The ACQUIRE mechanism for efficient querying in sensor networks

Narayanan Sadagopan [†], Bhaskar Krishnamachari [§†], and Ahmed Helmy [§]

[†]Department of Computer Science, University of Southern California, Los Angeles, CA 90089, USA
narayans@usc.edu

[§]Department of Electrical Engineering - Systems, University of Southern California, Los Angeles, CA 90089, USA
{bkrishna, helmy}@usc.edu

***Abstract*— We propose a novel and efficient mechanism for obtaining information in sensor networks which we refer to as ACQUIRE. In ACQUIRE an active query is forwarded through the network, and intermediate nodes use cached local information (within a look-ahead of d hops) in order to partially resolve the query. When the query is fully resolved, a completed response is sent directly back to the querying node.**

We take a mathematical modelling approach in this paper to calculate the energy costs associated with ACQUIRE. The models permit us to characterize analytically the impact of critical parameters, and compare the performance of ACQUIRE with respect to alternatives such as flooding-based querying (FBQ) and expanding ring search (ERS). We show that with optimal parameter settings, depending on the update frequency, ACQUIRE obtains order of magnitude reduction over FBQ and potentially over 60% reduction over ERS in consumed energy".

I. INTRODUCTION

Wireless sensor networks are envisioned to consist of large numbers of devices, each capable of some limited computation, communication and sensing, operating under energy constraints in an unattended mode. These networks are intended for a broad range of environmental sensing applications from weather data-collection to vehicle tracking and habitat monitoring [1], [2], [3].

With large-scale networks of energy-constrained sensors it is not feasible to collect all measurements from each device for centralized processing. It has been argued that it is best to view such sensor networks as distributed databases [8], [9]. A central querier/data sink (or collection of queriers/sinks) issues queries that sources in the network respond to. Due to energy constraints it is desirable for much of the data processing to be done in-network, and this has led to the concept of *data-centric* information routing, in which the queries and responses are for named data. Depending on the applications, there are likely to be different kinds of queries in these sensor networks. The types of queries can be categorized in many ways, for example:

- *Continuous queries*, which result in extended data flows (e.g. "Report the measured temperature for the next 7 days with a frequency of 1 measurement per hour") versus One-shot queries, which have a simple response (e.g. "Is the current temperature higher than 70 degrees?")
- *Aggregate queries*, which require the aggregation of information from several sources (e.g. "Report the calculated average temperature of all nodes in region X") versus *Non-aggregate Queries* which can be responded to by a single node (e.g. "What is the temperature measured by node x?")
- *Complex queries*, which consist of several nested or batched sub-queries (e.g. "What are the values of the following variables: X, Y, Z?") versus *simple queries*, which have no sub-queries (e.g. "What is the value of the variable X?") [1]
- *Queries for replicated data*, in which the response to a given query can be provided by many nodes (e.g. "Is there at least one target in the area?") and *queries for unique data*, in which the response to a given query can be provided only by one node.

Flooding-based query mechanisms such as the Directed Diffusion data-centric routing scheme [4] are well-suited for continuous, aggregate queries. This is because the cost of the initial flooding of the interest can be amortized over the duration of the continuous flow from the source(s) to sink(s). *However, keeping in mind the severe energy constraints in sensor networks, a one-size-fits-all approach is unlikely to provide efficient solutions for other types of queries.*

In this paper we propose a new data-centric querying mechanism, ACtive QUery forwarding In sensoR nEtworks (ACQUIRE). We shall show that ACQUIRE is well-suited for one-shot, complex queries for replicated data. As a motivation for ACQUIRE, we describe a scenario which involves such a query:

- **Bird Habitat Monitoring:** Imagine a network of acoustic sensors deployed in a wildlife reserve. The processor associated with each node is capable of analyzing and identifying bird-calls. Assume each node stores any bird-calls heard previously. The task "obtain sample calls for the following birds in the reserve: Blue Jay, Nightingale, Cardinal, Warbler" is a good example of a *complex* (because information is being requested about four birds), *one-shot*

[1]We consider a query to be complex if it consists of several sub-queries that are combined by conjunctions or disjunctions in an arbitrary manner. Each sub-query in turn is a query for some variable tracked by the sensor network.

(because each sub-query can be answered based on stored and current data) query, and is *for replicated data* (since many nodes in the network are expected to have information on such birds). Another example of a complex, one-shot query in this network might be "return 5 locations where a Warbler's call has been recorded" (the request for each location is a sub-query).

The principle behind ACQUIRE is to inject an active query packet into the network that follows a (random or guided) trajectory through the network. At each step, the node which receives the active query performs a triggered, on-demand, update obtaining information from all neighbors within a look-ahead of d hops. As this active query progresses through the network it gets progressively resolved into smaller and smaller components until it is completely solved and is returned back to the querying node as a completed response.

While most prior work in this area has relied on simulations in order to test and validate data-querying techniques, we take here a mathematical modelling approach that allows us to derive analytical expressions for the energy costs associated with ACQUIRE and compare it with other mechanisms, and to study rigorously the impact of various parameters such as the value of the look-ahead parameter, and the ratio of query rate to update rate.

The rest of the paper is organized as follows: in section II we describe some of the related work in the literature. We provide a basic description of the ACQUIRE mechanism in section III. In section IV we develop our mathematical model for ACQUIRE and in section V we compare it with two alternative mechanisms: flooding based queries (FBQ) and expanding ring search (ERS). We discuss these results and describe the future work we are planning to undertake along with our concluding comments in section VI.

II. Related Work

The ACQUIRE mechanism we describe in this paper is compatible with a database perspective on sensor networks, such as has been outlined by Bonnet, Gehrke, Seshadri and Yao in [9], [14] and by Govindan, Hellerstein, Hong *et al.* in [8]. ACQUIRE can be viewed as a data-centric routing mechanism that provides superior query optimization for responding to particular kinds of queries in sensor networks: complex, one-shot queries for replicated data.

Intanagonwiwat, Govindan, Estrin and Heidemann propose and study Directed Diffusion [4], [5], a data-centric protocol that is particularly useful for responding to long-standing/continuous queries. In Directed Diffusion, an interest for named data is first distributed through the network via flooding (although optimizations are possible for geographically localized queries), and the sources with relevant data respond with the appropriate information stream. Also related to our work are the Information Driven Sensor Querying (IDSQ) and Constrained Anisotropic Diffusion Routing (CADR) mechanisms proposed by Chu, Hausseker and Zhao [12].

One recent technique that is close in spirit to ACQUIRE is the rumor-routing mechanism proposed recently by Braginsky and Estrin in [16]. Their approach is quite interesting - sources with events launch mobile agents which execute random walks in the network resulting in event-paths. The queries issued by the querier/sink, in a manner somewhat similar to ACQUIRE, are also mobile agents that follow random walks. Whenever a query agent intersects with an event-path, it uses that information to efficiently route itself to the location of the event. Rumor routing is, however, primarily a mechanism to lower the interest-flooding cost for Directed Diffusion in situations where geographical information may not be available. It is conceivable to combine rumor routing with ACQUIRE in order to guide the trajectory taken by queries towards regions of the network with relevant information. Another approach for guiding the queries might be the idea of routing along curves, described by Nath and Niculescu in [15].

The recent work by Ratnasamy, Karp *et al.* [11] presents a geographic hash table technique for data-centric storage (DCS) in sensor networks. In estimating the cost of local storage the authors of [11] assume the use of flooding-based queries, to which we provide an alternative in this paper. It is also possible to conceive of using our ACQUIRE scheme in conjunction with any DCS techniques that result in replication (e.g. for robustness reasons).

Our work also has some similarities to techniques proposed for searching in unstructured peer-to-peer (P2P) overlay networks on the Internet. In particular, [17] discusses the possibility of launching k-random walks through the unstructured P2P network for discovering required files/data.

Our ACQUIRE mechanism combines a trajectory for active queries with a localized update mechanism whereby each node on the path utilizes information about the all nodes within a look-ahead of d hops. The size of this look-ahead parameter effects a tradeoff between the information obtained (which helps reduce the length of the overall trajectory) and the cost for obtaining the information. This look-ahead region is somewhat similar in spirit to the notion of zones in the Zone Routing Protocol (ZRP) [13] and to the notion of neighborhoods in the Contact-based Architecture for Resource Discovery (CARD) [10] developed for mobile ad-hoc networks.

III. Basic Description of ACQUIRE

In order to explain ACQUIRE, it is best to begin first with an overview of traditional flooding-based query techniques. In these techniques, there is a clear distinction between the query dissemination and response gathering stages. The querier/sink first floods several copies of the query (which is an interest for named data). Nodes with the relevant data then respond. If it is not a continuous/persistent query (i.e. one that calls for data from sensors for an extended period of time as opposed to a single value), then the flooding can dominate the costs associated with querying. In the same way, even when data aggregation

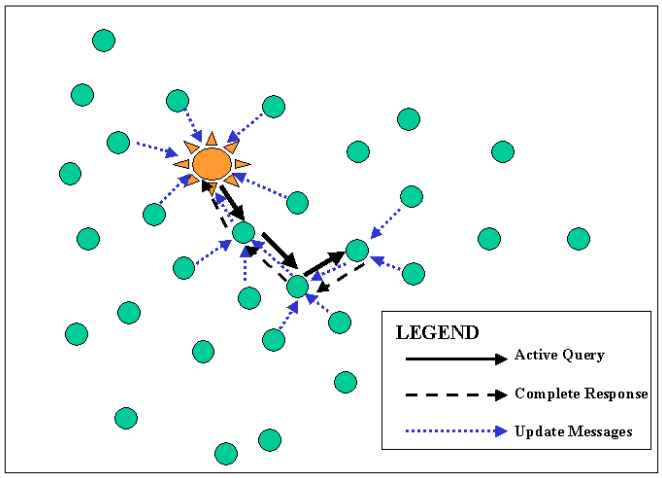

Fig. 1. Illustration of ACQUIRE with a one-hop lookahead ($d = 1$). At each step of the active query propagation, the node carrying the active query employs knowledge gained due to the triggered updates from all nodes within d hops in order to partially resolve the query. As d becomes larger, the active query has to travel fewer steps on average, but this also raises the update costs. When d becomes extremely large, ACQUIRE starts to resemble traditional flooding-based querying.

is employed, duplicate responses can result in suboptimal data collection in terms of energy costs.

By contrast, in ACQUIRE there are no distinct query/response stages. The querier issues an *active query* which can be a complex query, i.e. can consist of several sub-queries, each corresponding to a different variable/interest. The active query is forwarded step by step through a sequence of nodes. At each intermediate step, the node which is currently carrying the active query (the *active node*) utilizes updates received from all nodes within a lookahead of d hops in order to resolve the query partially. New updates are triggered reactively by the active node upon reception of the active query only if the current information it has is obsolete (i.e. if the last update occurred too long ago). After the active node has resolved the active query partially, i.e. after it has utilized its local knowledge to answer as much of the complex query as possible, it chooses a next node to forward this active query to. This choice may be done in a random manner (i.e. the active query executes a random walk) or directed intelligently based on other information, for example in such a way as to guarantee the maximum possible further resolution of the query. Thus as the active query proceeds through the network, it keeps getting "smaller" as pieces of it become resolved, until eventually it reaches an active node which is able to completely resolve the query, i.e. answer the last remaining pieces of the original query. At this point, the active query becomes a *completed response* and is routed back directly (along either the reverse path or the shortest path) to the originating querier.

The ACQUIRE scheme with a look-ahead of 1 is illustrated in figure 1.

IV. Analysis of ACQUIRE

A. Basic Model and Notation

Consider the following scenario: A sensor network consists of X sensors. This network tracks the values of certain variables like temperature, air pressure, humidity, etc. Let $V = \{V_1, V_2, ...V_N\}$ be the N variables tracked by the network. Each sensor is equally likely to track any of these N variables. Assume that we are interested in finding the answer to a query $Q = \{Q_1, Q_2, ...Q_M\}$ consisting of M sub-queries, $1 < M \leq N$ and $\forall i : i \leq M, Q_i \in V$. Let S_M be the average number of steps taken to resolve a query consisting of M sub-queries. We define the number of steps as the number of nodes to which the query is forwarded before being completely resolved. Define d as the look-ahead parameter. Let the neighborhood of a sensor consist of all sensors within d hops away from it. We model the size of a sensor's neighborhood (the number of nodes within d hops) as a function of d, $f(d)$, which is assumed to be independent of the particular node in question. We also assume that all possible queries Q are resolvable by this network (i.e. can be responded to by at least one node in the network).

Initially, let sensor x^* be the querier that issues a query Q consisting of M subqueries. Let d be the look-ahead parameter i.e each sensor can request information from sensors d hops away from it. In general when a sensor x gets a query it does the following:

1) *Local Update:* If its current information is not up-to-date, x sends a request to all sensors within d hops away. This request is forwarded hop by hop. The sensors who get the request will then forward their information to x. Let the energy consumed in this phase be E_{update}. Detailed analysis of E_{update} will be done in section IV-C.
2) *Forward:* After answering the query based on the information obtained, x then forwards the remaining query to a node that is chosen randomly from those d hops away.

Since the update is only triggered when the information is not fresh, it makes sense to try to quantify how often such updates will be triggered. We model this update frequency by an average *amortization factor* c, such that an update is likely to occur at any given node only once every c queries. In other words the cost of the update at each node is amortized over c queries, where $0 < c \leq 1$. For example, if on average an update has to be done once every 100 queries, $c = 0.01$. [2] After the query is completely resolved, the last node which has the query returns the completed response[3] to the querier x^* along

[2] It might be convenient to think of every datum having a time duration during which it is valid. During this period, all queries for the corresponding variable could be answered from the value cached from previous triggered updates. E.g. a sample bird call might have a longer validity period than a temperature reading.

[3] We note that it also makes sense to return partial responses back to the querier, as each sub-query is resolved along the way. This would reduce the energy and time costs of carrying partial responses along with the partial query. Our analysis thus overestimates the energy cost, and could be tightened further in this regard.

the reverse path [4]. We use α to denote the expected number of hops from the node where the query is completely resolved to x^*.

Let S_M be the average number of steps to answer a query of size M. Thus, the average energy consumed to answer a query of size M with look-ahead d can be expressed as follows:

$$E_{avg} = (cE_{update} + d)S_M + \alpha \quad (1)$$

Now, if $d = D$, where D is the diameter of the network, x^* can resolve the entire query in one step without forwarding it to any other node. However, in this case, E_{update} will be considerably large. On the other hand, if d is too small, a larger number of steps S_M will be required. In general, S_M reduces with increasing d, while E_{update} increases with increasing d. It is therefore possible, depending on other parameters, that the optimal energy expenditure is incurred at some intermediate value of d. One of the main objectives of our analysis is to *analyze the impact of parameters such as M, N, c, and d upon the energy consumption E_{avg} of ACQUIRE*.

B. Steps to Query Completion

Consider the following experiment. Each sensor tracks a value chosen between 1 and N with equal probability. Fetching information from each sensor can be thought of as a trial. Define a "success" as the event of resolving any one of the remaining queries. Thus, if there are currently M queries to be resolved, then the probability of success in each trial is $p = \frac{M}{N}$ and the probability of failure is $q = \frac{N-M}{N}$. Thus, the expected number of trials till the first success is $\frac{1}{p} = \frac{N}{M}$. Now the whole experiment can be repeated again with one less query, and the time to answer another query is $\frac{N}{M-1}$ and so on. Let σ_M be the number of trials till M successes i.e. the resolution of the entire query. It can be seen that

$$E(\sigma_M) = N \sum_{i=1}^{M} \frac{1}{M-i+1} = NH(M) \quad (2)$$

where $H(M)$ is the sum of the first M terms of the harmonic series. It is known that $H(M) \approx \ln(M) + \gamma$, where $\gamma = 0.57721$ is the Euler's constant. Thus

$$E(\sigma_M) \approx N(\ln M + \gamma) \quad (3)$$

Since we consider fetching information from $f(d)$ sensors as 1 step, then the number of steps to query completion S_M is given by [5]:

$$S_M = \frac{E(\sigma_M)}{f(d)} \approx \frac{N(\ln M + \gamma)}{f(d)} \quad (4)$$

[4] If additional unicast or geographic routing information is available, the completed response can also be sent back along the shortest path back from the final node to the querier.

[5] Here, we make an assumption that $f(d)$ new nodes will be encountered at every node where the query is forwarded. However, due to overlap, the number of new nodes actually encountered might be a fraction of f(d) i.e. $(1-\delta)f(d)$, where $\delta \in (0,1)$, is a measure of the average overlap of the neighborhoods of successive nodes handling the query. Note that δ should be low for ACQUIRE to perform well.

C. Local Update Cost

The energy spent in updating the information at each active node that is processing the active query E_{update} can be calculated as follows. Assume that the query Q is at the active node x. Given a look-ahead value d, x can request information from sensors within d hops away. This request will be forwarded by all sensors within d hops except those that are exactly d hops away from x. Thus the number of transmissions needed to forward this request is the number of nodes within $d-1$ hops, $f(d-1)$. The requested sensors will then forward their information to x. Now, the information of sensors 1 hop away will be transmitted once, 2 hops away will be transmitted twice,... d hops away will be transmitted d times. Thus,

$$E_{update} = (f(d-1) + \sum_{i=1}^{d} iN(i)) \quad (5)$$

where $N(i)$ is the number of nodes at hop i.

D. Total Energy Cost

We make the assumption that each active node forwards the resolved query to another node that is exactly d hops away, requiring d transmissions. Hence the average energy spent in answering a query of size M is given as

$$E_{avg} = (cE_{update} + d)S_M + \alpha \quad (6)$$

where α is the expected number of hops from the node where the query is completely resolved to the querier x^*. This is the cost of returning the completed response back to the querier node. This response can be returned along the reverse path in which case α can be at most dS_M. Thus,

$$E_{avg} = (cE_{update} + 2d)S_M \quad (7)$$

Special Case: $d = 0$ - Random Walk If the look-ahead $d = 0$, the node x will not request for updates from other nodes. x will try to resolve the query with the information it has, and will forward the query to a randomly chosen neighbor. Thus, in this case, ACQUIRE reduces to a random walk on the network. On an average it would take $E(\sigma_M)$ steps to resolve the query and $E(\sigma_M)$ steps to return the resolved query back to the querier x^*. Thus,

$$E_{avg} = 2E(\sigma_M) \quad (8)$$

E. Optimal Look-ahead

If we ignore boundary effects, it can be shown that $N(i) = 4i$ and $f(d) = (2d(d+1)) + 1$ for a grid of sensors (each node having 4 immediate neighbors). By combining the expressions in equations 4, 5, 7 and these expressions for $N(i)$ and $f(d)$, it can be shown that for such a grid of sensors:

$$E_{avg} \approx \{\frac{cN(\ln M + \gamma)}{3} \frac{4d^3 + 12d^2 - 4d + 3}{2d^2 + 2d + 1} + N(\ln M + \gamma) \frac{2d}{2d^2 + 2d + 1}\} \quad (9)$$

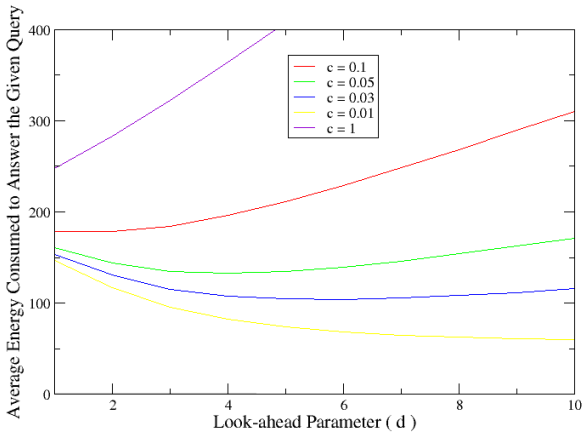

Fig. 2. Effect of c and d on the Average Energy Consumption of the ACQUIRE scheme. Here, $N = 100$ and $M = 20$

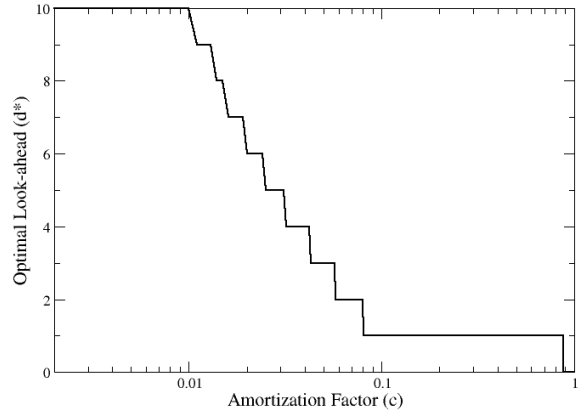

Fig. 3. Effect of c on d^* for $N = 100$ and $M = 20$. The x-axis is plotted on a log scale.

In order to determine the value of the look-ahead parameter which minimizes this energy cost, we need to take the derivative of this expression with respect to d and set it equal to zero. Performing this differentiation, we find that the optimal look-ahead d^* is the real solution to the following equation:

$$4cd^4 + 8cd^3 + 22cd^2 + 6cd - 5c - 6d^2 + 3 = 0 \qquad (10)$$

The expression shows that d^* varies only with the amortization factor c and not with the parameters M or N. In general the lower c is, the higher will be the look-ahead parameter d^*. The variation of E_{avg} for ACQUIRE with respect to d for different c and the impact of c on the optimal look-ahead parameter d^* are shown in figures 2 and 3 respectively.

We can now explain why ACQUIRE is well-suited for complex one-shot queries for replicated data. Other schemes such as flooding based querying (which we examine in greater detail next) are better suited for continuous queries because they incur lower delay and the initial cost of flooding the interest can be negligible compared to the total information flow between sources and the sink/querier. ACQUIRE is good at solving complex queries because, as our analysis shows, its energy costs scale logarithmically with the size of the query M. Finally, data replication is important to reduce the energy costs of ACQUIRE - the analysis here essentially assumes that the fraction $\frac{M}{N}$ of the nodes have the data being queried for. Hence the energy costs (which were shown to be linear in N) are inversely proportional to the degree of replication.

V. Comparison

We now analyze two other approaches, flooding-based querying (FBQ) and the expanding ring search (ERS), in order to compare them with ACQUIRE.

A. Flooding-based Querying (FBQ)

In FBQ, the querier x^* floods a request to all nodes in the network. All nodes with relevant variables respond.

Let $N_{avg}(i)$ be the expected number of nodes at hop i that can resolve some part of the query. This is equal to $N(i)\frac{M}{N}$, where $N(i)$ is the total number of nodes at hop i. Assume all nodes are within R hops of the querier, then it can shown that for FBQ,

$$\begin{aligned} E_{avg} &= (f(R) + \sum_{i=1}^{R} iN_{avg}(i))c \\ &= (f(R) + \frac{M}{N}\sum_{i=1}^{R} iN(i))c \end{aligned} \qquad (11)$$

For a grid with X nodes, $R \propto \sqrt{X}$, and from equation (11), it follows that for a given M, N and c, $E_{avg} \propto X^{3/2}$.

B. Expanding Ring Search (ERS)

In ERS, at stage 1, the querier x^* will request information from all sensors exactly one hop away. If the query is not completely resolved in the first stage, x^* will send a request to all sensors two hops away in the second stage. Thus, in general at stage i, x^* will request information from the $N(i)$ sensors exactly i hops away. The average number of stages t_{min} taken to completely resolve a query of size M can be approximately determined as follows (assuming a grid where $N(i) = 4i$):

$$\begin{aligned} \sum_{i=1}^{t_{min}} N(i) &= N(\ln M + \gamma) \\ \Rightarrow 2(t_{min})^2 + 2t_{min} - N(\ln M + \gamma) &= 0 \end{aligned} \qquad (12)$$

In ERS, at stage i, all nodes within $i-1$ hops of the querier x^* will forward the x^*'s request. Let $N_{avg}(i)$ be the expected

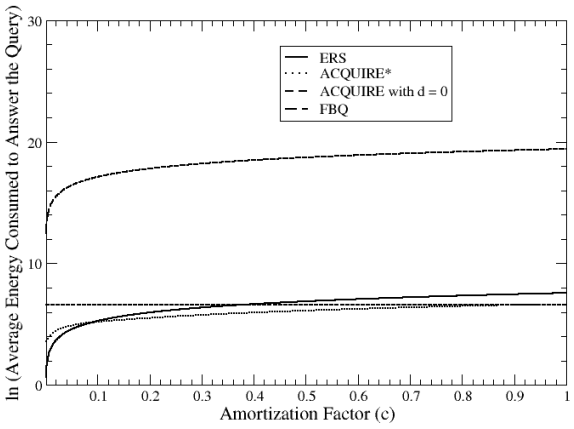

Fig. 4. Comparison of $ACQUIRE^*$, ERS and ACQUIRE with $d = 0$. Here, $N = 100$ and $M = 20$. The y-axis is $\ln(E_{avg})$

number of nodes at hop i that will resolve some sub-query. The response from these nodes will be forwarded over i hops. There are a total of t_{min} stages. Thus, the total energy cost is given as follows:

$$\begin{aligned} E_{avg} = cE_{update} &= (\sum_{i=1}^{t_{min}} (f(i-1) + iN_{avg}(i)))c \\ &= (\sum_{i=1}^{t_{min}} f(i-1) + \sum_{i=1}^{t_{min}} iN_{avg}(i))c \end{aligned} \quad (13)$$

It can be shown that for ERS, $N_{avg}(i) \approx N(i)(\frac{M - e^{\frac{f(i-1)}{N} - \gamma}}{N})$.

C. Comparison of ACQUIRE, FBQ and ERS

These schemes were compared across different values of c chosen in the range of $[0.001, 1]$. For ACQUIRE, the look-ahead parameter was set to d^* for a given value of c. We refer to this version of ACQUIRE as $ACQUIRE^*$. Equations (13), (11) and (9) (with $d = d^*$) were used in the comparative analysis. For the initial comparisons, $N = 100$ and $M = 20$. Using these values for M and N in equation (12), we obtain $t_{min} = 13$. This value of t_{min} is then used in equation (13).

As figure 4 shows that ACQUIRE with look-ahead 0 (i.e. random walk) performs at least as worse as ACQUIRE with the optimal look-ahead ($ACQUIRE^*$). $ACQUIRE^*$ outperforms ERS for higher values of the amortization factor (in this particular case, where $N = 100$ and $M = 20$, $ACQUIRE^*$ outperforms ERS if $c > 0.08$, $d^* \leq 1$). When $c = 1$, ACQUIRE gives more than 60% energy savings over ERS. It can be shown that this saving improves even more when N and M are larger.

As figure 4 shows, FBQ, on an average, incurs the worst energy consumption which is several orders of magnitude higher than the other schemes. This is mainly because of a very large number of nodes ($X = 10^6$) used in our calculations.

VI. Discussion and Conclusions

In this paper, we have proposed ACQUIRE - a novel mechanism for data extraction in energy-constrained sensor networks. The key features of ACQUIRE are the injection of active queries into the network with triggered local updates. We believe that ACQUIRE is likely to perform in an energy-efficient manner compared to other approaches on complex, one-shot, non-aggregate queries for replicated data.

We have developed a fairly sophisticated mathematical model that allows us to analytically evaluate and characterize the performance (in terms of energy costs) of ACQUIRE, as well as alternative techniques such as flooding-based queries (FBQ) and expanding ring search (ERS). As far as we are aware, there are very few similar results in the literature that provide similar mathematical characterizations of the performance of query techniques for sensor networks.

Partly for ease of analysis, we have described and modelled a very basic version of the ACQUIRE mechanism in this paper. While our analysis assumed a regular grid topology, these results can be easily extended for other topologies, so long as a reasonable model for $f(d)$ can be developed. Our major next step will be to convert ACQUIRE into a functional protocol that can be validated on an experimental sensor network testbed. There are a number of ways in which our analysis can be improved, and a number of additional design issues need to be considered in our future work, some of which we outline here.

- The efficiency of ACQUIRE can be improved if the neighborhoods of the successive active nodes in the query trajectory have minimal overlap. Making use of additional topological/geographical information to guide the trajectory would help reduce the overlap.
- Guided trajectories may also be helpful in dealing with non-uniform data distributions, ensuring that active queries spend most time in regions of the network where the relevant data are likely to be.
- We found that the optimal choice of the look-ahead parameter d^* is very much a function of the amortization factor c, and (somewhat surprisingly) independent of M, N, and the total number of nodes X. This lends itself to the possibility of using distributed algorithms in which localized estimates of c are used to determine the value of d at each step without global knowledge of system parameters.
- In this study we have only considered transmission costs in measuring the energy expenditure for the different querying mechanisms. In the future, we would like to enrich the analysis with energy metrics that incorporate reception costs, and undertake a study of the fundamental energy-latency tradeoffs involved in querying sensor networks.

In comparing ACQUIRE with other alternative strategies we found that ACQUIRE with optimal parameter settings outperforms all the other schemes for complex, one-shot queries, even

when the other schemes too are enhanced with cached updates. In particular, optimal ACQUIRE performs many orders of magnitude better than flooding-based schemes (such as Directed Diffusion) for such queries in large networks. We also observed that optimal ACQUIRE can reduce the energy consumption by more than 60% as compared to expanding ring search. The energy savings can be higher when $N \ln M$ is greater.

To conclude, we believe that there is no one-size-fits-all answer to the question: "How do we efficiently query sensor networks?" We propose ACQUIRE as a highly scalable technique that deserves to be incorporated into a portfolio of query mechanisms for use in real-world sensor networks.

REFERENCES

[1] J. Warrior, "Smart Sensor Networks of the Future," *Sensors Magazine*, March 1997.

[2] G.J. Pottie, W.J. Kaiser, "Wireless Integrated Network Sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 551-8, May 2000.

[3] A. Cerpa *et al.*, "Habitat monitoring: Application driver for wireless communications technology," *2001 ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, Costa Rica, April 2001.

[4] C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *ACM/IEEE International Conference on Mobile Computing and Networks (MobiCom 2000)*,August 2000, Boston, Massachusetts

[5] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of Network Density on Data Aggregation in Wireless Sensor Networks" , In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, Vienna, Austria. July, 2002.

[6] B. Krishnamachari, D. Estrin, and S. B. Wicker, ""The Impact of Data Aggregation in Wireless Sensor Networks," *International Workshop on Distributed Event-Based Systems, (DEBS '02)*, Vienna, Austria, July 2002.

[7] D. Estrin, R. Govindan, J. Heidemann and S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks," *ACM/IEEE International Conference on Mobile Computing and Networks (MobiCom '99)*, Seattle, Washington, August 1999.

[8] R. Govindan, J. Hellerstein, W. Hong, S. Madden, M. Franklin, S. Shenker, The Sensor Network as a Database, Technical Report 02-771, Computer Science Department, University of Southern California, September 2002.

[9] P. Bonnet, J. E. Gehrke, and P. Seshadri, "Querying the Physical World, " *IEEE Personal Communications*, Vol. 7, No. 5, October 2000.

[10] S. Garg, P. Pamu, N. Nahata, A. Helmy, "Contact Based Architecture for Resource Discovery (CARD) in Large Scale MANets", USC-TR, July 2002. (Submitted for Review).

[11] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, S. Shenker, "GHT – A Geographic Hash-Table for Data-Centric Storage," *First ACM International Workshop on Wireless Sensor Networks and their Applications*, 2002.

[12] M. Chu, H. Haussecker, F. Zhao, "Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks." Int'l J. High Performance Computing Applications, to appear, 2002. Also, Xerox Palo Alto Research Center Technical Report P2001-10113, May 2001.

[13] Zygmunt J. Haas, Marc R. Pearlman, and Prince Samar, "The Zone Routing Protocol (ZRP) for Ad Hoc Networks," *IETF MANET Internet Draft*, July 2002.

[14] Y. Yao and J. Gehrke, "The Cougar Approach to In-Network Query Processing in Sensor Networks," *SIGMOD*, 2002.

[15] Badri Nath and Dragos Niculescu, "Routing on a curve," *HotNets-I*, Princeton, NJ, October, 2002.

[16] David Braginsky and Deborah Estrin, "Rumor Routing Algorithm For Sensor Networks," *First Workshop on Sensor Networks and Applications (WSNA)*, September 2002.

[17] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In ICS'02, New York, USA, June 2002.