Mobile Networks and Applications 11, 341–350, 2006

DOI: 10.1007/s11036-006-5187-8

Decentralized Utility-based Sensor Network Design

NARAYANAN SADAGOPAN *
Department of Computer Science, University of Southern California, 941 W. 37th Place, Los Angeles, CA 90089-0781, USA

MITALI SINGH
Department of Computer Science, University of Southern California, 3740 McClintock Ave, EEB 226, Los Angeles, CA 90089-0781, USA

BHASKAR KRISHNAMACHARI
Dept. of Electrical Engineering - Systems, University of Southern California, 3740 McClintock Avenue, EEB 342, Los Angeles, CA 90089, USA

Published online: 28 April 2006

Abstract. Wireless sensor networks consist of energy-constrained sensor nodes operating unattended in highly dynamic environments. In this paper, we advocate a systematic decentralized approach towards the design of such networks based on utility functions. A local utility function is defined for each sensor node in the network. While each sensor node "selfishly" optimizes its own utility, the network as a "whole" converges to a desired global objective. For the purpose of demonstrating our approach, we consider the following two separate case studies for data gathering in sensor networks: (a) construction of a load balanced tree and (b) construction of an energy balanced tree. Our work suggests a significant departure from the existing view of sensor networks as consisting of cooperative nodes, i.e. "selfish"sensor nodes is a useful paradigm for designing efficient distributed algorithms for these networks.

1. Introduction

Wireless sensor networks provide us with an attractive option of being able to "instrument" the physical world. These networks are envisioned to support a wide range of applications including wild life tracking, microclimate monitoring, target tracking, and structural health monitoring. Several real world deployments of these networks are described in [2]. One of the first deployments was at the Great Duck Island, about 50 miles off the coast of Maine. This network was used for microclimate monitoring. The initial deployment consisted of 50 sensor nodes, which was later increased to 100 nodes [3]. Future deployments are envisioned to have a few hundreds of these nodes. Due to the monitoring activities, these networks collect and store large quantities of data. Data stored by these networks can be extracted in different ways such as one-shot querying [16], *en masse* data gathering [17] and continuous data gathering [10]. The severe energy constraints of the sensor nodes engender the need for designing efficient (or optimal) data extraction algorithms based on mathematical modeling and optimization techniques. The optimization criteria depend on the specific application and the mode of data gathering. For example, an *en masse* data gathering application might be interested in maximizing the total data collected (without caring about the remaining life of the network), while a continuous data gathering application may want to ensure the longevity of the network. Moreover, the autonomous mode of operation and the highly dynamic deployment environments of these networks motivate the need for designing robust, online, distributed optimization algorithms.

In this study, we apply techniques from Mechanism Design and Game Theory to facilitate the design of decentralized algorithms for optimizing data gathering in sensor networks. Informally, we investigate the following problem: *Given a global objective function, can suitable local utility functions be designed such that each sensor node while "selfishly" optimizing its own local utility function leads to optimizing the desired global objective*. In this paper, we design efficient algorithms for continuous data gathering in sensor networks. Specifically, we consider continuous data gathering applications that construct a spanning tree rooted at the data sink in the network [10]. The desirable characteristics of this tree depend on the nature of the query being answered. Consider two specific examples:

1. "Report the maximum temperate in the network for the next 30 minutes at the rate of 1 reading every 30 seconds". In this case, each node just needs to send a single unit of data, which is the maximum temperature in its sub tree. We call such an aggregation as "perfect" aggregation.
2. "Report the temperature from all the nodes for the next 30 minutes at the rate of 1 reading every 30 seconds". In this case, each node needs to send the temperature of all the nodes in its sub tree and there is no aggregation.

*Corresponding author.

We assume that the energy dissipation at a sensor node is proportional to the number of data units received and transmitted by it. The end user is interested in ensuring a fair utilization of the energy resources in the network such that no sensor node suffers an early energy depletion. Consider a data gathering tree used to execute a query of the first type. Each sensor node (other than the root) transmits one unit of data. The number of data units received by a sensor node is proportional to the number of its children. For such a query, it is useful to construct a load balanced tree, where the load (number of children attached to a parent) is balanced at each level of the tree. Alternatively, consider a sensor node executing a query of the second type. The number of units received and transmitted by it is proportional to the size of its subtree. In such a scenario, it is of interest to construct an energy balanced tree in the network that ensures that the energy dissipation in routing data is uniform among all the nodes in the network.

For the purpose of demonstrating our approach, we focus on the construction of continuous data gathering trees that are optimized for the global objectives described above: load balance and energy balance. For each of these objectives, we design a local utility function for each sensor node, such that the desired objective is attained while the sensor nodes selfishly optimize their local utility functions. We propose an efficient, distributed heuristic *DistributedParentBid* for constructing a load balanced tree in the network. The performance of *DistributedParentBid* is evaluated by comparing against an optimal centralized algorithm over several scenarios involving random deployment of sensor nodes in the network. Our simulation results show that the heuristic produces the desired load balanced tree for around 90% of the simulation scenarios. We also show that an energy balanced tree can be built over the network, provided each sensor node routes its data along the shortest path (using an edge length metric that is weighed inversely to the current energy of the sensor node) to the sink. This can be achieved in a simple manner by using a distributed distance vector algorithm with the appropriate edge length metric.

The rest of the paper is organized as follows: Section 2 briefly describes the related work. The formal description of the load balanced data gathering tree construction problem is given in Section 3. We describe and analyze our decentralized algorithm for construction of a load balanced tree in Sections 3.1, 3.2 and 3.3. Section 4 describes our second case study, which is construction of an energy balanced data gathering tree in the network. Finally, we conclude in Section 5 with some discussion on our future work.

2. Related work

Due to the autonomous mode of operation and highly dynamic operating conditions, it is desirable to develop robust, distributed algorithms for several tasks of a sensor network like target tracking, edge detection, data gathering, localization, etc. [4]. Several studies have proposed distributed algorithms for each of these tasks. Most of these algorithms view the sensor network as a cooperative entity, operating towards achieving the desired global objective. In this study, we show that viewing sensors as "selfish" entities is also a useful abstraction for designing distributed algorithms for these inherently cooperative networks. We borrow techniques from Algorithmic Mechanism Design (AMD), which is a useful tool for distributed decision making. AMD ensures that the desired global objective is attained in the presence of selfish agents, where the utility function of an agent is assumed to be known *a priori* [5]. This is usually done by offering incentives to the agents in the form of payments. Game Theory also deals with situations involving selfish agents. Recently, networking research has started using game theory to model the impact of social behavior on the Internet. Papadimitriou gives a very good exposition of the applicability of game theory and algorithms to the Internet [14]. Most of the game theoretic literature study the properties of the Nash Equilibrium resulting from the interaction of selfish agents (whose utility functions are known *a priori*), without dealing with any particular global objective. Kannan, et al. examine the Nash Equilibrium arising from interactions of selfish sensors in the context of reliable data gathering in sensor networks [6]. In this study, unlike the above approaches that assume a utility function for an agent, we are interested in designing local utility functions for each sensor node such that the network as a "whole" attains the desired global objective when each sensor node selfishly seeks to maximize its local utility. Our utility function based approach is similar to the one adopted by Byers and Nasser [12]. Their study is targeted to applications that assume "perfect" aggregation and do not require the collection of data from all the sensor nodes at each round. This is captured by a global objective function that is concave in the number of nodes involved in the sensing operation at each round. They then propose a distributed algorithm in which each sensor node decides whether it should be involved in the sensing operation at each round. This algorithm maximizes the total utility of the sensing operation over time. Our study considers continuous data gathering applications that assume no aggregation and require data to be collected from all the sensor nodes. We propose distributed algorithms (by designing local utility functions) such that each sensor node decides its parent on the data gathering tree depending on the desired properties of the tree. As mentioned in Section 1, we are interested in two specific global objectives: load balance and energy balance.

Several algorithms have been proposed for load balancing: distributed algorithms (in the context of selfish users) [9], online algorithms [8] and offline centralized algorithms [7]. Grosu and Chronopoulos assume a local utility function (*a priori*) for each user and compare the resulting Nash Equilibrium with the desired global optimal solution [9]. The load balancing problem is essentially finding

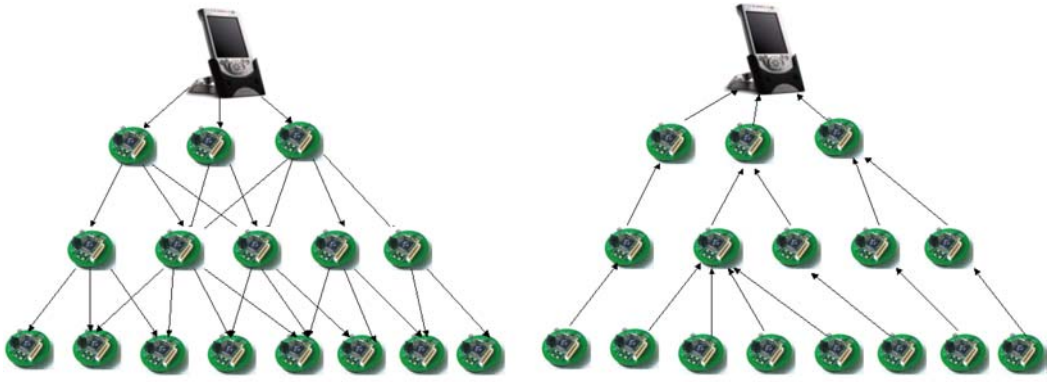


Figure 1. An illustration of load balancing. (b) depicts an arbitrary parent selection strategy. Notice that level 1 is load balanced, while level 2 is not, given the graph connectivity restrictions in (a)

an optimal semi-matching in a bipartite graph. Harvey et al. show that this problem can be reduced to a minimum cost maximum flow problem [7]. They propose efficient centralized offline algorithms for finding the optimal semi-matching. They also show that the resulting optimal semi-matching minimizes all norms including the variance of the load (number of children) i.e. the $\|L_2\|$ and the maximum load ($\|L_\infty\|$). In this study, we propose *DistributedParentBid*, a distributed iterative algorithm for the load balanced data gathering tree construction in sensor networks. The proposed algorithm specifies a local utility function for each sensor node such that the selfish behavior of the sensor nodes converges to a load balanced data gathering tree. Simulations indicate that in majority of the cases, our algorithm produces the same load balanced tree as that built by the centralized optimal semi-matching algorithm.

Energy efficiency has been one of main focal points for research in ad hoc wireless networks and sensor networks. Raghavendra et al. propose some metrics which address issues of energy efficiency and node life times [15]. Their work mainly focuses on pairs of communicating nodes. They show that some of their optimization metrics such as maximizing network partition time, minimizing variance in node energy levels, are NP-Complete [13]. In this paper, we propose a polynomial time algorithm for constructing an energy balanced tree, where no sensor node suffers early energy depletion.

3. Load balanced data gathering tree construction

Spanning trees have been widely used for gathering data from a large multi hop network to the base station/sink node. The following two phase process can be used for constructing a spanning tree in a sensor network as shown in Figure 1. The first phase involves flooding of a message from the sink node to all the sensor nodes in the network. This is followed by the second phase, whereby the sensor nodes organize themselves into levels based on their distance to the sink node. The sensor nodes at lower levels correspond to those closer to the sink node. Each sensor node other than the sink node selects one node from the previous level (with which it can communicate) as its parent [11].

In this section we consider a data gathering application that assumes "perfect" aggregation (see Section 1). The leaf nodes of the tree transmit one unit of data to their parent. The parent nodes (except the sink node) receive all the data from their children, aggregate the gathered information, and transmit one unit of data to their respective parent. One unit of energy is dissipated at the leaf nodes, while the parent nodes dissipate energy proportional to the number of children. Our goal is to construct a load balanced spanning tree that ensures fair utilization of the energy resources at all the sensor nodes.

Due to "perfect" aggregation, a key feature of the balanced data gathering construction problem is that the decisions taken by sensor nodes at one level of the tree are independent of those taken at another level. Thus, at each level, ensuring that each sensor node selects its parent such that none of the parents are overloaded will lead to a global tree in which none of the sensor nodes are overloaded. Thus, as mentioned in Section 1, we are interested in constructing a load balanced tree in the network that has the following property: Given a set of candidate parent nodes, each sensor node must select as parent, the sensor node at the previous level with the smallest number of children. In other words, if a sensor node discovers a candidate parent node that has fewer children than the current parent node, the sensor node must select this node as its new parent. As discussed earlier, the choices made by a sensor node at one level in the tree do not affect the choices made by the sensor nodes at the other levels. Hence, in the remaining part of this section, we will focus on a single level of the hierarchical structure created by the initial flood i.e. a set of children nodes with their corresponding sets of possible parents.

Before formally describing the problem, we introduce some preliminary notation:

1. M is the set of all parents at a level.
2. N is the set of children at a level.
3. M^j is the set of potential parents of sensor node j, such that $\forall i \in N : \bigcup_{i=1}^{|N|} M^i = M$.
4. $G = (M \bigcup N, E)$ is a bipartite graph such that if an edge $(i, j) \in E$, then $i \in M$ and $j \in N$ or vice-versa.
5. $|E| \leq M \times N$.

It is useful to abstract load balancing as a bandwidth allocation problem. Assume that a parent allocates its bandwidth (of 1 unit) equally to all its children. For example, if a parent has 3 children in the optimal tree then each child gets a bandwidth of $\frac{1}{3}$. In the optimal (load balanced tree), let $x_i{}^*$ be the bandwidth allocated to a sensor node i. Consider the optimal allocation vector $x^* = (x_1^*, x_2^*, \ldots x_N^*)$, such that $\sum_{i=1}^{N} x_i^* = |M|$. Using simple arguments, it can be shown that the optimal (load balanced) allocation vector x^* is max-min fair.

Thus, the global objective is to attain a max-min fair allocation for all the children (at each level). The mechanism design for the load balanced tree is described over the next three sections, Section 3.1 specifies the strategy space and the local utility function for each sensor node. The algorithm in described in Section 3.2and analyzed in Section 3.3.

3.1. Game description

We consider an iterative game. We define an iteration as a round in which parents announce their bandwidth guarantees and the children decide on the parent they want to attach. This game is played on the edges $e \in E$ of the bipartite graph mentioned in Section 3. The players in this game are the children. For each player i such that $i \in N$, let $S_k^i \subseteq M^i$ be the strategy space at iteration k. $u_k^i(p)$ denotes the utility of sensor node i for choosing sensor node p as its parent in iteration k. We define $u_k^i(p) = C_k^p$ where C_k^p is the bandwidth guaranteed by parent p at iteration k. i.e. the parent p is committed to provide a bandwidth of *at least* C_k^p to child i for all iterations after k as long as i is a child of p.

Let

$$\begin{aligned} u_k^i &= \text{Max}_{p \in S_k^i}\{u_k^i(p)\} \\ &= \text{Max}_{p \in S_k^i}\{C_k^p\} \end{aligned} \tag{1}$$

This utility function in Eqn. 1 dictates that at each iteration, a sensor node prefers to connect to a parent that gives it the maximum bandwidth guarantee. i.e. each sensor node is "selfish" about the bandwidth guarantee it receives. At every iteration, each parent p gives equal bandwidth guarantees to its children.

In Section 3.2, we describe the algorithm that constructs a load balanced tree in the presence of "selfish"sensor nodes.

3.2. Algorithm for load balanced tree construction

In this section, we describe *DistributedParentBid*, an iterative distributed algorithm for constructing a load balanced spanning tree in the network. The algorithm assumes that each sensor node's decision making is governed by the utility function described by Eqn. 1 in Section 3.1. Before describing the algorithms of the parent and child, we introduce some notations and definitions. As described in Section 3.1, we define one iteration to be one round of communication between the parents and the children. This round consists of the following steps:

1. A candidate parent announces its bandwidth guarantee.
2. Each child responds to all candidate parents that offer the best bandwidth guarantee.
3. A candidate parent chooses a subset of the responding children informing them to attach to it.
4. Each child then chooses one of the candidate parents to attach and informs the rest that it does not want to attach to them.

Let

1. deg(j) be the degree of a node j in the bipartite graph G described in Section 3.
2. P_k^i be the parent of node i at iteration k.
3. Y_k^p be the number of additional children that parent p can take during iteration k, giving a bandwidth guarantee of C_k^p.
4. n_k^p be the number of **new** children that request to attach to parent p during iteration k.
5. a_k^p be the number of **new** children that actually attach to parent p during iteration k[1].
6. z_p^k be the number of children that leave parent p during iteration k.
7. N_k^p be the set of children currently attached to parent p at the end of iteration k.
8. B_k^p be the total bandwidth auctioned by a parent p at the end of iteration k. i.e.

$$B_k^p = |N_k^p| C_k^p \quad \text{if} \quad |N_k^p| > 0 \tag{2}$$

$$= C_k^p \quad \text{otherwise} \tag{3}$$

Definition 1. A parent p is considered to be saturated at iteration k iff $B_k^p = 1$.

i.e. a parent p is saturated at iteration k iff its total auctioned bandwidth equals the maximum bandwidth of 1. On saturation at iteration k, parent p will offer a bandwidth guarantee of $C_k^p = \frac{1}{|N_k^p|}$ to each of the children attached to it.

[1] A child might request to be attached to several parents during iteration k, but will ultimately choose exactly one parent that is willing to take it at the end of iteration k.

If $|N_k^p| > 0$, saturation implies that p cannot take any more children at the guaranteed bandwidth of C_k^p. If $|N_k^p| = 0$, $B_k^p = C_k^p = 1$ implies that each of p's children have a bandwidth guarantee of 1 from some other parent p'. $\forall\ p$: $p \in M, y_k^p = 0$ iff p is saturated at iteration k.

DistributedParentBid consists of algorithms for **Parent** and **Child**, which are described below:

Parent: A parent p successively increases its bandwidth guarantees from $\frac{1}{deg(p)}$, $\frac{1}{deg(p)-1}$, $\frac{1}{deg(p)-2}$, so on until it gets saturated. In a system implementation, this can be achieved by the parent p broadcasting a message containing the bandwidth guarantee so that all children within its radio range can receive the bandwidth announcement. Each child that is not already attached to p, will respond to p if p guarantees the highest possible bandwidth among all its potential parents. If p cannot guarantee the announced bandwidth to all the children that respond, it will allow a subset of the responding children to select it as a parent and reject the rest. For example, consider a parent p that has 5 children currently. If p announces a bandwidth guarantee of $\frac{1}{7}$ and 3 additional children respond to it, p can only choose 2 out of the 3 children. This is because on choosing 2 children, p saturates with 7 children, each getting a bandwidth of $\frac{1}{7}$. An unsaturated p does not increase its bandwidth guarantee from iteration k to $k+1$ iff any of the following conditions are satisfied:

1. At least one of its current children switches to some other parent during iteration k.
2. At least one child that is not currently attached to it during iteration k requests to switch to it.

These conditions ensure that a parent p will increase its bandwidth guarantee iff no additional children want to switch to it at its current guarantee. While this is apparent for the second condition, the first condition is for letting a child that was rejected earlier by p to switch to it at the same bandwidth guarantee if possible (which may happen due to some already attached child leaving p for some other parent).

Parent p stops issuing the bandwidth announcement messages once it is saturated.

Initially, a parent p sets $C_k^p = \frac{1}{deg(p)}$. While p is not saturated (may also happen if a child leaves a saturated parent making $B_k^p < 1$), it executes the following iterative algorithm (here, k is a global variable that keeps track of iterations previously executed by this parent):

Parent(p){
  while ($B_k^p < 1$) {
    $C_{k+1}^p \leftarrow C_k^p$
    announce} C_{k+1}^p to all its children
    /* *Find the upper bound on the number of children that can be taken* */
    $y_{k+1}^p \leftarrow \frac{1}{C_{k+1}^p} - |N_k^p|$.
    if ($n_{k+1}^p > y_{k+1}^p$) reject the excess children
    $|N_{k+1}^p| \leftarrow |N_k^p| + a_{k+1}^p - z_{k+1}^p$.
    if ($B_{k+1}^p = 1$) exit
    /* *If a child left or responded to the announcement, don't increase the bandwidth guarantee* */}
    if (($z_{k+1}^p > 0$) or ($n_{k+1}^p > 0$)) $\in \{C_{k+1}^p \leftarrow C_k^p\}$
    else $\{C_{k+1}^p \leftarrow \frac{1}{\frac{1}{C_k^p}-1}\}$
    $k \leftarrow k+1$
    }
}

Child: At an iteration k, the strategy space of child i, $S_k^i = \{p | p \in M^i,\ B_k^p < 1\}$. i.e. at each iteration k, each child i will try to connect to an unsaturated parent p that provides the best bandwidth guarantee. In a system implementation, this can be achieved by each child sending a message in response to the bandwidth announcement from a parent p. If there are multiple such parents, the child will request to connect to all of them[2]. If multiple parents are willing to accept it, the node will choose one of the parents and will inform the others that it does not wish to connect to them.

Termination: *DistributedParentBid* terminates at the smallest iteration k at which all parents are saturated i.e. $\sum_{p=1}^{M} B_k^p = M$.

3.3. Analysis

In this section, we show that *DistributedParentBid* algorithm terminates and analyze its running time. We show that a Nash Equilibrium exists on termination of the algorithm. We also outline a candidate scenario where this algorithm might not produce a load balanced tree. However, by comparing it against a centralized algorithm for load balancing through simulations, we show that the proposed algorithm produces a load balanced tree for around 90% of the scenarios studied.

Let $S_k = \sum_{i=1}^{M} B_k^i$ be the total auctioned bandwidth across all the parents at the end of iteration k. Let k^* be the smallest iteration at which all parents p are saturated i.e. $\sum_{p=1}^{M} B_{k^*}^p = M$.

Lemma 1. *The following quantities are non-decreasing functions of k:*

1. *For each parent p,* C_k^p.
2. *For each child i,* u_k^i.

Proof:

1. The algorithm *Parent(p)* never decreases C_k^p from one iteration to the next. Thus, $C_{k+1}^p \geq C_k^p$
2. By definition, $u_k^i = \text{Max}_{p \in S_k^i}\{C_k^p\}$. If i is attached to the same parent at iteration k and $k+1$, from the first part

[2]The child has to respond to all such parents because a previously unsaturated parent might have to reject some children if it gets saturated during the current iteration.

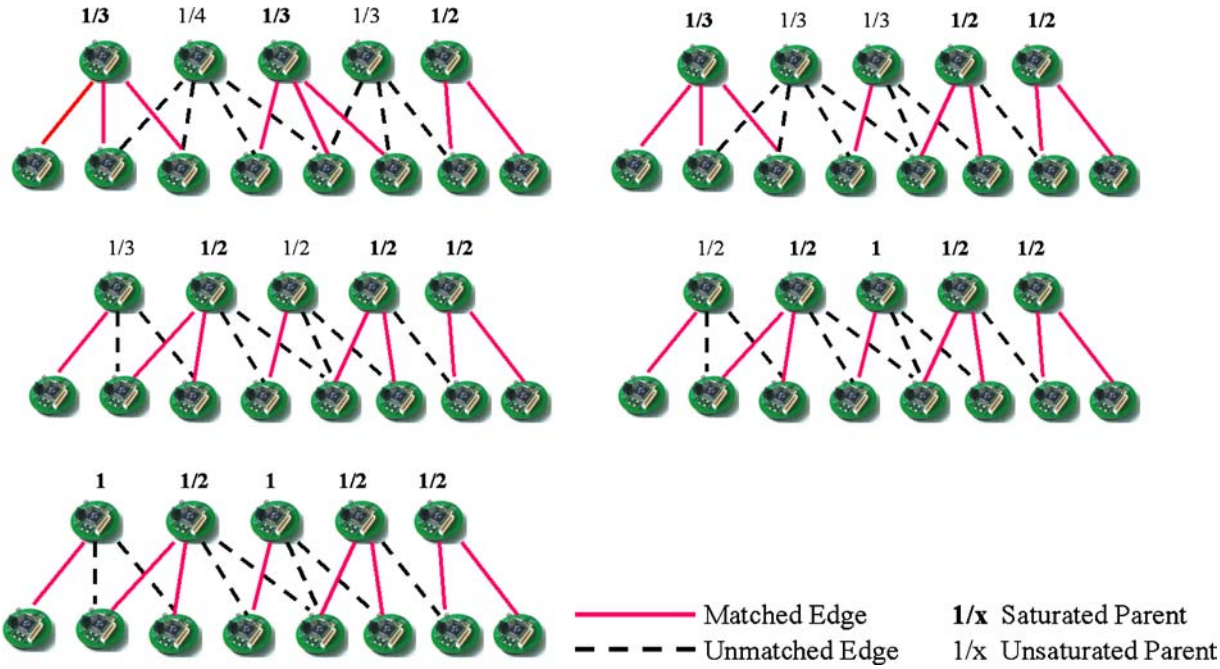


Figure 2. Load balancing level 2 of the tree show in figure 1 (b) using *DistributedParentBid*.

of this Lemma, $u^i_{k+1} \geq u^i_k$. i switches from p to p' during iteration $k+1$ iff $C^{p'}_{k+1} > C^p_{k+1} \geq C^p_k$. Hence, $u^i_{k+1} > u^i_k$. □

Lemma 2. *If at least one child switches from one parent to the other during iteration k, $S_k > S_{k-1}$.*

Proof: Let $N' \subseteq N$ be the set of children that switch and $N - N'$ be the set of children that do not switch parents. Then

$$S_k - S_{k-1} \geq \sum_{i \in N} \left\{u^i_k - u^i_{k-1}\right\}$$
$$\geq \sum_{i \in N} \left\{u^i_k - u^i_{k-1}\right\} + \sum_{j \in N-N'} \left\{u^j_k - u^j_{k-1}\right\}$$

From Lemma 1, we observe that $u^i_k \geq u^i_{k-1}$ for all i. Specifically for $i \in N'$, $u^i_k > u^i_{k-1}$. Hence, if at least one child switches parents during iteration k ($N' \neq \phi$), then $S_k > S_{k-1}$. □

Lemma 3. *If $S_k > S_{k-1}$, $\delta = S_k - S_{k-1} \geq \frac{1}{(\gamma)(\gamma-1)}$, where $\gamma = Max_{p \in M}\{deg(p)\}$ is the maximum degree of a parent in the graph G.*

Proof: If $S_k > S_{k-1}$, two cases arise:

1. A child i switches from parent p to q during iteration k.
2. No child switches, but an unsaturated parent p increases its bandwidth guarantee from C^p_{k-1} to C^p_k.

In the first case, a child i switches from parent p to q during iteration k iff $C^q_k > C^p_k$. Now, $C^p_k \geq C^p_{k-1}$ implies $u^i_k > u^i_{k-1}$. From the proof of lemma 2, we obtain that $S_{k-1} - S_k \geq \{u^i_k - u^i_{k-1}\}$. This difference is the smallest when exactly one child i switches from a parent p that offers it a bandwidth of $\frac{1}{\gamma}$ to a parent q that offers it a bandwidth of $\frac{1}{\gamma-1}$. Hence, the smallest difference $\delta = \frac{1}{(\gamma)(\gamma-1)} \leq S_k - S_{k-1}$. Hence, in general, if at least one node switches at iteration k, $S_k > S_{k-1}$ and difference $\delta = S_k - S_{k-1} \geq \frac{1}{(\gamma)(\gamma-1)}$. For the second case, we have the following:

$$S_k - S_{k-1} \geq \sum_{q \in M} \left\{C^q_k - C^q_{k-1}\right\}$$
$$\geq \left\{C^p_k - C^p_{k-1}\right\}$$
$$\geq \frac{1}{\gamma - 1} - \frac{1}{\gamma}$$
$$\geq \delta$$

where p is an unsaturated parent.

Leema 4. *The total auctioned bandwidth across all parents (S_k) is a non-decreasing function of k. Moreover, either $S_{k+1} > S_k$ or $S_{k+2} > S_{k+1} = S_k$.*

Proof: Consider an iteration $k < k^*$. There exists at least one unsaturated parent p at the endof iteration k. Two possibilities arise:

1. None of the unsaturated parents increase their bandwidth guarantee during iteration $k+1$. Two sub-cases arise:
   (a) At least one child switches during this iteration then from Lemma 2, $S_{k+1} > S_k$.
   (b) No child switches during this iteration$S_{k+1} = S_k$. Now, during iteration k + 2, all unsaturated parents increase their bandwidth guarantees. Again if

a child switches parents, then$S_{k+2} > S_{k+1}$. If no child switches during iteration k + 2 then,

$$S_{k+2} - S_{k+1} \geq \sum_{j \in M} \{C^j_{k+2} - C^j_{k-1}\}$$

Since $C^j_{k+2} > C^j_{k+1}$ for each unsaturated parent j, $S_{k+2} > S_{k+1} = S_k$.

2. At least one unsaturated parent increases its bandwidth guarantee during iteration $k + 1$. Then, as seen from the proof of 1 (b), $S_{k+1} > S_k$. □

We now prove our main result about the running time of*DistributedParentBid*

Theorem 1. *The total number of iterations taken by DistributedParentBid is $O(M\gamma^2)$. At termination, a Nash Equilibrium exists.*

Proof: Initially, each parent p announces a bandwidth guarantee of $C^p_0 = \frac{1}{deg(p)} \geq \frac{1}{\gamma}$. Thus, $S_0 \geq \frac{M}{\gamma}$. If k^* is the smallest iteration at which all parents are saturated, $S_{k^*} = M$. From Lemma 4, we see that there are at most 2 iterations when S_k remains constant (i.e. k and $k + 1$). Moreover, whenever S_k increases it increases by at least δ as seen from Lemma 3. Thus, the total number of iterations (T) for the termination of *DistributedParentBid* is given by:

$$T \leq 2\left\{\frac{S_{k^*} - S_0}{\delta}\right\}$$
$$\leq 2M\gamma^2$$

Next, we prove the existence of a Nash Equilibrium on the termination of *DistributedParentBid*.

On termination at iteration k^*, all parents are saturated i.e. $\forall p$, $B^p_{k^*} = 1$. Every saturated parent p offers a bandwidth of $\frac{1}{|N^p_{k^*}|}$ where N^p_{k*} is the number of children currently attached to it. Consider any child i. The current parent of i at iteration k^* is $q = P^i_{k^*}$. The bandwidth allocated by q to all its children (including i) is $\frac{1}{|N^q_{k^*}|}$. Then at iteration K^*, every other potential parent q' of i must have at least $|N^q_{k^*}| - 1$ children. This can be proved by contradiction. Let us assume that a parent q' is saturated with at most $|N^q_{k^*}| - 2$ children each getting a bandwidth of at least $\frac{1}{|N^q_{k^*}|-2}$. Since the bandwidth guaranteed by a parent is non-decreasing in the number of iterations as shown by Lemma 1, there would have been an iteration $k' < K^*$ such that q' would have offered a bandwidth of at least $\frac{1}{|N^q_{k^*}|-1}$ while q would have offered a bandwidth of at most $\frac{1}{|N^q_{k^*}|}$. At this iteration i would have switched and attached to q'. But this contradicts the fact that i is attached to parent q and not q'. Thus, every potential parent q' of i must have at least $|N^q_{k^*}| - 1$ children and can offer a bandwidth of at most $\frac{1}{|N^q_{k^*}|}$ to i (if i switches from q to q'), which is the bandwidth offered by q. Hence, at termination each child is attached to the parent that allocates the highest possible bandwidth, implying the existence of a Nash Equilibrium. □

As mentioned in Section 2, a load balanced tree corresponds to an optimal semi-matching. Our approach to proving the optimality of *DistributedParentBid* was based on the work of Harvey et al. which characterizes an optimal semi-matching using the notion of cost reducing paths (defined below) [7].

Definition 2. Given a semi-matching $S \subseteq E$ in $G = (V, E)$, a cost reducing path R is a sequence of alternating matched and unmatched edges given by $R = \{(v_1, u_1), (u_1, v_2), \ldots (u_{z-1}, v_z)\}$ where $v_i \in M, u_i \in N$ and $(v_i, u_i) \in S$ for all i, such that $deg_S(v_1) - deg_S(v_z) > 1$.$deg_S(v_i)$ is the number of children matched to parent v_i in the semi-matching S.

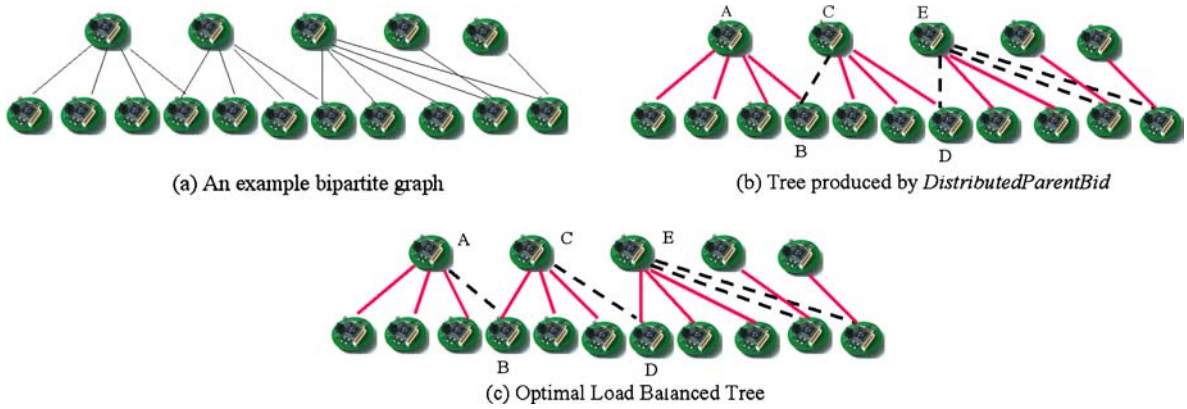
Moreover, the authors of [7] also prove the following:

Theorem 2. *A semi-matching is optimal iff no cost-reducing path exists.*

We now show that *DistributedParentBid* does not always produce a load balanced tree as shown in Figure 3:

Notice that in Figure 3 (b), the maximum load is 4, while in (c), it is 3. The above sub optimality occurs due to the existence of the cost reducing path A-B-C-D-E in the bipartite graph i.e. the path consisting of parent A – child B – parent C – child D – parent E as shown in Figure 3 (b). The matched and the unmatched edges along this path can be switched to obtain a better load balanced tree as shown in Figure 3 (c). In general, the sub-optimality depends on the number of such paths existing in the bipartite graph after *DistributedParentBid* terminates. In the worst case scenario, the maximum load produced by *DistributedParentBid* can be a worse off by a factor of M from the optimum, where M is the number of parents i.e. the maximum load can be equally distributed across all the parents.

We attempted to quantify the sub-optimality (in random scenarios) by simulations where we compared the tree resulting from an implementation of *DistributedParentBid* and the centralized algorithm for producing an optimally load balanced tree described in [7]. 7 parent and 30 child nodes were randomly deployed in an area of 500 m × 500 m. For each value of the transmission range used (250 m and 200 m), 5000 simulations (using random seeds) were run. In 90% of the simulation runs, *DistributedParentBid* produced trees that were isomorphic to the centralized algorithm, while in the remaining 10% of the simulations, the maximum (minimum) load produced by *DistributedParentBid* differed by a maximum of 1 from the centralized algorithm. These results give us reasonable confidence that *DistributedParentBid* is a good heuristic for load balancing.



(a) An example bipartite graph

(b) Tree produced by *DistributedParentBid*

(c) Optimal Load Balanced Tree

Figure 3. Sub optimality of *DistributedParentBid*.

4. Energy balanced data gathering tree construction

In this section, we discuss the construction of a spanning tree in the network for a continuous data gathering application that assumes no data aggregation. An example of such a query is described in Section 1. We consider an arbitrary deployment of n sensor nodes in the monitored terrain. Each sensor node samples one unit of data from the environment, which must be transmitted to the sink node in the network. Note that one unit of data can be one or several bytes. In order to route all the data, the sensor nodes form a spanning tree rooted at the sink.

Each leaf sensor node in the tree transmits one unit of data. The non-leaf sensor nodes in the tree route their own data as well as the data collected by all the sensor nodes in their subtree. Thus, each leaf sensor node dissipates one unit of energy, and the non-leaf sensor nodes dissipate energy proportional to the size of their subtree. To ensure the longevity of the network, our goal is to construct a spanning tree such that all the sensor nodes are fairly utilized on the average. In this study, we define fair utilization as utilizing a sensor node commensurate to its current energy i.e. a sensor node having greater energy should have a larger sub tree (as permitted by the topology).

Let T denote a spanning tree constructed over the network, rooted at the sink node. Consider any node i in the spanning tree. Let e^i denote its current energy and $S(T, i)$ represent the size of its subtree. As discussed earlier, energy dissipation at a sensor node is proportional to the size of its subtree. After one iteration of data routing, the remaining energy of the sensor node is given by $e^i - S(T, i)$, and the fractional remaining energy is defined as $\frac{e^i - S(T,i)}{e^i}$ [3]. To ensure fair utilization of energy resources at the sensor nodes, our goal is to construct an *energy balanced* spanning tree in the network, which maximizes the summation of the fractional remaining energy of all the sensor nodes[4]. Hence, our desired global objective function is to construct a tree T' such that:

$$T' = \arg\max_{T} \left\{ \sum_{i=1}^{n} \frac{(e^i - S(T, i))}{e^i} \right\}$$

Here $< e^1, \ldots, e^n >$ represent the current energy of the sensor nodes. Figure 4 illustrates an energy balanced tree.

We represent the network by the graph $G = (V, E, B)$, where the set of vertices represent the sensor nodes. An edge $(i, j) \in E$ exists between two vertices $v_i, v_j \in V$, provided that the sensor node represented by vertex v_j lies within the transmission range of the sensor node represented by vertex v_i. $B \in V$ denotes the sink node in the network. Theorem 3 states the key result that enables the design of a distributed mechanism for constructing an energy balanced tree.

Theorem 3. *Given the network graph $G = (V, E, B)$, construct a weighted directed graph $G' = (V, E_d, B, w)$ such that*

1. *If $(i, j) \in E$, then $(i, j) \in E_d$.*
2. *$\forall (i, j) \in E_d, i \neq B : w(i, j) = \frac{1}{e^i}$.*

The energy balanced tree is the sink-rooted Shortest Path Tree (SPT) in the weighted graph G'.

Proof: Let $T' = (V, E', B)$ represent the SPT, and $T'' = (V, E'', B)$ be any other spanning tree for the graph $G' = (V, E_d, B, w)$, rooted at B. Let $P'_{i,B}$ and $P''_{i,B}$ denote the paths from any vertex i to B in T' and T'' respectively. The weights (length) associated with these paths is defined as follows.

- $w(P'_{i,B}) = \sum_{(i_1, i_2) \in P'_{i,B}} w(i_1, i_2)$.
- $w(P''_{i,B}) = \sum_{(i_3, i_4) \in P''_{i,B}} w(i_3, i_4)$

Since, T' is the SPT for G', we know that

$$\forall i \in V, w(P'_{i,B}) \leq w(P''_{i,B}) \tag{4}$$

[3] We assume that the remaining energy is always positive. i.e. all sensor nodes have sufficient energy to accomplish the data gathering.

[4] This objective is reasonable if we view the entire sensor network as a distributed energy source.

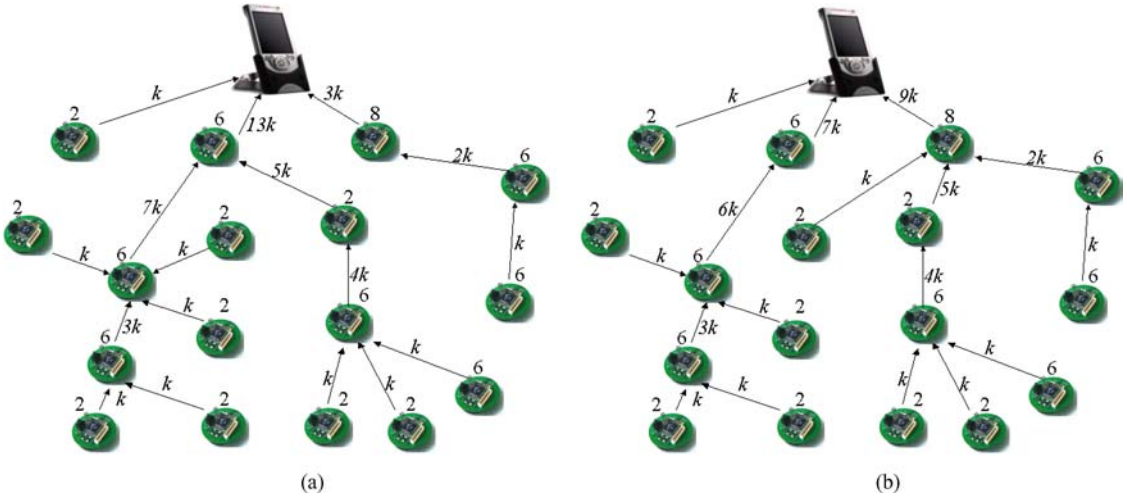


Figure 4. An illustration of energy balancing. The number on the sensor node shows the remaining energy, while the number on the edge shows the amount of data bytes transmitted. Each sensor node is assumed to generate one unit of data consisting of k bytes. (a) depicts the tree resulting from an arbitrary parent selection strategy, while (b) shows the energy balanced tree. In (b), notice the distribution of load commensurate to the energy on the 2 first hop nodes having energy of 6 and 8 units as opposed to (a).

Summing up over all the nodes in G', we get

$$\sum_{i \in V} w(P'_{i,B}) \leq \sum_{i \in V} w(P''_{i,B}) \tag{5}$$

$$\sum_{i \in V} \sum_{(i_1,i_2) \in P'_{i,B}} w(i_1, i_2) \leq \sum_{i \in V} \sum_{(i_3,i_4) \in P''_{i,B}} w(i_3, i_4) \tag{6}$$

Interchanging the summations,

$$\sum_{(i_1,i_2) \in E'} N(T', (i_1, i_2)) * w(i_1, i_2) \leq \sum_{(i_3,i_4) \in E''} N(T'', (i_3, i_4)) * w(i_3, i_4) \tag{7}$$

where for each edge (y, z) in a spanning tree T, $N(T,(y,z))$ is the number of nodes in T for which (y, z) occurs on the path to B. In the SPT T', this path is the shortest path, while in an arbitrary spanning tree T'', this is not necessarily the shortest path to B. Since, T is a tree there is no alternative path in T for these nodes to reach B. Thus, these nodes will be in the sub tree rooted at a node y. Hence,

$$N(T, (y, z)) = S(T, y)$$

On replacing $N(T, (y, z))$ by $S(T, y)$ and $w(y, z)$ by $\frac{1}{e^y}$ in Eqn 7, we get

$$\sum_{i \in V} \frac{S(T', i)}{e^i} \leq \sum_{j \in V} \frac{S(T'', j)}{e^j} \tag{8}$$

Thus,

$$T' = \arg\max_{T} \left\{ \sum_{i=1}^{X} \frac{(e^i - S(T, i))}{e^i} \right\}$$

□

From the above analysis, we conclude that the desired energy balanced tree is the shortest path tree in the weighted graph G'. The shortest path tree for any given graph can be computed in polynomial time.

Theorem 3, gives us the key insight for designing local utility functions for the sensor nodes for achieving our global objective. If each sensor node i "selfishly" routes its data to the sink over the shortest path using $\frac{1}{e^i}$ as the edge length metric, it results in the desired energy balanced data aggregation tree. The cost function $c(i, j) = \frac{1}{e^i}$ has the nice property that sensor nodes with lower energy will have a higher cost for their outgoing link and hence are less likely to have a large sub tree rooted at them. This ensures longevity of the data gathering tree.

We describe a simple mechanism for constructing an energy balanced tree in the following section.

4.1. Mechanism design

In this section, we describe a mechanism for constructing the energy balanced data gathering tree. Consider an iterative game on the edges of the graph $G = (V, E, B)$. Since we are interested in balancing the utilization of all the nodes in the network, this graph corresponds to the entire deployment topology (unlike the bipartite graph used in the load balanced tree construction in Section 3.1).

The strategy space S^i_κ of each sensor node i at iteration κ is given as follows:

$$S^i_\kappa = \{j \mid (i, j) \in E\} \tag{9}$$

The utility function of each sensor node i is given as follows:

$$u^i_\kappa = \text{Min}_{j \in S^i_\kappa} \left\{ \frac{1}{e^i} + Q^j_\kappa \right\} = \text{Min}_{j \in S^i_\kappa} \{Q^j_\kappa\} \tag{10}$$

where Q_κ^j is the length of the shortest path from sensor node j to the sink B at iteration κ. $\forall\kappa : Q_\kappa^{\text{sink}} = 0$. Thus, at each iteration, a sensor node i chooses a node j that offers it the shortest path (using $\frac{1}{e^z}$ as the metric of edge length for any edge $(z, w) \in E$) to B

The above mechanism can be easily implemented by a distributed distance vector algorithm. An iteration κ corresponds to an iteration of the distance vector algorithm. The energy balanced tree construction terminates when the distance vector algorithm terminates.

5. Conclusions and Future Work

In this study, we advocate a utility function based approach for designing sensor networks. Unlike previous related studies, we illustrate that treating the sensor nodes as "selfish" (using appropriate utility functions) enables the design of distributed algorithms for optimizing the network performance as a "whole". This approach is illustrated by two case studies of constructing a load balanced data gathering tree and an energy balanced data gathering tree in a sensor network.

As part of future work, it would be interesting to extend our mechanism of load balancing and energy balancing for more general scenarios that account for heterogeneity in quality of the links, more sophisticated models of data aggregation, etc. We also plan to investigate other desired global objectives of data collection trees for which decentralized mechanisms using local utility functions can be designed.

Acknowledgment

We thank Prof. A. Goel (Stanford University), Prof. D. Kempe (Cornell University) and Prof. B. Hajek (University of Illinois, Urbana Champaign) for helpful discussions and suggestions. This work was supported in part by NSF under grant 0325875, and by a 2003 USC Zumberge Grant.

References

[1] N. Sadagopan and B. Krishnamachari, Decentralized Utility-based Design of Sensor Networks, *WiOpt*'04: Second Workshop on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, University of Cambridge, UK (March, 2004).

[2] P. Zhang, C.M. Sadler, S. Lyon and M. Martonosi, "Hardware Design Experiences in ZebraNet", *ACM SenSys* (2004).

[3] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring and D. Estrin, "Habitat Monitoring with Sensor Networks", *Communications of the ACM*, 47(6) (2004).

[4] D. Estrin, R. Govindan, and J. Heidemann, Scalable Coordination in Sensor Networks, Technical Report 99-692, University of Southern California, (Jan. 1999). (appeared in Mobicom '99).

[5] N. Nisan and A. Ronen, Algorithmic Mechanism Design, In Proc. 31st *ACM Symposium of Theory of Computing (STOC)* (1999) 129–140.

[6] R. Kannan, S. Sarangi, S.S. Iyengar and L. Ray, Sensor Centric Quality of Routing in Sensor Networks, *IEEE Infocom* (2003).

[7] N.J. Harvey, R.E. Ladner, L. Lovasz and T. Tamir, Semi-Matchings for Bipartite Graphs and Load Balancing, *Workshop of Algorithms and Data Structures (WADS)*, (2003).

[8] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin and O. Waarts. On-line Machine Scheduling with Applications to Load Balancing and Virtual Circuit Routing, In Proc. *ACM Symposium of Theory of Computing (STOC)*, (1993).

[9] D. Grosu and A. Chronopoulos, A Game-Theoretic Model and Algorithms for Load Balancing in Distributed Systems, *International Parallel and Distributed Processing Symposium (IPDPS)* Workshop, (2002).

[10] C. Intanagonwiwat, R. Govindan and D. Estrin, Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks, *Mobile Computing and Networking*, (2000) 56–67.

[11] C. Zhou and B. Krishnamachari, Localized Topology Generation Mechanisms for Self-Configuring Sensor Networks, *IEEE Globecom*, San Francisco (December 2003).

[12] J. Byers and G. Nasser, Utility-Based Decision-Making in Wireless Sensor Networks, *IEEE Mobihoc* (2000).

[13] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, Introduction To Algorithms, Second Edition, (McGraw Hill, 1998).

[14] C.H. Papadimitriou, Algorithms, Games and the Internet, *ACM Symposium of Theory of Computing (STOC)* (2001).

[15] S. Singh, M. Woo, and C.S. Raghavendra, Power Aware Routing In Mobile Adhoc Networks, Proceedings of *ACM Mobicom*, Dallas, October (1998).

[16] N. Sadagopan, B. Krishnamachari and A. Helmy, Active Query Forwarding in Sensor Networks (ACQUIRE), *Elsevier journal on Ad Hoc Networks* (2003).

[17] N. Sadagopan and B. Krishnamachari, Maximizing Data Extraction in Energy-Limited Sensor Networks, *IEEE Infocom* (2004).

Narayanan Sadagopan received the B.S. degree in computer science from the Regional Engineering College, Trichy, India, in 1998, and the M.S. degree in computer science from University of Southern California (USC), Los Angeles, in 2001. He is currently working toward the Ph.D. degree in the Computer Science Department, USC. His research is focused on theoretical aspects of wireless ad hoc and sensor networks.
E-mail: narayans@usc.edu

Mitali Singh received the BTech. degree in Computer Science and Engineering from the Indian Institute of Technology, New Delhi, India in 2000, and the M.S. degree in Computer Science from the University of Southern California, Los Angeles, USA. She is currently working towards the Ph.D. degree in Computer Science at the University of Southern California. Her research interests lie in the area of applied theory and networks. Presently, her work is focused on high level modeling and distributed algorithm design for wireless sensor systems.
E-mail: mitali@halcyon.usc.edu

Bhaskar Krishnamachari received the B.E.E.E. degree from The Cooper Union for the Advancement of Science and Art, New York, in 1998, and the M.S.E.E. and Ph.D. degrees in electrical engineering from Cornell University, Ithaca, NY, in 1999 and 2002, respectively. He is now an Assistant Professor in the Department of Electrical Engineering, University of Southern California, Los Angeles, where he also holds a joint appointment in the Department of Computer Science. His current research is focused on the discovery of fundamental principles and the analysis and design of protocols for next-generation wireless sensor networks.
E-mail: bkrishna@usc.edu