription*: As a corollary of Property 3.2, satisfying the condition $\hat{\rho} \cap kzone(X) \neq \varnothing$ is equivalent to satisfying two conditions: (1) $\hat{\rho} \cap pzone(X) \neq \varnothing$, and (2) $\hat{\rho} \not\subset pzone(Y)$ for every $Y \in successor(X)$. Therefore, given a query $\hat{\rho}$, each node X that receives this query, initially the subscriber node, will store $\hat{\rho}$ if both of these conditions are satisfactory. For query forwarding, X forwards the query to $predecessor(X)$, its predecessor node, if $\hat{\rho} \not\subset pzone(X)$, and to each successor node $Y \in successor(X)$ if $\hat{\rho} \cap pzone(Y) \neq \varnothing$. It is noted that a node can only accept the query once (the first that arrives; duplicates are ignored).
- *Event publication*: The publication protocol is simpler because an event is just a single binary key. Given an event ρ, it is routed separately to each of its m designated nodes. Towards each designated node, $node(\rho : \Lambda_i)$, the two-phase routing mechanism in Section 3.1 is applied: (1) following the shortest-path phase to route ρ towards Λ_i, stopping when reaching the first node of cluster $CL(\Lambda_i)$; and (2) inside $CL(\Lambda_i)$, using prefix routing to route ρ towards the node whose VA is the largest prefix of ρ; this node is $node(\rho : \Lambda_i)$.

Fig. 2 shows an example of query subscription, here assuming that $k = 7$. Suppose that node N wants to subscribe a query $\hat{\rho} = [\text{'0101001'}, \text{'0101111'}]$, thus looking to be notified upon any of the following events {'0101001', '0101010', '0101011', '0101100', '0101111'}. Therefore, this query will be stored at nodes $J(va(J) = \text{'0101'}, kzone(J) = \{\text{'0101'}, \text{'01010}^*\text{'}\})$ and $W(va(W) = \text{'01011'}, kzone(W) = \{\text{'01011}^*\text{'}\})$, whose key zone intersects $\hat{\rho}$. The path for this query to find these nodes is $N \to M \to H \to D \to U \to J \to W$ (represented by the solid arrow lines in Fig. 2).

Fig. 3 shows an example of event publication, supposing that node C wants to publish an event ρ = '0101101'. This event will routed separately on two paths to find the corresponding designated nodes in clusters $CL(K)$ and $CL(D)$. The first path, $C \to I \to L$, is formed by prefix routing towards L – the designated node in the local cluster $(CL(K))$. At node L, the event will find all the matching queries locally subscribed in $CL(K)$. The second path, $C \to E \to G \to J \to W$, consists of two segments: the shortest-path segment, $C \to E \to G$, towards the reference D trying to get to the first node in $CL(D)$, and the prefix-routing segment, $G \to J \to W$, from G towards W – the designated node of the event in $CL(D)$. At node W, the event will find all the matching queries subscribed in cluster $CL(D)$.

Once an event reaches each of its designated nodes, this node needs to search its local query storage to find those queries that match the event. For each query found, the

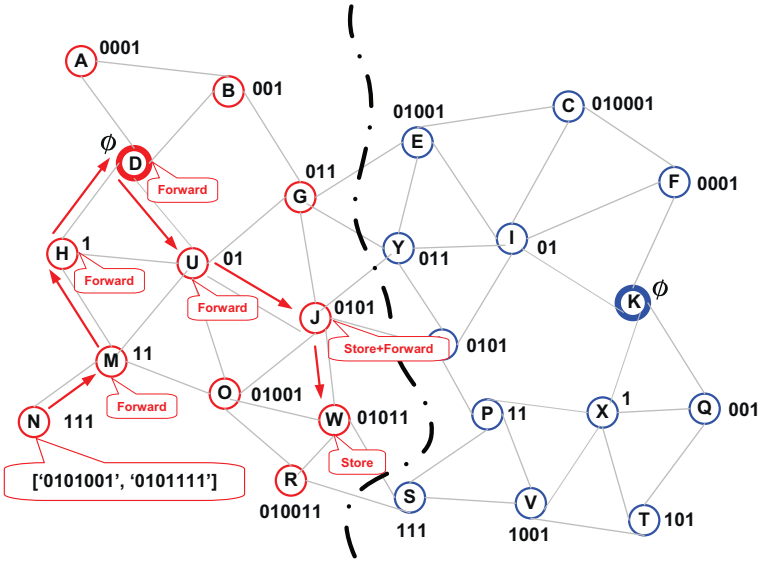


Fig. 2. A query $\hat{\rho}$ = ['0101001', '0101111'] subscribed by node N is stored at nodes J and W, whose key zone intersects $\hat{\rho}$. The path for this query to find these nodes is $N \to M \to H \to D \to U \to J \to W$ (represented by the arrow links).

Please cite this article in press as: C. Pham, D.A. Tran, DIBS: Efficient distributed information brokerage in large-scale sensor networks, Ad Hoc Netw. (2012), http://dx.doi.org/10.1016/j.adhoc.2012.08.008

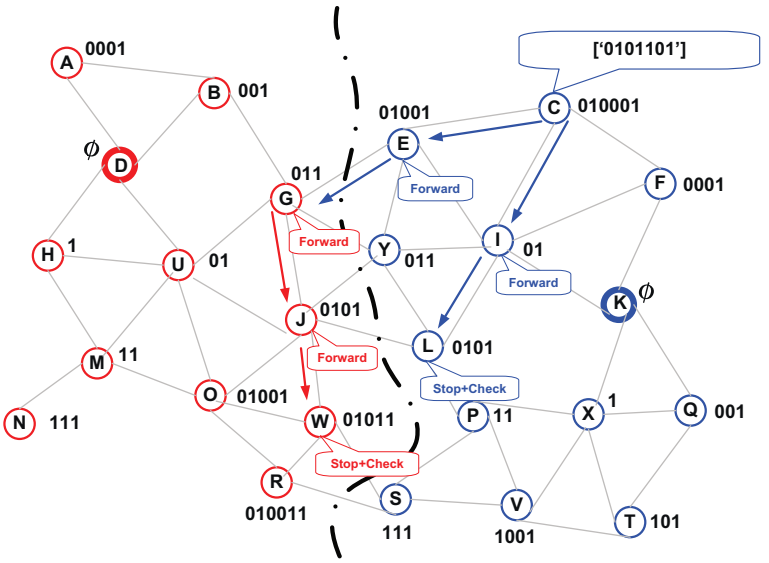


Fig. 3. An event ρ = '0101101' published by node C is routed separately on two paths. The first path, $C \rightarrow I \rightarrow L$, is formed by prefix routing towards $node(\rho : K)$ which is L. The second path consists of two segments: the shortest-path segment, $C \rightarrow E \rightarrow G$, trying to get to the first node in $CL(D)$, and the prefix-routing segment, $G \rightarrow J \rightarrow W$, trying to reach $node(\rho : D)$ which is W.

corresponding subscriber node needs to be notified. The notification protocol works simply as follows:

- *Event notification*: If an underlying routing protocol driven by physical addresses is available, we can simply rely on this protocol to route the event to the subscriber node using its physical address. Otherwise, CRES routing is applied to route the event towards the subscriber node; the VA of this node is used as the destination VA address.

The benefit of CRES to the proposed SPN protocols is multi-fold. First, it decouples the task of maintaining the connectivity in the network with the task of running publish/subscribe services. Therefore, multiple applications can run simultaneously on this substrate. Second, CRES can be thought of as a DHT-like scheme that allows hashing a single key (i.e., event) as well as a range of keys (i.e., query). Third, for a large network, CRES can be used to organize it into several disjoint clusters and because the subscription protocol advertises a query to only a subset of nodes in its local cluster, these nodes are a short distance away from the query subscriber. The size of this subset depends on the query range. In practice, a vast majority of the queries are rather specific than generic, and, consequently, the communication and storage costs involved in the subscription protocol should be small. In addition, although the publication protocol needs to advertise an event to all clusters, the publication path to each cluster and then to the corresponding designated node is just a single path, resulting in a small communication cost. The computation cost involved is also small because the event only needs to be evaluated at its designated nodes. These properties are substantiated in our performance study to be discussed in Section 4, which also shows that the hop-count distance to send an event to a matching query using CRES routing is reasonably close to the optimal distance (i.e., shortest hop-count distance between the corresponding publisher node and subscriber node in the connectivity graph).

3.3. Maintenance mechanisms

There may be changes in the network such as when a new node is added or an existing node fails. These changes are addressed in DIBS as follows.

3.3.1. Node addition

A new node, say node X_{new}, follows a simple procedure to join the network. First, it needs to discover the neighbor nodes which are the existing nodes inside its communication range. X_{new} then selects a neighbor as its predecessor node, from which it gets assigned a VA and a cluster to belong. The VA of X_{new} is determined according to the VA prefix rule. Its cluster is the same as that of the selected neighbor. By communicating with all the neighbors, X_{new} can also fill out the distance information. For the best performance, the predecessor node, $X_{predecessor}$, should be the neighbor node with the shortest VA, tie broken randomly. This is because the variance among the node VA lengths will be lesser, resulting in fairer-sized distribution of the nodes' key zones. Finally, as X_{new} becomes a new successor node of $X_{predecessor}$, the latter's key zone is changed. $X_{predecessor}$ needs to delete those queries that no longer intersect $kzone(X_{predecessor})$, and, in addition, forwards to X_{new} those queries that now intersect $kzone(X_{new})$.

3.3.2. Node failure

As discussed in Section 3.1, a node periodically exchanges information with its neighbors. Therefore, if a

Please cite this article in press as: C. Pham, D.A. Tran, DIBS: Efficient distributed information brokerage in large-scale sensor networks, Ad Hoc Netw. (2012), http://dx.doi.org/10.1016/j.adhoc.2012.08.008

node fails, say node X_{fail}, each neighbor node will detect the failure and, subsequently, need to update its state. If the neighbor node is not a successor node of X_{fail}, the update is simple; only the distance information and neighbor information need to be updated. If the neighbor node is a successor node, say node $X_{successor}$, this node needs to find a new predecessor and adjust its VA and cluster information accordingly. Similar to the case of adding a new node, the new predecessor node is chosen to be the alive neighbor of $X_{successor}$ with the shortest VA.

The new VA and cluster information of $X_{successor}$ is subsequently advertised to its neighborhood and all its successor nodes will need to adjust their VA information to be consistent with the VA prefix rule. This updating process takes place at all the nodes that are reachable recursively by following the successor links from $X_{successor}$. These nodes must set their cluster to the cluster of $X_{successor}$. In addition, if a node's VA has been changed, it may have to delete some queries and forward some to its successor nodes, as in the case of adding a new node. The failure of a node has only local impact on the cluster where the failure occurs and only to a small subset of nodes in this cluster. In a typical sensor network, nodes might fail at any time but we do not expect a high rate of failure. The cost to recover a failure in our scheme is reasonable for typical networks, which is substantiated in our performance study.

3.4. Deployment of the reference nodes

A requirement for DIBS is the existence of a small number of reference nodes. Although we do not require location information about these nodes, they need to be stable (i.e., almost never fail) and more resourceful than a normal sensor node. For the best performance, they should be deployed in the network field so that the sizes of the corresponding clusters are as similar as possible. The actual implementation of this deployment policy is beyond the scope of the DIBS framework.

An example of the type of sensor networks that our solution is best fit for is the CitySense network currently deployed in Cambridge, MA [33]. 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 reference nodes.

4. Evaluation study

We developed a discrete event-driven simulator to evaluate DIBS. To simulate a large network, we considered a 1479-node network topology generated with WSNET – a topology generator for sensor networks (http://wsnet.gforge.inria.fr/). The nodes of this network are uniformly placed in a 500 m × 200 m field, each node having a communication radius of 20 m (see Fig. 4). In a 64-bit key space ($k = 64$), we randomly generated 5000 events (uniformly) and 5000 queries (range uniformly distributed in [1]). Thus, a query could be as large as the entire event domain size (2^{64}) or just one (i.e., exact match).

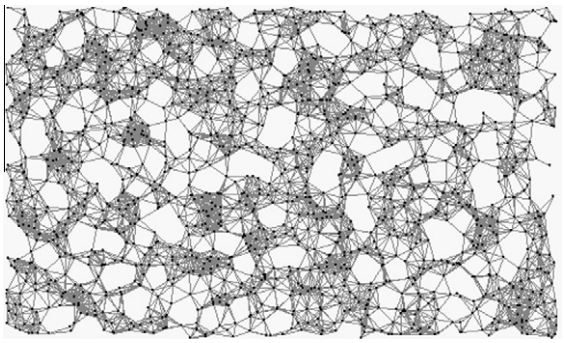
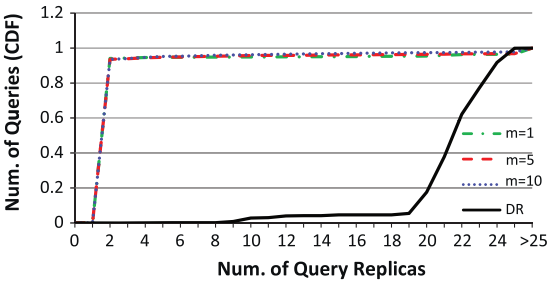


Fig. 4. Simulated network: 1479 nodes, communication radius 20 m, uniformly places in a 500 m × 200 m area.

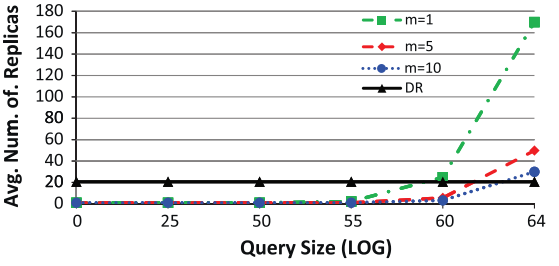
We evaluated DIBS in three cases of $m \in \{1, 5, 10\}$, the number of reference nodes; for each case, the reference nodes were chosen randomly among the 1479 nodes. The following aspects were investigated: subscription efficiency, publication efficiency, notification delay, and effect of failure. We also compared DIBS to a double ruling method in which every node is aware of its location and a query is replicated on all the nodes along the vertical line crossing the subscriber node and an event is published to all the nodes along the horizontal line crossing the publisher node. This method serves as basis for location-based publish/subscribe techniques such as [16] and location-less techniques such as [23,24]. For simplicity of presentation, we refer to this double ruling method as DR.

4.1. Subscription efficiency

To measure subscription efficiency, we compute the following costs for a query: (1) storage cost: the number of query replicas that need to be stored in the network; and (2) communication cost: the number of forwarded messages during the dissemination of the query. Fig. 5a shows the cumulative density function of the number of query replicas, which clearly shows that DR incurs a much higher storage cost than DIBS. While more than 90% of the queries in DIBS require no more than 2 replicas, more than 90% of the queries in DR require more than 18 replicas. To see how the query size affects the storage cost, we plot the average storage cost for each bucket of query ranges in Fig. 5b, where there are five buckets $[2^0, 2^{25}]$, $[2^{25}, 2^{50}]$, $[2^{50}, 2^{55}]$, $[2^{55}, 2^{60}]$, and $[2^{60}, 2^{64}]$. The storage cost in DR is similar for each query because DR replicates a query blindly of its content. Storage in DIBS is much cheaper than that in DR for query size ranging from 1 to 2^{60}. Only when the query size is extremely large (larger than 2^{60} when $m = 1$, 2^{61} when $m = 5$, and 2^{62} when $m = 10$), DR would require less storage cost than DIBS.



(a) Cumulative density function (CDF)



(b) Effect of query range

Fig. 5. Subscription efficiency: storage cost as number of replicas stored per query.

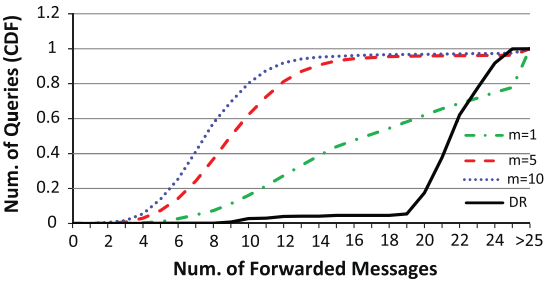
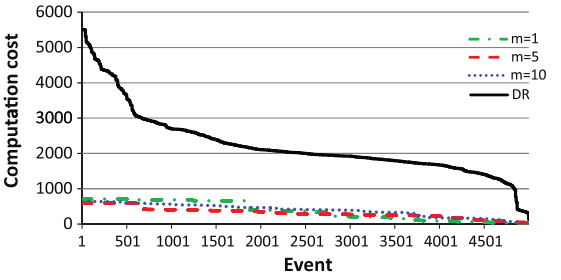
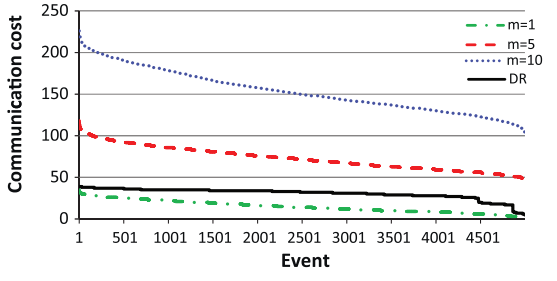


Fig. 6. Subscription efficiency: communication cost as number of forwarded messages.



(a) Computation cost as number of queries evaluated



(b) Communication cost as number of forwarded messages

Fig. 7. Publication efficiency.

However, this case is rare in practice and should not be allowed in any search system, especially for sensor networks. Fig. 5b also demonstrates that by increasing the number of reference nodes in DIBS, we can cut down the storage cost significantly when queries are range queries; exact match queries result in the same cost (1 replica) regardless of the number of reference nodes.

As observed in Fig. 6, among different versions of DIBS ($m = 1, 5$, or 10), a larger m results in better communication efficiency. This is because when m is large, each cluster is small and because the destinations for a query are in the local cluster, the number of messages involved must be small as well. Compared with DR, DIBS is substantially better. While more than 90% of the queries require more than 18 message transmissions in DR, they require less than 12 message transmissions in DIBS when $m = 5$ or $m = 10$. Even when $m = 1$, the worst version of DIBS, only 40% of the queries require more than 18 message transmission (vs. 90% in DR).

4.2. Publication efficiency

To measure publication efficiency, we compute the following costs for an event: (1) computation cost: the total number of queries stored at the nodes visited by the event, that need to be evaluated to find those matching the event; and (2) communication cost: the number of forwarded messages during the dissemination of the event. The costs for individual events are plotted in Fig. 7a and b. In terms of computation cost, DR is expensive because every time an event message visits a node, this node has to evaluate all the stored queries to find those matching the event. In contrast, this evaluation procedure in DIBS is required only at the m designated nodes of the event. Fig. 7a shows that for a vast majority of the events, the computation cost incurred by DIBS is at least eight times less than that incurred by DR. In DIBS, the computation cost does not change much when m varies.

In terms of communication cost, it is expected that if more reference nodes are used, an event needs to be routed to more clusters, thus leading to more message transmissions; this increase is linear with m as shown in Fig. 7b. DIBS requires less traffic than DR when $m = 1$, but more when m is 5 or 10. However, we expect in many publish/subscribe applications that events occur much less frequently than queries and as such the costs due to query subscription should be the dominant costs. We thus consider the increased communication cost due to event publication acceptable, taking into account the efficiency in query subscription.

Please cite this article in press as: C. Pham, D.A. Tran, DIBS: Efficient distributed information brokerage in large-scale sensor networks, Ad Hoc Netw. (2012), http://dx.doi.org/10.1016/j.adhoc.2012.08.008

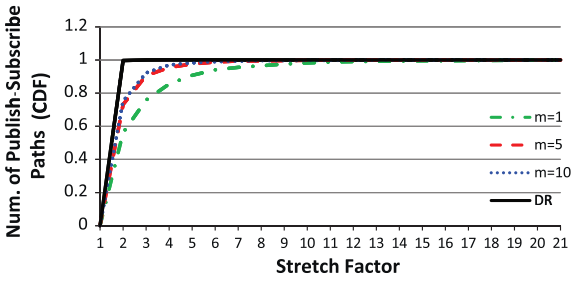


Fig. 8. Notification delay: stretch factor.

4.3. Notification delay

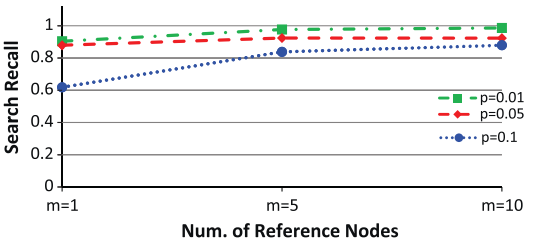
For each matching query-event pair, the notification delay comprises the delay to send the event from the publisher node to its designated node and the delay to send the event from the designated node to the matching query's subscriber node. The "perfect" delay is the delay to send the event directly from the publisher to the subscriber on their shortest path. This, however, requires that the publisher already know the subscriber, which is not the case in practice. Thus, to fairly measure the notification delay, we represent it by the *stretch factor* which is the ratio of the total hopcount distance from the publisher node to the designated node to the subscriber node to the shortest hopcount distance from the publisher node directly to the subscriber node.

Fig. 8 shows that when $m = 1$, 60% of publisher-subscriber paths in DIBS have a stretch factor less than two. The stretch factor of DIBS is improved if we use more than one reference node; when m is 5 or 10, the corresponding percentage is 80%. On average, DIBS has a stretch factor of 2.5, 1.9, 1.8 when m is 1, 5, and 10, respectively. Although this is still worse than DR (average stretch factor 1.3), the gap is small. It is noted that DR requires the location of every node to be known while DIBS does not require any location information.

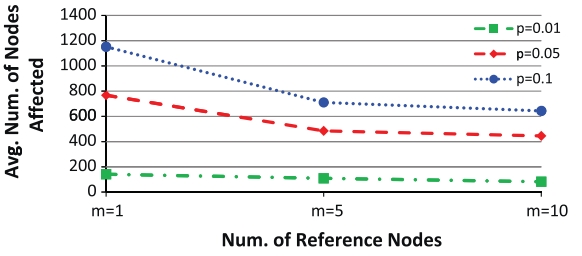
4.4. Effect of failure

We let a random fraction, $p \in \{1\%, 5\%, 10\%\}$, of the network down simultaneously and investigate the impact in terms of the following measures: (1) search recall: the ratio between the number of matching query/event pairs found to the (perfect) number of matching pairs as if there were no failure, and (2) repair cost: the average number of nodes that need to update state as a result of a node failure.

When $m = 1$, for each event there is only one event publication path which sends the event from its publisher node to its only designated node. The designated node stores all the queries that match the event. If the event is dropped along this path, no matching queries will be notified. When $m > 1$, there are multiple designated nodes for the event, each storing a subset of the queries that match the event. If one path fails to deliver the event to the corresponding designated node, the event still has other chances to reach the other designated nodes. Consequently, we expect that the search recall is higher if the network is divided into multiple clusters rather than one single cluster. This is observed in Fig. 9a; for example, while the search recall is 60% for $m = 1$ if 10% of the nodes fail, this recall is improved to 81% and 90% when $m = 5$ and $m = 10$, respectively.



(a) Search recall



(b) Repair cost

Fig. 9. Effect of failure.

When nodes fail, the repair cost is illustrated in Fig. 9b. While the cost to insert a node is very small because it involves only a single VA assignment, removal of a node is more sophisticated due to its effect on the nodes downstream the successor tree. Each of these downstream nodes needs to update its state information including its VA. In the case of single cluster ($m = 1$), when 10% of the network fails, less than 1200 nodes are affected, resulting in an average of $\frac{1200}{1479 \times 10\%} \approx 8.1$ nodes per node failure, whereas the average number is 13.5 nodes per failure if 1% of the network fails. The effect is less significant if the network is organized into more clusters. In the case of five clusters ($m = 5$), the average number of nodes affected per failure is 4.7 nodes (when 10% of the network fails) or 6.7 nodes (when 1% of the network fails). These numbers are even smaller when $m = 10$. Our conclusion is that the removal of a node does little effect on the remainder of the network, especially when the network is divided into more clusters. The effect averaged for each failure even decreases if the network experiences a more frequent failure rate.

5. Conclusions

DIBS is a novel design to enable efficient information brokerage in large-scale sensor networks. Relying on an application-independent content-based routing substrate (CRES), DIBS allows multiple publish/subscribe applications to run on the same network simultaneously without network reprogramming. Our evaluation study has substantiated the efficiency of DIBS and its capability to cope

Please cite this article in press as: C. Pham, D.A. Tran, DIBS: Efficient distributed information brokerage in large-scale sensor networks, Ad Hoc Netw. (2012), http://dx.doi.org/10.1016/j.adhoc.2012.08.008

with failures. DIBS can also notify a subscriber node reasonably quickly when an event of its interest emerges anywhere in the network. Although DIBS does not require location information, it has been shown to outperform a representative location-based technique, especially in terms of storage cost and computation cost. DIBS is most efficient if it is adopted for publish/subscribe applications where events occur less frequently than queries. In our future work, we will extend DIBS to address cases where events are more frequent; for example, dealing with stream events. We will also investigate how this framework can be extended to incorporate mobile nodes; they would be treated as new nodes when they enter a cluster or departing nodes when they leave a cluster.

Acknowledgments

This work was supported in part by the National Science Foundation awards CNS-1116430 and CNS-0753066 and by the UMass Boston's Proposal Development Grant. 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 National Science Foundation.

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, 2005, p. 70.

[3] 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, The ACM Transactions on Sensor Networks 2 (1) (2006) 1–38, http://dx.doi.org/10.1145/1138127.1138128.

[4] I.F. Akyildiz, I.H. Kasimoglu, Wireless sensor and actor networks: research challenges, Ad Hoc Networks 2 (4) (2004) 351–367.

[5] T. Melodia, D. Pompili, V.C. Gungor, I.F. Akyildiz, Communication and coordination in wireless sensor and actor networks, IEEE Transactions on Mobile Computing 6 (10) (2007) 1116–1129, http://dx.doi.org/10.1109/TMC.2007.1009.

[6] M. Castro, P. Druschel, A. Kermarrec, A. Rowstron, SCRIBE: a large-scale and decentralized application-level multicast infrastructure, IEEE Journal on Selected Areas in communications (JSAC) 20 (8) (2002) 1489–1499.

[7] A. Gupta, O.D. Sahin, D. Agrawal, A.E. Abbadi, Meghdoot: content-based publish/subscribe over p2p networks, in: Middleware '04: Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware, Springer-Verlag New York, Inc., New York, NY, USA, 2004, pp. 254–273.

[8] I. Aekaterinidis, P. Triantafillou, Internet scale string attribute publish/subscribe data networks, in: CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management, ACM Press, 2005, pp. 44–51.

[9] E. Fidler, H.-A. Jacobsen, G. Li, S. Mankovski, The padres distributed publish/subscribe system, in: FIW, 2005, pp. 12–30.

[10] L. Fiege, M. Cilia, G. Muhl, A. Buchmann, Publish-subscribe grows up: support for management, visibility control, and heterogeneity, IEEE Internet Computing 10 (1) (2006) 48–55, http://dx.doi.org/10.1109/MIC.2006.17.

[11] A. Carzaniga, D.S. Rosenblum, A.L. Wolf, Design and evaluation of a wide-area event notification service, ACM Transactions on Computer Systems 19 (3) (2001) 332–383. citeseer.ist.psu.edu/482106.html.

[12] X. Li, Y.J. Kim, R. Govindan, W. Hong, Multi-dimensional range queries in sensor networks, in: SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems, ACM, New York, NY, USA, 2003, pp. 63–75, http://dx.doi.org/10.1145/958491.958500.

[13] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, F. Yu, Data-centric storage in sensornets with ght, a geographic hash table, Mobile Networks and Applications 8 (4) (2003) 427–442, http://dx.doi.org/10.1023/A:1024591915518.

[14] D. Ganesan, B. Greenstein, D. Estrin, J. Heidemann, R. Govindan, Multiresolution storage and search in sensor networks, IEEE Transactions on Storage 1 (3) (2005) 277–315, http://dx.doi.org/10.1145/1084779.1084780.

[15] Y.-C. Chung, I.-F. Su, C. Lee, Supporting multi-dimensional range query for sensor networks, in: ICDCS '07: Proceedings of the 27th International Conference on Distributed Computing Systems, IEEE Computer Society, Washington, DC, USA, 2007, p. 35, http://dx.doi.org/10.1109/ICDCS.2007.143

[16] Q. Fang, J. Gao, L.J. Guibas, Landmark-based information storage and retrieval in sensor networks, in: IEEE INFOCOM, 2006.

[17] 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, http://dx.doi.org/10.1145/345910.345920, <http://portal.acm.org/citation.cfm?id=345920>.

[18] 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.

[19] 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, http://dx.doi.org/10.1145/1282380.1282387.

[20] 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.

[21] 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, <http://portal.acm.org/citation.cfm?id=345953>.

[22] 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, http://dx.doi.org/10.1145/1161089.1161122.

[23] S. Funke, I. Rauf, Information brokerage via location-free double rulings, in: ADHOC-NOW, 2007, pp. 87–100.

[24] C.-H. Lin, J.-J. Kuo, M.-J. Tsai, Reliable gps-free double-ruling-based information brokerage in wireless sensor networks, in: INFOCOM (miniconference), San Diego, CA, 2010.

[25] D.A. Tran, C. Pham, A content-guided publish/subscribe mechanism for sensor networks without location information, Computer Communications 33 (13) (2010) 1515–1523.

[26] M. Caesar, M. Castro, E.B. Nightingale, G. O'Shea, A. Rowstron, Virtual ring routing: network routing inspired by dhts, SIGCOMM Computer Communication Review 36 (4) (2006) 351–362, http://dx.doi.org/10.1145/1151659.1159954.

[27] J. Garcia-Luna-Aceves, D. Sampath, Scalable integrated routing using prefix labels and distributed hash tables for manets, in: Mobile Adhoc and Sensor Systems, 2009. MASS '09. IEEE 6th International Conference on, 2009, pp. 188–198.

[28] M. Albano, S. Chessa, Publish/subscribe in wireless sensor networks based on data centric storage, in: Proceedings of the 1st International Workshop on Context-Aware Middleware and Services: affiliated with the 4th International Conference on Communication System Software and Middleware (COMSWARE 2009), CAMS '09, ACM, New York, NY, USA, 2009, pp. 37–42, http://dx.doi.org/10.1145/1554233.1554243.

[29] M. Albano, S. Chessa, F. Nidito, S. Pelagatti, Dealing with nonuniformity in data centric storage for wireless sensor networks, IEEE Transactions on Parallel and Distributed Systems 22 (2011) 1398–1406, http://dx.doi.org/10.1109/TPDS.2011.18. http://doi.ieeecomputersociety.org/.

[30] C.G. Rezende, B.P.S. Rocha, A.A.F. Loureiro, Publish/subscribe architecture for mobile ad hoc networks, in: Proceedings of the 2008 ACM Symposium on Applied Computing, SAC '08, ACM, New

York, NY, USA, 2008, pp. 1913–1917, http://dx.doi.org/10.1145/1363686.1364149.
[31] C.E. Perkins, Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers, in: ACM SIGCOMM, 1994, pp. 234–344.
[32] J.K. Lawder, P.J.H. King, Using space-filling curves for multi-dimensional indexing, in: BNCOD 17: Proceedings of the 17th British National Conferenc on Databases, Springer-Verlag, London, UK, 2000, pp. 20–35.
[33] 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.

Cuong (Charlie) Pham is a PhD candidate in the Department of Computer Science at the University of Massachusetts at Boston and a research member of NISLab. He received a BS degree in Computer Science from Bowman Technical State University in Moscow, Russia in 2007. His research interests are P2P networks and wireless sensor networks. He was a research intern at EMC and Huawei Technologies during summers of 2010 and 2011. He received a Student Travel Award from the NSF and a Research Excellence Award from the Department of Computer Science (UMass Boston), both in 2009.

Duc A. Tran is an Assistant Professor in the Department of Computer Science at the University of Massachusetts at Boston, where he leads the Network Information Systems Laboratory (NISLab). He received a PhD degree from the University of Central Florida. His current research projects are focused on data management and networking designs for decentralized networks such as P2P networks, sensor networks, and disruption-tolerant networks. He has received research awards from the National Science Foundation (NSF), a Best Paper Award at ICCCN 2008, and a Best Paper Recognition at DaWak 1999. Dr. Tran has engaged in many professional activities, serving as an Editor for the Journal on Parallel, Emergent, and Distributed Systems (2010-date), Guest-Editor for the Journal on Pervasive Computing and Communications (2009), TPC Co-Chair for CCNet (2010, 2011), GridPeer (2009, 2010, 2011), and IRSN 2009, TPC Vice-Chair for AINA 2007, TPC member for 50+ international conferences, and referee and session chair for numerous journals/conferences. He is a Senior Member of ACM and a Professional Member of the IEEE.

Please cite this article in press as: C. Pham, D.A. Tran, DIBS: Efficient distributed information brokerage in large-scale sensor networks, Ad Hoc Netw. (2012), http://dx.doi.org/10.1016/j.adhoc.2012.08.008