Maximizing Data Extraction in Energy-Limited Sensor Networks

Narayanan Sadagopan[*] and Bhaskar Krishnamachari[*†]
[*]Department of Computer Science
[†]Department of Computer Engineering
University of Southern California,
Los Angeles, 90089
{narayans, bkrishna}@usc.edu

***Abstract*— We examine the problem of maximizing data collection from an energy-limited store-and-extract wireless sensor network, which is analogous to the maximum lifetime problem of interest in continuous data-gathering sensor networks. One significant difference is that this problem requires attention to "data-awareness" in addition to "energy-awareness." We formulate the maximum data extraction problem as a linear program and present a $1+\omega$ iterative approximation algorithm for it. As a practical distributed implementation we develop a faster greedy heuristic for this problem that uses an exponential metric based on the approximation algorithm. We then show through simulation results that the greedy heuristic incorporating this exponential metric performs near-optimally (within 1 to 20% of optimal, with low overhead) and significantly better than other shortest-path routing approaches, particularly when nodes are heterogeneous in their energy and data availability.**

Keywords: Sensor Networks, Mathematical Programming/Optimization, Network Flows.

I. Introduction

In many sensor network applications involving environmental monitoring in remote locations, planetary exploration and military surveillance, it is neither necessary nor even possible for a user to obtain data from the network continuously, in real time. In such applications, the information from the entire network can be extracted *en masse* after a prolonged period of sensing and local storage. This process of extraction can be done by a stationary sink or a mobile sink (robot). However, since communication is often the most expensive operation for a sensor node, the limited batteries may make it impossible to collect all the data stored in the network. We examine the problem of maximizing the data extracted (by a stationary sink) from such an energy-limited sensor network consisting of heterogeneous nodes. The maximum data extraction problem is an analog of the maximum lifetime problem of interest in continuous data-gathering sensor networks [3], [5], [14] that has been studied previously. However, this problem introduces an additional element of "data-awareness" that must be considered in addition to "energy-awareness."

We first show how the maximum data extraction problem can be formulated as a Linear Program. We then adapt and extend techniques for multi-commodity flow problems first developed by Garg and Koenemann [1] to develop an iterative algorithm for our problem with a provable $(1+\omega)$ approximation. This algorithm suggests a new link metric (involving the remaining energies of both sender and receiver nodes, the distance between them, and the data level at the sender) that we then used to develop a fast, practically implementable, distributed heuristic that we refer to E-MAX. The heuristic employs a selfish strategy in that each sensor gives priority to transmitting its data before relaying that of other nodes. Our simulations show that this sophisticated heuristic offers near-optimal performance under all conditions, and significantly better compared to other naive greedy solutions such as shortest-hop-count and shortest-distance routing, particularly when nodes are heterogeneous in their energy and data availability.

The rest of the paper is organized as follows. In section II, we discuss related work to place our contributions in context. We define the problem and present the LP formulation and its dual in section III. An interpretation of the LP dual suggests the $1+\omega$ iterative approximation algorithm that we present and analyze in section IV. We discuss the implementation of this algorithm in section V. We present fast implementable heuristics in section VI. Simulation results comparing these implementations are presented in section VII. Concluding comments are provided in section VIII.

II. Related Work

Our work is inspired by a vast body of literature related to optimizing the performance of ad hoc and sensor networks. We outline a few of these studies that are very close in spirit to our work.

A. Energy Aware Routing

Most of the literature in this area has focused on routing techniques that extend the life time of a sensor or ad hoc network by taking into account remaining battery energy. In [8], Toh has proposed the Conditional Max-Min Battery Capacity Routing (CMMBCR) which selects the shortest path for routing data from one node to the other in an ad hoc network such that all nodes on the path have remaining battery power above a certain threshold. Singh *et al.* [7] present an elaborate study of 5 different metrics which are all a function of the node battery power and conclude that these metrics can give significant energy savings over naive hop-count-based metrics. In [11], Kar *et al.* propose an online algorithm for

0-7803-8356-7/04/$20.00 (C) 2004 IEEE

routing messages in an ad hoc network , also based on the remaining battery energy of a node.

Energy efficient routing techniques have also been proposed in several studies on sensor networks. Heinzelman *et al.* propose a family of adaptive protocols called SPIN for energy efficient dissemination of information throughout the sensor network [6]. In [12], Heinzelman *et al.* propose LEACH, a scalable adaptive clustering protocol in which nodes are organized into clusters and system lifetime is extended by randomly choosing the cluster-heads. Lindsey, *et al.* propose an alternative data gathering scheme called PEGASIS in [13], in which nodes organize themselves in chains, also with rotating elections, for communicating data. Lindsey, *et al* [4] study data gathering schemes that explore the trade-off between energy consumed and delay incurred.

The problem of maximum data extraction introduces ”data-awareness” as an important factor for routing data in addition to ”energy-awareness” as we discuss in section II-B. This problem can also be formulated as a multi-commodity flow problem. There is a vast literature on algorithms for multi-commodity flow problems and their application to networking. Hence we next discuss studies in this area which are relevant to our work.

B. Multi-commodity Flow Algorithms

The multi-commodity flow problem is of great practical importance and theoretical interest. The problem deals with finding a routing scheme to maximize the total quantity of several different commodities (each possibly having different sources and sinks) sent over a network with restricted capacity.

Maximizing the lifetime of a sensor network can be formulated as a multi-commodity flow problem. In [9] Chang *et al.* also use the multi-commodity flow formulation for maximizing the lifetime of an ad hoc network. They propose a class of flow augmentation and flow redirection algorithms that balance the energy consumption rates across nodes based on the remaining battery power of these nodes. This approach seems to considerably increase the network lifetime. Bhardwaj and Chandrakasan [14] examine feasible role assignments (FRA) of nodes as a means of maximizing the lifetime of aggregating as well as non-aggregating sensor networks, and also make use of linear programs based on network flows. Kalpakis *et al.* examine the MLDA (Maximum Lifetime Data Aggregation) problem and the MLDR (Maximum Lifetime Data Routing) problem in [5], again formulating it as an LP using multi-commodity network flows. They observe that as the network size increases, solving the LP takes considerable time and propose some clustering heuristics to achieve near-optimal performance.

As the size of the LP increases, it becomes desirable to solve this problem approximately but quickly. Garg, *et al.* present an excellent discussion of the current fast approximation techniques for solving the multi-commodity flow problem [1]. They propose a simple polynomial time iterative algorithm that gives a $(1 + \epsilon)$ approximation to the multi-commodity flow problem and some other fractional packing problems. The algorithm associates a length with each edge. In each iteration, the algorithm routes flow over the shortest path. After routing the flow, the algorithm increases the length of all the edges along the shortest path. This is done so that subsequent flow may be routed over an under-utilized path. This process continues till the shortest path (using the length metric) exceeds 1. However, at this point, it is possible that some links might be over-utilized, i.e. in excess of their capacity. The algorithm then scales down all the flows by a factor of the maximum over-utilization. This is a beautiful algorithm that explains the nature of the routing scheme needed to maximize the flow in a multi-commodity flow problem. The algorithm as stated needs some modifications for it to be applied to an ad hoc network context. While Chang *et al.* in [3] have also previously applied the Garg-Koenemann algorithm to ad-hoc network lifetime maximization, there are significant difference between [3] and our work that we describe below.

In this paper, we modify and extend the application of the Garg-Koenemann algorithm to the problem of maximizing data collection in store-and-extract sensor networks. Receptions are assumed to consume battery power, unlike [3]. Also, the above studies in the multi-commodity flow problem do not restrict the amount of flow of each commodity [1] [3] [9] [5]. With unrestricted flows, the solution for the maximum data extraction problem would be trivial in that nodes near the sink would monopolize the entire flow. We are interested in the case where sensors generate a finite amount of data i.e. the flow of each commodity is finite. Thus as mentioned at the end of section II-A, the maximum data-extraction problem introduces the data availability at each node as an important routing concern, in addition to the “energy-awareness” previously discussed in the literature.

Besides developing an LP formulation for the maximum data-extraction problem and extending the application and analysis of the Garg-Koenemann approximation algorithm for it, we also present a practical distributed heuristic based on this algorithm that is shown to outperform other shortest-path routing approaches, particularly in heterogeneous conditions where nodes have varying data and energy availability.

III. Problem Definition

Consider a scenario where several sensors that are deployed in a remote region have completed their sensing task and have some locally stored data. We are interested in collecting the maximum amount of data possible from all these sensors at a sink node T, given some remaining energy constraints in each of these sensors.

Figure 1 shows a sample scenario. Each node i is labelled with its (x,y) coordinates, its available data and remaining energy. The goal is to extract all this data to the sink node. The arrows and the indicated flows on each indicate the optimal solution for this particular example obtained by using the LP we describe in the next section.

We now present the formal model for the problem.

A. Model

Let N be the total number of sensors. Let T be the target (sink) to which the data is to be sent. Let D^j_{max} be the amount of data (bytes) collected by sensor j, where $D^j_{max} > 0$. e^j_{max} is the residual energy of sensor j. These sensors are arbitrarily

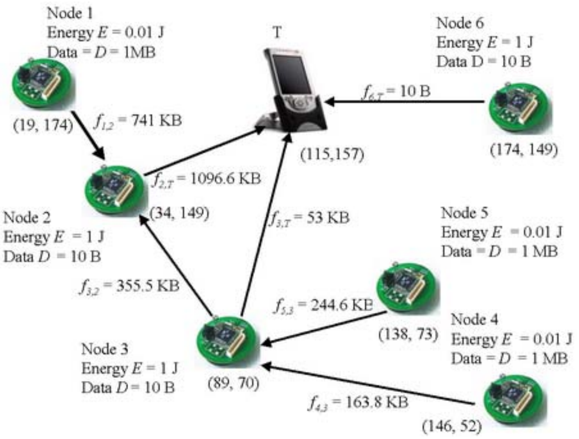

Fig. 1. Illustration of a sample scenario with optimal solution to the maximum data extraction problem. Solution assumes $\beta_t = 32.4\mu J/byte/km^2$ and $\beta_r = 400nJ/byte$

deployed in a region. d_{ij} is the Euclidean distance between sensors i and j. A sensor i can communicate with any sensor j which is within the communication range R from it. Thus, this communicating range overlays a connectivity graph $G = (V, E)$, where $|V| = N$. An edge $(i, j) \in E$ iff $d_{ij} \leq R$.

The energy consumed in transmitting a unit byte from one sensor to the other depends on the distance between them.Tx$_{ij}$ is the energy consumed in transmitting a single byte from sensor i to j. Tx$_{ij}$ is assumed to be proportional to the d_{ij}^2 i.e. Tx$_{ij} = \beta_t d_{ij}^2$, where $\beta_t > 0$. R_{ij} is the energy consumed at sensor j for receiving a single byte of data from sensor i. R_{ij} is assumed to be independent of the distance d_{ij}. Let $R_{ij} = \beta_r$, where $\beta_r > 0$.

For the ease of modelling, we add a fictitious source S, such that there exists an edge from S to every other node in V, except T. Also, add an edge from T to S. Let this new graph be G'. Thus, $G' = (V', E')$, where $V' = V \cup S$ and $E' = E \cup \{(S, i)\} \cup \{(T, S)\}$, where $i \in V$, $i \neq T$. As will be shown later in section III-B, the location of the fictitious source S can be arbitrary.

The problem of collecting the maximum possible data from these energy constrained sensor nodes can be formulated as a multi-commodity flow problem. As shown later in sections III-B and III-C, the addition of S and its associated links simplifies the LP formulation for the multi-commodity flow problem and our analysis.

B. Linear Program (LP) Formulation

In this section, we formulate the maximum data collection problem as an LP. The constraints of the LP are

1) *Flow Conservation:* The amount of data transmitted by a node is equal to sum of the amount of data received by the node and the amount of data generated by the node itself.
2) *Energy Constraint:* The amount of data received and transmitted by a node is limited by the energy of the node. However, there are no energy constraints for the fictitious source S and the sink T.

Let $f_{i,j}$ be the amount of data transmitted from node i to node j. The LP can now be formulated as follows:

$$\text{Maximize } \; f_{T,S} \; \text{ such that}$$

$$\begin{aligned} \sum_{j=1}^{N} f_{j,i} - \sum_{j=1}^{N} f_{i,j} &= 0 \\ i \neq S,T : \beta \sum_{(i,j)\in E} f_{i,j} d_{i,j}^2 + \sum_{(j,i)\in E} f_{j,i} &\leq e^i \\ i \neq S,T : f_{S,i} &\leq D_{max}^i \\ (i,j) \in E' : f_{i,j} &\geq 0 \end{aligned} \tag{1}$$

where

$$\beta = \frac{\beta_t}{\beta_r} \tag{2}$$

$$i \neq S,T : e_i = \frac{e_{max}^i}{\beta_r} \tag{3}$$

Note that we have normalized the energy in terms of receptions i.e. each reception costs a unit of energy, while each transmission from (i, j) costs βd_{ij}^2 units, where $\beta \geq 1$.

By adding the fictitious source S and its associated links, we have made the following transformations to the multi-commodity flow problem:

1) The amount of data transmitted by S to any node i is equal to the amount of data D_{max}^i generated at node i. The reception of this data from S does not incur any energy consumption. Hence, as we mentioned in section III-A, the placement of S does not affect the solution.
2) Note that since the flow conservation is satisfied by both the fictitious source S and the target T, all the data received by the target will be transmitted to the source S, which in turn will be equal to the amount of data transmitted by the source S.
3) The problem now becomes one of maximizing the circulation of the commodities from S to T and back. The advantage of this will become apparent in section III-C.

The above LP can be solved to compute the maximum amount of data that can be collected from the N sensors. It would also give the amount of data (flow) that should be sent from sensor i to sensor j.

However, in this study we are interested in proposing a constructive algorithm that maximizes the amount of data collected. For this purpose, we attempt to understand the structure of the primal LP solution by examining the dual LP in section III-C

C. Dual LP Formulation

The dual of the LP is as follows:

$$\begin{array}{rcl} \text{Minimize} \quad \sum_{i\neq S,T} b^i e^i + c^i D^i_{max} & & \\ i \neq S,T,(i,j)\in E : a^j - a^i + \beta b^i d^2_{ij} + b^j & \geq & 0 \\ a^S - a^T & \geq & 1 \\ i \neq T : a^i - a^S + c^i & \geq & 0 \\ i \neq S,T : b^i & \geq & 0 \\ i \neq S,T : c^i & \geq & 0 \end{array} \tag{4}$$

Let a, b and c be vectors such that their i'th element is denoted by a^i, b^i and c^i respectively. Let

$$A(b,c) = \sum_{i\neq S,T} b^i e^i + c^i D^i_{max} \tag{5}$$

i.e. $A(b,c)$ is the objective function of the dual LP. The above dual has the following interesting interpretation:
Let $l(b,c)$ be a length metric and let $l_{i,j}(b,c)$ be the length of edge (i,j) in this metric. Then, if

$$l_{i,j}(b,c) \;=\; \begin{cases} \beta b^i d^2_{ij} + b^j & \text{if} \quad i \neq S \\ c^i & \text{if} \quad i = S, j \neq T \end{cases} \tag{6}$$

The dual LP can be re-written as follows:

$$\begin{array}{rcl} \text{Minimize} \quad A(b,c) \quad \text{such that} & & \\ i \neq T,S : l_{i,j}(b,c) & \geq & a^i - a^j \\ a^S - a^T & \geq & 1 \\ l_{S,i}(b,c) & \geq & a^S - a^i \end{array} \tag{7}$$

Consider an arbitrary S-T path P. Let P be $S, i_1, i_2, ...i_k, T$. Now, the length of path P can be written as follows:

$$\begin{array}{rcl} l(P) & = & l_{S,i_i}(b,c) + l_{i_k,T}(b,c) + \sum_{z=1}^{k-1} l_{i_z,i_{z+1}}(b,c) \\ & \geq & (a_S - a_{i_1}) + (a_{i_k} - a_T) + a_{i_1} - a_{i_k} \\ & \geq & a_S - a_T \\ & \geq & 1 \end{array} \tag{8}$$

Thus, the length of any arbitrary S-T path in the $l(b,c)$ metric is greater than or equal to 1, which implies that the length of the shortest path in the $l(b,c)$ metric should also be greater than equal to 1. Thus, by transforming the primal LP in section III-B into a circulation problem, we get a very simple value i.e. 1 for the length of the shortest path in the $l(b,c)$ metric.

Let $P(b,c)$ be the shortest path in the metric $l(b,c)$ and $\alpha(b,c)$ be the length of this shortest path. Thus, the objective of the dual LP is to minimize $A(b,c)$ such that $\alpha(b,c) \geq 1$. This is equivalent to minimizing $\frac{A(b,c)}{\alpha(b,c)}$.

Let

$$\zeta \;=\; \text{Min}\{\frac{A(b,c)}{\alpha(b,c)}\} \tag{9}$$

The above interpretation of the dual LP leads to a simple algorithm which approximates the optimal value of the primal. We specify the algorithm and analyze it in section IV.

IV. Approximation Algorithm

We propose an iterative algorithm which is a modification of the Garg-Koenemann algorithm [1].

Before proposing the algorithm, we introduce a few notations. Let

- b^i_j be the value of b^i in the j'th iteration.Initially, $b_0 = c_0 = \delta$ i.e. $\forall i : i \in V' : b^i_0 = c^i_0 = \delta$. The choice of δ is discussed in section IV-B.
- f_j be the total T-S (or S-T) flow till the j'th iteration. Initially, this flow is zero i.e. $f_0 = 0$.
- $A(j) = A(b_j, c_j)$ be the value of the dual objective function after iteration j.
- $P_j = S, i_1, i_2, ...i_k, T$ be the shortest S-T path in the $l(b_j, c_j)$ metric (at iteration j)[1]. On P_j, let $i_0 = S$ and $i_{k+1} = T$.
- $\alpha(j) = \alpha(b_j, c_j)$ be the length of the shortest S-T path after iteration j.

Moreover, $\forall i \in V'$, node i has a capacity associated with it. The capacity of a node depends on where it appears in an S-T path. Hence, we define the capacity of a node with respect to some path $P = S, i_1, i_2, ...i_k, T$ in which it appears. The capacity $\kappa(i)$ of each node in this path is modelled as follows:

$$\begin{array}{rcl} \kappa(S) & = & D^{i_1}_{max} \\ \kappa(i_1) & = & \frac{e^{i_1}_{max}}{\beta d^2_{i_1,i_2}} \\ \kappa(i_z) & = & \{\frac{e^{i_z}_{max}}{1 + \beta d^2_{i_z,i_{z+1}}}\} \quad \text{for} 1 < z \leq k \end{array} \tag{10}$$

The algorithm is as follows:

```
1) j = 0
2) while($\alpha(j) < 1$) do
    a) Select the shortest S-T path $P_j$.
    b) Route $c$ units of data along this path, where $c$ is given as follows:
           $c \;=\; \text{Min}_{0 \leq z \leq k}\{\kappa(i_z)\}$
       i.e. “saturate” the shortest path.
    c) Update the vectors b and c as follows:
           For $1 < z \leq k$,
               $b^{i_z}_j \;=\; b^{i_z}_{j-1}\{1 + \frac{\epsilon c \beta d^2_{i_z,i_{z+1}}}{e^{i_z}} + \frac{\epsilon c}{e^{i_z}}\}$
           For $z = 1$,
               $c^{i_1}_j \;=\; c^{i_1}_{j-1}\{1 + \frac{\epsilon c}{D^{i_1}_{max}}\}$
               $b^{i_1}_j \;=\; b^{i_1}_{j-1}\{1 + \frac{\epsilon c \beta d^2_{i_1,i_2}}{e^{i_1}}\}$
    d) $f_j = f_{j-1} + c$
    e) $j = j + 1$
```

[1] The number of nodes along the shortest S-T path changes from iteration to iteration i.e. it should be k_j. However for notational convenience, we drop the subscript j.

On termination, f_j gives the value of the total data collected. While choosing the shortest S-T path, the algorithm accounts for the battery utilization (fraction of the battery power utilized) as well as the data utilization (fraction of the data sent) at each sensor. Thus, the algorithm, as mentioned earlier in section II is both "energy-aware" and "data-aware".

We analyze the approximation ratio of the algorithm in section IV-A.

A. Analysis

The objective function of the dual LP at the beginning of iteration j is given as follows:

$$\begin{aligned} A(j) &= \sum_{i\neq S,T} b^i_j e^i + c^i_j D^i_{max} \\ &= \sum_{i\neq S,T} b^i_{j-1} e^i + c^i_{j-1} D^i_{max} + \\ &\quad \epsilon c\{c^{i_1}_{j-1} + b^{i_1}_{j-1}\beta d^2_{i_1,i_2} + \\ &\quad \sum_{(i_k,i_{k+1})\in P_{j-1}}^{k\neq 1} b^i_{j-1}(1+\beta d^2_{i_k,i_{k+1}})\} \\ A(j) &= A(j-1) + \epsilon(f_j - f_{j-1})\alpha(j-1) \qquad (11) \end{aligned}$$

Solving the above recurrence, we get

$$A(j) = A(0) + \epsilon\sum_{l=1}^{j}(f_l - f_{l-1})\alpha(l-1) \qquad (12)$$

If each element of b_j and c_j is decreased by $b_0 = c_0 = \delta$, then the objective function of the dual LP is given as follows:

$$\begin{aligned} &\text{Let } x = A(b_j - b_0, c_j - c_0) \\ &= \sum_{i\neq S,T}(b^i_j - b^i_0)e^i + (c^i_j - c^i_0)D^i_{max} \\ &= \sum_{i\neq S,T} b^i_j e^i + c^i_j D^i_{max} - \sum_{i\neq S,T} b^i_0 e^i + c^i_0 D^i_{max} \\ &= A(j) - A(0) \end{aligned}$$

$$\text{i.e. } A(b_j - b_0, c_j - c_0) = A(j) - A(0) \qquad (13)$$

Similarly, if each element of b_j and c_j is decreased by $b_0 = c_0 = \delta$, then the length of the shortest path can be computed as follows:

The shortest path P_j of length α_j is $S, i_1, i_2, ...i_k, T$. Now, let $y = l_{P_j}(b_j - b_0, c_j - c_0)$ be the length of P_j in the metric $l(b_j - b_0, c_j - c_0)$,where $b_0 = c_0 = \delta$. Then

$$\begin{aligned} y &= l_{S,i_1}(b_j - b_0, c_j - c_0) + l_{i_1,i_2}(b_j - b_0, c_j - c_0) + \\ &\quad \sum_{(i_k,i_{k+1})\in P_j}^{k\neq 1} l_{i_k,i_{k+1}}(b_j - b_0, c_j - c_0) \\ &= (c^{i_1}_j - c^{i_1}_0) + (b^{i_1}_j - b^{i_1}_0)\beta d^2_{i_1,i_2} + \\ &\quad \sum_{(i_k,i_{k+1})\in P_j}^{k\neq 1} (b^{i_k}_j - b^{i_k}_0)(\beta d^2_{i_k,i_{k+1}} + 1) \\ &= (c^{i_1}_j - \delta) + (b^{i_1}_j - \delta)\beta d^2_{i_1,i_2} + \\ &\quad \sum_{(i_k,i_{k+1})\in P_j}^{k\neq 1} (b^{i_k}_j - \delta)(\beta d^2_{i_k,i_{k+1}} + 1) \\ &= c^{i_1}_j + \beta b^{i_1}_j d^2_{i_1,i_2} + \\ &\quad \sum_{(i_k,i_{k+1})\in P_j}^{k\neq 1} b^{i_k}_j(\beta d^2_{i_k,i_{k+1}} + 1) - \\ &\quad \delta\{1 + \beta d^2_{i_1,i_2} + \\ &\quad \sum_{(i_k,i_{k+1})\in P_j}^{k\neq 1} (\beta d^2_{i_k,i_{k+1}} + 1)\} \\ &= \alpha(j) - \delta\{1 + \beta d^2_{i_1,i_2} + \\ &\quad \sum_{(i_k,i_{k+1})\in P_j}^{k\neq 1} (\beta d^2_{i_k,i_{k+1}} + 1)\} \end{aligned}$$

Thus,

$$\alpha(b_j - b_0, c_j - c_0) \geq \alpha(j) - \delta L \qquad (14)$$

where L is the largest value of

$$1 + \beta d^2_{i_1,i_2} + \sum_{(i_k,i_{k+1})\in P_j}^{k\neq 1} (\beta d^2_{i_k,i_{k+1}} + 1)$$

Now, from Eqn. 9 in section III-C

$$\begin{aligned} \zeta &\leq \frac{A(b_j - b_0, c_j - c_0)}{\alpha(b_j - b_0, c_j - c_0)} \\ &\leq \frac{A(j) - A(0)}{\alpha(j) - \delta L} \\ \alpha(j) &\leq \delta L + \frac{\epsilon}{\zeta}\sum_{i=1}^{j}(f_i - f_{i-1})\alpha(i-1) \qquad (15) \end{aligned}$$

Now, in order to get an upper bound on $\alpha(j)$, we let all the $\alpha(i)$'s, where $1 \leq i \leq j-1$ to be as large as possible i.e. $\forall i : 1 \leq i \leq j-1$,

$$\alpha(i) = \delta L + \frac{\epsilon}{\zeta}\sum_{l=1}^{i}(f_l - f_{l-1})\alpha(l-1) \qquad (16)$$

Thus,

$$\begin{aligned} \alpha(j) &\leq \alpha(j-1) + \frac{\epsilon}{\zeta}(f_j - f_{j-1})\alpha(j-1) \\ &\leq \alpha(j-1)(1 + \frac{\epsilon}{\zeta}(f_j - f_{j-1})) \\ &\leq \alpha(j-1)e^{\frac{\epsilon}{\zeta}(f_j - f_{j-1})} \\ \alpha(j) &\leq \alpha(0)e^{\frac{\epsilon}{\zeta}f_j} \\ \alpha(j) &\leq \delta L e^{\frac{\epsilon}{\zeta}f_j} \qquad (17) \end{aligned}$$

Thus, at iteration t, when the shortest S-T path has length greater than or equal to 1, the following inequality holds

$$\begin{aligned} 1 \leq \alpha(t) &\leq \delta L e^{\frac{\epsilon}{\zeta}f_t} \\ \frac{\zeta}{f_t} &\leq \frac{\epsilon}{ln(\frac{1}{\delta L})} \qquad (18) \end{aligned}$$

Thus, despite the modifications to the Garg-Koenemann algorithm to adapt it to our problem, we see that the bound obtained in the above inequality is exactly the same as that obtained in [1].

Although the algorithm chooses the shortest path based on data and energy utilization of nodes along the path, neither the remaining energy nor the remaining data at each sensor is actively monitored. Hence, it would be interesting to see if the flow f_t is feasible. We analyze the feasibility of the flow in section IV-B.

B. Scaling

Whenever, b^i of a node i has increased, the node has been on a path whose length is strictly less than 1. If this had not been the case, the shortest S-T path had a length greater than 1 and the algorithm should have terminated. Moreover, the b^i is increased by a factor of at most $(1+\epsilon)$. Thus $b_t^i < (1+\epsilon)$. Now, if b^i is increased x times during the execution of the algorithm, in the worst case all these x increase operations completely utilize the energy e^i of node i. Thus, in the worst case, $b_t^i = \delta(1+\epsilon)^x < 1+\epsilon$, which implies that $x < log_{1+\epsilon}(\frac{1+\epsilon}{\delta})$. The factor by which the node i is over-utilized is i is x. Thus by scaling f_t down by a factor of $log_{1+\epsilon}(\frac{1+\epsilon}{\delta})$, we can ensure that the energy constraint is not violated.

If b^i is increased x times, c^i is also increased at most x times. Thus, it can be seen that even the data constraint is violated by at most a factor of $log_{1+\epsilon}(\frac{1+\epsilon}{\delta})$. Thus, a single scaling operation ensures that both the energy and the data constraints are satisfied.

Thus, there exists a feasible flow of value $\frac{f_t}{log_{1+\epsilon}(\frac{1+\epsilon}{\delta})}$.

Similar to [1], the ratio of the values of the dual to the primal solutions is

$$\gamma = \frac{\zeta}{f_t} log_{1+\epsilon} \frac{1+\epsilon}{\delta}$$

As shown in [1], if

$$\delta = (1+\epsilon)((1+\epsilon)L)^{-\frac{1}{\epsilon}} \tag{19}$$

$$\gamma \leq (1-\epsilon)^{-2} \tag{20}$$

Thus to get within a factor of $(1+\omega)$ of the optimal, ϵ is given as follows:

$$\epsilon \leq \sqrt{\frac{1}{1+\omega}} \tag{21}$$

Thus, the algorithm mentioned in section IV can solve the multi-commodity flow problem and produce results arbitrarily close to the optimum.

Number of Iterations:

From the analysis in this section, we observe that the factor by which a node i's energy is over-utilized is at most $log_{1+\epsilon}\frac{1+\epsilon}{\delta}$. Thus, the number of iterations in which i is the bottleneck on the shortest path is at most $log_{1+\epsilon}\frac{1+\epsilon}{\delta}$. There are a total of N nodes, and thus, the total number of iterations is $O(Nlog_{1+\epsilon}\frac{1+\epsilon}{\delta})$.

In the next section, we discuss a few issues which might be important for the implementation of the approximation algorithm in a sensor network.

V. Implementation of Approximation Algorithm

We now describe an iterative implementation of the algorithm mentioned in section IV. We refer to this implementation as A-MAX. In this implementation, the edge length metric is the one used by the approximation algorithm. The energy of each node i is initialized to e^i, while the data of each node is initialized to D_{max}^i. b_i^0 and c_i^0 are initialized to δ.

In each iteration the sink chooses a node with the shortest path to it as a *candidate* nodes. A sensor node is called *active* if it has been chosen by the sink as a *candidate* node at any iteration. Sensor nodes that are 1 hop away from the *active* nodes are called *threshold* nodes. Initially only the sink is *active*, while all the nodes within a distance R from the sink are *threshold* nodes. Each iteration k consists of the following steps:

1) The sink sends a message containing the iteration number k to its neighbors. It also advertises its shortest path to the sink having length 0.
2) Each neighbor i initializes its shortest path to the sink i.e. p_k^i to $\beta b_k^i d_{i,sink}^2$.
3) The *active* and the *threshold* nodes execute a distance vector algorithm using $\beta b_k^i d_{ij}^2 + b_k^j$ as the length of edge (i,j). Updates from downstream nodes are ignored to avoid loops. After the algorithm terminates, each of these nodes has the length of the shortest path and the next hop to the sink.
4) Each node i that is either a *active* or a *threshold* node sends a response towards the sink. This response contains the value of $p_k^i + c_k^i$. The response is also used to find the capacity of the path along which it is forwarded. The capacity is found as per Eqn. 10 in section IV. The forwarding process sets up a reverse path state in all the nodes along the path.
5) The sink selects the node z with the minimum p_i^k as a *candidate* node. If p_z^k is greater than or equal to 1, the sink sends a message to all active nodes to scale down their flows by the scaling factor mentioned in section IV-B and then send their data towards the sink. In this case the algorithm terminates.
6) If p_z^k is less than 1, the sink sends a message addressed to the node z containing c i.e. the minimum capacity along the path. The message is forwarded based on the state created in an earlier step. As the message is being forwarded back along the reverse path, each node sets z to be downstream of itself. This is used to avoid loops. For each node i_1 along the path, the value of $b_{k+1}^{i_1}$ is updated using the value of $b_k^{i_1}$ according to the algorithm.
7) The node z updates its routing table to augment the flow to its current next hop to the sink by c. It also updates c_{k+1}^z using c_k^z according to the algorithm.
8) Go to step 1.

Message Complexity:

In each iteration, in the worst case all the N nodes will participate in the distance vector algorithm. Thus, from standard analysis of distributed Bellman Ford algorithm, the number of messages exchanged due to the distance vector algorithm in each iteration is $O(N|E|)$, where $|E|$ is the number of edges in the graph G [15]. For step 4, the response of each *active* or *threshold* node would be forwarded $O(N)$ times. Since, there are a total of N nodes, the total control overhead in step 4 is $O(N^2)$. Steps $5-6$ in the implementation require $O(E)$ messages in each iteration. Since $|E| = O(N^2)$, for each iteration, the total message complexity of A-MAX is $O(N^3)$.

As shown in section IV-B, the number of iterations is $O(N \log_{1+\epsilon} \frac{1+\epsilon}{\delta})$, and hence the total message complexity of A-MAX is $O(N^4 log_{1+\epsilon} \frac{1+\epsilon}{\delta})$.

A. Parameter Settings and Implementation Concerns

In A-MAX, each node i initializes $b_0^i = c_0^i = \delta$. As shown in Eqn. 20 in section IV-B, δ is dependent on ϵ and L. ϵ depends on the $(1+\omega)$ approximation needed as shown in Eqn. 21 in section IV-B. As shown in section IV-A, L is the largest value of $\{1 + \beta d_{i_1,i_2}^2 + \sum_{(i_k,i_{k+1}) \in P_j}^{k \neq 1} (\beta d_{i_k,i_{k+1}}^2 + 1)\}$. Thus, L depends on the topology of the network. However, it is possible to get a loose bound on L given as follows:

$$L \quad \leq \quad (N-1)\{\beta R^2 + 1\} \tag{22}$$

i.e. L is maximum when all the nodes are arranged in a linear fashion and just within range of each other.

In simulating this implementation however, we find that this loose bound on L results in extremely small values of δ (in fact so close to 0 that it has to be manually set to an arbitrary small positive value to make it work). As a result, A-MAX takes a large number of iterations to converge for small values of ω. Hence, as we shall see in the simulations section VII, A-MAX may not be suitable for practical implementations. This motivated us to look for the fast, practical heuristics described in the next section.

VI. Fast Greedy Heuristics

A. Motivation

We can observe that the implementation of the approximation algorithm A-MAX consists of 2 phases. First, there is a *negotiation phase*, in which nodes do not actually transmit data but try to make decisions about how much data to send and how much data they will relay for other nodes. Since neither the remaining energy nor the remaining data at a sensor is actively monitored at the negotiations phase, some nodes may be over-committed (i.e. may have negotiated to send or receive more data than allowed by their energy and data constraints). In the second, *data transmission phase*, each node scales down the data that it has to send or receive from each of its neighbors to accommodate the constraints and only then starts the data transmission towards the sink (T). This cumbersome two-phase process is one reason A-MAX is inefficient in implementations.

Another observation that can be made about the maximum data extraction problem is that the reception costs make it expensive for nodes to relay other nodes' data. As a result, one of the characteristics of the optimal solution is that each node should only commit to relay another nodes' data if it has energy remaining after transmitting all its own data - in other words, it behaves selfishly.

Motivated by these observations, we seek to develop efficient heuristics that avoid the 2-phase overshoot and scale down iterations of A-MAX and incorporate selfish, greedy behavior. It turns out that these heuristics are much faster in implementation and with the right metric can perform very well in practice.

B. Description of Heuristics

The key features of the heuristics we develop are:

1) Each sensor is greedy. i.e if a sensor has the shortest path to the sink, it accords priority to sending its data first. If after sending its data, the sensor exhausts its battery, it disconnects itself from the network.
2) There is no scaling operation.
3) Each sensor actively monitors its remaining battery power and remaining data.

We explore three variants of the greedy strategy. Each variant differs from the other in the link metric it chooses for distance vector routing.

Exponential Metric: This metric is based on the algorithm mentioned in section IV. At iteration k, if a sensor i receives or sends data, it updates its b^i as follows:

$$b_k^i \quad = \quad b_{k-1}^i \{e^{\epsilon \lambda_1^i}\} \tag{23}$$

where λ_1^i is the current battery utilization of sensor i i.e. the ratio of the battery power spent (till and including iteration k) to the total battery power. This update is approximately the same as that mentioned in section IV, when $\epsilon < 1$ (as $1 + x \approx e^x$, for small x). If the data sent is its own data, the sensor i also updates its c^i in the same manner:

$$c_k^i \quad = \quad c_{k-1}^i \{e^{\epsilon \lambda_2^i}\} \tag{24}$$

where λ_2^i is the current data utilization of sensor i i.e. the ratio of quantity of its data sent (till and including iteration k) to its total data quantity.

However, the initial values of b^i's and c^i's are still to be chosen. Intuitively, if all sensors have the same value for the ratio of $\frac{D_{max}^i}{e^i}$, all these sensors can be treated the same. However, if a few sensors have a low value for this ratio, which implies that they can carry data from other sensors, it would be advantageous to choose these sensors as the next hop towards the sink. Hence, we set the initial values of b^i's and c^i's are follows:

$$b_0^i \quad = \quad \frac{D_{max}^i}{e^i} \tag{25}$$

$$c_0^i \quad = \quad \frac{D_{max}^i}{e^i} \tag{26}$$

We will refer to the *Exponential Metric* based implementation as E-MAX. For this implementation we assume $\epsilon < 1$.

To illustrate the importance of this metric, we choose 2 other variants of the distance vector implementation. Both these variants do not use b^i's and c^i's in the implementation:

Distance Metric: The length of an edge (i, j) is d_{ij}. This metric is constant across iterations. We refer to this variant of the implementation as DIST-MAX.

Hop Count Metric: The length of an edge (i, j) is 1 iff $d_{ij} \leq R$. This metric is also constant across iterations. We refer to this variant of the implementation as H-MAX.

Number of Iterations:

In all the above greedy heuristics, at each iteration, a node i's energy or data is completely exhausted. Since, there are a total of N nodes, the number of iterations is $O(N)$.

C. Implementation of Heuristics

The implementation mentioned in section V can be easily extended to incorporate the greedy heuristics with the appropriate edge length metrics. However, there are some differences:

1) In E-MAX, each node i initializes b_0^i and c_0^i according to Eqn. 25 and Eqn. 26 respectively. At each iteration k, E-MAX updates the value of b_k^i and c_k^i as per Eqn. 23 and Eqn. 24 respectively. H-MAX and DIST-MAX do not need any specific initializations for these values.
2) Each node i actively monitors its remaining battery and remaining data levels. While calculating the capacity of the path, the remaining battery power and remaining data are used as opposed to e^i and D_{max}^i used in A-MAX. If the battery level of a sensor hits 0, it ceases to be a part of the network and hence does not execute the distance vector algorithm.
3) While sending a response to the sink, each *active* and *threshold* node sends its remaining data in addition to the length of its path to the sink. The sink chooses a *candidate* from those nodes that still have remaining data to send.
4) At each iteration, the *candidate* node immediately sends data to the sink.
5) There is no scaling operation involved. The algorithm terminates if the sink detects that all the nodes[2] in the network do not have data to send or if the sink is disconnected from the network due to its first hop neighbors dying out.

Message Complexity:

For these greedy heuristics, the message complexity at each iteration is the same as that for A-MAX i.e. $O(N^3)$. Since, as shown in section VI-B, the number of iterations is $O(N)$, the total message complexity is $O(N^4)$.

In section VII, we describe our simulation scenarios used to compare the different approaches and the results.

VII. Simulations and Results

We simulated A-MAX, E-MAX, DIST-MAX and H-MAX using a high level simulator. We ignore MAC effects in this simulator.

The simulation set up consists of 50 nodes randomly deployed in an area of 0.5 km x 0.5 km. Each node has a radio range of 0.2 km. We use the first order radio model used in [5]. In this model, a sensor consumes $\epsilon_{elec} = 400nJ/byte$ for running the transmitter or receiver circuitry and $\epsilon_{amp} = 800pJ/byte/m^2$ for running the amplifier circuitry. Thus $\beta_r = \epsilon_{elec}$. In order to send a single byte, the sensor has to run its transmitter and amplifier circuitry. Now, a sensor can receive data only from sensors within its range i.e. within a distance of R. Hence,

$$\beta_t \quad \approx \quad \epsilon_{elec} + \epsilon_{amp} R^2$$

Thus, using the definition of β from Eqn. 2 in section III-B,

$$\beta \quad \approx \quad 1 + \frac{\epsilon_{amp}}{\epsilon_{elec}} R^2$$

Converting distances into km units, for $R = 0.2$ km, we get $\beta = 81$. We use this value of β in our simulations. We assume that the maximum energy is 1J as in [5]. As we mentioned earlier in section III-A, our model assumes that a reception of a single unit(byte) consumes one unit of energy, we normalize the maximum energy in terms of bytes that can be received. From the value of ϵ_{elec}, it can be shown that 1J of energy allows $1250K$ receptions. Thus, in our simulations we set the maximum energy to $1250K$. The maximum data is set to $1000K$ bytes. We also count the number of transmissions and receptions due to exchange of control messages in the various implementations (that are appropriate modifications of A-MAX). These control messages include the distance vector updates (step 3), messages forwarded during the response of an *threshold* node or *active* node to the sink (step 4) and the messages forwarded during the response of the sink to the candidate node (step 5 and 6). These control messages are assumed to be of 1 byte. This is a reasonable assumption as the control messages usually contain a single real valued quantity. Moreover, if i transmits a control message to j, we compute the energy spent in the control message as βd_{ij}^2 for node i's transmission and 1 for node j's reception. This helps us to quantify the overhead of A-MAX, E-MAX, DIST-MAX and H-MAX in terms of the percentage of energy at each node.

As mentioned at the end of section II, the amount of data collected in our problem scenario is mainly dependent of the energy of the first hop sensors and the amount of data they have. Thus, in our simulations we define a few *good* nodes. A node i is a *good* node if it has a very low value of $\frac{D_{max}^i}{e^i}$. These *good* nodes do not have any other special capabilities. For a *good* node i, $e^i = 1250000$ (1J of energy) and $D_{max}^i = 10$ (bytes) of For a node j that is not *good*, $e^i = 1250$ (0.01J of energy) and $D_{max}^i = 1000K = 1M$ (bytes). As we show in our results, when all nodes have the similar values of energy and data, the three heuristic implementations perform very similar and give close to optimal performance. Varying the energy of the *good* nodes, their number and their placement helps us to outline scenarios where E-MAX significantly outperforms DIST-MAX and H-MAX.

A. Objectives

We were interested in studying the performance of A-MAX, E-MAX, DIST-MAX and H-MAX in terms of optimality of solution produced and the overhead. We compared the solutions produced by these implementations with that produced by solving the LP specified in section III-B using *lp_solve*. The objectives of our simulations were to study the following:

1) Effect of δ and ω on the performance of A-MAX.

[2] This might need the sink to have an approximate estimate on the number of nodes in the network.

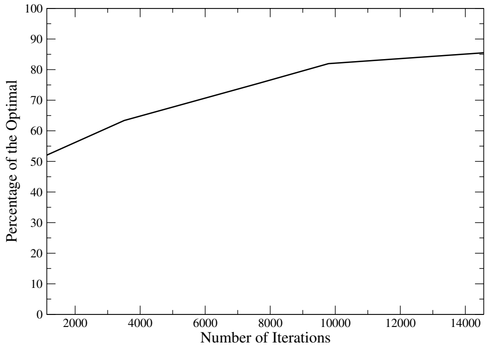

Fig. 2. Effect of number of iterations on the solution produced by A-MAX due to varying δ. Here $\omega = 0.1$. The x-axis is plotted on a log scale

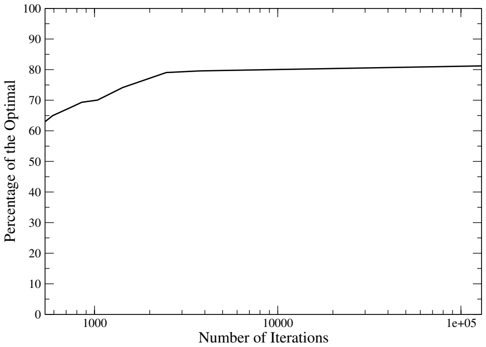

Fig. 3. Effect of number of iterations on the solution produced by A-MAX due to varying ω. Here $\delta = 0.0000001$. The x-axis is in log-scale

2) Effect of *good* nodes on the performance of E-MAX, DIST-MAX and H-MAX.
3) Effect of density on the performance on E-MAX, DIST-MAX and H-MAX.

1) Effect of δ and ω on A-MAX: We discuss the scenario when there are 20 *good* nodes in the network and 4 of them are in range of the sink. However, the conclusions are applicable to most of the scenarios we examined.

For a given ω, ϵ can be easily determined using Eqn. 21 in section IV-B. We set ϵ to the maximum value permitted by Eqn. 21 for a given ω. In our case, if ϵ is very small, due to a very loose bound on L, $\delta \approx 0$. Hence, for the implementation to work correctly, we set δ to small non-zero values. As mentioned in section V-A, the value of δ affects the number of iterations of A-MAX and the optimality of the solution produced.

For the aforementioned scenario, we use the following values of δ: $0.0000001, 0.000001, 0.0001, 0.01$ and 0.1. We set $\omega = 0.1$. We count the number of iterations taken by A-MAX across these values. We notice that as δ decreases, the number of iterations increases.

Figure 2 shows the effect of the number of iterations (by varying δ) on A-MAX. As the number of iterations increases (δ decreases), A-MAX produces solutions that are closer to the optimal.

We also varied ω by fixing $\delta = 0.0000001$. Figure 3 shows the effect of the number of iterations (by varying ω) on the optimality of the solution produced by A-MAX. We used the following values of ω: $0.01, 0.1, 1, 3, 5, 7$ and 10. As ω is increased, the number of iterations decrease. We also observe that at $\omega = 10$, the solution produced by A-MAX is around 65% of the optimal, while at $\omega = 7$, the solution is around 80% of the optimal.

Figures 2 and 3 demonstrate the trade-off between the number of iterations and the quality of solution obtained by A-MAX. In general, for solutions that are arbitrarily close to the optimum, a greater number of iterations are needed. This translates to a higher number of control messages and hence a greater overhead. In all these scenarios, the overhead of A-MAX measured from simulations precludes its implementation in an energy constrained sensor network.

These curves justify the need for fast heuristics that we developed. We evaluate the performance of E-MAX, DIST-MAX and H-MAX next.

2) Effect of good *nodes on E-MAX, DIST-MAX and H-MAX:* Our initial aim was to understand the effect of ϵ on E-MAX. In a sample scenario with 10 good nodes placed at random, we varied ϵ from 0.1 to 1 in steps of 0.2. Across all these scenarios, the solution produced by E-MAX remains unchanged. So, in the rest of our simulations with E-MAX, we arbitrarily set $\epsilon = 0.1$.

From our preliminary study, we observe that if all the nodes in the network have similar amount of energy and data, then all these greedy implementations perform similarly. However, the importance of the exponential metric becomes apparent as we vary the energy, the number and the placement of *good* nodes in the network.

Initially, we choose these good nodes at random. Figure 4 shows the effect of the energy of the *good* nodes. In this scenario, we fix the number of good nodes to be 20. We vary their energies from 0.1 J to 1 J (and appropriate reception units) in steps of 0.2. We observe that the energy of the *good* nodes has a significant impact on the relative performance of E-MAX, DIST-MAX and H-MAX. The performance gap between E-MAX and the other heuristics increases as the energy of *good* nodes increases, all other factors remaining the same. At lower values (around 0.1 J) of energy for *good* nodes, all the 3 heuristics perform similarly and close to optimal. But we would like to see if the significant difference between E-MAX and the other heuristics at high energy values of the *good* nodes is independent of other factors like fraction of *good* nodes and their placement. Hence in the subsequent simulations, we set the energy of the *good* nodes at 1J.

Figure 5 shows the effect of increasing the ratio of the *good* nodes in the network. In these scenarios, E-MAX produces solutions that are within 1-20% of the optimal. It also gives up to 240% improvement (more data collected) over both DIST-MAX and H-MAX. This happens when the fraction of *good* nodes is 0.6. At the extremes of the x-axis, when almost all nodes are similar in their energy and data levels, all the 3 implementations perform equally good and give very close to optimal solutions. From the topology, we were able to deduce that the increase in the number of *good* nodes leads to an increase in the number of *good* nodes close to the sink. While E-MAX is able to take advantage of these nodes, the other

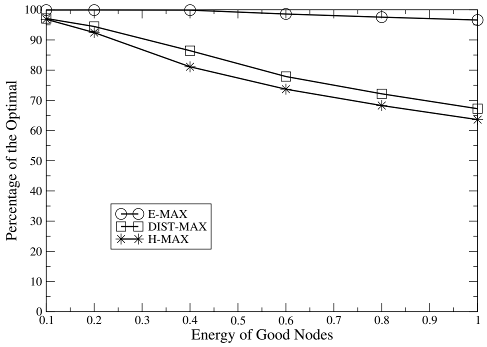

Fig. 4. Effect of the energy of *good* nodes on E-MAX, DIST-MAX and H-MAX.

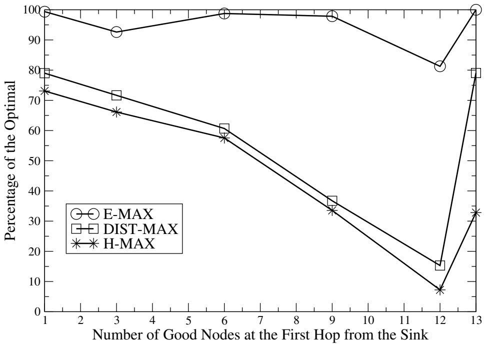

Fig. 6. Effect of placement of *good* nodes on E-MAX, DIST-MAX and H-MAX.

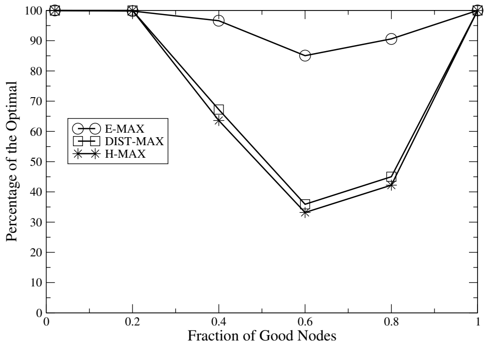

Fig. 5. Effect of fraction of *good* nodes on E-MAX, DIST-MAX and H-MAX. The area of deployment is 0.5 km x 0.5 km

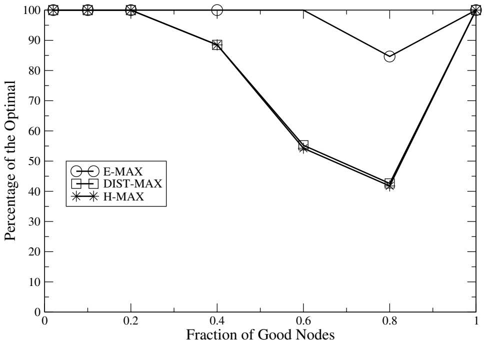

Fig. 7. Effect of fraction of *good* nodes on E-MAX, DIST-MAX and H-MAX. The area of deployment is 1 km x 1 km.

naive heuristics do not. As the fraction of *good* nodes increases beyond 0.8, the amount of data to be collected is far lower than the amount of energy in the network. Hence all the schemes yield close to optimal solutions.

These observations suggest that not only the fraction but also the placement of these *good* nodes relative to the sink affects the relative performance of the 3 greedy implementations.

To understand the effect of the placement of the *good* nodes relative to the sink, we fix the number of *good* nodes to 15 and increase the number of *good* nodes within the range of the sink (in our scenario, the sink had 13 neighbors). These *good* nodes are chosen at random. Intuitively, as the number of *good* nodes within the range of the sink increases, H-MAX and DIST-MAX should perform as good as E-MAX. But figure 6 shows otherwise. This illustrates the importance of the placement of the *good* nodes. If there is a connected "back-bone" of *good* nodes i.e. a series of *good* nodes in range of each other and in range of the sink, E-MAX is able to take advantage of them as opposed to DIST-MAX and H-MAX. This "back-bone" helps in gathering more data. E-MAX is able together around 500% more data than DIST-MAX (when the number of first hop good nodes is 12). The metric used by E-MAX helps to route the data along such "back-bones". Even if there are only $5-6$ *good* nodes (out of 13 neighbors) at the first hop, the high density of the scenario considered leads to the existence of such "back-bones". When the majority of the nodes i.e. 13 (maximum possible in this case) of the 15 are within range of the sink, DIST-MAX and H-MAX start performing better. It would be interesting to examine the existence of such "back-bones" in lower density deployment. Hence, we next vary the size of the deployment area and study the performance of E-MAX, DIST-MAX and H-MAX

3) Effect of Density on E-MAX, DIST-MAX and H-MAX: We randomly deploy the same number of nodes, using the same communication range in a bigger area of 1km x 1km.

The performance trends shown in Figure 7 are very similar to those in Figure 5. Again E-MAX gives solutions that are within 1 to 20% of the optimal solution. E-MAX can also give more than 200% improvement over H-MAX and DIST-MAX. This occurs when the fraction of good nodes is 0.8, while in the higher density case, this occurs at 0.6. This is because in a sparser network, due to nodes being more spread out, a greater fraction of good nodes is needed to create a set of well connected good nodes or the "back-bone".

Across all of the scenarios mentioned in sections VII-A.2 and VII-A.3, E-MAX, H-MAX and DIST-MAX took significantly lesser number of iterations as compared to A-MAX. For example, in the scenario used for the performance of A-MAX in section VII-A.1, E-MAX took only 36 iterations and gives a solution that is 99.9% of the optimum. Such significant reduction in iterations were observed across all scenarios used in our study. This translated to an overhead in the range of 5-10%. This overhead shows that these greedy heuristics are suitable for implementation in an energy constrained sensor network.

When all nodes have similar data and energy levels, all these greedy implementations perform similarly. However, if there are nodes in the network with very high energy and low data, E-MAX (due to the metric based on the Garg Koenemann algorithm) significantly outperforms the other approaches. In some scenarios it results in collecting up to 500% more data than H-MAX and DIST-MAX. In most of the scenarios we used, E-MAX gives flows that are within 15% of the optimum.

VIII. Conclusions

In remote monitoring sensor network applications where data does not need to be gathered continuously, the key problem of interest is that of extracting the maximum information from the local storage of network nodes, post-sensing. We have formulated this problem as a linear program in this paper, and presented and analyzed a $1 + \omega$ iterative approximation algorithm for it. For ease of distributed implementation, we then developed a greedy heuristic that incorporates the link metric suggested by the approximation algorithm. Our simulation results enable us to conclude that this heuristic, the E-MAX algorithm, performs near-optimally (within 1 to 15% in most scenarios) and significantly better than other shortest-path routing approaches.

While the problem formulation we presented does not explicitly incorporate fairness, it could be extended with some modifications to incorporate different priorities or even equal priorities on the data from sensor nodes by varying the data constraints on individual nodes. A thorough examination of fairness issues is one of our areas for future work.

Other future work could involve the incorporation of data aggregation. Existing results pertaining to maximum lifetime problems with aggregation, such as [5] and [14], suggest ways in which our work may be extended in this direction.

IX. Acknowledgements

We thank Prof. Ashish Goel for pointing us towards relevant results on the multi-commodity flow problem. We would also like to thank the anonymous reviewers for their insightful comments.

References

[1] N. Garg and J. Koenemann. Faster and Simpler Algorithms for Multicommodity Flow and Other Fractional Packing Problems. FOCS 1998.

[2] F. Shahrokhi and D. Matula. The Maximum Concurrent Flow Problem. J. ACM, 37(2): 318-334, 1990.

[3] J. H. Chang and L. Tassiulas. Fast Approximate Algorithm for Maximum Lifetime Routing in Wireless Ad-hoc Networks. Networking 2000, vol. 1815 of Lecture Notes in Computer Science, pp. 702-713, Paris, France, May 2000.

[4] S. Lindsey, C. Raghavendra and K. Sivalingam. Data Gathering Algorithms in Sensor Networks Using the Energy*Delay Metric. In Proceedings of the IPDPS Workshop on Issues in Wireless Networks and Mobile Computing, 2001.

[5] K. Kalpakis, K. Dasgupta and P. Namjoshi. Efficient Algorithms for Maximum Lifetime Data Gathering and Aggregation in Wireless Sensor Networks. To appear in the Computer Networks Journal. Also available as UMBC CS TR-02-13,2002.

[6] W. Heinzelman, J. Kulik and H. Balakrishnan. Adaptive Protocols for Information Dissemination in Wireless Sensor Networks. In Proceedings of 5'th ACM/IEEE Mobicom Conference, Seattle, WA, August 1999.

[7] S. Singh, M. Woo and C. S. Raghavendra. Power-Aware Routing in Mobile Ad Hoc Networks. Mobile Computing and Networking. 181-190(1998).

[8] C. Toh. Maximum Battery Life Routing to Support Ubiquitous Mobile Computing in Wireless Ad Hoc Networks. IEEE Communications Magazine, June 2001.

[9] J. Chang and L. Tassiulas. Energy Conserving Routing in Wireless Ad Hoc Networks. IEEE Infocom 2000. pp. 22-31.

[10] Q. Li, J. Aslam and D. Rus. Online Power-Aware Routing in Wireless Ad-Hoc Networks. Mobicom. pp 97-107. 2001.

[11] K. Kar, T. V. Lakshman, M. Kodialam and L. Tassiulas. Online Routing in Energy Constrained Ad Hoc Networks. IEEE Infocom 2003.

[12] W. Heinzelman, A. Chandrakasan and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In Proceedings of the Hawaii Conference on System Sciences, Jan. 2000.

[13] S. Lindsey and C. S. Raghavendra. PEGASIS: Power Efficient GAthering in Sensor Information Systems. ICC 2001.

[14] M. Bhardwaj and A.P. Chandrakasan, Bounding the Lifetime of Sensor Networks Via Optimal Role Assignments. Infocom 2002.

[15] T. H. Cormen, C. E. Leiserson, R. Rivest and C. Stein. Introduction to Algorithms. Second Edition, Mc Graw Hill, 1998.