

An Adaptive Framework of Multiple Schemes for Event and Query Distribution in Wireless Sensor Networks

Vincent Tam, Keng-Teck Ma and King-Shan Lui

Department of Electrical and Electronic Engineering
The University of Hong Kong, Pokfulam, Hong Kong.
{vtam,ktma,kslui}@eee.hku.hk

Abstract—Wireless sensor networks are useful for many real-world applications including environmental monitoring, military applications, disaster management, etc. In many cases, interesting events detected by sensors are disseminated to some targeted node(s) for storage whereas queries for specific event types would be directed by a data dissemination protocol aiming to the “right” storage node(s) for definite answers. Nevertheless, the query/event ratio can be changing over time in many practical applications like the tracking of endangered animals in an open safari due to the seasonal trend or other factors. This varying query/event ratio will significantly affect the overall performance of different data dissemination schemes in the underlying sensor networks. In this paper, we consider an adaptive framework that can flexibly switch from one scheme to another based on the results quickly evaluated by our cost models. We implemented several GHT-based schemes including our adaptive GHT (AGHT) for event distribution in the JSim packages, and compared their performance on an example safari application with changing query/event ratios. In our simulation results, the AGHT clearly excelled the other GHT-based schemes with the lowest storage and communication overheads. More importantly, these promising results shed light on many possible directions for future investigation.

I. INTRODUCTION

Sensors are small devices that have capability to detect objects, collect information, and communicate with each other. A wireless sensor network is a network consisting of hundreds or thousands of sensors that span a large geographical region. These sensors are able to communicate with each other to collaboratively detect objects, collect information, and transmit messages. Sensor networks have become an important technology especially for environmental monitoring, military applications, disaster management, etc [4], [9].

Sensor networks differ from traditional wireless networks in many aspects. Therefore, many mechanisms developed for traditional wireless networks cannot be directly applied in sensor networks. In this paper, we study the problem of distributing events and queries in wireless sensor networks. An event is something of interest, such as a sharp change in temperature or the appearance of some endangered animals in a safari, detected by a sensor node. A query is a request of information of a certain event issued by a node in the sensor network. Different nodes may be interested in different events.

A node knows what events it is interested in but does not know where these events happen. To get an answer, the query has to be sent to a node that possesses the information. A traditional approach to facilitate query nodes to acquire what they want is flooding. In the *push-based* approach, whenever an event happens, the information is flooded to all the nodes in the network. Then, every node knows the information when it needs it and no query is needed. In the *pull-based* approach, a node that detects an event keeps the information in its own storage without sending the information to any other node. When a node wants that information, it floods the network with a query. The node that possesses the requested information replies the query. Obviously, energy can be saved if information is kept and distributed in a more intelligent manner.

To allow efficient routing of queries, the concept of *data-centric storage (DCS)* is introduced [13]. When a sensor detects an event, it decides which location is responsible for keeping the information and the information is sent to a node in that location. The location depends on the type of information and can be computed using a globally known hash function as used in the Geographic Hash Table (GHT) approach [10]. Then, when a node requested a particular piece of information, it can find out the location by the hash function and send a query to that location. A number of protocols have been developed based on the idea of DCS [2], [6], [8], [10], [12], [14]. An important issue of DCS is where to keep event information since it affects how a query is propagated. This in turn affects the number of messages it takes to retrieve event information. For example, if an event is kept in a remote node in the network, a query may have to travel a long way, so does the answer message. On the other hand, if the event is kept in a node sitting in the center of the network, no matter which node issues a query, the query does not have to go through a very long path. Although keeping information in the center seems to reduce the message overhead of query propagation, keeping all the information in the center will highly increase the workload of the nodes there and is not appropriate in sensor networks. Therefore, it is necessary to strike a balance.

Nevertheless, in many real-life applications such as the

tracking of endangered animals in an open safari or the monitoring of indoor conditions including humidity and temperature for plants inside different greenhouses in a farm, the query/event ratio can be changing over time due to the seasonal or other factors. This varying query/event ratio will significantly affect the overall performance of various data dissemination schemes in the underlying sensor networks. In this paper, we consider an adaptive framework that can flexibly switch from one scheme to another based on the results quickly evaluated by our cost models. We implemented several GHT-based schemes including our adaptive GHT (AGHT) for event and query distribution in the JSim packages, and compared their performance on an example safari application with changing query/event ratios. In our simulation results, the AGHT clearly excelled the GHT-based or other schemes considered with the lowest storage and communication overheads. More importantly, these promising empirical results shed light on many possible directions for future investigation.

This paper is organized as follows. Several data dissemination schemes over wireless sensor networks are considered in Section II. Section III presents the pseudo-code of our adaptive framework which can flexibly switch from one scheme to another after comparing their latest costs returned by the predetermined cost functions. Simulation results of our adaptive framework utilizing multiple schemes against those of the individual schemes are evaluated in Section IV. Lastly, we conclude our work in Section V.

II. RELATED WORK

Some protocols have been developed based on the data-centric storage [2], [6], [8], [10], [12], [14].

[6] studies how to efficiently answer queries about events that have scalar attributes. An example query provided in the paper is "list all events whose temperature lies between 50° and 60° , and whose light levels lie between 10 and 15." The authors develop a hash function such that events having similar attribute values are kept in nearby nodes to allow more efficient querying.

[2] develops a *multi-resolution querying* scheme. An event is detected by a set of nodes instead of one. Different subsets of these nodes provide different resolutions of the event. Some nodes, called registration points, are designated to keep the sources of events but not the actual information of the events. When a node requests information about the event, it also has to specify the resolution it wants. The query is sent to the registration points and the registration points forward the query to the event sources. The event sources will then send replies to the requesting node based on the resolution requested.

The event model in this paper is similar to those used in [10], [8], [7], [12], [14] that an event is something of interest and can be any kind of information. Events are independent of each other and occur randomly at anywhere.

GHT [10] is one of the earliest works on event and query distribution for DCS. Each event is kept in a particular geographical location. This location is computed using a globally-known hash function. Since it is possible that there is no

node in the designated location, GHT develops a mechanism to assign a node in the neighborhood to keep the event. GPSR [5], a routing protocol that routes packets according to geographical location instead of IP addresses, is enhanced to route messages to such a node. When a node searches for an event, it computes the location of keeping the event using the hash function and send a query using the enhanced GPSR. GEM [8] follows similar idea except that it works in a network where geographical location information is not available. The authors propose to embed a tree in the network to provide relative location information. The authors also develop a routing algorithm to route messages on the tree.

It is not difficult to see that if the network is static and nodes never fail, GHT is able to answer all queries. However, when sensors are highly mobile, it is very likely that information is not kept in the calculated geographical location anymore by the time the query arrives. To solve the problem, [12] suggests to divide the network into geographical regions. Information of an event is kept in all nodes in that region instead of in only one node. Then, as long as there is a node having the information stays in the region, a query can get an answer. Nevertheless, similar to [8] and [10], [12] has the hot spot problem. To simplify our discussion, we call the node that detects an event *event node*, the node that keeps the information *designated node*, and the node that issues a query *query node*. When an event is very popular, nodes surrounding the designated node and the designated node will have to transmit a lot of queries, which uses up energy quickly.

To provide different resiliency levels for different events, keeping an event in multiple locations is suggested in [14]. More important information is kept in more hash locations to enhance resiliency. This mechanism alleviates the hot-spot problem to a certain extent due to the duplication of event storage. Nevertheless, an event may be sent to several locations which are very far apart, the number of messages to distribute an event is also a multiple of what are needed in GHT.

To solve the message hot-spot problem, keeping the same information in multiple locations is the only way for DCS. [1] proposes a variant of GHT that distributes events only horizontally and queries only vertically across the whole sensor network. And the number of messages required is proportional to the dimension of the network. Basically, path that an event passes through is the event distribution path whereas the path that a query traverses is the query distribution path. When the event distribution path and query distribution path intersect, the query can be answered.

On the other hand, to reduce the storage hotspot, we may devise an alternative (non-DCS) scheme, namely the Local Storage (LS), in which each event detected is stored locally by the detecting node. When a query is routed to a home node that cannot answer the query, a flooding over the whole network will be needed to answer that particular query. Clearly, this strategy will significantly raises the message overheads. Nevertheless, in any DCS, message and storage overheads are always trade-offs to each other. Thus, in the next section, we are going to consider an adaptive framework embedding

multiple schemes that can flexibly switch from one scheme to another according to their relative costs computed on-the-fly.

III. OUR ADAPTIVE FRAMEWORK USING MULTIPLE SCHEMES

In this paper, we propose an adaptive framework utilizing various event and query distribution schemes that may work perfectly to reduce the total message overheads at different query-event ratios. Based on the relative values of the total message overheads estimated dynamically, our adaptive framework can flexibly switch from one scheme to another in order to minimize the overall message overheads in any location-aware sensor network as long as the switching costs are relatively small, and will never outweigh the gains obtained from the switching. After all, our adaptive framework can easily be generalized to consider other performance criteria such as the storage or other costs formulated as a weighted and multi-objective optimization function for any real-world application in a wireless sensor network.

In our subsequent discussion, we focus on two major event and query distribution schemes, namely the GHT for DCS and the (non-DCS) Local Storage (LS) schemes. To enable our adaptive framework to precisely determine which scheme to switch to, we need to firstly perform a careful cost analysis as follows. Basically, following the notations used in [10], we define n as the number of nodes equipped to detect T event types, D_{total} as the total number of events detected, Q as the number of event types for which queries are issued, and lastly D_q is the number of events detected for the types of events being enquired for. When we assume **there is only one query per event type**, there are altogether Q queries in total. The total message cost for the Local Storage (LS) scheme will be :

$$Q \times n + D_q \times \sqrt{n}$$

The first part of the above cost function considers the message overheads of flooding to n nodes for all the Q queries whereas the second part gives the averaged message cost to answer all the queries. On the other hand, the total message cost for the GHT scheme as according to [10] is :

$$Q \times \sqrt{n} + D_{total} \times \sqrt{n} + D_q \times \sqrt{n}$$

in which the first and last parts of the formulae concern about the averaged message costs for routing the queries to the hashed nodes and vice versa so as to answer the queries, while the middle part gives the averaged message costs associated with the storage of events being detected.

As obvious from the above cost functions, when $Q \times \sqrt{n} + D_{total} \times \sqrt{n} < Q \times n$ implying $Q/D_{total} > 1/(\sqrt{n} - 1)$, the GHT outperforms the LS with a smaller overall message overheads. This occurs when the number of nodes (and so \sqrt{n}) or the ratio of (*query : event*) is relatively large. On the other hand, when $Q/D_{total} < 1/(\sqrt{n} - 1)$, the LS excelled the GHT with respect to the above cost functions, i.e. with relatively few queries when compared to the number of events detected. After all, in many real-world applications of wireless sensor

networks, such ratio of (*query : event*) can be dynamically changing over time due to the seasonal trends of involved events or other environmental factors such as the availability of users to pose queries, or sometimes failure of sensors to detect certain events due to obstacles, diminishing battery power or physical locations. An illustrative example is the tracking of endangered animals in open safaris that is basically subject to seasonal changes.

Nevertheless, this motivates us to look into an adaptive framework to exploit the effectiveness of both GHT and LS that works on a event-per-type basis, and aims to minimize the message overheads in the following manner.

- 1) Initially, our adaptive framework arbitrarily starts with the LS since there will be in general relatively few events detected in the beginning.
- 2) When any event is detected, it will simply be stored in the detecting node;
- 3) When any query is generated, it will be routed to the home node as used in the GHT scheme.
- 4) Upon a query is received, the home node will compute the cost of LS against that of GHT according to the cost functions defined previously. If the cost of LS is higher than that of GHT, the home node will try to switch **from LS to GHT**. Conversely, when the home node is already in the GHT mode, and the cost of LS is lower than that of GHT, the home node will switch **from GHT to LS** in which the home node will flood the whole network with the query.
- 5) Any node with the locally stored event being enquired will reply to the home node which will update the count for the events being detected, and then relay the event (i.e. answer) back to the query node.
- 6) goto step 2. until no more event or query remains, or any of the preset (time or iteration) limits is already reached.

The protocol to implement the switching mechanism in step 4) is explained in greater detail as follows.

- **Switching from LS to GHT:** a) the home node will start the process of switching from LS to GHT by firstly setting the GHT flag in the next arriving query packet to be broadcasted to the whole network as still under the LS mode. After that, the home node will switch itself to the GHT mode. It is worth noting that since events are stored locally while in the LS mode, to switch from LS to GHT, the arrival of new events in the detecting nodes cannot reset the whole network. Thus, our protocol must rely solely on the next arriving query at the home node to initiate broadcasting so as to reset the whole network; b) upon receiving the broadcasted query packet, each node will reset itself to the GHT mode and then actively route any future event detected to the home node as according to the GHT scheme; c) once the home node is switched to the GHT mode, it can directly answer any query by itself without broadcasting. Basically, our proposed protocol aims to minimize the message overheads when switching from the LS to GHT mode.

- **Switching from GHT to LS:** a) the home node will firstly switch itself to LS mode after answering the particular query at hand. Before the next query arrives, the home node will set the LS flag embedded in the ACK packets for all the future events being detected; b) upon receiving such ACK packet for a particular event type, the individual node will switch itself to LS mode and then simply store any future event of that event type instead of routing to the home node for storage. This has the overall effect of incrementally turning each detecting node into LS mode until the next query arrives. However, if the next query arrives before any event being detected, it will naturally flood the whole sensor network with the LS flag already set in its broadcasted query. Here, our protocol also aims to achieve the minimal message overheads for switching.

Furthermore, each home node will maintain a list of the recently received events and queries, with one for each event type. Each entry in the list is time-stamped so that a sliding window mechanism can be implemented to filter out those “expired” events or queries that may be no longer relevant to our latest decision for switching. The list will then be used to compute the number of “relevant” events or queries as determined by the sliding window per designated event type for the calculation of the predefined cost functions. It is worth considering how the home nodes collect the global information such as the number of relevant events to compute the cost functions under the LS mode. In fact, the LS mode still make use of any broadcasted query to collect the number of events occurred for that specific event type. Each node that stores some events of the queried type will reply to the home node. Therefore, the home node will ultimately have the total number of events for each event type that has been queried. After all, it is important to determine an appropriate size of the sliding window that will affect our switching decisions and thus the overall performance of our adaptive framework. When the size of the sliding window is too small, our adaptive framework can only compute the message costs based on the most recent events or queries. Otherwise, lots of irrelevant events or queries will be included in a relatively large sliding window for our switching decisions.

IV. SIMULATION

We evaluate the performance of our adaptive framework, namely the AGHT, and compare it against the original LS and GHT schemes using simulations.

A. Application Scenario

Obviously, our proposed framework can be most beneficial for applications with gradual changes of the *query/event* ratios over time. For applications with very much constant *query/event* ratios, the overheads in storage and computation as demanded by our approach may simply adversely affect the overall performance due to the extra costs incurred. On the other hand, for applications with rapidly changing *query/event* ratios, our adaptive framework may perform

worse than a fixed protocol such as the GHT for event and query distribution since the uses of sliding windows may reduce the responsiveness of our approach to continuous changes in the *query/event* ratios. On top of it, the costs incurred for frequent switching may not be justifiable. After all, there are still lots of real-life applications in which we can find gradual change patterns of the *query/event* ratios over time.

An example is the open safari with many different species of animals. Some of them can be the endangered species. A sensor network is thus deployed to detect and identify the different species of animals. An observer can pose a query anywhere in the sensor network for the detection of a particular type of animals. In general, the number of detected animals (events) can be n times of the number of queries issued over the sensor network. In our simulations below, we arbitrarily set n to be 20 in which the range of n as 5...20 is not uncommon in real-life applications involving regular monitoring. Furthermore, the observers in open safari may tend to pose queries for some particular species in summer, and other different species in winter.

B. Simulation Setup

The setup for our simulations using JSim [3] is summarized as Table I. Queries are generated randomly, both in locations and the events being enquired. Similarly, events are randomly generated in both locations and time. To prevent flooding, the new query (or event) will only be added after an REPLY (or ACK) packet is received. For any unanswered query, the query node will retry 10 times at an interval of 0.5 second before giving up. Events will be re-sent at double of the previous interval until an ACK message is received. At each iteration, events and queries will have equal chances of 50% being generated. The costs of packet transmission and reception is normalized as according to [11]. For the first 25 queries in

Node Density	1 node / 256 m^2
Radio Range	40m
Number of Nodes	50, 100, 150, 200
Simulation Time	500s
Query Nodes	1
Number of Queries	50
Event Types	2
Events Detected Per Type	100, 500, 1000
Sliding Window Size	5.0s

TABLE I
SIMULATION SETUP

our simulation, the ratios of the two event types are 5 : 1. For the last 25 queries, the ratios of the two event types are reversed as 1 : 5. This settings may simulate the changing of *query/event* ratios over the various seasons in our open safari application described above.

C. Simulation Results

We measure (1) the averaged maximum and overall average numbers of messages involved for event and query distribution,

(2) the averaged maximum and averaged total of events being stored in the wireless sensor network for the GHT, LS and AGHT schemes over 5 runs for each setting as specified in Table I. The first set of performance parameters shows the maximum and averaged message overheads required by each individual scheme for event and query distribution while the second set reflects whether the individual scheme is demanding in storage costs or not. Since our proposed framework mainly works to minimize on the message costs, we focus on comparing the message overheads of the GHT, LS and AGHT schemes in in Table II, III and IV respectively when the number of events increases from 100 to 500 and lastly to 1,000. Nevertheless, as far as the storage costs are concerned, the AGHT scheme is generally outperformed by the original GHT scheme. When compared to the LS scheme, our proposed AGHT scheme is much less demanding in storage costs when the number of events increases from 100 to 1,000. After all, we can always fine-tune the cost functions of our adapted framework to minimize on the storage costs as well in our future investigation.

In the 3 tables, the lowest figures in message overheads among the 3 schemes for comparison are marked with asterisks. Our adaptive scheme requires fewer number of messages while reducing the averaged number of messages a node has to handle in most cases when the number of events increases from 500 to 1,000. We therefore conclude that our adaptive framework is an effective scheme for event and query distribution in wireless sensor networks.

Size	Max. Message Cost			Avg. Message Cost		
	GHT	LS	AGHT	GHT	LS	AGHT
50	*13845	24228	14310	*1975	7221	2233
100	*12464	23017	12570	*632	5560	763
150	*12989	25052	13283	*518	5826	665
200	*12602	25353	12767	*410	5412	540

TABLE II
50 QUERIES AND 100 EVENTS

Size	Max. Message Cost			Avg. Message Cost		
	GHT	LS	AGHT	GHT	LS	AGHT
50	26680	24295	*23706	3399	5708	*2874
100	28233	28295	*25313	2428	6211	*2318
150	*26277	28768	27108	*1253	5941	1425
200	*24988	30752	25691	*939	5376	1073

TABLE III
50 QUERIES AND 500 EVENTS

V. CONCLUSION

In this paper, we study the problem of distributing events and queries in sensor networks using data-centric storage or local storage (LS) approach when the query-to-event ratio can be changing over time in many real-life applications. We first identify the weaknesses of existing approaches and then describe our new adaptive framework which can flexibly

Size	Max. Message Cost			Avg. Message Cost		
	GHT	LS	AGHT	GHT	LS	AGHT
50	45722	28730	*24163	6795	6193	*3256
100	44342	33581	*30450	3095	5971	*2146
150	44825	*34344	36536	2132	5889	*1811
200	41642	*34737	37397	1649	5408	*1611

TABLE IV
50 QUERIES AND 1000 EVENTS

switch from one scheme to another based on the message costs or possibly other criteria. We measure the performance of our proposal using extensive simulations and compare with the LS and GHT, a pioneer work in this area. Simulation results show that our adaptive framework outperforms GHT in cases where the number of events increases from 500 to 1,000 with the total number of queries as 50, i.e. at a diminishing ($query : event$) ratio. Therefore, we conclude that our adaptive framework is a promising mechanism for event and query distribution for many real-world applications with changing query-to-event ratios in wireless sensor networks.

ACKNOWLEDGMENT

The authors would like to thank Professor Yi Shang for his fruitful discussion.

REFERENCES

- [1] M. Chan, K. Lui, and V. Tam. Efficient event and query distribution in sensor networks. In *CollaborateCom 2005*, pages 251 – 258, 2005.
- [2] J. Chen, Y. Guan, and U. Pooch. An efficient data dissemination method in wireless sensor networks. In *IEEE Global Telecommunications Conference (Globecom)*, volume 5, pages 3200 – 3204, 2004.
- [3] Development Team of the Jsim. Jsim: A java-based simulation and animation environment. available at : <http://chief.cs.uga.edu/jam/jsim/> (lastly visited: September 12, 2006).
- [4] D. Estrin, D. Culler, K. Pister, and G. Sukhatme. Connecting the physical world with pervasive networks. In *IEEE Pervasive Computing*, pages 59 – 69, 2002.
- [5] B. Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *ACM Mobicom*, pages 243 – 254, 2000.
- [6] X. Li, Y. J. Kim, R. Govindan, and W. Hong. Multi-dimensional Range Queries in Sensor Networks. In *ACM SenSys*, pages 63 – 75, 2003.
- [7] X. Liu and Q. H. Y. Zhang. Combs, Needles, Haystacks: Balancing Push and Pull for Discovery in Large-Scale Sensor Networks. In *ACM SenSys*, 2004.
- [8] J. Newsome and D. Song. GEM: Graph EMbedding for Routing and Data-Centric Storage in Sensor Networks Without Geographic Information. In *ACM SenSys*, pages 76 – 88, 2003.
- [9] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. In *Communications of the ACM*, volume 43, pages 51 – 58, 2000.
- [10] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: A Geographic Hash Table for Data-Centric Storage. In *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, pages 78 – 87, 2002.
- [11] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava. Topology management for sensor networks: Exploiting latency and density. In *Mobihoc 2002*, 2002.
- [12] K. Seada and A. Helmy. Rendezvous Regions: A Scalable Architecture for Service Location and Data-Centric Storage in Large-Scale Wireless Networks. In *International Parallel and Distributed Processing Symposium (IPDPS)*, 2004.
- [13] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin. Data-Centric Storage in Sensornets. In *ACM SIGCOMM HotNets*, 2002.
- [14] R. Tamishetty, L. Ngoh, and P. Keng. An efficient resiliency scheme for data centric storage in wireless sensor networks. In *IEEE 60th Vehicular Technology Conference*, volume 4, pages 2936 – 2940, Fall 2004.