Wireless Sensor Networks and Applications
SECTION III Data Management
Y. Li, M. Thai and W. Wu (Eds.) pp. 572-591

Chapter 15

Modeling Data Gathering in Wireless Sensor Networks

Bhaskar Krishnamachari
Department of Electrical Engineering-Systems
Viterbi School of Engineering, University of Southern California, Los Angeles, CA 90089
E-mail: bkrishna@usc.edu

1 Introduction

The predominant protocol development methodology in the area of wireless sensor networks today can be characterized as being one of *design-by-intuition/validate-through-simulation*. However, a concurrent development of complementary theoretical models for these protocols is essential for the rapid advancement of this technology. This is because the severe resource constraints on energy, computation, and storage that are characteristic of these networks make it crucial to optimize protocols, in order to maximize network lifetime while providing an acceptable quality of sensed information.

We survey in this chapter some examples drawn from recent studies pertaining to data gathering in sensor networks. These examples demonstrate the importance of optimizing protocol parameters carefully to maximize performance. We emphasize the use of a *first-order mathematical modeling approach*. The optimizations are all based on simple expressions that are intended to capture the impact of essential environmental and protocol parameters. This approach is motivated by the desire to understand easily the key design tradeoffs that are involved in each context. As we shall see, even such simple models can yield fundamental design insights to improve the performance of practical protocols.

This style of modeling is nicely described in an essay written by the noted economist, Prof. Hal Varian, titled "How to build an economic model in your spare time" [1]:

> "The critical advice here is KISS: keep it simple, stupid. Write down the simplest model you can think of, and see if it still exhibits some interesting behavior. If it does, then make it even simpler.
>
> ... keep at it till it gets simple. The whole point of a model is to give a simplified representation of reality. Einstein once said 'Everything should be as simple as possible... but no simpler.' A model is supposed to reveal the essence of what is going on: your model should be reduced to just those pieces that are required to make it work."

The methodology employed in these studies can be summarized as follows:

1. Identify the unique functionality of the protocol to be modeled. What exactly does it do — does it provide for routing with data aggregation from a set of sources to a common sink, or is it for gathering information from the network to resolve a query for a specific named attribute?

2. Identify the primary performance metric of interest in analyzing this protocol. For battery-constrained sensor networks this often translates to minimizing the total number of transmissions required to accomplish the specified networking task. This is because under the assumption that idle listening and overhearing can be minimized through appropriate MAC-level scheduling, transmissions of packets (and their corresponding receptions) are the primary source of energy consumption.

3. Identify the building blocks of the model. It is typical in these models to assume some simple topology such as uniform random placement with a fixed radius for connectivity, or a carefully placed grid of sensors, each communicating with just their four cardinal neighbors. Other significant building blocks for the model are the environmental and protocol parameters that have an significant impact on performance. As we are aiming to build an abstracted, simplified model of reality, this need not be an exhaustive list, but should include key parameters. This is more of an art than a science in many respects, and often entails an iterative process to refine the components of the model.

4. Derive a simple mathematical expression that gives the performance metric for that protocol, as a function of the variables corresponding to key environmental, network, and protocol parameters. Along with the previous step, this is part of the core work of building the model.

5. Refine the model, by adding, discarding, or modifying variables corresponding to environmental, protocol or network settings. The goal is to obtain an expression for the protocol performance metric that illustrates the core tradeoff.

6. Solve for the value of the protocol parameter which optimizes the performance metric of interest. There are often opposite trends in different components of the model that are in tension with each other. As a result, performance loss may be incurred in setting a key protocol parameter to too low a value or too high a value. In most cases, determining the optimum parameter setting requires just finding the zero of the derivative of the metric with respect to the parameter in question. The obtained result reveals how the optimal protocol parameter setting depends upon environmental and network conditions.

In the following sections, we shall present models for three specific problems pertaining to data gathering in wireless sensor networks. In the first case study, we optimize the look-ahead parameter for an active querying mechanism that provides a tunable tradeoff between trajectory-based and flooding-based querying. In the second case study, we

optimize the cluster size for joint routing and compression that minimizes the total transmitted information for a prescribed level of correlation between the sources. Finally, in the third case study, we look at a problem of querying for replicated information and identify the optimal number of replicas that minimizes the total energy cost involved.

2 Active Querying with Look-Ahead

The ACQUIRE mechanism [2] is an active querying technique designed for sensor networks. In essence, it consists of the following repeated sequence of steps: (i) An active node which receives the query checks its local cache to see if it contains the information requested. (ii) If the information in the local cache is not fresh, the active node sends a controlled flood of the query to all nodes within d hops of it to obtain a fresh update of the information. (iii) If the query is still not resolved, the query is forwarded to a succeeding active node that is $2d$ hops away along a trajectory (which could be random or guided in some way). Finally, when the information being sought is obtained, the query response is routed back to the original querying node. This is illustrated in figure 1.

One interesting observation about the ACQUIRE querying mechanism is that the look-ahead parameter d essentially allows for tuning across a wide range of behaviors. When $d = 0$, then the query is simply forwarded along some path until it is resolved (e.g., a random walk query, or a geographic trajectory based query). When d is large enough to be the diameter of the network, then the query is

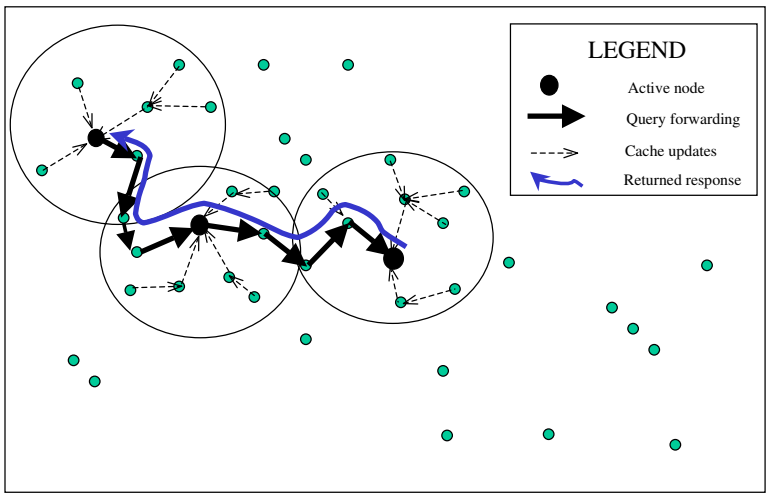


Figure 1: Illustration of ACQUIRE

essentially performed as a network-wide cache-based flooding.

A key question in this setting that we shall address with first-order analysis is what should determine the optimal setting of this look-ahead parameter d. It turns out that, since caching is employed, this is affected primarily by the ratio of updates to queries, which we denote by c. (When $c = 0.01$, for example, on average one update is requested every 100 queries. Alternately, we could say that the cache at each active node remains valid on average for 100 queries). This parameter quantifies the level of dynamics in the environment relative to the rate at which queries are posed.

We will use, as the metric of interest, the average total number of transmissions required to resolve a query. For ACQUIRE this is essentially the product of two factors, the expected number of steps (i.e. the number of active

query nodes visited in the trajectory), and the expected total number of transmissions incurred at each step. Interestingly, each of these factors depends in a different manner on the look-ahead parameter d. The expected number of steps is smaller when the look-ahead parameter is large, because each step would cover a larger portion of the network and make it more likely that the query is resolved in fewer steps. However, with a larger look-ahead parameter, the expected number of transmission incurred at each step is larger as the controlled flood has to reach a larger number of nodes.

Let $S(d)$ be the expected number of steps, and $T(d)$ be the expected number of transmissions incurred at each step. Let us denote by η the expected number of nodes that must be searched in order to resolve the query (we assume here that this is a constant regardless of how the query is implemented in practice. This is reasonable if the query is essentially a blind, unstructured search). For randomly deployed nodes with a uniform distribution, the number of nodes "covered" at each step with a look-ahead of d is $\gamma \cdot d^2$ (here $\gamma = \rho\pi R^2$, where ρ is the deployed density of nodes per square meter and R the nominal radio range for each hop). Then the expected number of steps needed to resolve the query can be expressed as:

$$S(d) = \frac{\eta}{\gamma d^2} \tag{1}$$

The expected number of transmissions incurred at each step depends on c, since this determines how frequently the local controlled flood is invoked. When the flood is invoked, we assume that all nodes in the $d-1$ hops forward

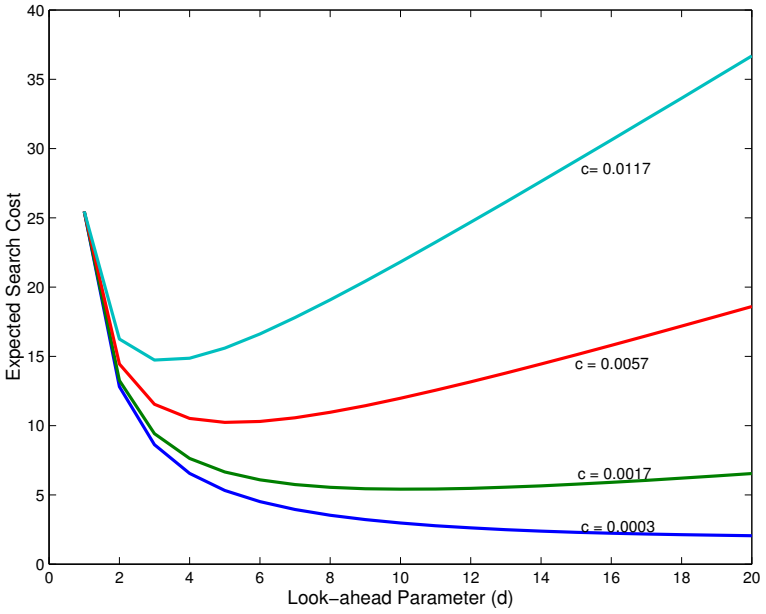
the flood, and all nodes within the d hops respond back with messages providing their respective information. The expected number of nodes at hop i is $\gamma i^2 - \gamma(i-1)^2 = \gamma \cdot (2i-1)$. These must all send their information back to the active node through i transmissions. The expected number of transmissions at each step is therefore:

$$T(d) = c(\gamma(d-1)^2 + \gamma \sum_{i=1}^{d}(2i-1)i) = c\gamma((d-1)^2 + \frac{1}{3}d(d^2-1)) \tag{2}$$

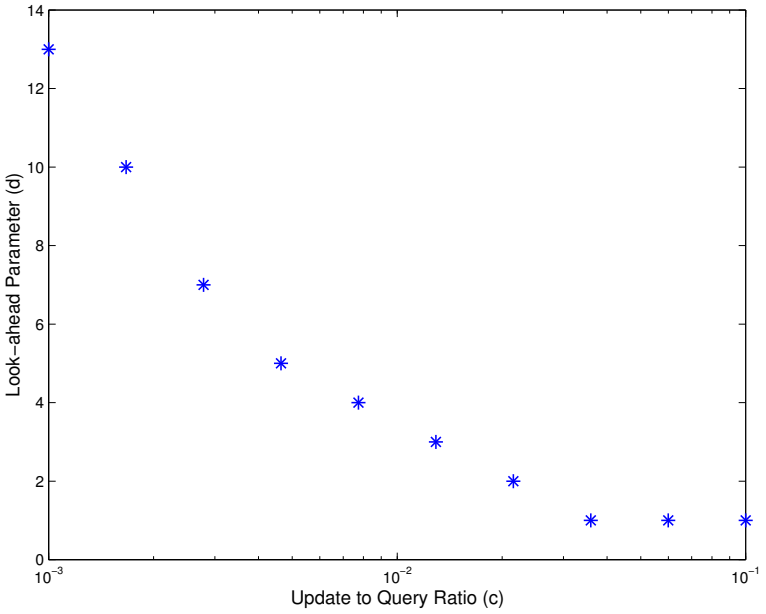
Now taking into account that the resolved response to the query must then be returned to the original querying node by incurring about $S(d) * 2d$ additional transmissions (assuming $S(d) > 1$, else it would be 0), we have the total expected number of transmissions $N(d)$ required by ACQUIRE with a look-ahead setting of d to be as follows:

$$N(d) = S(d)(T(d)+2d) = \frac{\eta}{\gamma d^2}(c\gamma((d-1)^2 + \frac{1}{3}d(d^2-1)) + 2d) \tag{3}$$

This expression is plotted in figure 2(a) for different values of c (assuming $\eta = 400, \gamma = 10\pi$. We can see that for a fixed value of c, the optimal setting of the look-ahead parameter that minimizes the total number of transmissions varies. We can determine the optimal d by taking the derivative of the above expression with respect to d and setting it to zero (the resulting real value is then rounded to the nearest integer). The numerical solution for the optimal d is plotted as a function of c in figure 2(b). This figures quantifies the insight that a smaller look-ahead (corresponding to a trajectory based search) is favored when



(a)



(b)

Figure 2: Performance of ACQUIRE

the environmental dynamics are so high that caching is not effective (high c), whereas a larger look-ahead (resembling flooding) is favored when caches can be used with high frequency (low c).

3 Cluster-Based Joint Routing and Compression

Because of their application-specificity, sensor networks are capable of performing data-centric routing, which allows for in-network processing of information. In particular, to reduce total energy consumption, data from correlated sensors can be compressed at intermediate nodes even as they are routed. We examine now how the appropriate joint routing and compression strategy can depend on the degree of correlation between the sources.

We first need a model to quantify the amount of information generated by a set of sources. We use here a simple model that has been previously validated with some real data [3]. In this model, there is a tunable parameter δ which varies from 0 to 1 and provides an indicator of the level of correlation between the sources. We use the joint entropy H_n of the sources as the measure of the total information they generate, assuming that each individual source has an identical entropy of H_1:

$$H_n(\delta) = H_1(1 + \delta(n - 1)) \tag{4}$$

Thus, when $\delta = 0$, the correlation is the highest (the sources sensing identical readings), resulting in a joint entropy that is equal to the entropy of a single source. On the other extreme, when $\delta = 1$, there is no correlation at

all, with a joint entropy that is equal to the sum of the entropies of each source.

To illustrate the tradeoffs involved in joint routing and compression, we consider a simple network scenario illustrated in figure 3. The n sources are equally spaced and each having D additional intermediate nodes between them and the sink. The way the information is routed from each source to the sink is as follows.

First the set of sources is divided into clusters of s nodes. Within each cluster, the data is routed sequentially from node to node, with compression at each successive step. Thus the H_1 bits of data from the first sensor move to the second sensor, where they are compressed jointly with the data at that node (we assume an idealized entropy coding that achieves the maximum compression possible); then H_2 bits are transmitted from the second sensor to the third, and so on till the last node within the cluster. Then the jointly compressed H_s bits of data from each cluster are routed to the sink along the shortest path. Thus we have a family of cluster-based joint routing and compression strategies that span from one extreme ($s = 0$) where no compression is performed and each node routes its information along the shortest path to the sink, to the other extreme ($s = n$) where the data from every source is compressed sequentially before routing to the sink.

The key question we address in this modeling effort is: what is the correct setting of the cluster size? The energy metric we use is the total number of bits that are transmitted over the air to deliver sensed data from all sources to the sink. We can intuit the tradeoff that is involved here:

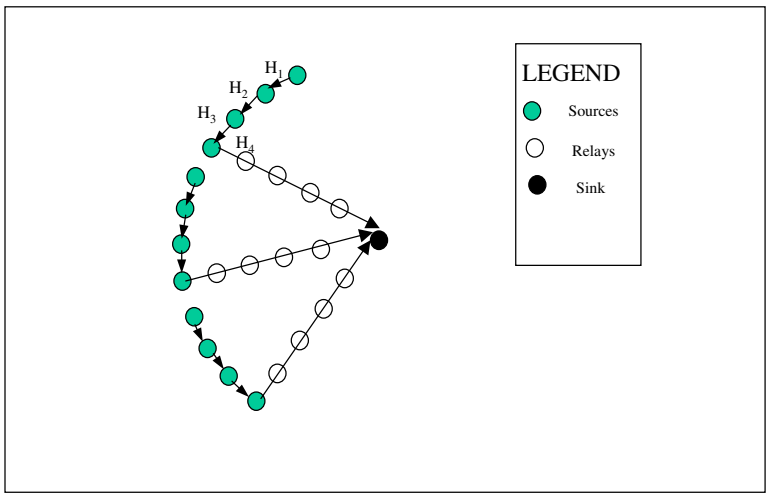


Figure 3: Illustrative Scenario for Cluster-based Routing with Compression

a strategy that uses a small cluster size favoring shortest path routing may perform best when the correlation is low (high δ), while a strategy using a large cluster size may perform best when the correlation is high (low δ). This is because when the correlation is high, the savings due to compression of data near the sources outweigh the benefits of shortest-path routing.

We need to consider the two components of the cost in terms of the total number of bits required to transport information from all sensors to the sink. Within each cluster the cost is $\sum_{i=1}^{s} H_i(\delta)$. To carry the combined information from each cluster to the sink requires a cost of another $H_s D$, and there are n/s clusters in all. Therefore the total cost for first compressing within clusters of size s and then transporting the information to the sink is given by the following expression:

$$C_{total}(s,\delta) = \frac{n}{s}(\sum_{i=1}^{s} H_i(\delta) + H_s D) \quad (5)$$

$$= nH_1 \frac{(s - s\delta + \delta s(s-1)/2 + D + D\delta(s-1))}{s} \quad (6)$$

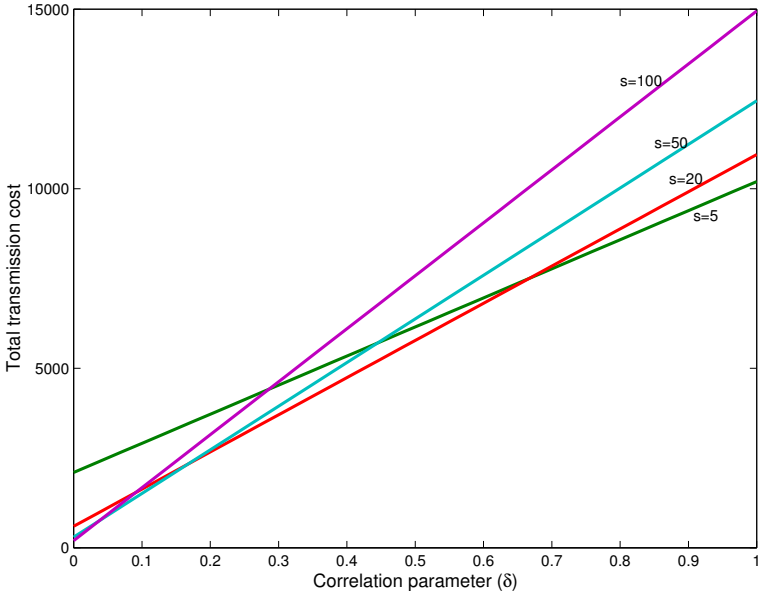
$$= nH_1(1 - \delta + D\delta + (s-1)\delta/2 + D(1-\delta)/s) \quad (7)$$

The minimization of the above expression for the total cost yields the optimal cluster size to be

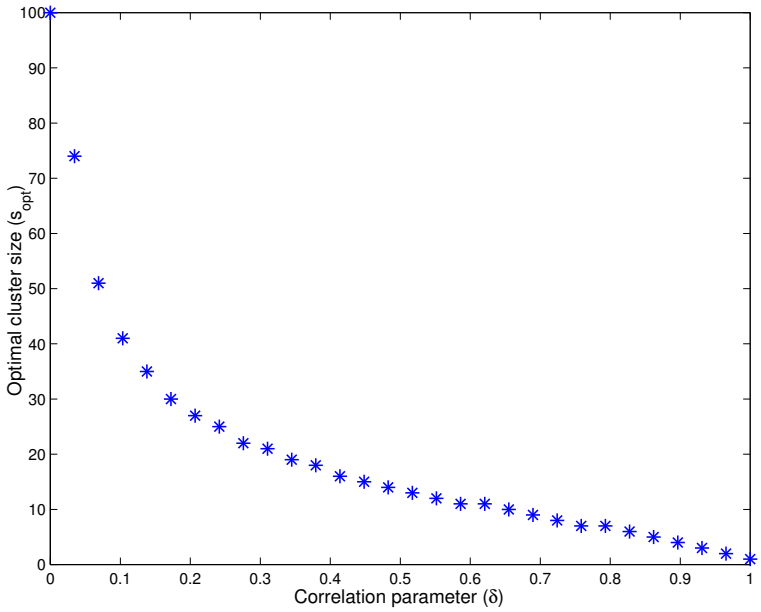
$$s_{opt}(\delta) = \sqrt{2D\frac{1-\delta}{\delta}} \quad (8)$$

The above applies for all intermediate values of δ in $(0, 1)$. For the two extreme cases, we get that when $\delta = 0$, $s_{opt}(0) = n$ and when $\delta = 1, s_{opt} = 1$. Figure 4(a) shows the performance for different cluster sizes as a function of the correlation level, for a scenario with $n = 100, D = 100$. Figure 4(b) shows the optimal cluster size s_{opt} decreasing with δ. This quantifies the tradeoff mentioned above, that a high correlation favors large clusters, while low correlations favor small clusters.

In [3], this analysis is validated for more general topologies through simulations involving random placement of sensors in a 2D region. Another interesting finding of that study is that while the optimal cluster size is indeed a function of the level of correlation, it is also possible to use a static cluster size that provides near-optimal performance regardless of the correlation. While we do not go into the analysis of the near-optimal clustering here, it is interesting to note that figure 4(a) suggests such a result — the clus-



(a)



(b)

Figure 4: Performance of Clustering

ter size of 20 provides good performance for all correlation levels.

4 Joint Search and Replication

As a third case study to illustrate first-order modeling, we examine the problem of querying a sensor network for information that can be replicated.

In this scenario, each node that senses a unique event (e.g. "there is a bird at location (x,y)") not only stores this information at its own location, but also creates additional replicas of this information and sends it to $k-1$ other (randomly selected) locations in the network for a total of k replicas of the information. In this problem, we assume that any random node (not just a single pre-identified sink) can be the source of a query for this information, so that there is no incentive to store the replicas in any particular locations.

To simplify the analysis we will focus on a simple grid network where each node can communicate with its four cardinal neighbors. A more sophisticated version of the analysis described here, considering expanding ring searches for a randomly deployed network, is presented in [4]. In the simple grid network, we assume that each query proceeds sequentially in a pre-determined trajectory that (if a solution is not obtained) eventually visits all nodes in the network. Figure 5 shows how several queries for the same event originating from different location are resolved at different replicas of that event.

We aim to minimize the total expected cost of search and

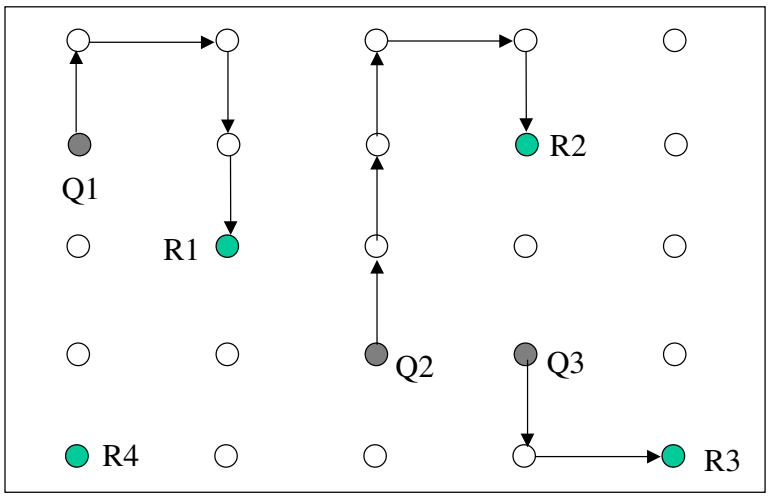


Figure 5: Illustration of Search with Replication

replication for each event. While there is an energy cost to be paid for moving each replica of the information to its location, having more replicas reduces the expected search energy cost. The search energy cost is measured in terms of the total number of transmissions needed for the query to locate the nearest replica. We could also account for the number of transmissions needed to return the response back to the querier by doubling this number, assuming that the response is returned along the reverse path.

Let us first consider the cost of replication. The expected number of transmission required to place each replica at a randomly selected location is the expected Manhattan distance (i.e. the L_1 distance, measured as the sum of the absolute distance in the x-coordinate and the absolute distance in the y-coordinate) between any pair of nodes in the $n \times n$ grid. The expected x-distance is $n/3$ and the expected y-distance is $n/3$, hence $2n/3$ transmissions are required on

average to place each replica. To place $k-1$ replicas, this cost is then:

$$C_{replication} = \frac{2}{3}n(k-1) \tag{9}$$

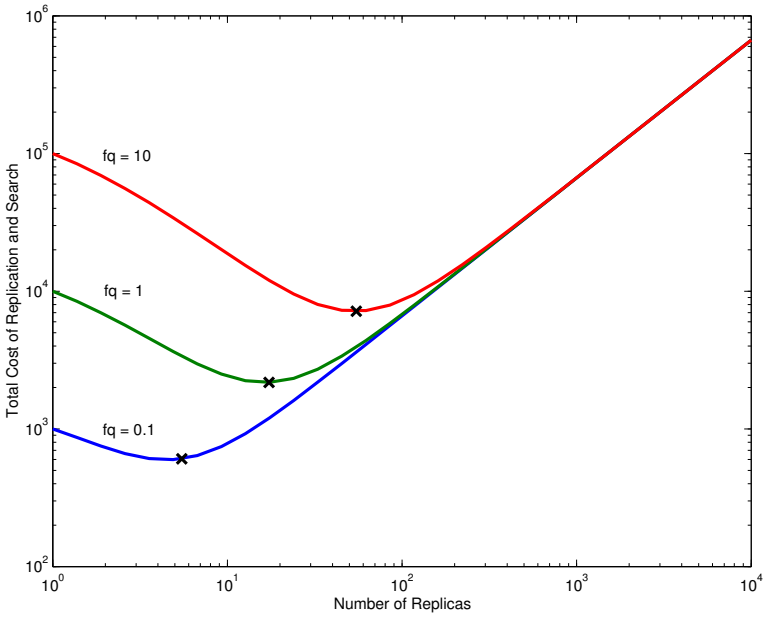
We now look at the search cost. The expected number of nodes visited on the trajectory till the nearest replica is obtained is the expected value of the minimum of k discrete random variables chosen from values between 1 and n^2 without replacement. A good approximation for this minimum is $n^2/(k+1)$ (which can be obtained, for instance, by considering a continuous distribution and using an integral to compute the corresponding integral expression for the expected value). Taking into account the same cost for the returned response, the expected search energy cost is therefore

$$C_{search} = 2\frac{n^2}{k+1} \tag{10}$$

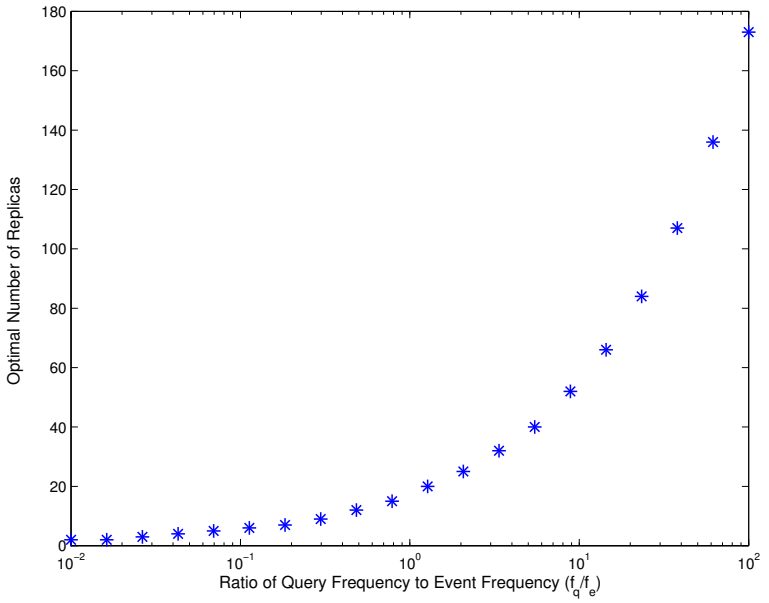
Combining the two components with variables f_e and f_q to denote, respectively, the frequency with which the event is generated (we can assume that an update of each replica occurs whenever new information is obtained about that event) and the frequency with which a query for the event is sent, we have:

$$C_{total}(k) = 2f_q\frac{n^2}{k+1} + \frac{2}{3}f_e(k-1)n \tag{11}$$

The combined cost of search and replication for different values of the query frequency f_q (assuming $f_e = 1$) is shown in figure 6(a) for a 100×100 grid. The optimal replication



(a)



(b)

Figure 6: Performance of Search with Replication

size can be determined from the above expression as

$$k_{opt} = \sqrt{\frac{3f_q}{f_e}n} \tag{12}$$

The variation of the optimal replication size with respect to the ratio of query frequency to event frequency is shown in figure 6(b).

5 Conclusions

We have shown several examples of first-order analysis for data gathering mechanisms in sensor networks. Such modeling can provide a clear understanding of the key design tradeoffs underlying such mechanisms, and give an insight into how protocol parameters should be optimized to provide efficiency in terms of energy savings or latency improvements, or other relevant metrics.

We should caution, however, that the first order protocol analysis methodology, which emphasizes tractability, abstraction, and simplicity, is by no means a complete substitute for other analytical and experimental tools which emphasize detail, realism, and sophistication. While such modeling provides a good starting point for understanding the key design tradeoffs involved in parameter selection, it is entirely complementary to other approaches. The translation of the insights obtained through mathematical analysis into practical protocols will certainly involve further evaluation through more detailed simulations, as well as through experiments on a test-bed and real-world implementations.

6 Acknowledgement

This work has been supported in part by the following grants from the National Science Foundation: CNS-0435505 (NeTS-NOSS), CNS-0347621 (CAREER), CCF-0430061, and CNS-0325875 (ITR). The author would like to acknowledge the input of several colleagues and students in the original development of the models described in this chapter, including Narayanan Sadagopan, Ramesh Govindan, Sundeep Pattem, Joon Ahn, and Ahmed Helmy. More details regarding these models can be found in the corresponding papers listed as references.

References

[1] Hal R. Varian, "How to Build an Economic Model in your Spare Time," *Passion and Craft: Economists at Work*, Ed. M. Szenberg, University of Michigan Press, 1997.

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

[3] Sundeep Pattem, Bhaskar Krishnamachari, and Ramesh Govindan, "The Impact of Spatial Correlation on Routing with Compression in Wireless Sensor Networks," *ACM/IEEE International Symposium on Information Processing in Sensor Networks (IPSN)*, April 26-27, Berkeley, CA 2004.

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