Comparative Analysis of Push-Pull Query Strategies for Wireless Sensor Networks

Shyam Kapadia[1], Bhaskar Krishnamachari[1,2]
[1]Department of Computer Science
[2]Department of Electrical Engineering
Viterbi School of Engineering
University of Southern California
Los Angeles, CA 90089, USA
{kapadia, bkrishna}@usc.edu

***Abstract*— We present a comparative mathematical analysis of two important distinct approaches to hybrid push-pull querying in wireless sensor networks: structured hash-based data-centric storage (DCS) and the unstructured comb-needle (CN) rendezvous mechanism. Our analysis, which is based on a single-sink square-grid deployment, yields several interesting insights. For ALL-type queries pertaining to information about all events corresponding to a given attribute, we examine the conditions under which the two approaches outperform each other in terms of the average query and event rates. For the case of ANY-type queries where it is sufficient to obtain information from any one of the desired events for a given attribute, we propose and analyze a modified sequential comb-needle technique (SCN) to compare with DCS. We find that DCS generally performs better than CN/SCN for high query rates and low event rates, while CN/SCN perform better for high event rates. Surprisingly, for the cases of ALL-type aggregated queries and ANY-type queries, we identify the existence of "magic number" event rate thresholds, independent of network size or query probability, which dictate the choice of querying protocol.**

I. Introduction

The primary function of a sensor network is to enable information gathering. The simplest strategy is to have all sensors provide a continuous stream of all the data that they gather to a sink node. However, for many classes of applications where only a small subset of the collected information is likely to be useful to end-users, the simple approach can become very inefficient. Researchers have therefore advocated the use of data-centric techniques which allow for efficient in-network storage and retrieval of named data using queries [1]. A number of data-centric querying and routing techniques have been proposed and examined in recent years: directed diffusion [2], TAG/TinyDB [3], rumor routing [4], hash-based data centric storage [5], hybrid push-pull [8], comb-needles [9], ACQUIRE [10], TTL-based expanding search [11], [12].

With the presence of an increasing number of choices of data-centric storage and querying techniques, it becomes of crucial importance to understand and quantify their performance (both in absolute terms and with respect to each other) with respect to key application, network, and environmental parameters. In particular, carefully developed mathematical models can provide deep practical design insights on protocol selection as well as protocol parameter optimization for different sensor network deployments.

There are several interesting prior studies on analytical modeling of query strategies [5], [10], [8], [7], [6], [9], [12]. The energy costs of data centric storage are compared with the two extremes of external storage and local storage in [5]. A hybrid push-pull query processing strategy is proposed and analyzed in [8]. Push and pull alternatives of directed diffusion are also analyzed in [7]. Shakkotai [6] has presented a comparison of the asymptotic performance of three random walk-based query strategies, showing that a push-pull rendezvous-based sticky search has the best success probability over time. The optimal parameter setting for the comb-needles approach is analyzed in [9]. The optimal replication level for queries disseminated using expanding ring searches is analyzed in [12]. A common thread through much of this literature on the analysis of query techniques is the argument that tunable hybrid push-pull strategies offer significant advantages. Our work builds on and complements these existing studies, as we aim to compare two distinct and important approaches to hybrid push-pull querying.

Following the nomenclature used to classify peer-to-peer networks, we can distinguish between two main categories of hybrid push-pull query strategies: structured and unstructured. The structured approach is exemplified by geographic hash table-based data centric storage technique [5]. The data from sources is placed at a location using the same hash that the sink uses to retrieve it. This significantly simplifies the query since the sink effectively "knows" exactly where to look for the stored information. The unstructured approach to push-pull querying is exemplified by the comb-needle approach [9]. In this approach the absence of a hash implies that the sink does not have prior knowledge of the location of the information. In that case, the queries are disseminated in the form of a comb with horizontal teeth, while the sources send event notifications independently in the form of limited vertical needles in either direction. The inter-teeth spacing and needle size are chosen and optimized to ensure that sources and sinks can rendezvous with each other efficiently. To the best of our knowledge, these two distinct and important approaches

This work has been supported in part by NSF grants numbered CNS-0435505 (NeTS NOSS), CNS-0347621 (CAREER), CCF-0430061, and CNS-0325875 (ITR).

to hybrid push-pull querying — the structured DCS and the unstructured comb-needle technique — have never been compared to each other. This is our objective in this paper.

We undertake a mathematical analysis comparing the expected total energy costs of both these approaches on a grid-based sensor deployment. Our modelling of these query strategies allows us to study the impact of several key parameters such as the size of the network, the event and query rates, the use of data aggregation (using summaries), as well as the type of queries. For a fair comparison, we carefully select optimized versions of each strategy. In particular, we allow the storage location to be chosen optimally for the hash based data-centric storage scheme, and we use optimized inter-tooth spacing for the comb-needle approach.

We consider two important types of one-shot queries in this paper. We refer to the first query type as an ALL-type query. These are global discovery-type queries, such as 'Give me the location of all the lions in the sensor deployed area?' or 'Return the locations that have temperature $\geq 60^\circ$ F'. In this case, the desired information must be obtained from all nodes in the network with relevant event information. The second type of query, which we refer to as an ANY-type query, is a one-shot query where any event that has the information can reply to the querier. Examples of such queries are 'Give me any location where a lion has been spotted in the sensor deployed area?' or 'Give me any location where the measured temperature is greater than 60° F'. For the ANY-type queries, we find that the entire network need not be necessarily covered by the combs in the comb-needle strategy. Based on this insight, we propose and analyze a modified sequential comb-needle querying scheme (see Section II-C).

Our analysis yields a number of useful insights into the relative performance of structured and unstructured approaches to hybrid push-pull querying. In all cases, we find that the unstructured comb-needle approach outperforms the data centric storage strategy when the number of events per epoch is large, while the reverse is true for small number of events, particularly for higher query rates. A particularly surprising and strong finding of our analysis is that under the assumptions of our modelling (large square grid network with a single caching-enabled querying sink located at bottom left) for the cases of aggregated ALL-type queries as well as the ANY-type queries, there exist "magic numbers" dictating which approach should be used for a given application scenario. In particular, for ALL-type queries, we find that if the expected number of events per epoch is greater than about 40 (regardless of the query rate or the size of the network), the comb-needle strategy always outperforms data centric storage. For ANY-type queries, when the number of events per epoch is less than about 1.5 (regardless of the query rate or network size), the data-centric storage approach always outperformas great than about 3.2 (again, regardless of the query rate or network size), the sequential comb-needle strategy always outperforms data-centric storage.

The rest of the paper is organized as follows. In section II, we present a brief overview of the algorithms to be analyzed in our paper: data centric storage (DCS), the basic comb-needle (CN) algorithm, and sequential comb-needle (SCN) algorithm.

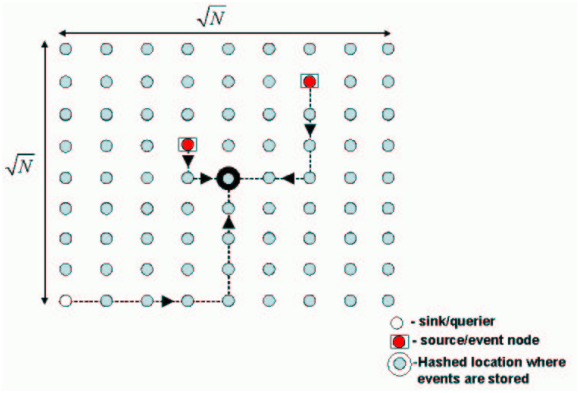


Fig. 1. Illustration of the data centric storage technique

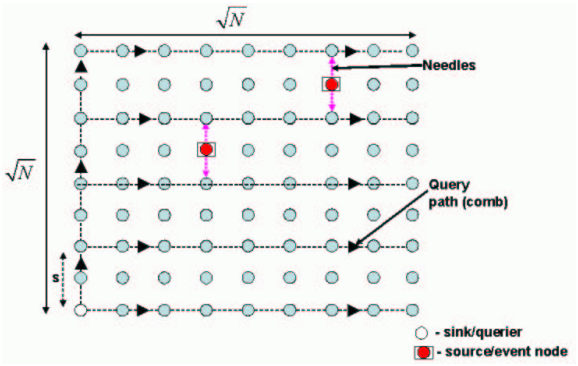


Fig. 2. Illustration of the comb-needles technique

We specify our modelling assumptions in section III. We derive and compare the costs of data centric storage and comb-needle strategies with and without summary aggregation for ALL-type queries in section IV. Then we analyze and compare data centric storage with the sequential comb-needle algorithm in section V. Finally, we discuss our key findings, along with directions for future work in the concluding section VI.

II. Overview of Algorithms

A. Data Centric Storage (DCS)

The data centric storage query dissemination strategy uses distributed hash tables to store the event data sensed by a particular node (see Figure 1). All the events of a particular event type (i.e. event having similar attributes) are hashed to the same node location. The data is then transported from the various event nodes along the shortest path to the node at the chosen location. Assuming the presence of location information, the authors propose to use GPSR to perform the routing. Queries for an event are then directed along the shortest path to these named location, since the query nodes also use the same hash function. The query responses are sent on the reverse path along which the query is forwarded.

B. Comb Needle (CN)

In this query dissemination strategy, the event nodes send out the sensed information vertically up and down like a spike (needle) of a certain length (see Figure 2). Let the length of the

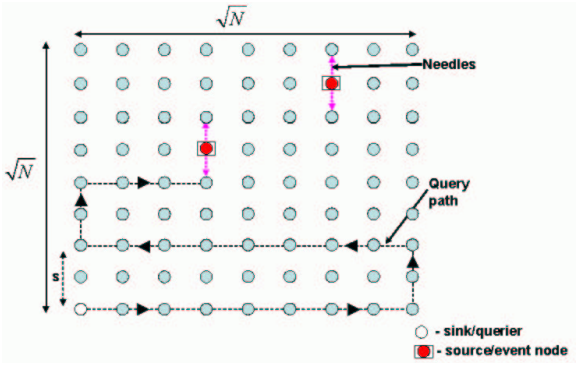


Fig. 3. Illustration of the sequential comb-needle technique

needle be denoted by s. The sink then sends out a query that traverses the network along a comb. The separation between the teeth of the comb is also s to ensure that at least one comb teeth hits each needle, so as to not miss out any event nodes. The information requested is then sent back to the sink along the shortest path. Note that this strategy is used when the average number of queries, Q, is less than or equal to the average number of events, E. When $Q > E$, a reverse comb-needle strategy is used where the query nodes form a needle and the event information is forwarded along a comb. Hence, the total cost for query dissemination in case of CN (C_{CN}) depends on the relationship between Q and E. However, in our case, since the sink is fixed and located at the left-bottom corner of the grid we do not use the reverse comb needle scheme.

C. Sequential Comb-Needle (SCN)

The motivation behind introducing this query scheme is to efficiently resolve the ANY type queries in which case the query terminates as soon as the query hits first event node of interest. In this way, the sequential comb needle scheme will always do better than the comb-needle scheme since it does not pay the extra cost incurred by the comb during query dissemination. In this scheme, similar to the comb-needle scheme, the event nodes form needles by spreading their information vertically to some nodes above and below them. The query originates from the sink and traverses the network as shown in Figure 3. Again, the size of the needles is denoted by s. Also, the distance between consecutive query horizontal traversals is s. The moment the query hits a node with the desired event information, the query path is truncated and the response is returned back to the sink.

III. Model and Assumptions

We first present our modelling assumptions:

- We consider a $\sqrt{N} \times \sqrt{N}$ regular grid comprising N nodes. Each node has 4 neighboring nodes adjacent to it. Hence, the distances between the nodes are evaluated as Manhattan distances.
- Queries only occur at the sink node located at the left-bottom corner of the grid. This represents the interface of the sensor network to the outside world.
- We consider a time period, T, defined as an epoch. This represents the period of time when the event information stored by the nodes will be valid.
- Our analysis aims at optimizing the total expected energy cost incurred during each epoch. We use the total number of required unicast transmissions as the indicator of energy costs.
- Without loss of generality, we focus the analysis on queries and events for a single generic event attribute (i.e. event type). Events corresponding to this attribute are assumed to occur uniformly across the sensor network.
- We denote by E the average number of events that occur during epoch T.
- We denote by Q the expected number of queries that occur within epoch T. Since the event information does not change over an epoch, Q is always between 0 and 1 and represents the probability that a query is issued during that epoch.
- We assume the presence of a suitable MAC layer to handle collisions and contention.

IV. Analysis of ALL-Type Queries

As mentioned earlier, ALL-type queries are of the type 'Give me all locations in the network where a lion was seen'. We first present the comparison of the data-centric storage and comb-needle scheme for such queries. We consider two cases: (a) When all the event information is sent to the sink (Without aggregation i.e. with no summaries). (b) When only an aggregated summary of the event information is sent to the sink (With aggregation i.e. using summaries).

A. Without Aggregation

1) Cost of DCS with optimized hash location: Below, we calculate the average cost incurred in case of the DCS strategy in terms of the number of hops needed for query resolution in the epoch T.

There are 3 different query costs involved, C_{st} to store events, C_{qd} the query dissemination cost and C_{qr} the cost for the query response. Hence, we have total cost in case of DCS, given by,

$$C_{DCS} = C_{st} + C_{qd} + C_{qr} \tag{1}$$

Since the position of the sink is fixed and known *a priori*, the DCS scheme can be optimized on the basis of the position of the hashed named location where all the event nodes send their data. Let $P_{opt}(x, y)$ denote the location of the node at that point. By symmetry, it is easy to see that P_{opt} will lie on the diagonal of the grid, otherwise, nodes on either side of the diagonal will have a larger distance to P_{opt}. Hence, they will pay more for transferring the event information to P_{opt} as compared to the other nodes. Hence, we have $x = y = p$ (say). Let $d_1(p)$ denote the distance from the sink to P_{opt}, and $d_2(p)$ the average distance between any node on the grid to the node located at point P_{opt}.

Note that without summaries, all the event information has to be sent out in the reply to the sink. Hence, we have,

$$C_{DCS} = \min_{p} (d_2(p) \cdot E + d_1(p) * Q + d_1(p) \cdot Q \cdot E) \quad \text{Without summaries} \tag{2}$$

We can determine the distance from the sink to P_{opt} trivially as

$$d_1(p) = x + y = 2 \cdot p \tag{3}$$

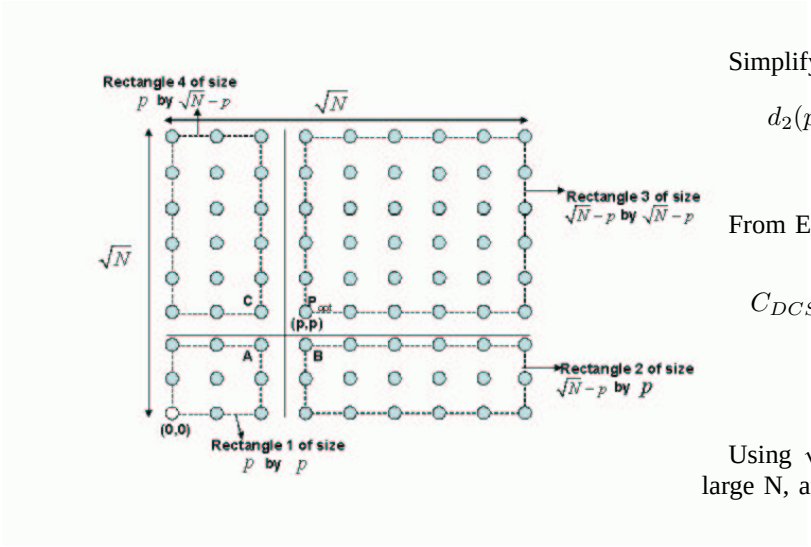


Fig. 4. Illustration of the four-rectangle decomposition for calculating $d_2(p)$ - the average distance between all nodes and a storage point located at $P_{opt}(p, p)$

The calculation of $d_2(p)$ is more involved. We can consider the grid as being divided into 4 rectangles as shown in Figure 4. The size of these rectangles is $p \times p$, $(\sqrt{N} - p) \times (\sqrt{N} - p)$, $p \times (\sqrt{N} - p)$ and $(\sqrt{N} - p) \times p$. The average distance from a node, located on a corner, to any node for a rectangle of size $X \times Y$ is given by Equation 39(see Appendix), $\overline{D_{rect}} = \frac{X \cdot Y \cdot (X+Y-2)}{2 \cdot (X \cdot Y - 1)}$.

For rectangle 1, the average distance between the node on its right-top corner and the other nodes is given by $\frac{p^2}{p+1}$. Note that there are $p^2 - 1$ nodes in this rectangle other than the node at the right-top corner. Hence, the total distance between any node and the node on the right-top corner is given by,

$$\frac{p^2}{p+1} \cdot (p^2 - 1) \tag{4}$$

Similarly, total distance for rectangle 3 is given by,

$$\frac{(\sqrt{N} - p)^2}{\sqrt{N} - p + 1} \cdot (\sqrt{N} - p)^2 - 1) \tag{5}$$

Since rectangles 2 and 4 are of the same size, we have their total distance given by,

$$2 \cdot \frac{p \cdot (\sqrt{N} - p)}{2} \cdot \frac{\sqrt{N} - 2}{p \cdot (\sqrt{N} - p) - 1} \cdot (p \cdot (\sqrt{N} - p) - 1) \tag{6}$$

Also, note that the distance from A to P_{opt} is 2, while the distance from B and C to P_{opt} is 1. Hence, we need to add an additional $2 \cdot p^2$ and $2 \cdot p \cdot (\sqrt{N} - p)$ to the numerator to account for the distances between all the points in rectangles 1, 2 and 4 to point P_{opt}.

From Equations 4, 5, and 6 we get,

$$\begin{aligned} d_2(p) &= \frac{1}{N-1} \cdot \Big[(\sqrt{N} - p)^2 \cdot (\sqrt{N} - p - 1) \\ &\quad + p \cdot (\sqrt{N} - p) \cdot (\sqrt{N} - 2) + p^2 \cdot (p - 1) \\ &\quad + 2 \cdot p^2 + 2 \cdot p \cdot (\sqrt{N} - p) \Big] \end{aligned} \tag{7}$$

Simplifying the above expression we get,

$$\begin{aligned} d_2(p) &= \frac{1}{N-1} \cdot \Big[N \cdot \sqrt{N} - N - 2 \cdot N \cdot p + \\ &\quad 2 \cdot \sqrt{N} \cdot p^2 + 2 \cdot \sqrt{N} \cdot p \Big] \end{aligned} \tag{8}$$

From Equation 2 we have,

$$\begin{aligned} C_{DCS} = \min_{p} \quad & (\tfrac{E}{N-1} \cdot \Big[N \cdot \sqrt{N} - N - 2 \cdot N \cdot p \\ & + 2 \cdot \sqrt{N} \cdot p^2 + 2 \cdot \sqrt{N} \cdot p \Big] \\ & + 2 \cdot Q \cdot (E + 1) \cdot p) \end{aligned} \tag{9}$$

Using $\sqrt{N} + 1 = \sqrt{N} - 1 \approx \sqrt{N}$ and $N - 1 \approx N$, for large N, and simplifying the above equation we get,

$$C_{DCS} = \min_{p} ((\sqrt{N} - 2 \cdot p + \frac{2}{\sqrt{N}} \cdot p^2) \cdot E + 2 \cdot Q \cdot (E + 1) \cdot p) \tag{10}$$

In order to determine the optimum value of p, we differentiate the above equation with respect to p and set it to 0. This yields the minimum value for C_{DCS} because the above expression is convex in p. Hence, we get the optimal value of p as,

$$\begin{aligned} p^* &= \frac{\sqrt{N} \cdot (\sqrt{N} - 1) \cdot E - Q \cdot (E + 1) \cdot (N - 1)}{2 \cdot E \cdot (\sqrt{N} + 1)} \\ &\approx \frac{\sqrt{N}}{2 \cdot E} \cdot (E - Q \cdot (E + 1)) \end{aligned} \tag{11}$$

In the above expression, if $p^* \leq 0$, this implies that the event nodes should send all their information directly to the sink. This resembles the external storage scheme. In that case, the expression for C_{DCS} reduces to $\sqrt{N} \cdot E$. Hence, the cost for query dissemination then goes to 0. Also, the condition for which $p^* > 0$ is given by,

$$Q < \frac{E}{E + 1} \tag{12}$$

Putting the optimal value of p^* obtained from Equation 11 into Equation 10, we get the total cost for the DCS scheme (without summaries) as,

$$C_{DCS} = \begin{cases} \sqrt{N} \left[Q \cdot (1 + E) - \frac{Q^2 \cdot (1+E)^2}{2 \cdot E} + \frac{E}{2} \right] & \text{if } Q < \frac{E}{E+1} \\ \sqrt{N} \cdot E & \text{Otherwise} \end{cases} \tag{13}$$

2) Cost of CN with optimized inter-tooth spacing: The derivation for the analysis of the comb-needle strategy is adapted from [9], however, here we use the exact expressions in case of the grid. First, we consider the case without summaries. As with DCS, there are 3 different costs involved, C_{needle} represents the needle costs for forwarding the event information to a subset of nodes, C_{comb} represents the query dissemination cost and C_{qr} represents the cost for the query response. Below, we present expressions for each of them.

$$C_{CN} = C_{needle} + C_{comb} + C_{qr} \tag{14}$$

Let s be the length of the needle formed by each node that senses an event. Then, the total needle cost is given by,

$$C_{needle} = s \cdot E \tag{15}$$

In the comb-needle strategy, the query is first sent out vertically upward from the sink and then fans out horizontally (see Figure 2). The distance between consecutive horizontal fan outs is also s, also known as the teeth separation for the comb.

$$\begin{aligned} C_{comb} &= (\sqrt{N} - 1 + (\sqrt{N} - 1) \cdot (\lceil \frac{\sqrt{N} - 1}{s} \rceil + 1)) \cdot Q \\ &\approx 2 \cdot \sqrt{N} \cdot Q + \frac{N \cdot Q}{s} \end{aligned} \tag{16}$$

Note that the ceil is present because there is a horizontal fan out at (0,0) and ($\sqrt{N}$ − 1,0). Assuming, that each node where the comb tooth intersects with the needle, replies along the shortest path to the sink (see Appendix VIII), we have the total query response cost given by,

$$\begin{aligned} C_{qr} &= \frac{N}{\sqrt{N} + 1} \cdot E \cdot Q \\ &\approx \sqrt{N} \cdot E \cdot Q \end{aligned} \tag{17}$$

Hence, the total cost for the comb needle strategy is given by,

$$C_{CN} = s \cdot E + 2 \cdot \sqrt{N} \cdot Q + \frac{N \cdot Q}{s} + \sqrt{N} \cdot E \cdot Q \tag{18}$$

Now we find the value of s that minimizes this total query cost. On solving we get, $s^* = \sqrt{N} \cdot \sqrt{\frac{Q}{E}}$

Hence, the total cost with the comb needle scheme without summaries is given by,

$$C_{CN} = \sqrt{N} \cdot (2 \cdot Q + 2 \cdot \sqrt{Q \cdot E} + \cdot E \cdot Q) \tag{19}$$

3) Comparison of DCS and CN: Figure 5 (a) and (b) compares the normalized expected cost of querying (which is calculated as the total expected cost divided by the square-root of the number of nodes) with the DCS and CN strategies with respect to the two key parameters E and Q. We observe that CN outperforms DCS as the average number of events per epoch increases, while DCS outperforms CN when the per-epoch query probability increases.

Figure 6 shows the regions in the E-Q plane where DCS and CN outperform each other. This is generated by obtaining the zero-contour of the surface representing the difference in cost between DCS and SCN as a function of E and Q. We note that the equal-cost curve grows slowly with respect to E[1]. In particular, there is no threshold event rate beyond which CN is always better regardless of the query rate — we shall see later that this is not always the case.

B. With Aggregation

1) Cost of DCS with optimized hash location: With summaries, all the event information can be compressed into a single packet and sent out to the sink, hence, we have,

$$\begin{aligned} C_{DCS} &= \min_p (d_2(p) \cdot E + d_1(p) \cdot Q + d_1(p) \cdot Q) \\ &\quad \text{With summaries} \end{aligned} \tag{20}$$

Using a similar procedure to that used above for the case without summaries, since only the reply cost is different and everything else is the same, we obtain the total cost for DCS with summaries as,

$$C_{DCS} = \begin{cases} \sqrt{N}\left[2 \cdot Q - \frac{2 \cdot Q^2}{E} + \frac{E}{2}\right] & \text{if } Q < \frac{E}{2} \\ \sqrt{N} \cdot E & \text{Otherwise} \end{cases} \tag{21}$$

2) Cost of CN with optimized inter-tooth spacing: We now describe the CN cost with summaries. The only change that occurs in the cost of the reply path. The query dissemination cost, C_{qd} and the needle cost C_{needle} remain the same as was the case without summaries. For the reply cost, we note that reply from the various events can be aggregated on the way back to the sink. To account for this aggregation we approximate the reply cost to be the same as the cost for the comb i.e. the cost for query dissemination. This is because the events need only send their data horizontally toward the sink, the vertical path downward toward the sink will account for the aggregation. Hence, now we have the total cost for CN given by,

$$C_{CN} = C_{needle} + C_{comb} + C_{qr} = C_{needle} + 2 \cdot C_{comb} \tag{22}$$

$$\begin{aligned} C_{CN} &= s \cdot E + 4 \cdot Q \cdot (\sqrt{N} - 1) \\ &\quad + 2 \cdot (\sqrt{N} - 1) \cdot (\lceil \frac{\sqrt{N} - 1}{s} \rceil) \cdot Q \\ &\approx s \cdot E + 4 \cdot Q \cdot \sqrt{N} + 2 \cdot Ns \cdot Q \end{aligned} \tag{23}$$

Again, we solve for the optimum s to get, $s^* = \sqrt{N} \cdot \sqrt{\frac{2 \cdot Q}{E}}$

Hence, the total cost with the comb needle scheme with summaries is given by,

$$C_{CN} = 2 \cdot \sqrt{2 \cdot N \cdot Q \cdot E} + 4 \cdot \sqrt{N} \cdot Q \tag{24}$$

[1]This can be shown rigorously in terms of the derivative of that curve, but we do not present that analysis here due to lack of space

3) Comparison of DCS and CN: Figures 7(a) and (b) compare the normalized expected cost of storage and querying with the DCS and CN strategies with respect to the two key parameters E and Q. We observe that even with summaries CN outperforms DCS as the average number of events per epoch increases, while DCS outperforms CN when the per-epoch query probability increases.

Figure 8 shows the regions in the E-Q plane where DCS and CN outperform each other. We can see that (unlike in the case without summaries) there exists an threshold Θ for the event rate beyond which CN is always better. This threshold can be derived analytically.

First, we can prove that when $Q \geq E/2$, $C_{DCS} = \sqrt{N}E$ is always smaller than C_{CN}, hence there is no solution for $C_{DCS} - C_{CN} = 0$ in this case. When $Q < E/2$, then we can write the expression for the equal-costs curve as follows:

$$\sqrt{N}\left[(2Q - \frac{2Q^2}{E} + \frac{E}{2}) - (2\sqrt{2QE} + 4Q)\right] = 0 \qquad (25)$$

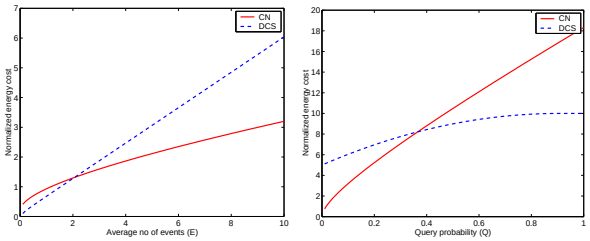
As can be seen from the figure, the threshold event rate corresponds to the point when there is a query at every epoch. Setting $Q = 1$, and solving the above expression for E, we find that the threshold $\Theta \approx 39.78$. An important point to note is that this threshold is a "magic number" that is independent of the size of the network. It tells us a surprising design lesson: for a grid-based network where ALL-type queries are always injected from the bottom left corner, if there are more than 40 events on average in each epoch that must be aggregated in response to queries, then a comb-needle approach is preferable in terms of total energy cost to a hash-based data-centric storage approach.

V. Analysis of ANY-Type Queries

Recall that in case of ANY-type queries, the query need not visit every node in the network, it should be terminated as soon as it hits a node that has the desired information. Here, for such query types, we obtain the expressions for the data-centric storage scheme and the modified comb and needle scheme which we call the sequential comb-needle (SCN) scheme.

4) Cost of DCS: The cost for ANY-type queries remains the same as that obtained for ALL-type queries with summaries. This is because the data centric storage scheme stores all the information about a given event type at a named location. Hence, the reply to the ANY-type query can be considered similar to just returning the summary. Hence, the cost in case of DCS can be obtained from Equation 21.

5) Cost of SCN: We now derive the cost for the sequential comb-needles (SCN) approach. To determine the cost for the query transmission we need to obtain the average number of hops/transmissions till a node with the desired event information is hit. Since each event node replicates the data to s other nodes, and the separation between successive horizontal traversals along the query path is also s, the original grid with N nodes can be transformed to a new grid with $\frac{N}{s}$ nodes. The sequential comb-needle scheme then traverses this new grid as a chain of $\frac{N}{s}$ nodes. Denote $n = \frac{N}{s}$. Let X be a random variable that determines the number of hops till a event node



(a) (b)

Fig. 5. Cost of CN and DCS for ALL-type queries, without summary aggregation, with respect to E (for Q = 0.1) and Q (for E = 10)

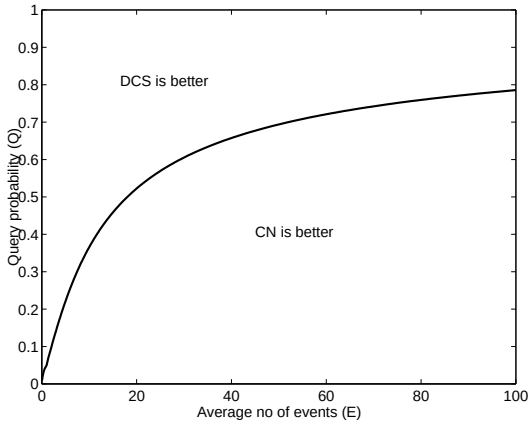
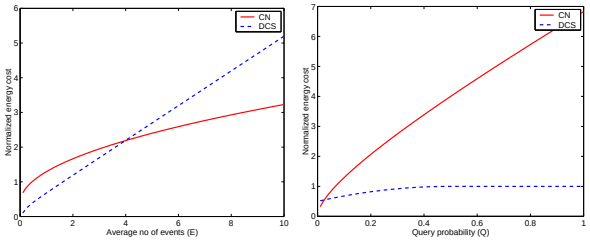


Fig. 6. Relative Performance of CN and DCS for ALL-type queries, without summary aggregation, with respect to event and query rates



(a) (b)

Fig. 7. Cost of CN and DCS for ALL-type queries, with summary aggregation, with respect to E (for Q = 0.1), and with respect to Q (for E = 1)

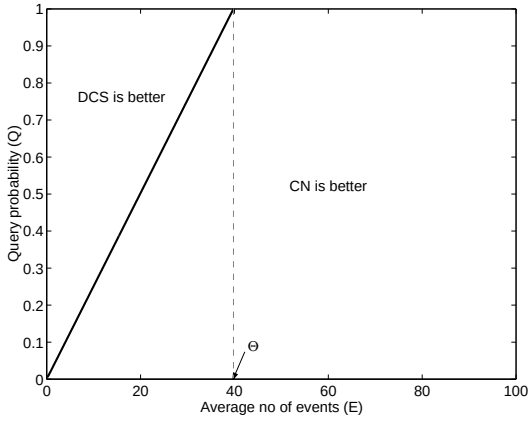


Fig. 8. Relative Performance of CN and DCS for ALL-type queries, with summary aggregation, with respect to event and query rates

is hit by the query. Note that, even in this compressed chain, $\frac{E \cdot s}{s} = E$ is the number of event nodes. Now, we have the cdf of X given by,

$$cdf(X) = P(X \leq k) = 1 - (\frac{n-k}{n})^E \tag{26}$$

We can now obtain the pmf of X as,

$$\begin{aligned} pmf(X) &= P(X \leq k) - P(X \leq k-1) \\ &= (1 - \frac{k-1}{n})^E - (1 - \frac{k}{n})^E \end{aligned} \tag{27}$$

Now the expected value of X can be obtained by using Equation 27 as follows,

$$Exp(X) = \sum_{k=1}^{n-1} k \cdot ((1 - \frac{k-1}{n})^E - (1 - \frac{k}{n})^E) \tag{28}$$

Let $f(k) = (1 - \frac{k}{n})^E$. Then we get,

$$Exp(X) = \sum_{k=1}^{n-1} k \cdot (f(k-1) - f(k)) \tag{29}$$

Note that this summation can be opened up, so the consecutive terms can be grouped together to leave,

$$Exp(X) = \sum_{k=1}^{n-1} f(k) - n \cdot f(n) = \sum_{k=1}^{n-1} (1 - \frac{k}{n})^E \tag{30}$$

Note that $f(n) = 0$, hence, in the above expression by substituting $j = n - k$, we get,

$$Exp(X) = \sum_{j=1}^{n-1} (\frac{j}{n})^E = \frac{1}{n^E} \sum_{j=1}^{n-1} j^E \tag{31}$$

Approximating the summation by an integration, we get,

$$Exp(X) \approx \frac{1}{n^E} \cdot \frac{n^{E+1}}{E+1} = \frac{n}{E+1} = \frac{\frac{N}{s}}{E+1} \tag{32}$$

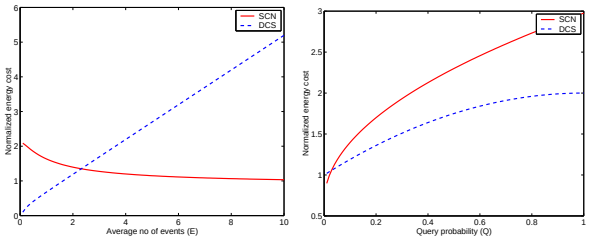
Note that Exp(X) just accounts for the number of horizontal steps taken by the SCN query path. We also need to account for the vertical steps that it takes. This can be approximated by determining the y-coordinate of the point where SCN hits the first event node. This is given by,

$$\overline{Y} = \frac{\frac{N}{s \cdot (E+1)}}{\sqrt{N}} \cdot s = \frac{\sqrt{N}}{E+1} \tag{33}$$

For simplicity, we assume that the query response path is the same as that taken by the query. Now, we can get the total cost in case of SCN as,

$$C_{SCN} = C_{needle} + C_{qd} + C_{qr} \tag{34}$$

$$C_{SCN} = s.E + \frac{\frac{N}{s}}{E+1} \cdot Q + \frac{\frac{N}{s}}{E+1} \cdot Q + 2 \cdot \frac{\sqrt{N}}{E+1} \tag{35}$$



(a) (b)

Fig. 9. Cost of SCN and DCS for ANY-type queries, with summary aggregation, with respect to E (for Q = 0.1), and with respect to Q (for E = 2)

Solving for the value of s that minimizes C_{SCN} we get $s^* = \sqrt{\frac{2 \cdot N \cdot Q}{E \cdot (E+1)}}$. Using this value, we get the total cost in case of the sequential comb-needle strategy as,

$$C_{SCN} = 2 \cdot \sqrt{\frac{2 \cdot N \cdot Q \cdot E}{E+1}} + 2 \cdot \frac{\sqrt{N}}{E+1} \tag{36}$$

6) Comparison of DCS and SCN: Figures 9 (a) and (b) compare the normalized expected cost of storage and querying with the DCS and SCN strategies with respect to the two key parameters E and Q. We observe that SCN outperforms DCS as the average number of events per epoch increases, while DCS outperforms SCN when the per-epoch query probability increases.

Figure 10 shows the regions in the E-Q where DCS and SCN outperform each other. We can see that in this case, there are two significant thresholds for the event rate. Below a lower threshold Θ_{lower}, we find that DCS is always better (regardless of the query probability), and above an upper threshold Θ_{upper}, SCN is always better (regardless of the query probability). These "magic numbers" can be derived analytically.

First, similar to the analysis of the DCS and CN strategies with aggregated responses for the ALL-type queries, we can prove that when $Q \geq E/2$, $C_{DCS} = \sqrt{N}E$ is always smaller than C_{SCN}. When $Q < E/2$, then we can write the expression for the equal-costs curve as follows:

$$\sqrt{N} \left[(2Q - \frac{2Q^2}{E} + \frac{E}{2}) - (2\sqrt{\frac{2QE}{E+1}} + \frac{2}{E+1}) \right] = 0 \tag{37}$$

As can be seen from the figure, the threshold event rate corresponds to the point when there is a query every epoch. Setting $Q = 0$, and solving the above expression for E, we get the lower threshold $\Theta_{lower} \approx 1.56$. And setting $Q = 1$, and solving the above expression for E, we find that the threshold $\Theta_{upper} \approx 3.16$.

VI. Conclusions and Future Research Directions

We have presented a comparative analysis of two distinct and important approaches to hybrid push-pull querying in wireless sensor networks - the structured hash-based DCS,

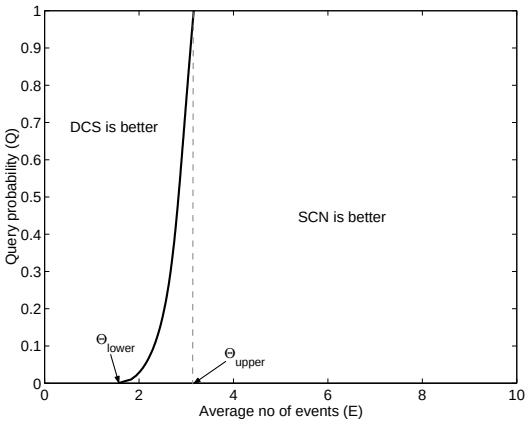


Fig. 10. Relative Performance of SCN and DCS for ANY-type queries with respect to event and query rates

and the unstructured CN/SCN. We have examined their performance with respect to key environment, network, and application parameters including the event and query rates, network size, type of query, and the use in-network aggregation.

We have found that the costs of DCS, CN, and SCN are all directly proportional to the square-root of the number of nodes in the network. Therefore the relative performance of DCS versus CN/SCN is unaffected by network size. The exact shape of the relative best performance regions for the two approaches do change depending on the query type (ALL, ANY) and the use/non-use of summary aggregation; however, we find in all cases that the unstructured CN/SCN approach generally outperforms the DCS strategy when the number of events per epoch is large, while the reverse is true for small number of events, particularly for higher query rates. A possible explanation for this is that, relatively speaking, the query cost burden is reduced in structured strategies like DCS when compared with an unstructured strategy like CN/SCN because the use of hashing provides a predetermined location to pick up information about all events. But this comes at the expense of a higher cost burden in event notification since all events must be transmitted to a generally non-local hash location. Thus a hash-based push-pull scheme like DCS favors high query rates but low event rates, compared to an path-intersection based push-pull scheme like CN/SCN.

Our analysis reveals the existence of event rate thresholds for aggregate ALL-type queries ($\Theta \approx 39.78$) as well for as ANY-type queries ($\Theta_{lower} \approx 1.56, \Theta_{upper} \approx 3.16$), that dictate which protocol should be used in a given application scenario regardless of the query probability. It is remarkable that these thresholds are also independent of the network size.

Besides offering some concrete guidelines for practitioners, this study suggests a number of interesting directions for future work. These include extensions of the analysis taking into account different deployment topologies, different cost metrics (including other energy models, as well as delay), and allowing multiple querying sinks. The theoretical results we present should also be validated through experiments on a real application/test-bed.

VII. Acknowledgements

We'd like to thank the members of the USC Autonomous Networks Research Group for their feedback on this paper. A special thanks to Joon Ahn, Sundeep Pattem, and Kiran Yedavalli for their technical input.

References

[1] R. Govindan, "Data-Centric Storage in Sensor Networks, in Wireless Sensor Networks", (T. Znati, K. Sivalingam, C. S. Raghavendra Ed.), Kluwer Publishers, 2003.

[2] C. Intanagonwiwat, R. Govindan, and D. Estrin, Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks, In Proceedings of the Sixth Annual International Conference on Mobile Computing and Networks (MobiCOM), August 2000.

[3] S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong, TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks, 5th Symposium on Operating System Design and Implementation (OSDI 2002), December 2002

[4] D. Braginsky and D. Estrin, "Rumor Routing Algorithm For Sensor Networks", The First Workshop on Sensor Networks and Applications (WSNA'02), October 2002.

[5] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin, "Data-Centric Storage in Sensornets", ACM SIGCOMM, Computer Communications Review, Vol. 33, Num. 1, January 2003.

[6] S. Shakkottai, Asymptotics of Query Strategies over a Sensor Network, INFOCOM'04, March 2004

[7] B. Krishnamachari and J. Heidemann, "Application-Specific Modelling of Information Routing in Wireless Sensor Networks," Workshop on Multihop Wireless Networks (MWN'04) held in conjunction with the IEEE International Performance Computing and Communications Conference (IPCCC), April 2004.

[8] N. Trigoni, Y. Yao, A. Demers, J. Gehrke, and R. Rajaraman. "Hybrid Push-Pull Query Processing for Sensor Networks", In Proceedings of the Workshop on Sensor Networks as part of the GI-Conference Informatik 2004. Berlin, Germany, September 2004.

[9] X. Liu, Q. Huang, Y. Zhang, "Combs, Needles, Haystacks: Balancing Push and Pull for Discovery in Large-Scale Sensor Networks", ACM Sensys, November 2004

[10] N. Sadagopan, B. Krishnamachari, and A. Helmy, "Active Query Forwarding in Sensor Networks (ACQUIRE)", *Ad Hoc Networks Journal-Elsevier Science*, Vol. 3, No. 1, pp. 91-113, January 2005.

[11] N. Chang and M. Liu, "Revisiting the TTL-based Controlled Flooding Search: Optimality and Randomization", Proceedings of the Tenth Annual International Conference on Mobile Computing and Networks (ACM MobiCom), September, 2004, Philadelphia, PA.

[12] B. Krishnamachari and J. Ahn, "Optimizing Data Replication for Expanding Ring-based Queries in Wireless Sensor Networks," *USC Computer Engineering Technical Report CENG-05-14*, October 2005.

VIII. Appendix

Average distance between a node located at the bottom-left corner and any other node within a $X \times Y$ rectangular grid

This can be expressed by the following summation:

$$\overline{D_{rect}} = \frac{\sum_{i=0}^{X-1}\sum_{j=0}^{Y-1}(i+j)}{X \cdot Y - 1} \tag{38}$$

Evaluating the above expression, we get

$$\overline{D_{rect}} = \frac{X \cdot Y \cdot (X+Y-2)}{2 \cdot (X \cdot Y - 1)} \tag{39}$$

Note that from this by setting $X = Y = \sqrt{N}$, we have the distance from the node at one corner to any point in the $\sqrt{N}$ by $\sqrt{N}$ square grid:

$$\overline{D_{square}} = \frac{N}{\sqrt{N}+1} \tag{40}$$