Decentralized Utility-based Sensor Network Design

Narayanan Sadagopan and Bhaskar Krishnamachari

University of Southern California, Los Angeles, CA 90089-0781, USA
narayans@cs.usc.edu, bkrishna@usc.edu

Abstract. Wireless sensor networks consist of energy constrained nodes operating typically in an unattended mode and highly dynamic environments. In this paper, we advocate a systematic decentralized approach to designing these networks based on utility functions. We investigate the design of appropriate local utility functions for each sensor such that while each sensor "selfishly" optimizes its own utility, the network as a "whole" converges to a desired global objective. Specifically, we design a mechanism for the problem of constructing a load balanced data gathering tree in a sensor network using this approach. We show that the tree obtained using this mechanism is optimally load balanced by comparison with a centralized offline algorithm. This study suggests a significant departure from the existing view of sensor networks as consisting of cooperative nodes i.e "selfish" sensors might be a useful paradigm for designing efficient distributed algorithms for optimizing the performance of these networks.

1 Introduction

Severe energy constraints of a sensor network engender the need for a principled approach to designing these networks based on mathematical and optimization techniques. These networks are assumed to operate autonomously in highly dynamic environments, thus rendering offline, centralized optimization techniques unsuitable for these systems. In this study, we apply techniques from Mechanism Design and Game Theory to facilitate the design of decentralized mechanisms for optimization in sensor networks. Informally, we investigate the following problem: *Given a global objective function that has to be optimized by a sensor network, can suitable local utility functions be designed such that each sensor while "selfishly" optimizing its own local utility function leads to optimizing the desired global objective.* In this paper, we examine the specific case study of constructing a load balanced data collection tree in sensor networks as our global objective.

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 and our proposed decentralized iterative mechanism and its analysis are described in section 4. The validation of our algorithm through simulations is described in section 5 while section 6 outlines our conclusions and future work.

2 Related Work

Game-theoretic Mechanism Design is a useful tool for distributed decision making. It 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* [1]. This is usually done by offering incentives to the agents in the form of payments. Game Theory also deals with situations involving selfish agents. Most of the game theoretic studies study the properties of the Nash Equilibrium resulting from 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 [2]. 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 such that the network as a "whole" attains the desired global objective when each sensor selfishly seeks to maximize its local utility. Our utility function based approach is similar to the one adopted by Byers and Nasser in [7]. They use this approach to assign tasks to the various sensors as either sensing only, sensing and routing, routing only or idle depending on the requirements of an application (which is modelled by a global utility function). However, their approach does not design local utility functions for each sensor.

Load balancing has been widely studied in the context of selfish users [5], online algorithms [4] and offline centralized algorithms [3]. The authors of [5] specify a local utility function for each user and compare the resulting Nash Equilibrium with the desired global optimal solution. The load balancing problem is essentially finding an optimal semi-matching in a bipartite graph. The authors of [3] show that this problem can be reduced to a minimum cost maximum flow problem. They propose faster centralized offline algorithms for finding the optimal load-balanced 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 a distributed iterative algorithm for the load balancing problem in the context of sensor networks. This algorithm specifies a local utility function for each sensor such that the selfish behavior of the sensors converges to a load balanced data gathering tree. Simulations of our distributed algorithm indicate that it produces exactly similar results compared to the centralized optimal semi-matching algorithm described in [3]. In the next section, we formally describe the problem of constructing a load balanced data gathering tree in the context of sensor networks.

3 Problem Description

One mechanism to organize data gathering from a large multi-hop sensor network, is to construct a tree rooted at the base-station/sink node. This can be done by a two phase process whereby the sink issues a flood, and nodes organize themselves by level (depending on how many hops they are from the sink) and select a neighboring node at the previous level to be their parent [6]. Although a priori there are a number of ways to perform this parent selection (some of which are explored in [6] via simulations), one key requirement is to generate a balanced tree where that no node is overloaded with more traffic than is necessary. If a sensor can connect to a parent (permitted by the topology) that has fewer children than the current one, it should connect to the parent with fewer children.

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 sensors select their parent such that none of their parents are overloaded at each level will lead to a global tree in which none of the sensors are overloaded. Hence, in the remaining part of this paper, 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.
2. N is the set of children, $|N| \geq |M|$.
3. M^j is the set of parents of sensor 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 an edge $(i, j) \in E$ iff $i \in M$ and $j \in N$ or vice-versa.
5. $|E| \leq M \times N$.

It is useful to consider the load balancing problem from the bandwidth allocation point of view. Assume that a parent allocates its bandwidth (of 1 unit) equally to all its children. For example, if in the optimal tree, a parent has 3 children, each child gets a bandwidth of $\frac{1}{3}$. Now, in the optimal (load balanced tree), let x_i^* be the bandwidth allocated to a sensor i. Consider the optimal allocation vector $x^* = (x_1^*, x_2^*, ...x_N^*)$, 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). In section 4, we show that viewing load balancing as a bandwidth allocation problem helps in designing simple localized rules that can be used to attain the global objective.

4 Mechanism Design

In this section, we describe an iterative distributed algorithm that converges to the load balanced tree. This algorithm treats each sensor as being "selfish". The strategy space and the local utility function of each sensor is described in section 4.1, while the algorithm is described in section 4.2.

4.1 Game Description

We consider an iterative game. Here, we define an iteration as a round in which all 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 i such that $i \in N$. For each player i, let $S_k^i \subseteq M^i$ be the strategy space at iteration k. Let $u_k^i(p)$ be the utility of sensor i to choose sensor p as its parent in iteration k. Further,

$$u_k^i(p) = C_k^p \tag{1}$$

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* $u_k^i(p)$ to child i for all iterations after k so long as i is a child of p.

Let

$$\begin{aligned} u_k^i &= \text{Max}_{p \in S_k^i}\{u_k^i(p)\} \\ &= \text{Max}_{p \in S_k^i}\{C_k^p\} \end{aligned} \tag{2}$$

This utility function in Eqn. 2 dictates that at each iteration, a sensor prefers to connect to a parent that gives it the maximum bandwidth guarantee. i.e. each sensor is "selfish" about the bandwidth guarantee it receives. At every iteration, each parent p gives equal bandwidth guarantees to its children.

In section 4.2, we describe the algorithm that constructs a load-balanced tree in the presence of "selfish" sensors.

4.2 Algorithm for Load Balanced Tree Construction

In this section, we describe an iterative distributed algorithm for the load balanced tree construction. This algorithm assumes that each sensor's decision making is governed by the utility function described in Eqn. 2 in section 4.1. Before describing the algorithms of the parent and child, we introduce a few notations and definitions. Let

1. deg(j) be the degree of a node j in the bipartite graph described in section 3.
2. N_k^p be the set of children currently attached to parent p at the end of iteration k.
3. n_k^p be the number of **new** children that wish to attach to parent p at iteration k.
4. a_k^p be the number of **new** children that actually attach to parent p at iteration k[1].
5. y_k^p be the number of additional children that parent p can take at iteration k, giving a bandwidth guarantee of C_k^p. Initially $y_0^p = deg(j)$ (as $C_0^p = \frac{1}{deg(p)}$)
6. z_p^k be the number of children that leave parent p at iteration k.
7. P_k^i be the parent of node i at iteration k.
8. B_k^p be the total bandwidth guaranteed by parent p at iteration k. i.e.

$$B_k^p = |N_k^p| C_k^p \text{ if } |N_k^p| > 0 \tag{3}$$

$$= C_k^p \text{ otherwise} \tag{4}$$

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

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.

We now describe the algorithms of the **Child** and the **Parent**.

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 possible bandwidth guarantee. If there are multiple such parents, the child will indicate a willingness to connect to all of them[2]. After it has been chosen by one parent, the child node will inform the other parents that it does not wish to connect to them.

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

[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.

Parent: Essentially, until a parent p saturates, it successively increases its bandwidth guarantees from $\frac{1}{deg(p)}, \frac{1}{deg(p)-1}, \frac{1}{deg(p)-2}$, so on to 1. 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 */

$y_{k+1}^p \leftarrow \frac{1}{C_{k+1}^p} - |N_k^p|$.

$x_{k+1}^p \leftarrow \min\{n_{k+1}^p, y_{k+1}^p\}$
$|N_{k+1}^p| \leftarrow |N_k^p| + a_{k+1}^p - z_{k+1}^p$.
If $(B_{k+1}^p = 1)$\{
exit
}

/* If some child left or a child joined
don't increase the bandwidth guarantee */

If $(z_{k+1}^p > 0)$ or $(n_{k+1}^p > 0)$ {
$C_{k+1}^p \leftarrow C_k^p$
}else{
$C_{k+1}^p \leftarrow \frac{1}{\frac{1}{C_k^p}-1}$.
}
$k \leftarrow k+1$
}

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

4.3 Analysis

In this section, we analyze the running time of the algorithm. We show that at termination, a Nash Equilibrium exists. We conjecture that the Nash Equilibrium coupled with the termination condition mentioned at the end of section 4.2 results in an optimally load balanced tree (or max-min allocation of bandwidths to the children).

Theorem 1. *The total number of iterations taken by the distributed algorithm is $O(N\gamma)$, where γ is the maximum degree of a parent in the graph G. At termination, a Nash Equilibrium exists.*

Proof:
In every iteration, at least one parent increases its bandwidth guarantee. Moreover, each parent can increase it bandwidth guarantee at most γ times.
On termination at iteration k^*, all parents are saturated i.e. $\forall p$, $B_{k^*}^p = 1$. Thus, the strategy space of every child i and iteration k^*, $S_{k^*}^i = \phi$. Hence, by definition, a Nash Equilibrium exists at termination.
□

We are also interested in proving the correctness of the algorithm. We are in the process of working out the details of the proof. However, we outline the preliminary progress made in this direction.

As mentioned in section 2, a load balanced tree corresponds to an optimal semi-matching. Harvey *et al.* characterize an optimal semi-matching using the notion of cost reducing paths (defined below) [3].

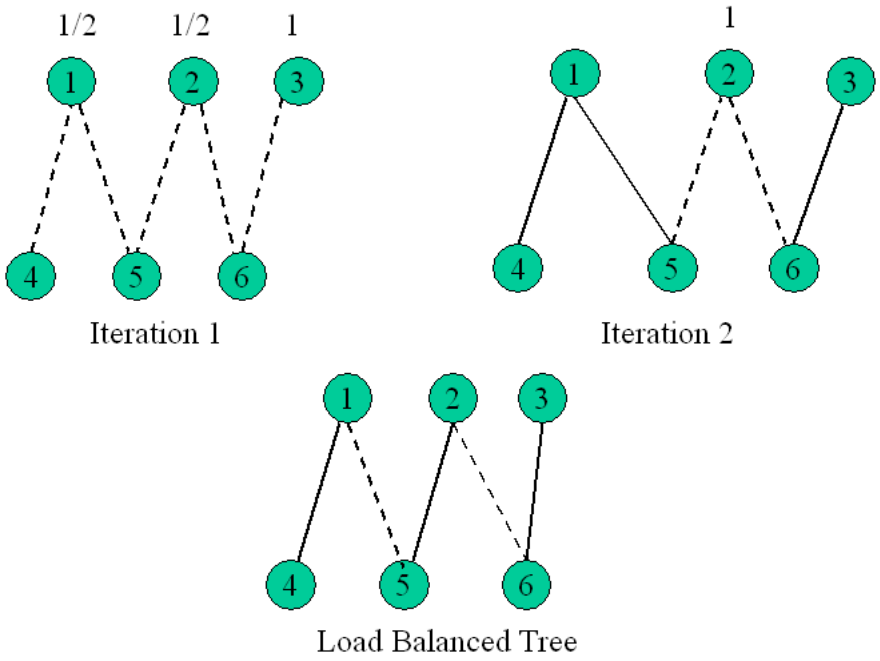

Figure 1. An illustration of the execution of the algorithm. The matched edges are marked as solid lines. The set of parents $M = \{1, 2, 3\}$, while the set of children $N = \{4, 5, 6\}$. The bandwidth guarantee offered by each parent is shown on top of the parent.

Definition 2. *Given a semi-matching S in $G = (V, E)$, a cost reducing path R is a sequence of alternating matched and unmatched edges given by $R = \{(v_1, u_1), (u_1, v_2), ...(u_{z-1}, v_z)\}$ 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 matching S.*

Moreover, the authors of [3] also prove the following:

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

We are interested in proving the following:

Conjecture 1. On termination at iteration k, the algorithm produces a load balanced tree (or an optimal semi-matching) i.e. the allocation of bandwidths $x_k = (x_k^1, x_k^2, ...x_k^N)$ is max-min fair.

For this purpose, we define a simple cost reducing path as follows:

Definition 3. *A simple cost reducing path is a cost reducing path that does not contain any other cost reducing path within it.*

Theorem 3. *If at each iteration, no simple cost reducing path terminates at a saturated parent, the algorithm on termination, results in a load-balanced tree (optimal semi-matching).*

Proof: On termination, all parents are saturated, implying no simple cost reducing paths can exist in the resulting matching and hence by theorem 2, the semi-matching produced is optimal. □

Moreover, simple cost reducing paths terminating at a saturated parent have the following definitive characteristic:

Theorem 4. *Given a semi-matching S in $G = (V, E)$, a simple cost reducing path $R = \{(v_1, u_1), (u_1, v_2), ...(u_{z-1}, v_z)\}$ where v_z is saturated should satisfy the following conditions:*

1. $k > 0$ i.e. the path should have at least one intermediate parent between v_1 and v_z.
2. $\forall i : i \neq 1, z : deg_S(v_i) = deg_S(v_z) + 1$

3. $deg_S(v_1) = deg_S(v_z) + 2$.

Proof:

1. If v_z is saturated with x children ($deg_S(v_z) = x$), v_{z-1} can have at most $x+1$ children ($deg_S(v_{z-1}) \leq x+1$). This can be proved by contradiction.
   Let $deg_S(v_{z-1}) > x+1$. Consider the child u_{z-1} such that $(v_{z-1}, u_{z-1}) \in E$ and $(u_{z-1}, v_z) \in E$. Let this child be attached to parent v_{z-1} in the current matching S, implying that u_{z-1} never got an opportunity to attach to a parent that offered it a bandwidth greater than or equal to $\frac{1}{x+1}$.
   At some iteration k, v_z would not have been saturated and hence increased its bandwidth guarantee from $\frac{1}{x+1}$ to $\frac{1}{x}$. i.e. $C_{k-1}^{v_z} = \frac{1}{x+1}$. Thus, at iteration $k-1$, u_{z-1} had at least one unsaturated parent v_z offering a bandwidth guarantee equal to $\frac{1}{x+1}$. This contradicts the claim that u_{z-1} never got an opportunity to attach to a parent that offered it a bandwidth greater than or equal to $\frac{1}{x+1}$.
   Thus, $deg_S(v_{z-1}) - deg_S(v_z) \leq 1$. Thus, v_{z-1} cannot be the starting vertex of the cost reducing path R, because the $deg_S(v_1) - deg_S(v_z) > 2$.
2. Consider v_i such that $i \neq 1, z$. Now, $deg_S(v_1) - deg_S(v_z) > 2$. Hence, $deg_S(v_i) \geq deg_S(v_z) + 1$. Otherwise if $deg_S(v_i) \leq deg_S(v_z)$, then $deg_S(v_1) - deg_S(v_i) \geq 2$. Thus, R contains a simpler cost reducing path that terminates at v_i.
   Also, $deg_S(v_i) \leq deg_S(v_z) + 1$. Otherwise, if $deg_S(v_i) - deg_S(v_z) \geq 2$, R contains a simpler cost reducing path from v_i to v_z.
   Hence $\forall i : i \neq 1, z$, $deg_S(v_i) = deg_S(v_z) + 1$.
3. For R to be a cost reducing path $deg_S(v_1) - deg_S(v_z) \geq 2$. If R is a simple cost reducing path terminating at a saturated parent v_z, $deg_S(v_1) - deg_S(v_z) = 2$. Otherwise, if $deg_S(v_1) - deg_S(v_z) > 2$, implies $deg_S(v_1) > (1 + deg_S(v_z)) + 1$. Thus, $deg_S(v_1) > deg_S(v_i) + 1$. Hence, R contains a simpler cost reducing path that terminates at one of the intermediate parents v_i, $i \neq z$ because, $deg_S(v_i) = deg_S(v_z) + 1$.

□

We are currently working on proving that such simple cost reducing paths do not arise during the execution of our algorithm.

Our confidence in conjecture 1 is increased by the simulations that we describe next.

5 Simulations

In this section, we compare the tree resulting from an implementation of our distributed algorithm and the centralized algorithm for producing an optimally load balanced tree described in [3]. In our simulation scenarios, sensors were randomly distributed in a 500 x 500 m area. There were a total of 7 parents and 30 children. In the first set of simulations (using 8 different random seeds), the transmission range $R = 250$m. In the second set simulations (using 8 different random seeds), $R = 200$m. In all these simulations, we observed that the tree produced by the distributed algorithm and the centralized algorithm were similar. i.e. the number of parents of degree x were same in both cases.

6 Conclusions & 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 sensors as "selfish" (using appropriate utility functions) enables the design of distributed algorithms for optimizing the network performance as a "whole". This is done for the case study of constructing a load-balanced data gathering tree in a sensor network.

As part of future work, we would like to prove the correctness of our algorithm. It would also be interesting to extend our mechanism of load balancing for more general scenarios that account for heterogeneity in the energy of the sensors, quality of the links, 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.

References

1. N. Nisan and A. Ronen. Algorithmic mechanism design. In Proc. 31śt ACM Symposium of Theory of Computing, 129-140, 1999.
2. R. Kannan, S. Sarangi, S. Sitharama Iyengar and L. Ray. Sensor Centric Quality of Routing in Sensor Networks, IEEE INFOCOM 2003.
3. 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.
4. 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.
5. 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.
6. C. Zhou and B. Krishnamachari, "Localized Topology Generation Mechanisms for Self-Configuring Sensor Networks," *IEEE Globecom*, San Francisco, December 2003.
7. J. Byers and G. Nasser. Utility-Based Decision-Making in Wireless Sensor Networks. IEEE MOBIHOC 2000.
8. P. Berenbrink, F. Heide and K. Schroder. Allocating Weighted Jobs in Parallel. ACM Symposium on Parallel Algorithms and Architectures. p 302-310, 1997.