Efficient and Robust Query Processing in Dynamic Environments Using Random Walk Techniques

Chen Avin *
Computer Science Department
University of California,
Los Angeles
avin@cs.ucla.edu

Carlos Brito †
Computer Science Department
University of California,
Los Angeles
fisch@cs.ucla.edu

ABSTRACT

Many existing systems for sensor networks rely on state information stored in the nodes for proper operation (e.g., pointers to parent in a spanning tree, routing information, etc). In dynamic environments, such systems must adopt failure recovery mechanisms, which significantly increase the complexity and impact the overall performance. In this paper, we investigate alternative schemes for query processing based on random walk techniques. The robustness of this approach under dynamics follows from the simplicity of the process, which only requires the connectivity of the neighborhood to keep moving. In addition we show that visiting a constant fraction of sensor network, say 80%, using a random walk is efficient in number of messages and sufficient for answering many interesting queries with high quality. Finally, the natural behavior of a random walk, also provide the important properties of load-balancing and scalability.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Distributed Systems; G.3 [**Probability and Statistic**]: Markov Processes, Probabilistic Algorithms

General Terms

Algorithms, Design, Performance, Reliability

Keywords

Sensor Networks, Random Walk, Query Processing

*This author was partially supported by AFOSR grant #F49620-01-1-0055, NSF grant #IIS-0091082, and ONR (MURI) grant #N00014-00-1-0617

†This author was partially supported by a CNPq PhD fellowship 200201/99-9.

IPSN'04, April 26–27, 2004, Berkeley, California, USA.

1. INTRODUCTION

Wireless sensor networks represent an emerging new type of networks and computing platform. They are constructed from small devices equipped with sensing capabilities, wireless communication and limited power supply, CPU and memory. These devices are expected to be embedded into the environment to create very dense networks. High level tasks, such as monitoring specific events, are accomplished by the cooperation of multiple nodes using their limited ability to collect and process information.

The execution of a task usually starts from base stations, which issue queries into the network. The sensor nodes are viewed as data sources that can provide relevant information for the querying agents. A distinctive characteristic of this framework is the focus on the data to be collected, as opposed to the particular entities that generate it.

Sensor networks are supposed to operate unattended and under strict energy constraints [14]. The natural problems that arise from such conditions are overcome by relying on a large amount of redundancy. This presents a very different scenario than the one encountered in traditional computing systems. On top of that, many such networks will also be subject to high dynamics, created by failures, mobility, temporary communication obstacles, nodes switching on and off [5], etc. Such adverse conditions make the design of energy efficient, robust and scalable systems a considerable challenge.

The extensive research in this field, though, allowed to learn a few principles for the design of efficient sensor networks. For example, solutions based on transferring the data generated by the sensors for centralized processing are not acceptable [17]. This requires an excessive amount of communication, which is a critical component in the energy consumption of the devices. Taking advantage of the fact that computation is much less energy expensive than communication [14], most existing designs consist of distributed systems, which delegate a large portion of the information processing to the sensor nodes themselves.

A common characteristic of existing sensor network systems is their dependence on state information stored in the nodes for proper operation. Examples range from pointers to parents in spanning tree structures [11] to routing information stored in the intermediate nodes of a path [8]. In related areas such as ad-hoc networks and distributed applications, this technique is used to obtain tight control over the process and achieve high levels of performance [3]. In

sensor networks, however, optimality is not a fundamental issue. More importantly, in a highly dynamic environment these ideas require the development of sophisticated failure recovery mechanisms, which significantly increase the complexity of the system, and have impact on the overall performance.

1.1 Overview of Our Approach

In the search for an alternative solution, we investigate the use of random walk techniques for the task of query processing in dynamic environments. The use of randomization in the context of sensor networks is not new and has been explored, for instance, in [4, 16, 15]. However, while those systems make a great effort to avoid some undesirable effects of the random steps, like visiting a node more than once, or moving in the "wrong" direction, we accept the natural behavior of the random walk and try to explore its good properties.

The first important advantage of our approach is related to critical points of failure. State based approaches usually introduce nodes whose failure has a great impact on the operation of the system. Examples include nodes close to the root in a spanning tree, and cluster-heads in cluster based designs. A random walk, on the other hand, only needs a connected neighbor to keep moving. There are no critical points, all the nodes are equally unimportant at all times.

To study the efficiency of the approach, we first observe that, due to the redundancy in the network, it is not necessary to consult every node to answer many interesting queries. This motivates the definition of Partial Cover Time (PCT) as the expected number of steps required by a random walk to visit a constant fraction of the nodes, for example 50%, 80% or 99%. This definition generalizes the well studied concept of Cover Time [1, 6, 18, 9], that is, the time required to visit every node in the network.

Our main analytical result provides an upper bound on the Partial Cover Time, which is asymptotically smaller than Matthews' bound on the Cover Time [12] (for any constant fraction c of the network). Intuitively, this means that, on sufficiently large graphs, almost all the time used by a walk to cover the entire graph is spent trying to reach the last $\log(n)$ nodes.

Clearly, results of this nature may have little consequences for practical applications, because the constants hidden in the asymptotic notation can be very large. Then, to get a better understanding of the efficiency of our approach in practice, we conduct an extensive, simulation study. Our results show that the approach is very competitive, performing better than a cluster based design on 2-dimensional grids (under the measure of number of messages per query).

To investigate the issue of dynamics in our approach, we consider two failure models. The first model assigns a uniform probability of failure p to all the nodes, and captures the effect of defective devices and the process of switching on and off in duty cycle. The second model specifies regions of high probability of failure, and capture the effect of disasters and areas under hostile conditions. As we should expect, the approach perform well as long as the network is reasonably connected and there are no bottlenecks. Indeed we only observe considerable degradation of performance under extreme situations.

Besides efficiency and robustness, we also study the quality of the answers that can be obtained with a partial cover of the network. For this purpose, we first investigate how well distributed are the nodes visited by a random walk that covers 80% of a typical sensor network. The results show that almost every unvisited node is with in distance 1 or 2 (in number of hops) from a visited node. Also, since geographically close nodes are likely to observe the same phenomena, this indicates that the answer should be very accurate. Actually, this is confirmed in another set of experiments, where we assume a non-uniform distribution of data values for the sensors, and obtain an approximation of this distribution from a partial cover of the network. We observe that the histogram computed by the random walk is very close to the actual distribution.

Finally, we present simulation results showing that the approach achieves very good load balancing among the nodes, and make some comments on how to attenuate the problem of latency, which is inherent to our approach.

1.2 Paper Organization

The rest of the paper is organized as follows. Section 2 review related work. In section 3 we discuss random walks, cover times, and prove the "Partial Cover Lemma" that is the basis for this paper. The following four sections focus on the properties of Partial Cover studied through simulations. In Section 4 we discuss efficiency and compare our result to the cluster head scheme. Section 5 talks about the robustness of our approach and shows the problem with spanning tree based systems. Section 6 explores the quality of the random walk and gives an application example. Section 7 discusses load balancing, scalability and latency. We conclude in section 8.

2. RELATED WORK

2.1 Query Answering Systems

Madden *et al.* [11] present a tiny aggregation (TAG) for sensor networks and describe a SQL-like query language for answering queries. Their basic approach consists of building a spanning tree using a flooding mechanism, and then uses this structure to answer queries in an efficient way. They show how to apply aggregation ideas on different types of queries, and propose methods to deal with dynamic environments. Heinzelman *et al.* [7] analyze the use of a cluster-based system for answering queries. They develop a randomized scheme for choosing cluster heads, and explore compression (aggregation) techniques, in order to achieve load balancing and reduce energy consumption. Chalermek *et al.* [8] provide a data-centric mechanism called "directed diffusion" to answer long term queries. Interest requests (queries) are flooded into the network leaving gradient paths back to the base station. When a node has data that matches the interest request, the gradient information stored in the intermediate nodes are used to establish a flow from the node to the base station. These routes are reinforced later to overcome failures and aggregation is performed along the path to increase efficiency.

2.2 Randomize Systems

Servetto and Barrenechea [16] propose the use of constrained random walks to perform routing in dynamic networks. They consider an $n \times n$ grid topology, where the nodes have precise knowledge about their location. This knowledge is used to compute transition probabilities at each

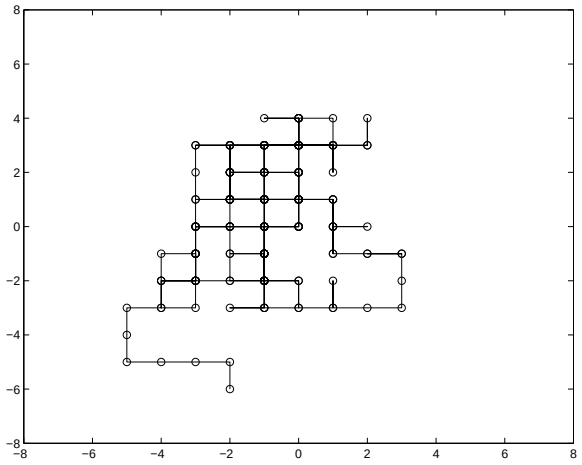

Figure 1: An example of short random walk on a grid starting at $(0, 0)$

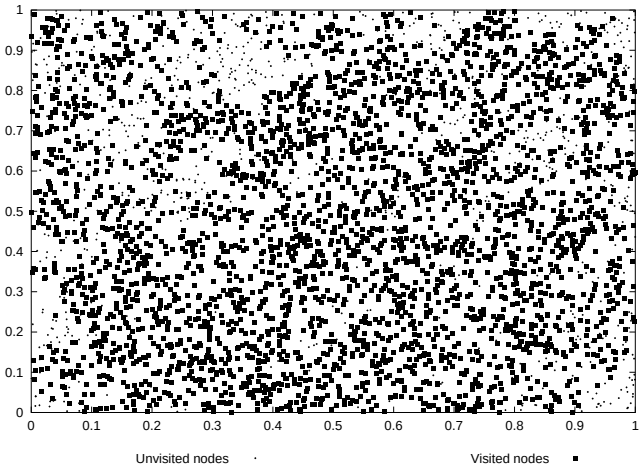

Figure 2: An example of random walk that covers 80% **of a random network with 4096 nodes**

node, in order to route the messages to their destination. In the situation analyzed in this paper (routing a message from location $(0, 0)$ to (n, n)), the message always follows a shortest path, and perfect load balancing among the intermediate nodes is achieved. The approach of Braginsky and Estrin [4] is closer to ours in the sense that they also employ random walks to process queries. Their solution, however, only relies on random walks to find a route between two points. The basic idea is that both the base station and a node that observes some event of interest start random walks that leave traces on the nodes that they visit. Whenever the traces intersect, a route is discovered and the information can be transferred between the two points. The most similar solution to ours is provided in AQUIRE [15]. this system uses random walks to answer one-shot, non-aggregate, complex, replicate data queries. AQUIRE adopts a look ahead mechanism in which at each step, information is collected from every node at distance at most d hops away from the current one, and then the walk jumps to a node at distance d. The goal of such a mechanism is to reduce the impact of the random steps. Their analysis, however, only investigates how long it takes for a random walk to reach a node with specific information, assuming a uniform distribution of data values over the nodes in the network. Here, we present a more complete and general study of the use of simple random walks to the task of query processing.

3. RANDOM WALKS AND COVER TIME

A Random Walk is the simple process of visiting the nodes of a graph G in some sequential random order. Basically, the walk starts at some fixed node, and at each step it moves to a neighbor of the current node randomly chosen according to an arbitrary distribution. Figure 1 is an example of such a process on a grid.

This procedure can be easily implemented in sensor networks as follows. In the beginning, the base station releases a special message, or token, with a description of a query to be performed. When the token arrives at node v, the information contained in the token is updated with the local data stored at node v. Then, a neighbor of v is randomly chosen and the token is sent to it. When the answer is satisfying or a sufficient number of steps have been taken, the token is sent back to the base station. For example, say we want to find the minimum temperature in the network. Each time the token sees a lower temperature it updates the answer. After seeing enough nodes the token reports the value back to the base station.

The simplicity of the process makes it an almost assumption free method. Taking advantage of the broadcast nature of the communication, it is possible to define a simple protocol for token passing that does not require nodes to have knowledge about their location, neighbors, transmission range or symmetric connection. In addition since there is only one token (or few in parallel) moving around, collision is not a problem.

However, an important question is: "How many steps should a random walk take in order to obtain a satisfactory answer to a query ?" In the following, we approach this problem from an analytical perspective, and obtain some answers for the case of a simple random walk, a walk that chooses the next node uniformly at random.

Let $G(V, E)$ be the graph on which the random walk is performed and let $n = |V|$. For arbitrary nodes $i, j \in V$, let h_{ij} be the expected number of steps (or messages in our case) for the random walk to move from i to j. Then h_{max} (h_{min}) is defined as the maximum (minimum) h_{ij} over all ordered pairs of nodes. Finally, the Cover Time C is define as the expected number of steps taken by a random walk to visit every node in G, (starting from the worst node in the graph). The following theorem provides bounds on the cover time C in term of h_{max}.

THEOREM 1 (MATTHEWS' THEOREM [12]). *For any graph G,*

$$h_{min} \cdot H_n \;\leq\; C \;\leq\; h_{max} \cdot H_n$$

where $H_k = ln(k) + \Theta(1)$ *is the k-th harmonic number.*

Notice that the bound provided above is not always tight, since in the line, for instance we have $C = h_{max}$. Known results for the cover time of specific graphs vary from the best case of $O(n \log(n))$ to the worst case of $O(n^3)$. The best cases correspond to dense, highly connected graphs such: the complete graph, d-regular graphs with $d > \frac{n}{2}$, and the

Hypercube. When connectivity decreases and "bottlenecks" exist in the graph, the cover time increases. In the line, for example, it is known to be $O(n^2)$.

3.1 Upper Bound on Partial Cover Time

In sensor network application, however, for many tasks it is not always necessary to consult every node in the network. So, in the following we investigate the Partial Cover Time (PCT), the time required to cover only a constant fraction of the network (see Figure 2 for an example of the result of a random walk that covers 80% of a random network , this should give intuition about how such walks may look). For $0 \leq c \leq 1$, let $PCT(c)$ be the expected time to cover $\lfloor cn \rfloor$ nodes of a graph G. In Lemma 1 we prove that as long as we want to cover a constant fraction of the graph we can reduce Matthews bound by an order of $\log(n)$ so the bounds becomes linear in h_{max}.

LEMMA 1 (PARTIAL COVER LEMMA). *For any graph G, and $0 \leq c \leq \frac{n-1}{n}$*

$$PCT(c) \;<\; 2 \cdot h_{max} \cdot \left\lceil \log_2(\frac{1}{1-c}) \right\rceil \;=\; O(h_{max})$$

PROOF. For simplicity of exposition we prove the result for special cases where c is of the form $c_l = \frac{2^l-1}{2^l}$. The general case will follow directly by taking the minimum l such that $\frac{2^l-1}{2^l} \geq c$. Let $n = 2^L k + 1$. The proof will be by induction on l. Let $c_l = \frac{2^l-1}{2^l}$. For a fixed random walk, let γ_l be the minimum number of steps to visit more than $c_l \cdot n$ of the nodes. Let α_v be the time (step number) when node v is visited for the first time and let $S_l = \{v \in V \mid \alpha_v \geq \gamma_l\}$ be the set of all nodes visited at time γ_l or later. Note that $|S_l| = n - \lfloor c_l n \rfloor = 2^{L-l}k + 1$ and that $S_0 = V$ is the set of all nodes.

Base Case: The base case is stated in Lemma 2.8 [10] where $l = 1$, $c_l = \frac{1}{2}$ and visiting more than half of the nodes take less then $2h_{max}$ steps. We will follow this proof here:

Let $k' = k^L$ and so $n = 2k' + 1$ is odd. The time γ_1 when we reach more than half of the nodes is the $(k'+1)$-st largest of the α_v. Hence

$$\sum_{v \in S_0} \alpha_v \geq (k'+1)\gamma_1$$

taking the expectation on both sides

$$\begin{aligned} PCT(c_1) = E(\gamma_1) &\leq \frac{1}{k'+1} \sum_{v \in S_0} E(\alpha_v) \\ &\leq \frac{2k'+1}{k'+1} h_{max} \\ &< 2h_{max} \log_2(\frac{1}{1-c_1}) = 2h_{max} \end{aligned}$$

Induction Step: Assume true for $1, \ldots, l \leq L-1$. We will prove true for $l+1$. Let α'_v be the number of steps until node v is first visited after γ_l steps.

$$\sum_{v \in S_l} \alpha_v = \sum_{v \in S_l} (\gamma_l + \alpha'_v) \geq k^{L-l}\gamma_l + (k^{L-l}+1)\gamma_{l+1}$$

Taking the expectation, the l.h.s is:

$$\begin{aligned} E\left[\sum_{v \in S_l} (\gamma_l + \alpha'_v)\right] &= \sum_{v \in S_l} E[\gamma_l + \alpha'_v] \\ &= (2^{L-l}k + 1)PCT(c_l) + \sum_{v \in S_l} E[\alpha'_v] \end{aligned}$$

The r.h.s will be:

$$E\left[k^{L-l}\gamma_l + (k^{L-l}+1)\gamma_{l+1}\right] =$$
$$k^{L-l}PCT(c_l) + (k^{L-l}+1)PCT(c_{l+1})$$

Putting it together:

$$PCT(c_{l+1}) \leq PCT(c_l) + \frac{1}{k^{L-l}+1} \sum_{v \in S_l} E[\alpha'_v]$$

Using the induction assumption we get:

$$\begin{aligned} PCT(c_{l+1}) &\leq 2h_{max} \log_2(\frac{1}{1-c_l}) + \frac{2^{L-l}k+1}{k^{L-l}+1} h_{max} \\ &< 2h_{max} \log_2(\frac{1}{1-c_l}) + 2h_{max} \log_2(2) \\ &= 2h_{max} \log_2(\frac{1}{1-c_{l+1}}) \end{aligned}$$

□

This implies the following interesting results: for graphs in which $h_{max} = n$, PCT becomes linear in n; known graph of this type are the complete graph, the Star and the Hypercube. Covering x percent of the nodes in the Hypercube, for example, is linear in n while the cover time is $O(n \log(n))$. The Hypercube is of interest since it is a d-regular graph with $d = \log(n)$. For the grid, which is also a d-regular graph ($d = 4$) the maximum hitting time is $n \log(n)$ [18] so PCT becomes $O(n \log(n))$. These two graphs are related since we think of sensor networks as "almost" d-regular graphs that lie between these two.

4. EFFICIENCY OF PARTIAL COVER TIME

In the following four sections, we are going to study in more detail some of the properties of random walks and PCT, using simulations. We performed experiments on grids, the Hypercube and random graphs. The random graphs were generated by placing n nodes uniformly at random in a 1×1 square, and making neighbors of every pair of nodes at distance less than R. When not explicitly mention we assume $n = 4096$ and $R = 0.04$. For each experiment, we took the average results of 100 runs.

In this section, we want to validate the analytical result on the efficiency of Partial Cover Time. Figure 3 compare the partial cover time curves for a few distinct networks of size 4096: a grid, a random network with $R = 0.035$ (average node degree about 15), a random network with $R = 0.04$ (average degree of about 19) and a Hypercube of degree 12. Observe that most of the steps required to cover these networks are performed after 80% of the network has been covered. Moreover, we call attention to the fact that the high density of sensor networks improves the result for random walk , as opposed to other methods such as flooding where the high density causes more problems, mostly because of collisions [13].

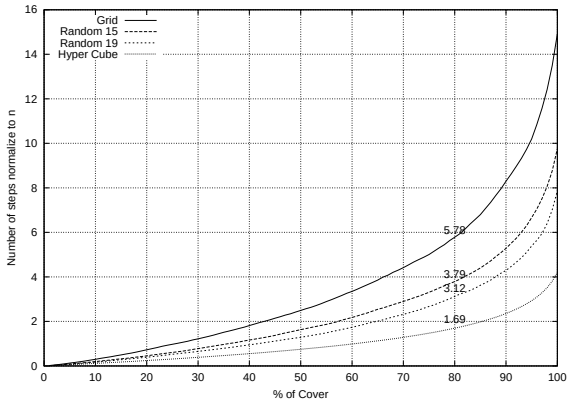

Figure 3: The Partial Cover time progress in four network of size 4096

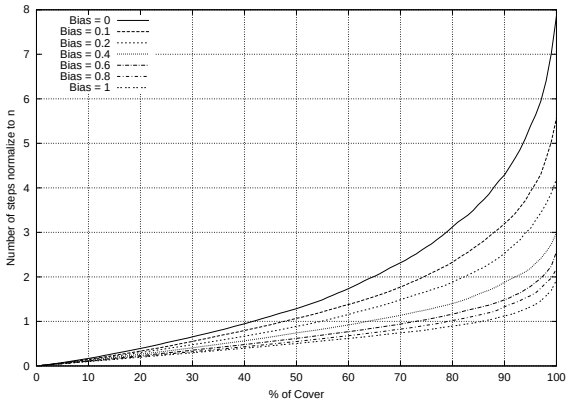

Figure 4: Partial Cover Time in random walks with increasing bias on random network

As we can see from the graph, the average degree is not the only parameter that influences the random walk. The average degree in the Hypercube is lower than in the random graph, but the connectivity is still higher and the maximum number of hops between any two nodes is 12. This means that if we lay out the Hypercube on a plane, many links will require long radio range which is not available in sensor networks.

4.1 Biased Random Walk

Can we improve on the previous result ? What if we can direct our random walk toward unvisited nodes ? Biased random walk gives priority to unvisited neighbors instead of choosing uniformly at random. We define a bias parameter $0 \leq bias \leq 1$ and select our next node according to the following rule: Let d be the number of neighbors of the current node, and let d_u be the number of unvisited neighbors. Then: (i) A visited neighbor is selected with probability $(1 - bias)/d$. (ii) An unvisited neighbor is selected with probability $(1 - bias)/d + bias/d_u$. If all neighbors are already visited and $bias = 1$, a neighbor selected uniformly at random is returned. When each node knows whether it has been visited then we can execute this protocol without knowledge of neighbors, again using the broadcast nature of the communication.

We realize that $bias = 1$ cannot be always maintained due to errors, but as we can see in Figure 4 the substantial improvement in the number of steps required is obtained even with a small bias. Notice also that covering 80% with bias greater than 0.8 requires less than n steps. A future improvement can be to have "super bias" which take the neighborhood into consideration the neighborhood. A node will decrease its priority even if it hears the token passing by in his neighborhood. We can do this easily, again, because of the broadcast channel.

4.2 Comparison with Cluster Head Scheme

In this subsection we compute the number of messages M that will be needed to collect data from the network by an optimal cluster head protocol with two-level hierarchy on a grid. The protocol goes as follow: each node sends its data to its cluster head via the shortest path. Cluster heads then aggregate all the information and sends it to the base station via the shortest path. Let n be the number of nodes in the grid and assume the base station is in the center of the grid. First we will consider the case of one cluster where the base station is the cluster head (this case is equal to the case where there are n cluster of size one). For simplicity let $\sqrt{n} = 2k + 1$. Then the number of messages required to collect all the information from the nodes is:

$$\begin{aligned} M &= 4 \cdot 1 + 8 \cdot 2 + \cdots + 4kk + 4k(k+1) + \ldots \\ &\ldots \quad +8(2k-1) + 4(2k) \\ &= 4\sum_{i=1}^{k} i^2 + 4\sum_{i=1}^{k} i(2k+1-i) = 4(2k+1)\sum_{i=1}^{k} i \\ &= 4\frac{(2k+1)k(k+1)}{2} \approx 2\sqrt{n}\frac{n}{4} = \frac{(\sqrt{n})^3}{2} \end{aligned}$$

Now assume we have square sub grids as clusters and cluster heads in the center of them and we want to optimize the number of clusters. Let x be the size of the side of a cluster. So there are $(\frac{\sqrt{n}}{x})^2$ clusters. In each cluster there are x^2 nodes. Let $f(x)$ be the number of messages sent using this protocol. This involves two quantities: (i) the number of messages sent from nodes to cluster heads, (ii) the number of messages sent from the cluster heads to the base station. Note that the cluster heads now form a grid with x as the new unit, so:

$$f(x) = \frac{x^3}{2}\left(\frac{\sqrt{n}}{x}\right)^2 + \frac{\left(\frac{\sqrt{n}}{x}\right)^3}{2}x = \frac{xn}{2} + \frac{n^{\frac{3}{2}}}{2x^2}$$

Taking the derivative we find that f is minimized at $x^* = (4n)^{\frac{1}{6}}$:

$$\begin{aligned} f'(x) = \frac{n}{2} - \frac{n^{\frac{3}{2}}}{x^3} &= 0 \\ n &= \frac{2n^{\frac{3}{2}}}{x^3} \\ x^3 &= \frac{2\sqrt{n}}{n} \\ x &= (4n)^{\frac{1}{6}} \end{aligned}$$

so the minimum number of messages this protocol will re-

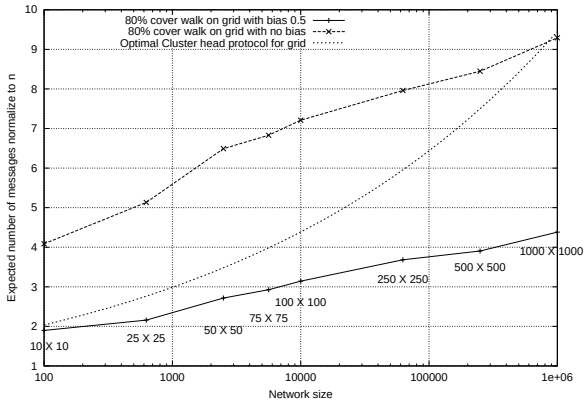

Figure 5: Comparing the number of messages required by a 80% random walk and optimal cluster head protocol on increasing grid

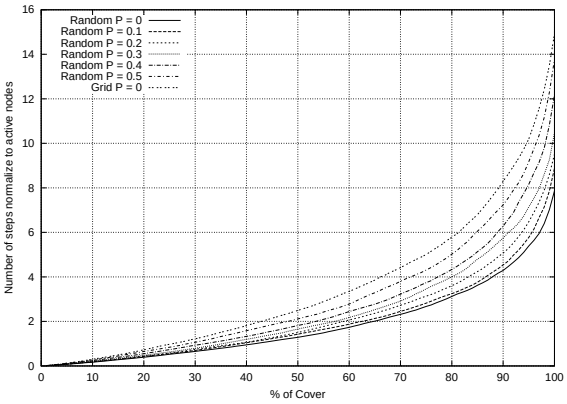

Figure 6: The Partial Cover time required when the probability p of each node to fail is increasing. The result are for 4096 nodes networks

quire is

$$f(x^*) = \frac{(4n)^{\frac{1}{6}}n}{2} + \frac{n^{\frac{3}{2}}}{2(4n)^{\frac{1}{3}}} \approx 0.945n^{\frac{1}{6}}n$$

From our analytical result we know that the PCT on the grid is $O(n\log(n))$ which is asymptotically better than $O(n^{\frac{7}{6}})$, the result for the cluster head. In Figure 5 we compare the cluster head result to the simulation result obtained using Partial Cover random walk on grid, one with no bias and one with bias 0.5. We examine the result of the methods on increasing grid sizes, plotted on a log scale. The cluster head curve does not include any additional calculation, just the optimal lower bound. In reality the result may be higher because of other factors, such as dynamics. The point we want to make is that the efficiency of PCT is on the same order as the cluster head protocol. The biased walk is doing better even on a small 10×10 grid.

5. ROBUSTNESS TO DYNAMICS

Dynamics in sensor networks are due to many different factors. Nodes can fail, turn off their radio in a duty cycle, or move. Wireless communication obstacles can disconnect links and more. How robust is our process to these dynamics ? We know that as long as the network is strongly connected with no bottlenecks, the random walk will do well. Actually we do not care about the whole network, since we only need this requirement in the area of the token. The nice thing is that this condition is, in many cases, orthogonal to the network dynamic. A high duty cycle rate, for example, can still leave the network strongly connected at any time, so the random walk is robust to that. No fault tolerance or recovery mechanism is in needed. As long as the token is alive, there is nothing to recover from.

To illustrate this point we choose the following model of failures. Each node can fail independently with probability p during the run (or nodes switching on-off in a duty cycle). The failure can occur at any time during the walk, so the probability that a node will fail exactly when it has the token is negligible . We simulated the worst case for the random walk, where all the failures happen at once before the walk starts. It is worth mentioning that for other methods all-fail-at-once may not be the worst case. For example for a spanning tree it will be the best case, while nodes turning on-off during the data collection will be worst. Figure 6 shows the result for this model for $p = \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$. The vertical axis is normalized to the number of active nodes in each run. We see that the walks in this case continue to perform well (better than the grid) even at high rate of failure.

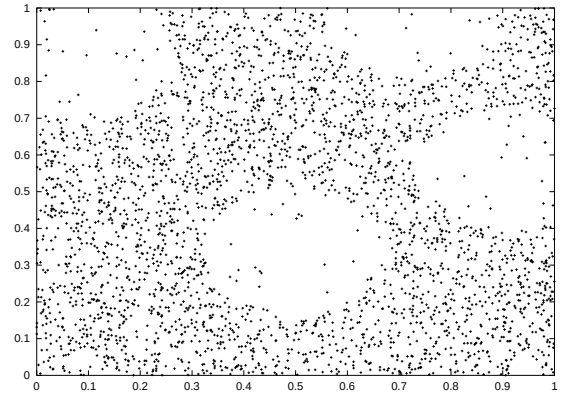

Figure 7: An example of a 4096 random network with 4 disaster areas. We can see the creation of bottlenecks

What about dependent, or correlated failures ? Our second model tries to answer that by creating random areas of "disaster" where at the center of each there is a source and the probability of failures decreases exponentially from the source with parameter α (up to r hops). Figure 7 shows an example of a 4096 nodes random network with 4 holes. As we increase the number of holes, we create more bottlenecks. Figure 8 supports that the random walk is highly robust to such failures. We can see that the effect of this type of failure is mostly on the covering of the last nodes and not on the Partial Cover.

But a token can still be lost, so what do we do then ? First, to solve the problem of unreliable communication which can destroy our token, we suggest using a reliable protocol. The cost of that will not be too high, assuming there is a probability p of each transmission to fail. The expected number

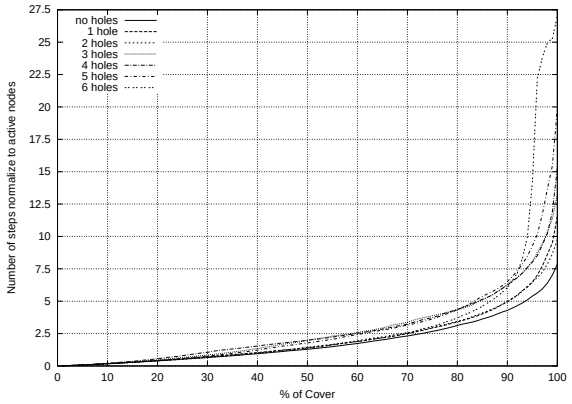

Figure 8: The Partial Cover time required when we increase the number of disaster areas in the network

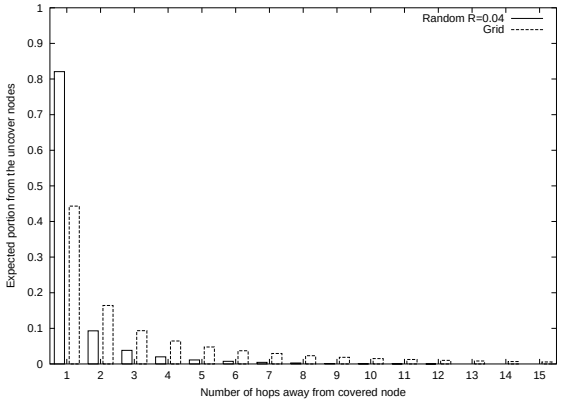

Figure 9: Histogram of the min distance of uncovered nodes from visited nodes after 80% of the network was covered. n=4096

of transmission to pass the token reliably (including ACK) will be $\frac{2-p}{(1-p)^2}$. For $p = 0.2$ for example we get 2.8125 transmissions. Second, if we still lost it, we will just issue another one. Since the probability for that is so small, it will not affect the overall performance of a few tokens traveling across the network at the same time.

5.1 Counter Example: Spanning Trees

Spanning trees are very powerful data structures that are used in many similar systems. They can also be used in sensor networks. For example, shortest path trees can be optimal for many types of query processing. The problem with this approach arises when dynamics are present in the network. Then, in order for the protocol to work, it has to maintain the tree at all times. Since there is only one path from each node in the tree to the root, when nodes or links fail they disconnect their sub tree from the root. Moreover, failures of nodes close to the root will lead to disconnectivity of a larger fraction of the network. To avoid this, the protocol has to include a recovery mechanism. This is a non-trivial task since we have to prevent loops during the recovery. In fact, this process must include all the nodes in the subtree and that significantly reduces the efficiency of the protocol and increase latency.

We conjecture that a spanning tree is not a scalable solution to a large, dense, dynamic sensor network. To show why, we compute the expected network fraction that will remain connected to the root given the probability of a node to fail during the query processing. Let p be the probability of a node to fail and $q = 1 - p$. To find a lower bound, we consider an optimal d-regular spanning tree where d is the maximum degree in the network. The data collection goes as follows: each node collect the values from all its unfailed children, aggregates it and sends it to its parent with probability q, or fails with probability p. Let T_i be a d-regular tree with height i and $n = \frac{d^{i+1}-1}{d-1}$ nodes. Let T_i^* be the same tree but the root is a base station that cannot fail. First we will find $E[T_i]$, the expected number of nodes in T_i whose data will reach the root (i.e if the root fails, no data will be available). We also assume that $qd > 1$ meaning that the expected number of children that will not fail is greater then 1. For the other case ($qd \leq 1$) the result will be worse. The base cases are: $E[T_0] = p \cdot 0 + q \cdot 1 = q$ and $E[T_1] = p \cdot 0 + q(1 + dE[T_0]) = q + q^2 d$. For the general case $i = h$ we have:

$$\begin{aligned} E[T_h] &= p \cdot 0 + q(1 + dE[T_{h-1}]) = q + dq(q + dq(E[T_{h-2}]) \\ &= \sum_{i=0}^{h} q^{i+1} d^i = q \sum_{i=0}^{h} (qd)^i \end{aligned}$$

Since $qd > 1$ we get

$$E[T_h] = q \frac{(qd)^{h+1} - 1}{qd - 1}$$

In the case of T_h^* the root cannot fail so the expected number of nodes that will report to the root is $E[T_h^*] = dE[T_{h-1}]$, and the total number of nodes except the root in T_h^* is $|T_h^*| = d(\frac{d^h-1}{d-1})$. The expected fraction of the tree that will report to the root is $E_{\%}[T_h^*] = \frac{E[T_h^*]}{|T_h^*|}$.

$$E_{\%}[T_h^*] = \frac{dq\frac{(qd)^h - 1}{qd - 1}}{d\frac{d^h - 1}{d - 1}} \approx q\frac{d-1}{qd-1}\frac{(qd)^h}{d^h} = O(q^h)$$

In sensor networks the radio transmission radius of the node is usually relatively small compared to the network size, and so the diameter of the network is not small. Since any spanning tree has height which is at least the diameter of the network divided by two, a spanning tree in sensor network will have considerable height which will cause a large fraction of the network to execute a recovery mechanism. In our simulation with R=0.04, the diameter is about 35, but even if we take, for example, $p = 0.1$ and $h = 10$ then 65% of the nodes will be disconnected from the root. We think that this property make the spanning tree not robust to dynamic large networks, and so not a scalable solution.

6. QUALITY OF PARTIAL COVER TIME

Partial Cover is fast, but what is the quality of the cover ? In Figure 9, we try to measure the quality of 80% cover by showing how far on average the uncovered nodes are from the ones that have been covered. We show a histogram of the minimum distance between uncovered to covered nodes. As we can see in the random graph, about 90% of the uncovered nodes are at most 2 hops away from a covered node. In the

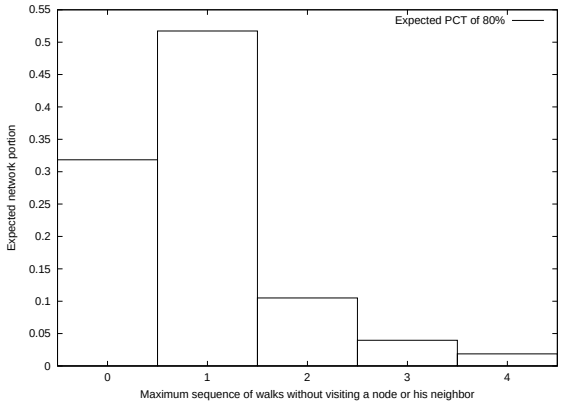

Figure 10: Histogram of the maximum number of consecutive runs without a node or his neighbor in 100runs of 80% cover

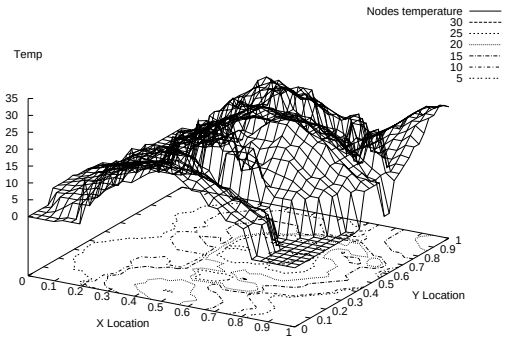

Figure 11: An example of the temperature in an area with six random light sources

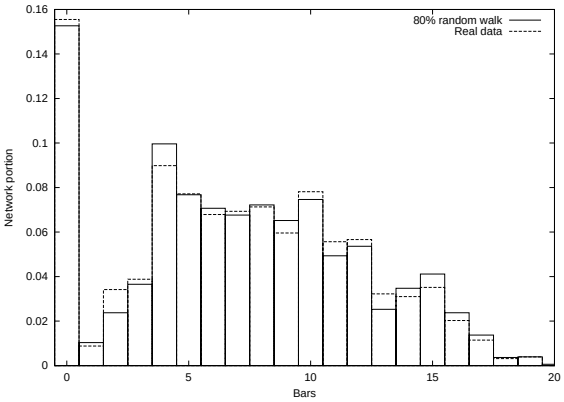

Figure 12: Comparing the histogram founded by the 80% random walk on the graph and the histogram of the real data from Figure 11

grid it is about 60%. This is not an obvious result: if we arrange the 80% covered nodes on a same size grid, in the worst case, less than 15% of the uncovered nodes will be at most 2 hops away. The random walk is leading to a much better result. This measure is important since sensors are densely spread with redundancy and phenomena will be probably detected by more than one node. This histogram shows that expected run on networks of this size will not leave large areas uncovered.

From Figure 9 we can also observe that the expected walk does leave some nodes far away from covered nodes. How many walks do we need until we will visit those nodes ? Figure 10 shows that if an area is left out in one walk it will be visited in the next few runs with very high probability, so all "neighborhoods" of the network will be covered very fast. This figure shows what is the maximum sequence of consecutive walks that does not visit a node or any of his neighbors (i.e neighborhood) in 100 runs. The neighborhood of nodes in the 0 column were visited in every run, for the nodes in the 1 column there was at least one run in which they were not visited, and no two consecutive runs without a visit and so on. In 100 runs we see that for about 85% of the nodes, a walk will visit their neighborhood at most every other walk. In the worst case, a node will have to wait 4 walks for his neighborhood to be visited.

6.1 Application Example

Until now we only mentioned briefly applications or tasks that can be done using random walks, but it is clear that many types of queries can be answered using this method, for example; finding the min, max or mean of the data, calculating statistics of the network, finding sensors with specific thresholds and so on. We can even use a SQL like language to describe these queries as offered in [11, 3].

To illustrate this ability we choose as an example a simple query, finding the histogram of the data in the network. The procedure is as described earlier, the token is moving in the network, each time it arrives at an unvisited node it updates its histogram (we note that in order to distinguish between visited and unvisited nodes, each node should keep a flag if it has been visited by this walk). After seeing 80% of the nodes the token reports its histogram back.

We wanted to make reasonable assumptions about the data observed by the sensors in order for this task not to be trivial. For example, assuming uniform distribution over the data did not seem reasonable, so we created a model of light sources that will produce a non-uniform temperature distribution over the network, Figure 11 is an example of such a distribution. Figure 12 compares the histogram obtained from the real data and the one collected by the random walk with 80% cover. As we can see from this example, the histogram is accurate with expected error as low as 0.37%.

7. OTHER PCT PROPERTIES

7.1 Load Balancing

Random walk is an uncontrolled process. It may be the case that the walk will go to a neighbor and return back to the same node after one step. Even if we forbid this move, we cannot prevent it from happening in small cycles. This leads to the intuition that there will be nodes that are much more visited than others, so consuming more energy. However, for very long random walks this is not necessarily true. Since this process is a Markov Chain, it is known that the stationary distribution π of it is $\pi = \{\pi_1, \ldots, \pi_n\}$ where $\pi_i = d_i/2m$, d_i is the number of neighbors of i and m is

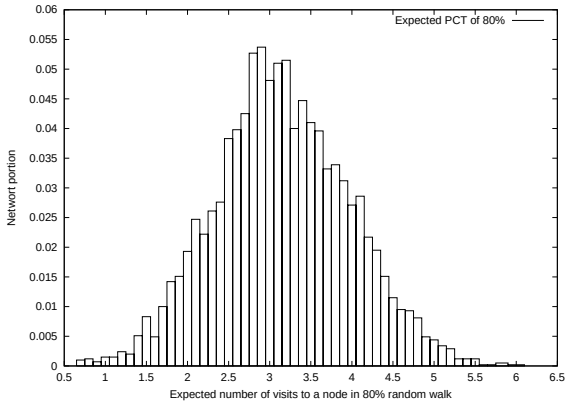

Figure 13: Histogram of the expected number of visits to a node in a 80% cover random walk

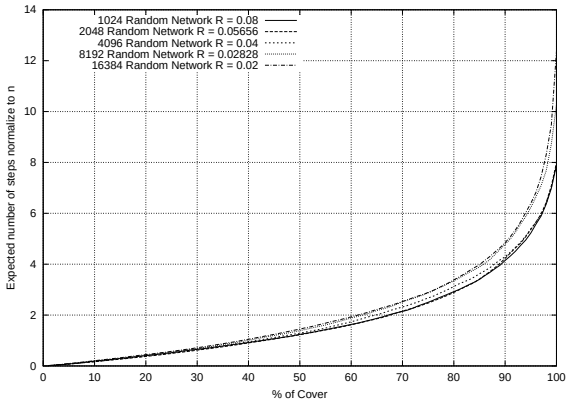

Figure 14: Partial Cover time in increasing size of random network with same density

the number of edges in the network [2]. So if the graph is regular the stationary distribution will be uniform. This means that after a long enough time the probability of the token to be at any node is the same. From this, it follows that the frequency of returning to a node is the same for all nodes, which gives us load balancing.

We see sensor networks as "almost regular" graphs where with very high probability each node degree is bounded by a constant, but in our case we are issuing "short" random walks. There is no guarantee that in any given walk there will not be nodes that are much more visited than others. Although this is true, our simulation results show that if we issue many (i.e 100) such "short" walks, we will still get very close to this property.

To measure the load balance we looked at the expected number of visits to each node in a 80% cover random walk. Figure 13 present the histogram of these values in a random walk with 13100 steps. The mean of the expected number of visits for each node is 3.19 which is approximately 13100/4096 and the standard deviation is 0.81606. This is showing that only small part of the network will be visited much less (more), and will use less(more) energy. For this histogram more than 95% of the nodes are in $mean(x) \pm std(x)$.

7.2 Scalability

Figure 14 validates our theoretical result that Partial Cover in sensor networks is scalable. The graph shows the increase in the Partial Cover normalized to n in increasing random network size with constant density. While increasing the network 16 times from 1024 nodes to 16384 the expected number of steps to cover 80% increases from $2.92n$ to $3.37n$. In the Hypercube for example the normalization factor for covering 80% will remain constant as n increases (for big enough n).

7.3 Latency

Random walk is a sequential process where no steps are performed in parallel. The latency therefor is proportional to the number of steps required to accomplish the task. In many cases this will be on the order of the size of the network. This is an inherent problem of the approach, and certainly reduces its range of applicability. One possible idea to mitigate this problem is to divide the network into regions, and perform random walks in parallel on each of them. However, it is not clear if every query can be answered in this way. We leave this and related questions to future work.

8. CONCLUSIONS

In this paper we studied the use of random walks for query processing in sensor networks. The approach is especially appealing for dynamic environments where approaches that maintain data structures must relay on recovery mechanisms which reduce their performance. Our analysis and simulation shows that the approach, while simple and achieve a high quality results, is very efficient and robust to failures. In particular, it is more efficient than the cluster head on a grid and more robust than the spanning tree in under dynamics. Clearly, this approach may not be a general solution for all types of queries. However, whenever it applies, it provide an elegant, simple and efficient solution.

9. ACKNOWLEDGMENT

We would like to thank Deborah Estrin for helpful discussions. We would also like to thank Mark Hopkins for his help.

10. REFERENCES

[1] D. J. Aldous. On the time taken by random on finite groups to visit every state. *Z. Wahrsch. Verw. Gebiete*, 62(3):361–374, 1983.

[2] R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovász, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *20th Annual Symposium on Foundations of Computer Science (San Juan, Puerto Rico, 1979)*, pages 218–223. IEEE, New York, 1979.

[3] P. Bonnet, J. Gehrke, and P. Seshadri. Querying the physical world. *Personal Communications, IEEE*, 7(5):10–15, 2000.

[4] D. Braginsky and D. Estrin. Rumor routing algorthim for sensor networks. In *Proc. of the 1st ACM Int. workshop on Wireless sensor networks and applications*, pages 22–31. ACM Press, 2002.

[5] D. Cerpa, A.and Estrin. Ascent: Adaptive self-configuring sensor networks topologies. In *Proc. of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. INFOCOM*, volume 3, pages 1278–1287, 2002.

[6] A. K. Chandra, P. Raghavan, W. L. Ruzzo, and R. Smolensky. The electrical resistance of a graph captures its commute and cover times. In *Proc. of the twenty-first annual ACM symposium on Theory of computing*, pages 574–586. ACM Press, 1989.

[7] A. Heinzelman, W.R.and Chandrakasan and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proc. of the 33rd Annual Hawaii Int. Conference on System Sciences*, pages 3005–3014, 2000.

[8] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking (TON)*, 11(1):2–16, 2003.

[9] J. Kahn, J. H. Kim, L. Lovasz, and V. H. Vu. The cover time, the blanket time, and the matthews bound. In *IEEE Symposium on Foundations of Computer Science*, pages 467–475, 2002.

[10] L. Lovász. Random walks on graphs: A survey. In *Combinatorics, Paul Erdős is eighty, Vol. 2 (Keszthely, 1993)*, volume 2 of *Bolyai Soc. Math. Stud.*, pages 353–397. János Bolyai Math. Soc., Budapest, 1996.

[11] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. *ACM SIGOPS Operating Systems Review*, 36(SI):131–146, 2002.

[12] P. Matthews. Covering problems for Brownian motion on spheres. *Ann. Probab.*, 16(1):189–199, 1988.

[13] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proc. of the 5th annual ACM/IEEE Int. conference on Mobile computing and networking*, pages 151–162. ACM Press, 1999.

[14] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, 2000.

[15] N. Sadagopan, B. Krishnamachari, and A. Helmy. Active query forwarding in sensor networks (acquire). *To appear in Elsevier journal on Ad Hoc Networks*, 2003.

[16] S. D. Servetto and G. Barrenechea. Constrained random walks on random graphs: routing algorithms for large scale wireless sensor networks. In *Proc. of the first ACM Int. workshop on Wireless sensor networks and applications*, pages 12–21. ACM Press, 2002.

[17] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin. Data-centric storage in sensornets. *ACM SIGCOMM Computer Communication Review*, 33(1):137–142, 2003.

[18] D. Zuckerman. A technique for lower bounding the cover time. In *Proc. of the twenty-second annual ACM symposium on Theory of computing*, pages 254–259. ACM Press, 1990.