

Maximum lifetime data sensing and extraction in energy constrained networked sensor systems[☆]

Bo Hong^{a,*}, Viktor K. Prasanna^b

^aDrexel University, Philadelphia, PA 19004, USA

^bUniversity of Southern California, Los Angeles, CA 90089, USA

Received 21 December 2004; received in revised form 12 October 2005; accepted 15 October 2005

Available online 27 December 2005

Abstract

We focus on data gathering problems in energy constrained networked sensor systems. The system operates in rounds where a subset of the sensors generate a certain number of data packets during each round. All the data packets need to be transferred to the base station. The goal is to maximize the system lifetime in terms of the number of rounds the system can operate. We show that the above problem reduces to a restricted flow problem with quota constraint, flow conservation requirement, and edge capacity constraint. We further develop a strongly polynomial time algorithm for this problem, which is guaranteed to find an optimal solution. We then study the performance of a distributed shortest path heuristic for the problem. This heuristic is based on self-stabilizing spanning tree construction and shortest path routing methods. In this heuristic, every node determines its sensing activities and data transfers based on locally available information. No global synchronization is needed. Although the heuristic cannot guarantee optimality, simulations show that the heuristic has good average case performance over randomly generated deployment of sensors. We also derive bounds for the worst case performance of the heuristic.

© 2005 Elsevier Inc. All rights reserved.

Keywords: Sensor networks; Data gathering; System lifetime; Network flow; Flow decomposition; Energy constraints

1. Introduction

A major application of the networked sensor systems is to periodically monitor the environment, such as vehicle tracking and classification in the battle field, patient health monitoring, pollution detection, etc. The basic operation in such applications is to sense the environment and eventually transmit the sensed data to the base station for further processing. The data gathering processes are usually implemented using multi-hop packet transmissions, since short-range communications can save energy and also reduce communication interferences in high density networked sensor systems [3].

Although the applications may be designed for different functionalities, the underlying data gathering processes share

a common characteristic: compared with sensing and computing, transferring data between the sensors is the most expensive (in terms of energy consumption) operation [3]. Techniques ranging from hardware development [15], MAC/network layer design [5,1], and application level optimization [6,16] have been proposed to improve the energy efficiency of networked sensor systems. In this paper, we consider the class of applications where the system works in rounds as in [10]. The event is sensed by a subset of the sensors, each of which generate certain number of data packets during each round and all the sensed data needs to be collected by the base station. The goal is to maximize the number of rounds that the system can operate.

Some variations of data gathering problems have been studied recently. In [13], a non-linear programming formulation is proposed to explore the trade-off between energy consumed and the transmission bandwidth. The radio transmission energy is modeled using Shannon's theorem. In [14], the data gathering problem is formulated as a linear programming problem and a $1 + \omega$ approximation algorithm is proposed. This algorithm further leads to a distributed heuristic. More closer to our

[☆] Supported by the National Science Foundation under award no. IIS-0330445 and in part by DARPA under contract F33615-02-2-4005. A preliminary version of this paper appeared in the Workshop on Algorithms for Wireless and Ad-hoc Networks 2004 (ASWAN 2004).

* Corresponding author.

E-mail address: bohong@coe.drexel.edu (B. Hong).

work is study in [10], where data gathering is also assumed to be performed in *rounds* and every sensor generates exactly one data packet per round. In [10], each sensor can communicate (in a single hop) with the base station and all other sensors. The total number of rounds is then maximized under a given energy constraint on the sensors.

Our study departs from the above with respect to the problem definition, system modeling, as well as the solution technique. The studies in [13,14,10] differentiate the energy dissipated for sending data packets to different neighbors. However, for short-range communications, the difference in the energy consumption when sending data packets to different neighbors is almost negligible. This is because a fixed amount of energy is consumed by the radio antenna to cover the (short) communication range. As long as the receiving sensor is within this range, it can receive data packets. We adopt the reasonable approximation that sending a data packet consumes a fixed amount of energy [3] of a sensor. Additionally, in [10], the system is assumed to be fully connected. This is inevitably based on long-range communications. Although it can be debated whether long-range or short-range communication is more suitable for wireless networked sensor systems, short-range communication is almost certainly, from interference avoidance and frequency re-use perspective, the only feasible solution when the sensors are densely deployed. In our sensor network model, each sensor communicates with a limited number of neighbors due to the short range of the communications, resulting in a general graph topology for the system.

By modeling the energy consumption associated with each send and receive operation, we formulate the data gathering problem as a restricted flow optimization problem. We show that the maximization of the number of rounds reduces to a modification of the standard network flow problem that has quota constraint (details can be found in Section 3) on the nodes. We develop an $O(|V_C| \times |V| \times |E|^2 \times \log(\min\{B_u/(S_u + T_u) | u \in V_C\}))$ algorithm for this problem, where $|V|$ is the total number of sensors, $|E|$ is the number of communication links, V_C is the set of sensors that collect data packets from the environment and $|V_C|$ is the number of such sensors, B_u is the energy budget of sensor u , S_u is the energy cost for u to sense one data packet, and T_u is the energy cost for u to send out one data packet.

While the above algorithm finds the optimal solution, it is centralized. We then develop a distributed heuristic for the data gathering problem where every node determines its sensing activities and data transfers based on locally available information. This heuristic is based on self-stabilizing spanning tree construction algorithms and the shortest path routing method. Although the heuristic cannot guarantee optimality, simulations show that the heuristic has good average case performance over randomly generated deployment of sensors.

The rest of the paper is organized as follows. The system model and problem formulation are discussed in Section 2. In Section 3, we reduce the maximization of rounds in data gathering to a restricted flow problem. In Section 4, we present our algorithm for the restricted flow problem and prove its optimality. In Section 5, we reconstruct the data flow for each round from the solution we find in Section 4. Section 6 compares the

worst case performance of our algorithm against that in [10]. Section 7 discusses the distributed heuristic, examines its worst case performance, and presents simulations results on its average case performance. Section 8 concludes this paper.

2. System model and problem statement

2.1. Model of networked sensor system

Suppose a network of sensors is deployed over a region. The location of the sensors are fixed and known a priori. The networked sensor system is represented by a graph $G(V, E)$, where V is the set of sensor nodes. $(u, v) \in E$ if $u \in V$, $v \in V$ and u is within the communication radius of v . σ_u , the set of successors of u , is defined as $\sigma_u = \{v \in V | (u, v) \in E\}$. Similarly, ψ_u , the set of predecessors of u is defined as $\psi_u = \{v \in V | (v, u) \in E\}$. The event is sensed by a subset of sensors $V_C \subset V$. t is the base station to which the sensed data are transmitted. Sensors $V - V_C - \{t\}$ in the network do not sense the event but can relay the data sensed by V_C . We further assume that no data aggregation is performed during the transmission of the data.

Data transfers are assumed to be performed via multi-hop communications where each hop is a short-range communication. This is due to the well-known fact that long-distance wireless communication is expensive in terms of both implementation complexity and energy dissipation, especially when using the low-lying antennae and near-ground channels typically found in sensor networks [3]. Additionally, short-range communication enables efficient spatial frequency re-use in sensor networks [3].

Each sensor $u \in V$ has an energy budget B_u . Base station t is assumed to have unlimited energy supply. Our energy model for radio transmissions of the sensors is based on the first order radio model described in [9]. If v is within the communication radius of u , the energy consumed by sensor u to transmit a k -bit data packet to v is $T_u = \varepsilon_{\text{elec}} \times k + \varepsilon_{\text{amp}} \times d_u^2 \times k$, where $\varepsilon_{\text{elec}}$ is the energy required for transceiver circuitry to process one bit of data, ε_{amp} is the energy required per bit of data for transmitter amplifier, and d_u is the communication radius of sensor u . Transmitter amplifier is not needed by u to receive data and the energy consumed by u to receive a k -bit data packet is $R_u = \varepsilon_{\text{elec}} \times k$. The system is considered to be heterogeneous, i.e. $T_u(R_u)$ can be different from $T_v(R_v)$. The energy consumed by sensor u to sense a k -byte data packet from the environment is $S_u = \varepsilon_{\text{sen}} \times k$, where ε_{sen} is the energy required for sensing circuitry to collect one bit of data from the environment.

2.2. Problem statement

We consider the class of applications where the system operates in rounds. Sensor $u \in V_C$ generates n_u data packets during each round. These data packets need to be gathered by the base station. The total number of rounds that the system can operate is limited by the energy budgets of the sensors as well as the routing of data packets through the system. Our goal is to maximize the number of rounds.

Let $f(u, v)$ denote the number of data packets transferred from u to v . The maximal data gathering (MDG) problem is formulated as follows:

Given: a networked sensor system represented by a graph $G(V, E)$ where each sensor $u \in V$ has energy budget B_u , cost of sending a data packet T_u , and cost of receiving a data packet R_u . $V_c \subset V$ is the set of sensors that collect data from the environment. $v \in V_c$ generates n_v data packets during each round. The cost for $v \in V_c$ to sense a data packet is S_u . Edge $(u, v) \in E$ if v is within the communication radius of u . $t \in V - V_c$ is the base station.

Maximize N

Subject to

- (1) $\sum_{v \in \sigma_u} f(u, v) - \sum_{v \in \psi_u} f(v, u) = N \times n_u$ for $u \in V_c$,
- (2) $\sum_{v \in \sigma_u} f(u, v) = \sum_{v \in \psi_u} f(v, u)$ for $u \in V - V_c - \{t\}$,
- (3) $\sum_{v \in \sigma_u} f(u, v) \times T_u + \sum_{v \in \psi_u} f(v, u) \times R_u + N \times n_u \times S_u \leq B_u$ for $u \in V_c$,
- (4) $\sum_{v \in \sigma_u} f(u, v) \times T_u + \sum_{v \in \psi_u} f(v, u) \times R_u \leq B_u$ for $u \in V - V_c - \{t\}$.

Function f in the MDG problem is called a data flow in G . Variable N in the problem statement represents the number of rounds the system can operate. Condition 1 requires that each sensor $u \in V_c$ generates n_u data packets for each rounds. Condition 2 says that all the intermediate sensors do not generate or drop data packets. Conditions 3 and 4 describe the energy constraints of the sensors. Nodes in V_c can sense the environment as well as relaying data packets for other nodes. Hence they may receive some data packets from the neighbors, as is shown in condition 3.

B_u in the MDG problem models the energy constraints of the sensor nodes. It does not have to be the total remaining energy of u . For example, when the remaining battery power of a sensor is lower than a particular level, the sensor may limit its contribution to the data gathering operation by setting a small value for B_u (so that this sensor still has enough energy for future operations). For another example, if a sensor is deployed in a critical location so that it is utilized as a gateway to relay data packets to a group of sensors, then it may limit its energy budget for a particular data gathering operation, thereby conserving energy for future operations. These considerations can be captured by energy budget B_u in the MDG problem.

Fig. 1 shows an example of the MDG problem. The system consists of nine nodes. $V_c = \{a, b, c\}$. Node t is the base station. Energy budgets are marked on the nodes. For simplicity, we consider $R_u = T_u = 1$ for all $u \in V$, $S_a = S_b = S_c = 1$, and $n_a = n_b = n_c = 1$. This example system can operate over a maximum of two rounds. The paths to route the data packets are illustrated in Fig. 1 as the four dotted lines. After two rounds, nodes a through h will have remaining energy 2.2, 0.3, 1.7, 1.4, 1.2, 0.3, 3.4, 0.3, respectively. Nodes a and c can still sense data and some of the sensed data can be transferred to t . But node b cannot sense any more data packets since it has only 0.3 unit of energy left, which is less than S_b . Even if b can sense, it does not have enough energy to transmit the data packets since $T_b > 0.3$. Since a round is defined as collection of all data packets sensed by all the nodes in V_c , it means that the loss

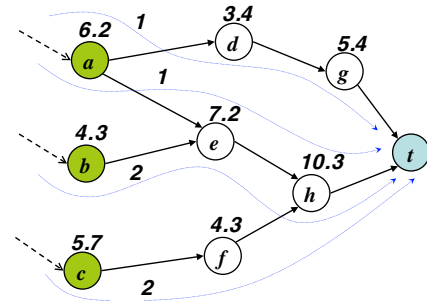


Fig. 1. An example of the MDG problem.

of node b prevents the system from successfully operating any more rounds. Another scenario that the system fails to operate a round occurs when the intermediate nodes fail (due to lack of energy) to transfer all the sensed data to t . This scenario is not shown in this example.

3. Reduction to a restricted flow problem

We can see that the energy budgets (condition 3) in the MDG problem is imposed on the nodes. In this section, we show that energy budgets of the sensors can be transformed to edge capacities.

The MDG problem is an optimization problem of finding the maximum number of rounds N that can be achieved in a given graph. We consider the corresponding decision problem: given a graph $G(V, E)$ and an integer N , can we achieve N rounds in graph G ?

With the existence of condition 1, condition 3 can be rewritten as

$$\sum_{v \in \sigma_u} f(u, v) \times T_u + \left(\sum_{v \in \sigma_u} f(u, v) - N \times n_u \right) \times R_u + N \times n_u \times S_u \leq B_u \quad \text{for } u \in V_c,$$

which is equivalent to

$$\sum_{v \in \sigma_u} f(u, v) \leq (B_u + N \times n_u \times (R_u - S_u)) / (T_u + R_u) \quad \text{for } u \in V_c.$$

With the existence of condition 2, condition 4 can be rewritten as

$$\sum_{v \in \sigma_u} f(u, v) \times T_u + \sum_{v \in \sigma_u} f(u, v) \times R_u \leq B_u \quad \text{for } u \in V - V_c - \{t\},$$

which is equivalent to

$$\sum_{v \in \sigma_u} f(u, v) \leq B_u / (T_u + R_u) \quad \text{for } u \in V - V_c - \{t\}.$$

Then condition 3 and 4 can be represented uniformly as

$$\sum_{v \in \sigma_u} f(u, v) \leq \beta_u \quad \text{for } u \in V - \{t\}, \quad (1)$$

where

$$\begin{aligned} \beta_u &= (B_u + N \times n_u \times (R_u - S_u)) / (T_u + R_u) \quad \text{for } u \in V_c, \\ \beta_u &= B_u / (T_u + R_u) \quad \text{for } u \in V - V_c - \{t\}. \end{aligned}$$

By introducing a pseudo source node s that connects to each node $u \in V_c$, we can state the decision version of the MDG problem as the following restricted flow problem with vertex capacity constraints (RFVC):

Given: a graph $G(V, E)$ with source s and sink t , a number N . Node $u \in V - \{s, t\}$ has capacity constraint β_u . $V_c \subset V - \{s, t\}$. Node $v \in V_c$ generates n_v data packets per round.

Determine: whether there exists a data flow $f : E \rightarrow R$ that satisfies the following conditions:

- (1) $f(s, u) = N \times n_u$ for $u \in V_c$,
- (2) $\sum_{v \in \sigma_u} f(u, v) = \sum_{v \in \psi_u} f(v, u)$ for $u \in V - \{s, t\}$,
- (3) $\sum_{v \in \sigma_u} f(u, v) \leq \beta_u$ for $u \in V - \{s, t\}$.

For the RFVC problem, suppose we split $u \in V - \{s, t\}$ into two nodes u_1 and u_2 , re-direct all incoming links to u to arrive at u_1 and all the outgoing links from u to leave from u_2 , and add a link from u_1 to u_2 with capacity β_u , then the vertex constraint β_u is fully represented by the capacity of link (u_1, u_2) . Actually, such a split transforms all the vertex constraints to the corresponding link capacities and leads to the restricted flow problem with edge capacities (RFEC). The RFEC problem is stated as follows:

Given: a graph $G(V, E)$ with source s and sink t , a number N , and $V_c \subset V - \{s, t\}$. Edge (u, v) has capacity constraint $c(u, v)$. Node $v \in V_c$ generates n_v data packets per round.

Determine: whether there exists a data flow $f : E \rightarrow R$ that satisfies the following conditions:

- (1) $f(s, u) = N \times n_u$ for $u \in V_c$,
- (2) $\sum_{v \in \sigma_u} f(u, v) = \sum_{v \in \psi_u} f(v, u)$ for $u \in V - \{s, t\}$,
- (3) $f(u, v) \leq c(u, v)$ for $u \in V - \{s, t\}$.

Formal proof of the correctness of reduction from RFVC problem to RFEC problem is straightforward and the outline is as follows. (1) Given an instance of the RFVC problem, if there exists a feasible data flow f , then we can construct a feasible solution to the corresponding RFEC problem.

Condition 1 in the RFEC problem is the *quota* constraint, which requires node $u \in V_c$ to generate n_u data packets per round. Condition 2 is the *flow conservation* constraint. It says that nodes do not generate or consume data packets except source s and sink t . (Node $u \in V_c$ does generate data packets in the MDG formulation. But with the introduction of pseudo source s , flow conservation is enforced on u in RFVC and RFEC problems.) Condition 3 is the *edge capacity* constraint.

The RFEC problem is very similar to the standard network flow problem [4], which enforces flow conservation and edge capacity constraints. The RFEC problem differs from the standard network flow problem w.r.t. the additional quota constraint.

We call both RFVC and RFEC problems restricted flow problems due to such a quota constraint.

4. Algorithm for the restricted flow problem

Given a RFEC problem with graph $G(V, E)$ and number N , we define the *source capped* graph induced by N as a graph $G^N(V, E^N)$ where $E^N = \{(u, v) | (u, v) \in E\}$ and the capacity $c^N(u, v)$ of the edge (u, v) in E^N is defined as follows: $c^N(s, u) = N \times n_u$ for $u \in \sigma_s$ and $c^N(u, v) = c(u, v)$ otherwise. Obviously, a feasible solution to a RFEC problem with graph $G(V, E)$ and number N must also be an optimal solution to the standard network flow problem in $G^N(V, E^N)$.

Before introducing the algorithm for the RFEC problem, let us review some notations and concepts for the standard network flow problem. The standard network flow problem is to find a maximum flow from s to t in graph $G(V, E)$, subject to the flow reservation and edge capacity constraints. For notational convenience, $c(u, v) \stackrel{\text{def}}{=} 0$ if $(u, v) \notin E$. If the actual data flow is from u to v , we define $f(v, u) = -f(u, v)$. With the definition of $f(u, v)$ and $c(u, v)$ thus expanded, if neither (u, v) nor (v, u) belongs to E , then $c(u, v) = c(v, u) = 0$, which implies that $f(u, v) = f(v, u) = 0$. In this way, we can define $f(u, v)$ over $V \times V$, rather than being restricted to E . $f(u, v) = -f(v, u)$ also allows us to represent the flow conservation constraint as $\sum_{v \in V} f(u, v) = 0$, which is equivalent to $\sum_{v \in \sigma_u} f(u, v) = \sum_{v \in \psi_u} f(v, u)$. Given a graph $G(V, E)$ and a flow f , the *residual graph* induced by f is a graph $G_f(V, E_f)$, where $E_f = \{(u, v) | u \in V, v \in V, c(u, v) - f(u, v) > 0\}$. Edge (u, v) in E_f has *residual capacity* $c_f(u, v) = c(u, v) - f(u, v)$. An *augmenting path* p is a simple path from s to t in the residual graph G_f . The *residual capacity* of augmenting path p is defined as $c_f(p) = \min\{c_f(u, v) : (u, v) \text{ is on } p\}$. A *cut* of $G(V, E)$ is a binary partition (S, T) of V such that $s \in S, t \in T$. The *capacity* of a cut (S, T) is defined as $c(S, T) = \sum_{u \in S, v \in T} c(u, v)$.

Our algorithm (denoted as the RFEC algorithm) for the RFEC problem is as follows:

1. $f(u, v) = 0$ for $\forall u, v \in V$
2. **for** each $u \in \sigma_s$
3. **while** $f(s, u) < N \times n_u$
4. find the shortest augmenting path p that has (s, u) as the first hop
5. **if** such a path p does not exist
6. **return** FAIL and **exit** the algorithm
7. **else**
8. $d = \min\{c_f(p), N \cdot n_u - f(s, u)\}$
9. **for** each edge (u, v) in p
10. $f(u, v) \leftarrow f(u, v) + d$
11. $c_f(u, v) \leftarrow c_f(u, v) - d$
12. **end for**
13. **end if**
14. **end while**
15. **end for**
16. **return** SUCCESS

Theorem 1. Given a RFEC problem with graph $G(V, E)$ and number N , the RFEC algorithm returns SUCCESS iff there exists a feasible data flow for the RFEC problem.

Proof. \Rightarrow : If the RFEC algorithm returns SUCCESS, the values of $f(u, v)$ upon termination of the algorithm actually constructs a feasible flow for the RFEC problem.

\Leftarrow : Suppose there exists a feasible data flow f for the given graph $G(V, E)$ and number N . It is easy to verify that f is also a maximum flow from s to t in $G^N(V, E^N)$. According to maximum-flow minimum-cut theorem [4], this maximum flow implies a minimum-cut $(s, V - \{s\})$ of $G^N(V, E^N)$ whose capacity is $N \times \sum_{u \in \sigma_s} n_u$.

Assume for the sake of contradiction that the RFEC algorithm returns FAIL. Without loss of generality, assume that the algorithm returns FAIL when checking $u^* \in \sigma_s$, i.e. when the algorithm cannot find any augmenting paths that has (s, u^*) as the first hop while $f(s, u^*)$ is still less than $N \times n_{u^*}$.

Now we consider the source capped graph $G^N(V, E^N)$. Upon termination of the algorithm, data flow f in $G(V, E)$ actually constructs a flow (not necessarily a maximum flow) in $G^N(V, E^N)$.

We construct a cut (S, T) of $G^N(V, E^N)$ as follows:

$S = \{s\} \cup \{v \mid \text{there exists a path } p \text{ from } s \text{ to } v \text{ in } G_f^N,$

and the first hop of p is $(s, u^*)\}$,

$T = V - S$.

G_f^N in the above equation denotes the residual graph of G^N , induced by f .

We claim that $t \notin S$. Otherwise there exists an augmenting path from s to t in G_f (and also in G_f^N) with (s, u) as the first hop, then the test at line 5 of the algorithm will be valid and the algorithm will not fail. The fact that $t \notin S$ implies that $t \in T$, which means that (S, T) is indeed a cut of G_f^N .

Let $S' = S - \{s\}$. We claim that $f(u, v) = c(u, v)$ for $\forall u \in S', v \in T$. Otherwise $f(u, v) < c(u, v)$ implies an edge (u, v) in G_f^N , which further implies the existence of a path from s to v ($s \rightarrow \dots \rightarrow u \rightarrow v$) in G_f^N . But this contradicts the assumption that $v \in T$.

Because flow conservation is always satisfied as we push data packets along the augmenting paths,

$$\begin{aligned} 0 &= \sum_{v \in V} \sum_{u \in S'} f(v, u) \\ &= \sum_{u \in S'} f(s, u) + \sum_{v \in T} \sum_{u \in S'} f(v, u) \\ &\quad + \sum_{v \in S'} \sum_{u \in S'} f(v, u) \\ &= \sum_{u \in S'} f(s, u) + \sum_{v \in T} \sum_{u \in S'} f(v, u) \\ &= \sum_{u \in S'} f(s, u) - \sum_{v \in T} \sum_{u \in S'} f(u, v), \end{aligned}$$

which means that

$$\begin{aligned} \sum_{u \in S'} f(s, u) &= \sum_{u \in S'} \sum_{v \in T} f(u, v) \\ &= \sum_{u \in S'} \sum_{v \in T} c(u, v). \end{aligned}$$

The capacity of cut (S, T) is calculated as follows:

$$\begin{aligned} c(S, T) &= \sum_{u \in S, v \in T} c(u, v) \\ &= \sum_{v \in T} c(s, v) + \sum_{u \in S', v \in T} c(u, v) \\ &= \sum_{v \in T} c(s, v) + \sum_{u \in S'} f(s, u) \\ &= \sum_{v \in \{\sigma_s \cap T\}} c(s, v) + \sum_{u \in S'} f(s, u) \\ &= \sum_{v \in \{\sigma_s \cap T\}} n_v + \sum_{u \in \{\sigma_s \cap S'\}} f(s, u). \end{aligned}$$

According to edge capacity constraint,

$$f(s, u) \leq c(s, u) = n_u \quad \text{for } u \in \sigma_s.$$

Particularly, for node u^* , which is by definition in $\sigma_s \cap S'$, we have

$$f(s, u^*) < c(s, u^*) = n_{u^*}.$$

Therefore,

$$\begin{aligned} c(S, T) &= \sum_{v \in \{\sigma_s \cap T\}} n_v + \sum_{u \in \{\sigma_s \cap S'\}} f(s, u) \\ &< \sum_{v \in \{\sigma_s \cap T\}} n_v + \sum_{u \in \{\sigma_s \cap S'\}} c(s, u) \\ &= \sum_{v \in \{\sigma_s \cap T\}} n_v + \sum_{u \in \{\sigma_s \cap S'\}} n_u \\ &= \sum_{u \in \{\sigma_s \cap (S \cup T)\}} n_u \\ &= \sum_{u \in \sigma_s} n_u. \end{aligned}$$

This is impossible since we have shown that the minimum cut of G_f^N has capacity $N \times \sum_{u \in \sigma_s} n_u$. The capacity of cut (S, T) cannot be smaller than that of the minimum cut. \square

Does the RFEC algorithm always terminate? By terminate, we mean that the algorithm either returns FAIL at line 6 or returns SUCCESS at line 16. The answer is yes. It can be seen that the while loop in the RFEC algorithm consisting of line 3–14 is pushing data packets along the shortest augmenting path whose first hop is (s, u) . Similar to the complexity analysis of the Edmonds–Karp algorithm [4], it can be shown easily that the complexity of the while loop is $O(|V| \times |E|^2)$. The complexity of the RFEC algorithm is therefore $O(|\sigma_s| \times |V| \times |E|^2)$.

Note that σ_s in the RFEC problem is mapped from V_c in the MDG problem. The mapping procedure can be completed in $O(|V| + |E|)$ time. Hence the decision version of the MDG problem can be solved in $O(|V_c| \times |V| \times |E|^2)$ time. We can apply binary search to find the maximum value of N for the original MDG problem. Because $\min\{B_u / (S_u + T_u) \mid u \in V_c\}$ is an obvious upper bound for N , the maximum number of rounds in the MDG problem can be found in $O(|V_c| \times |V| \times |E|^2 \times \log(\min\{B_u / (S_u + T_u) \mid u \in V_c\}))$ time.

In the RFVC problem, the capacity β_u of vertex u restricts the maximum number of the data packets that u can transfer. There can only be integer number of data packets. Hence β_u is

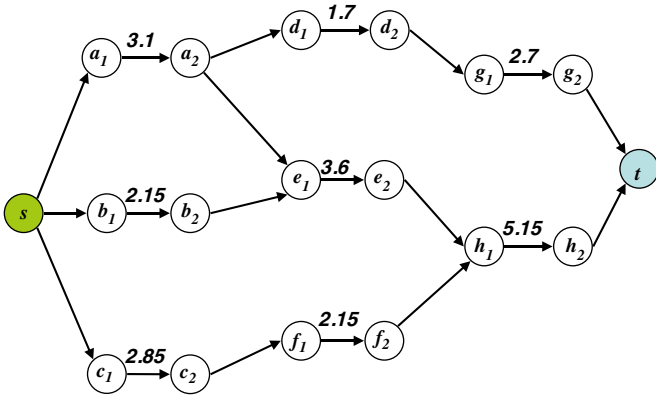


Fig. 2. An illustrative example of the RFEC algorithm.

effectively equivalent to $\lfloor \beta_u \rfloor$ where $\lfloor \beta_u \rfloor$ represents the largest integer smaller than or equal to β_u . Consequently, when mapping such a RFVC problem to a RFEC problem, a real-valued $c(u, v)$ is effectively equivalent to $\lfloor c(u, v) \rfloor$. It is well known that flow maximization using augmenting paths, which includes the scenario of our RFEC algorithm, generates integer valued solutions when the edge capacities are integers. Therefore, if the data packets are atomic and cannot be further divided, our algorithm is guaranteed to find integer valued optimal solution for the RFEC, and hence the RFVC and MDG problems.

We illustrate the execution of the RFEC algorithm using the example system in Fig. 1 where $V_c = \{a, b, c\}$. Remember that we consider $R_u = T_u = 1$ for $u \in \{a, b, c, d, e, f, g, h\}$, $S_u = 1$ and $n_u = 1$ for $u \in \{a, b, c\}$. $\min\{B_u/(S_u + T_u) | u \in V_c\} = 2.15$, which is the upper bound on the number of rounds. Hence we start the binary search with $N = 2$.

The first step is to transform the MDG problem formulation in Fig. 1 to the RFEC formulation. The result is shown in Fig. 2 where each node $u \in \{a, b, c, d, e, f, g, h\}$ in the MDG formulation is split into two nodes u_1 and u_2 , and a pseudo source node s is added. Weight of the newly added edge (u_1, u_2) is calculated according to Eq. (1). The value of n_u ($u \in \{a, b, c\}$) in the MDG formulation is inherited by n_u ($u \in \{a_1, b_1, c_1\}$) in the RFEC formulation, i.e. $n_u = 1$ for $u \in \{a_1, b_1, c_1\}$.

In order to check if the system can successfully operate two rounds, the RFEC algorithm is executed. After initialization, suppose in line 2, the procedure chooses to check node a_1 first. The algorithm then attempts to push $N \times n_{a_1} = 2$ units of flow from s to t along augmenting paths that have (s, a_1) as the first hop. One of the possible augmenting paths is $s \rightarrow a_1 \rightarrow a_2 \rightarrow e_1 \rightarrow e_2 \rightarrow h_1 \rightarrow h_2 \rightarrow t$. Suppose this path is chosen and 2 units of flow is pushed along this path. This fulfills the request of node a_1 . After pushing this flow, edges (a_1, a_2) , (e_1, e_2) , and (h_1, h_2) have remaining capacities 1.1, 1.6, and 3.15, respectively. Then, suppose the algorithm chooses to check node b_1 , which is the second neighbor of source s . $N \times n_{b_1} = 2$ units of flow need to be pushed from s to t via b_1 . $s \rightarrow b_1 \rightarrow b_2 \rightarrow e_1 \rightarrow a_2 \rightarrow d_1 \rightarrow d_2 \rightarrow g_1 \rightarrow g_2 \rightarrow t$ is one possible augmenting path from s to t with (s, b_1) being the first hop. Since the capacity of this augmenting path is 1, the algorithm can successfully push 1 units of flow along this

path. We have another augmenting path $s \rightarrow b_1 \rightarrow b_2 \rightarrow e_1 \rightarrow e_2 \rightarrow h_1 \rightarrow h_2 \rightarrow t$, along which the algorithm can push 1 unit of flow. Now the request of node b_1 is fulfilled. Similarly, for the third neighbor c_1 of source s , $N \times n_{c_1} = 2$ units of flow can be pushed from s to t along augmenting path $s \rightarrow c_1 \rightarrow c_2 \rightarrow f_1 \rightarrow f_2 \rightarrow h_1 \rightarrow h_2 \rightarrow t$. This completes the check for all the three neighbors of s and the algorithm returns ‘SUCCESS’, which means that the system indeed can operate over a maximum of two rounds. Hence we stop the binary search at $N = 2$. Note that during the execution of the RFEC algorithm, we do not enforce a specific order by which the neighbors of the source node are checked.

5. Reconstructing the data flow for each round

The RFEC algorithm (when used together with binary search) finds the maximum number of rounds N that the system can operate. Besides the total number of rounds N , the RFEC algorithm also finds the flow $f(u, v)$ for each edge (u, v) . The value of $f(u, v)$ specifies the total number of data packets that are transferred along edge (u, v) over the N rounds, but it does not specify how many data packets should be transferred along (u, v) during the i th round, $1 \leq i \leq N$. In this section, we address the problem of reconstructing the data flow for each round, given the result of the RFEC algorithm, $f(u, v)$.

The reconstruction is based on the *flow decomposition* [2] technique, which was originally developed for the standard flow maximization problem. Let us first briefly review the standard flow decomposition problem. Given graph $G(V, E)$ with source s and sink t , and any (not necessarily the maximum) flow $f(u, v)$ in G , the *flow graph* induced by f is defined as graph $G_{[f]}(V, E')$ where $E' = \{(u, v) \in E | f(u, v) > 0\}$. A *flow decomposition* of $G_{[f]}$ is a decomposition of $G_{[f]}$ into a certain number of ‘primitives’. Each primitive is either a simple path p from s to t where the flow along each edge of p is the same, or a simple circle γ where the flow along each edge of γ is the same. It is well known that there can be at most $|V| + |E|$ primitives when decomposing any flow graph $G_{[f]}$ [2].

Given an MDG problem, we have transformed its decision version to the RFEC problem, which enforces the quota constraint that differentiates the RFEC problem from the standard flow problem. The following discussion in this section considers the reconstruction of data flow for each round, given the solution (N and $f(u, v)$, $(u, v) \in E$) to the RFEC problem. The result of reconstruction can then be easily transformed and applied to the original MDG problem. Let $\varphi(p)$ denote the destination of the first hop of path p .

Our objective is to decompose the solution of the RFEC algorithm (in the form of $f(u, v)$ where $(u, v) \in E$) to N sets of paths. The i th set ($i = 1, 2, \dots, N$) corresponds to the i th round of data gathering. It specifies a set of paths $\Pi_i = \{p_{i1}, p_{i2}, \dots\}$ where each path is from s to t , and the data flow (denoted as $\delta(p)$) that should be transferred along each path $p \in \Pi_i$. We require that $\Pi_1 \cup \dots \cup \Pi_N$ is a decomposition of $f(u, v)$ (where $(u, v) \in E$). To construct a valid data flow for the i th round, we require that $\sum_{p \in \Pi_i \& \varphi(p)=u} \delta(p) = n_u$ for $\forall u \in \sigma_s$. The

various paths are not necessarily edge disjoint, i.e. an edge can be a member of multiple paths.

Given $G(V, E)$ and $f(u, v)$, we first construct the flow graph $G_{[f]}$. Then we perform depth first search (DFS) on $G_{[f]}$ and find all the circles. Suppose a circle γ is found, let $\delta(\gamma)$ denote the minimum flow along all the edges of γ . Then for each edge (u, v) on γ , we reduce $f(u, v)$ by $\delta(\gamma)$. In this way, we eliminate one edge from γ and hence break the circle γ . Repeating the above procedure, we can eliminate all the circles in $G_{[f]}$. In the following discussion, for the sake of simplicity, we assume that $G_{[f]}$ is acyclic.

To identify the paths, for each $u \in \sigma_s$, we split edge (s, u) into N edges $(s, u)_1, (s, u)_2, \dots, (s, u)_N$, each going from s to u and having flow $f(s, u)/N = n_u$, (recall that $f(s, u) = N \times n_u$ upon completion of the RFEC algorithm). Flow n_u along path $(s, u)_i$ corresponds to the quota constraint on u in the i th round of data gathering, i.e. node u needs to sense and send out n_u data packets in the i th round. Then, as long as there are edges leaving s in $G_{[f]}$, follow a path p out of s until we reach t (we must reach t because $G_{[f]}$ is acyclic and t is the unique sink node). Add path p thus found to Π_k if the first hop of p is $(s, u)_k$. Let the value of the path, $\delta(p)$, be the minimum flow along all the edges in p . Then we reduce the flow along every edge in p by $\delta(p)$. This eliminates one edge from $G_{[f]}$. We repeat the above procedure until there is no edge out of s in $G_{[f]}$.

Each time a path is added to Π_k ($k = 1, 2, \dots, N$), an edge is eliminated from $G_{[f]}$. Since there can be at most $|E|$ edges in $G_{[f]}$, the above procedure terminates in at most $|E|$ steps.

Next we show that the above procedure, upon termination, finds a reconstruction of the data flow for each round.

Upon termination, there is no edge out of s in $G_{[f]}$. Because all the nodes conserve flow and $G_{[f]}$ is acyclic, all the edges must have been eliminated from $G_{[f]}$. Therefore $\Pi_1 \cup \dots \cup \Pi_N$ is indeed a decomposition of $f(u, v)$ where $(u, v) \in E$.

Now we consider any individual set Π_i . Π_i consists of paths whose first hop is $(s, u)_i$ where $u \in \sigma_s$. Then, upon completion of the above procedure, we must have $\sum_{p \in \Pi_i \& \phi(p)=u} \delta(p) = n_u$ for each $u \in \sigma_s$, i.e. the sum of flows on all paths in Π_i whose first hop is u must be equal to n_u for each $u \in \sigma_s$. Otherwise the sum would be less than n_u , which means that edge $(s, u)_i$ is not eliminated from $G_{[f]}$, contradicting the assumption that the procedure terminated. $\{p | p \in \Pi_i \& \phi(p) = u\}$ is what we are looking for: the set of simple paths that can transfer n_u data packets from s to t . Therefore, we use Π_i to transfer the data packets in the i th round, $i = 1, 2, \dots, N$.

It is interesting to point out that the reconstruction is not unique. For example, the order in which the paths are found and added to $\Pi_1 \cup \dots \cup \Pi_N$ can be arbitrary; Π_i and Π_j can also be exchanged (when $i \neq j$).

We illustrate the above reconstruction procedure using the example system previously shown in Fig. 1.

Given the MDG problem shown in Fig. 1, we have illustrated the corresponding RFEC problem formulation in Fig. 2 and demonstrated the execution of the RFEC algorithm in Section 4. The solution generated by the RFEC algorithm is shown in Fig. 3 in the form of a flow graph, where the flow is marked on

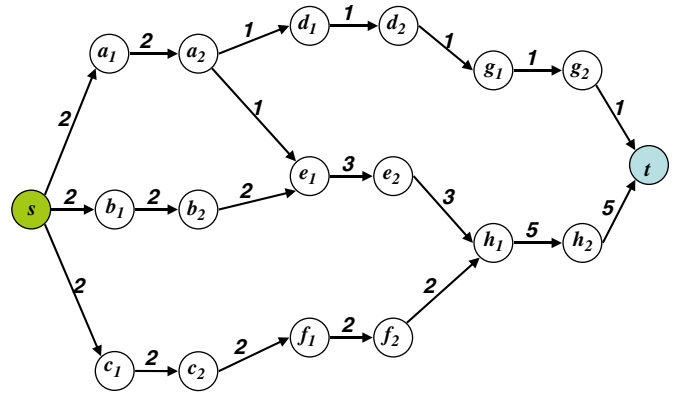


Fig. 3. Flow graph generated by the RFEC algorithm.

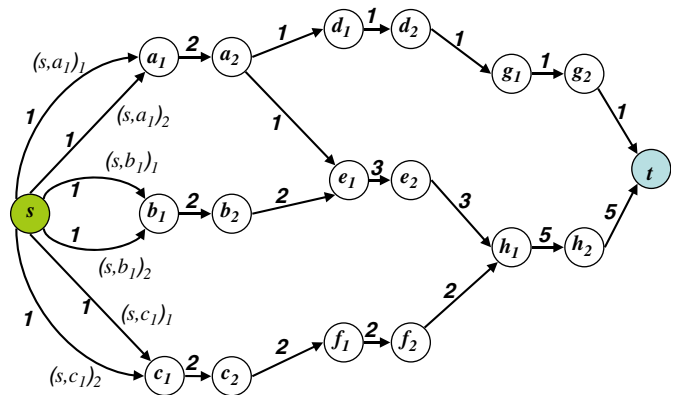


Fig. 4. Reconstructing the flow for each round. Step 1: split the edges out of s .

each edge. Note that the flow along any edge (u, v) indicates the total number of data packets going through (u, v) for all the $N (= 2)$ rounds. Edge (a_2, e_1) does not carry any flow so it does not appear in the flow graph.

It is a coincidence that Fig. 3 is already acyclic. So the removal of circles is skipped here. (Actually, we still need to perform a DFS to verify that the flow graph is indeed acyclic.) Note that the flow graph may not always be acyclic and the removal of circles then becomes necessary.

The next step is to split (s, u) into N edges for each $u \in \sigma_s$. In this example, $\sigma_s = \{a_1, b_1, c_1\}$ and $N = 2$. Therefore, (s, a_1) is split into $N (= 2)$ edges, $(s, a_1)_1$ and $(s, a_1)_2$. $f(s, a_1) = 2$ in Fig. 3 indicates that a total number of two data packets should be transferred from s to a_1 in two rounds. Therefore, both edge $(s, a_1)_1$ and $(s, a_1)_2$ in Fig. 4 have a flow of value $f(s, a_1)/N = n_a = 1$, representing that $n_a = 1$ data packets should be transferred from s to a_1 in the first and the second round. Edges (s, b_1) and (s, c_1) are also split in the same way. The result of the split is shown in Fig. 4. The names of the newly split edges are marked on the edges.

The paths to transfer the data packets are identified as follows:

As long as there are edges leaving s , we follow an arbitrary path till we reach t . Suppose the first path we choose is $s \rightarrow c_1 \rightarrow c_2 \rightarrow f_1 \rightarrow f_2 \rightarrow h_1 \rightarrow h_2 \rightarrow t$ and the first hop is

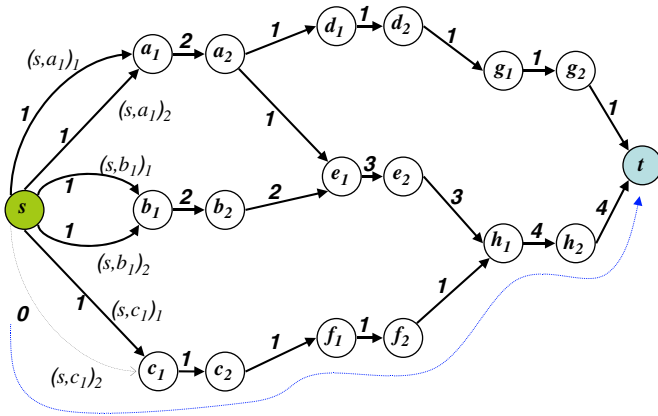


Fig. 5. Reconstructing the flow for each round. Step 2: find the paths.

along edge $(s, c_1)_2$. We add this path to Π_2 since the first hop is $(s, c_1)_2$. The minimum flow along the edges in this path is 1 (edge $(s, c_1)_2$). Therefore, the flow along this path is set to 1 and we reduce the flow along all edges in this path, each by 1. Since $(s, c_1)_2$ does not have any flow after the reduction, it is removed from the flow graph. The result at this stage is shown in Fig. 5. We repeatedly follow the paths from s to t and add the paths to Π_i according to their first hop, until there are no more edges out of s . It is easy to verify that upon termination of the above process, we have the following result:

$$\begin{aligned}
 p_{11} &= s \rightarrow a_1 \rightarrow a_2 \rightarrow d_1 \rightarrow d_2 \rightarrow g_1 \rightarrow g_2 \rightarrow t, \\
 p_{12} &= s \rightarrow b_1 \rightarrow b_2 \rightarrow e_1 \rightarrow e_2 \rightarrow h_1 \rightarrow h_2 \rightarrow t, \\
 p_{13} &= s \rightarrow c_1 \rightarrow c_2 \rightarrow f_1 \rightarrow f_2 \rightarrow h_1 \rightarrow h_2 \rightarrow t, \\
 \Pi_1 &= \{p_{11} |_{\delta(p_{11})=1}, p_{12} |_{\delta(p_{12})=1}, p_{13} |_{\delta(p_{13})=1}\}, \\
 p_{21} &= s \rightarrow a_1 \rightarrow a_2 \rightarrow e_1 \rightarrow e_2 \rightarrow h_1 \rightarrow h_2 \rightarrow t, \\
 p_{22} &= s \rightarrow b_1 \rightarrow b_2 \rightarrow e_1 \rightarrow e_2 \rightarrow h_1 \rightarrow h_2 \rightarrow t, \\
 p_{23} &= s \rightarrow c_1 \rightarrow c_2 \rightarrow f_1 \rightarrow f_2 \rightarrow h_1 \rightarrow h_2 \rightarrow t, \\
 \Pi_2 &= \{p_{21} |_{\delta(p_{21})=1}, p_{22} |_{\delta(p_{22})=1}, p_{23} |_{\delta(p_{23})=1}\}.
 \end{aligned}$$

For the sake of simplicity, we set $n_a = n_b = n_c = 1$ in this example, which means that nodes a , b , and c sense only one data packets in each round. Consequently, a single path is capable of transferring all the data packets sensed by a , b and c in each round. If $n_a > 1$ ($n_b > 1$, or $n_c > 1$), we may need multiple paths to transfer the data packets sensed by a (b , or c) for each round.

6. Performance comparison

In this section, we compare the performance of our algorithm against the method proposed in [10].

The study in [10] considers a similar data gathering problem as the problem studied here. In [10], it is assumed that each sensor generates exactly one data packet in each round, and all the data packets need to be transferred to the base station. The goal is to maximize the number of rounds the system can operate. Such a scenario reduces to the MDG problem. The system model used in [10] is generally the same as our model except for three aspects: the energy for sending a data packet is assumed to be dependent on the distance of the receiving node from the sender, the energy cost of sensing the environment is

ignored, and each sensor generates exactly one data packet per round. While it can be debated which system model accurately represents reality, in order to obtain a fair comparison, we consider the scenario that every sensor generates one data packet in each round, a sensor spends the same amount of energy when sending a data packets to any of its neighbors, and the energy cost sensing the environment is zero. For this scenario, the solution technique in [10] as well as the proposed technique can be applied.

A two stage method is used in [10] to obtain an integer valued solution. In the first stage, a *relaxed* linear programming formulation of the data gathering problem is solved (using some linear programming algorithm). We refer to this approach as the relaxed flow approach. The solution specifies how many data packets $f(u, v)$ should be transferred between sensors u and v (when communication link (u, v) exists). Since the result $f(u, v)$ may not be an integer, the solution is floor rounded to $f'(u, v)$. Note that this floor rounding may compromise the property of flow conservation. So $f'(u, v)$ is only used as the edge capacity constraint, which defines a new flow optimization problem. Then in the second stage, an integer valued solution is found for the new flow optimization problem, using the augmenting path method. It is claimed that the solution produced by such an approach is near-optimal.

Although experimental results in [10] illustrate that the two stage method can achieve close to optimal performance, in the following, we demonstrate the non-optimality of the relaxed flow approach.

Consider the simple example shown in Fig. 6. The system consists of four sensors (a, b, c, d) and the base station t . For a, b, c , and d , the cost of sending a data packet is equal to that of receiving a data packet. The energy budget of the sensors, in terms of the number of data packets that can be received and transferred, are marked by the nodes.

Obviously, the sample system can operate for a maximum number of two rounds (one of the optimal solutions is $f(a, b) = 2, f(c, d) = 2, f(b, t) = 4, f(d, t) = 4, f(a, d) = 0, f(c, b) = 0$). It is easy to verify that the proposed approach generates this solution. However, when we apply the relaxed flow approach, the first stage may generate the following real-valued solution: $f(a, b) = 1.5, f(c, d) = 1.5, f(b, t) = 4, f(d, t) = 4, f(a, d) = 0.5, f(c, b) = 0.5$, which rounds to $f'(a, b) = 1, f'(c, d) = 1, f'(b, t) = 4, f'(d, t) = 4, f'(a, d) = 0, f'(c, b) = 0$. If we define a new flow problem using these rounded values and solve it as in the second stage in [10], the maximum number of rounds that the system can operate, however, is only 1, which is only 50% of the optimal.

In the worst case, the behavior of the relaxed flow approach can be much worse than above. Suppose in Fig. 6, sensors a and c have energy budget 1, b and d have energy budget 2. It is easy to verify that the proposed approach finds the optimal solution. The number of rounds in the optimal solution is 1. The relaxed flow approach, however, may produce the following real-valued solution: $f(a, b) = 0.5, f(c, d) = 0.5, f(b, t) = 2, f(d, t) = 2, f(a, d) = 0.5, f(c, b) = 0.5$, which, after rounding, leads to an integer solution with 0 rounds. We denote a solution technique to have failed if it produces a solution with

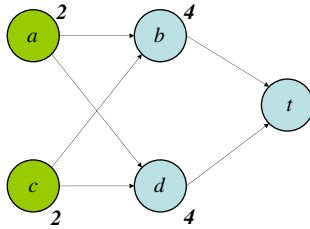


Fig. 6. Example for worst case performance comparison.

zero rounds (i.e. the solution does not gather all the packets in the first round) while there exists a solution with at least one round.

Let Ω represent the number of sensors in the system. The above scenario can be generalized to show:

Theorem 2. *For all $\Omega \geq 4$, there exist instances in which the relaxed technique fails to produce a solution.*

The theorem can be easily proved using the example above.

7. A distributed heuristic for the MDG problem

The proposed RFEC algorithm and the relaxed flow approach are centralized. Both can only be executed in a centralized fashion. However, in an actual deployment of a networked sensor systems, there is no central controller. It is prohibitively expensive, if not impossible, to construct a global view of the system from the distributed nodes and optimize the performance of the system accordingly. The MDG problem, like many other applications of networked sensor systems (e.g. [11]), needs to be solved in a distributed fashion.

Next we study the performance of an distributed heuristic for the MDG problem. In this heuristic, every node determines its activities (sensing and transferring) based on its own local information and the information available at its neighbors. The heuristic is based on the self-stabilizing spanning tree construction algorithms [8] and the widely used shortest path routing method. We denote the heuristic as the shortest path heuristic.

In this heuristic, each node $u \in V$ maintains a variable $d(u)$ which is used to record the distance from u to base station t . Node u also maintains a data buffer, which is used to store sensed or received data. We assume that every node knows an upper bound U on the total number of nodes. We also assume that a new round is triggered by some external event which notifies all the source nodes that a new round starts. (For example, the external event can be the approaching of a vehicle whose acoustic emission activates data sensing and extraction.) The time interval between two rounds is large enough so that the system can complete one round (if possible) before starting a new one. Let $e(u)$ denote the remaining energy of u and $b(u)$ denote the number of data packets in the data buffer of u . The heuristic is described as follows:

- (1) Initially, $d(u) = 0$ for all nodes $u \in V$.
- (2) Each $u \in V - \{t\}$ executes the following two operations:
 - (a) distance update

- (i) if $\min_{v \in \sigma_u, e(v) > R_v} \{d(v)\} < d(u) < U$, then set $d(u) \leftarrow \min_{v \in \sigma_u} d(v) + 1$,

- (ii) if $e(u) < T_u$, the set $d(u) \leftarrow U$,

- (b) data transfer

if $0 < b(u) < U$, $e(u) > T_u$ and $\exists v \in \sigma_u$ s.t. $d(u) = d(v) + 1$, and $b(v) < b(u)$ then u sends one data packet to v (and $e(u)$ reduces by T_u).

- (3) Each node $u \in V_c$ has one additional operation to execute: whenever a new round starts, u senses and puts n_u data packets into its data buffer.

- (4) Since the base station t is assumed to have unlimited energy supply, t simply receives any data packets intended for t .

In the shortest path heuristic, the ‘distance update’ step attempts to establish for each node a shortest path to the base station. Base station t is always at distance 0 (i.e. $d(t) = 0$). For every other node $u \in V - \{t\}$, it chooses the neighbor v with the smallest distance $d(v)$ as its successor and sets its own distance to $d(u) = d(v) + 1$. It can be shown that starting with $d(u) = 0$ for $u \in V$, the distance update step will eventually establish a breadth first search tree. A node simply chooses a neighbor which is one hop closer to the base station as its parent in the tree. This constructs a shortest path to the base station for each node.

As the data gathering proceeds, some nodes may deplete their energy due to sending and receiving data packets. Consequently, for any node, its shortest path to the base station may change. This has also been considered by the distance update step of the heuristic. If a node u does not have enough energy to send a data packet, then it sets its $d(u)$ to the upper bound U , indicating that u cannot reach t anymore. If a node u does not have enough energy to receive a data packet, then it will be excluded from being the parent of any other nodes. Actually, the execution of the distance update step continuously updates the breadth first tree rooted at the base station.

After the breadth first tree is constructed, the nodes start transferring the data packets. Node u will send one data packet to its neighbor v if the following conditions are met: (1) $d(u)$ is less than the upper bound U , (2) u has enough energy to send one more data packet, (3) v is one hop closer to the base station, and (4) v has less data packets in its buffer than u . Conditions 1 and 2 indicate that a node sends out a data packet only if a path to the base station exists. Condition 3 tells a node to send data packets only to its parent in the breadth first tree. Condition 4 prevents a node from receiving too many data packets but cannot deliver these data packets to the base station later on.

Because the shortest path heuristic continuously updates the breadth first tree according to the remaining energy of the nodes, it can be executed while the data transfer is performed. However, it cannot guarantee optimality. This is illustrated using the example system in Fig. 7. The system is shown in the MDG formulation. To simplify the notations and figures, we omit the transformation to RFEC representation and present the following discussions based on this MDG formulation directly. The system consists of 17 nodes $V = \{a_1, a_2, \dots, a_{16}, t\}$. $V_c = \{a_1\}$. $T_u = R_u = 1$ for $u \in V$.

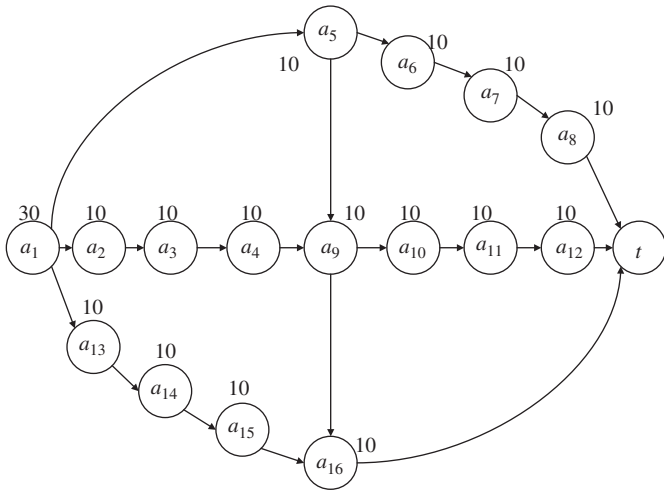


Fig. 7. Example demonstrating the non-optimality of the shortest path heuristic.

$S_{a_1} = 1$. $n_{a_1} = 1$. Energy budgets of the nodes are marked on the nodes. The system can operate over a maximum of 15 rounds. This optimal solution is obtained when $f(a_1, a_5) = f(a_5, a_6) = f(a_6, a_7) = f(a_7, a_8) = f(a_8, t) = 5$, $f(a_1, a_2) = f(a_2, a_3) = f(a_3, a_4) = f(a_4, a_9) = f(a_9, a_{10}) = f(a_{10}, a_{11}) = f(a_{11}, a_{12}) = f(a_{12}, t) = 5$, and $f(a_1, a_{13}) = f(a_{13}, a_{14}) = f(a_{14}, a_{15}) = f(a_{15}, a_{16}) = f(a_{16}, t) = 5$. It can be easily verified that the RFEC algorithm finds this solution.

However, using the shortest path heuristic, the following scenario will occur:

- (1) the distance update step will find $a_1 \rightarrow a_5 \rightarrow a_9 \rightarrow a_{16} \rightarrow t$ as the shortest path connecting a_1 and t ,
- (2) 5 units of data packets are transferred along this path,
- (3) no more data packets can be transferred, so the data gathering terminates.

In the above solution, step (2) utilizes the currently available shortest path to transfer the data packets. It uses up the energy of nodes a_5 , a_9 , and a_{16} . Then no more data can be transferred from a_1 to t because there does not exist any path between a_1 and t . The shortest path heuristic can achieve a total number of five rounds, which is only one third of the optimal. If the shortest path from a_1 to t contains n nodes instead of 3, we can construct a similar example system where the shortest path heuristic achieves only $1/n$ of the optimal.

As before, let Ω denote the total number of nodes. The above scenario can be generalized to show:

Theorem 3. For any real valued $\epsilon \in (0, 1)$, there exists an integer $\Omega_0 > 0$ s.t. for all $\Omega \geq \Omega_0$, there exist problem instance for which the shortest path heuristic produces a solution with system lifetime less than $(\epsilon \times \text{the optimal system lifetime})$.

However, this heuristic has very good average case performance. This is illustrated through the following simulations.

In the simulations, nodes are randomly deployed in a unit square with uniform distribution. The communication radius of all the nodes are assumed to be equal. The base station is

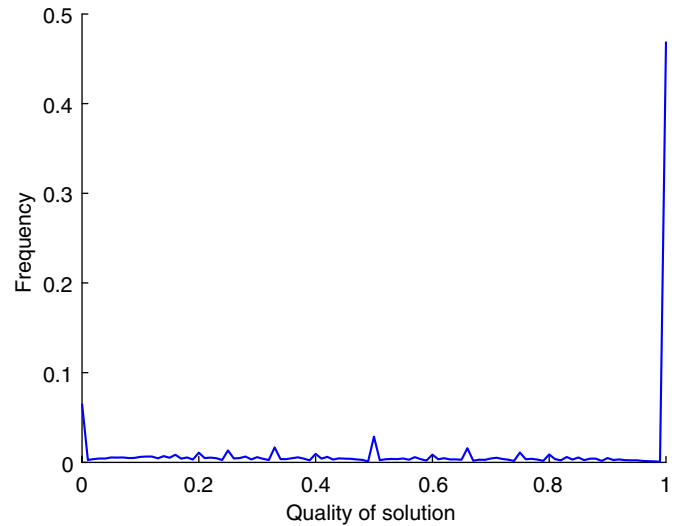


Fig. 8. The histogram of q over all the simulations. The value of q for each simulation is calculated as the ratio between the achieved system lifetime and the optimal lifetime. The y-axis has been normalized to the total number of simulations.

located at the lower left corner of the unit square. S_u , R_u and T_u are uniformly distributed in $[0, 1]$. A certain number of nodes are randomly chosen as the source nodes. The source nodes are not necessarily direct neighbors of each other. For each source node u , the number of data packets to be gathered per round, n_u , is uniformly distributed in $[1, n_{\max}]$ where n_{\max} is a parameter in the simulations. The initial energy at the nodes are uniformly distributed in $[0, e_{\max}]$ where e_{\max} is another parameter.

The following five sets of parameters were studied: (1) $|V|$, the total number of nodes, (2) communication radius of the nodes, (3) $|V_c|$, the number of source nodes (4) n_{\max} , the maximum number of data packets to be gathered per source node per round, and (5) e_{\max} , the maximum amount of energy initially available at the nodes. In the simulations, $|V|$ was selected from 40, 80, 120, and 160. Communication radius was selected from 0.2, 0.3, 0.4, and 0.5. $|V_c|/|V|$ was selected from 0.1, 0.2, 0.3, and 0.4. n_{\max} was selected from 5, 10, 15, and 20. And e_{\max} was selected from 1000, 2000, 3000, and 4000. In summary, there were 1024 combinations of the parameters. For each of the 1024 combinations, we simulated 200 randomly generated systems.

For each system simulated, we calculated the ratio between the lifetime achieved by the shortest path heuristic and the maximum lifetime obtained through the RFEC algorithm. Let q denote this ratio. Summarizing over all the simulation results, we observed that q has mean value 0.68 with standard deviation 0.36. Fig. 8 shows the histogram of q over the 20 4800 experiments. In 47% of the experiments, the heuristic achieved the maximum system lifetime. For the remaining 53% of the experiments, the quality of solution, represented by q , was roughly uniformly distributed between 0 and 1.

To identify possible directions to further improve the performance of the heuristic, we studied the impact of each individual parameter. The results are represented in Figs. 9–13.

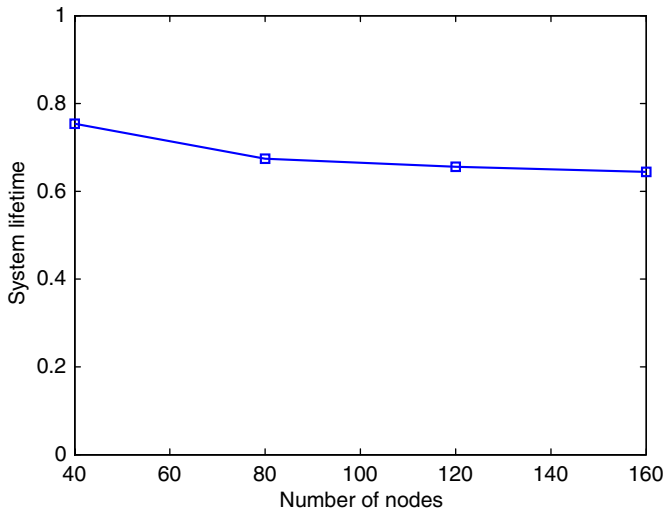


Fig. 9. The impact of the number of nodes on system lifetime. The y-axis has been normalized to the optimal system lifetime.

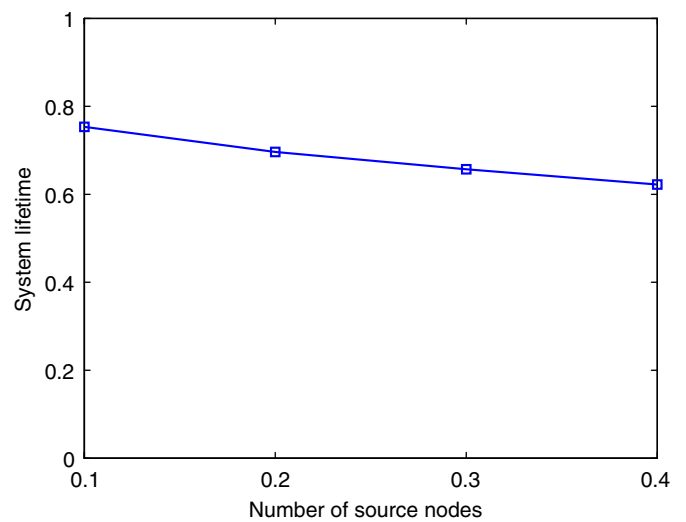


Fig. 11. The impact of the number of source nodes (normalized to $|V_c|/|V|$) on system lifetime. The y-axis has been normalized to the optimal system lifetime.

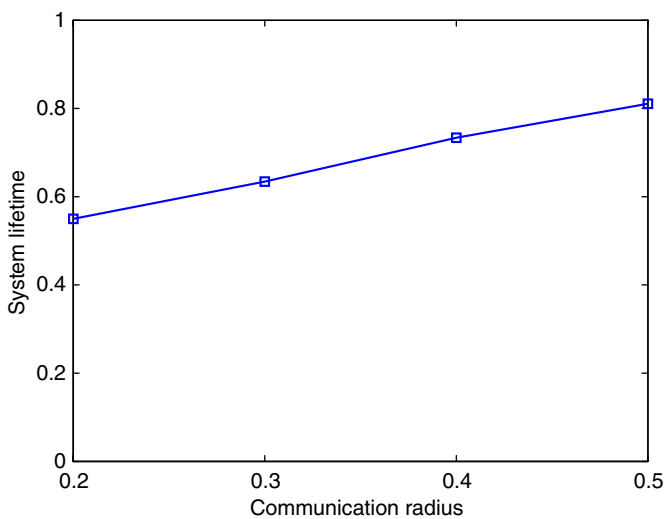


Fig. 10. The impact of communication radius on system lifetime. The y-axis has been normalized to the optimal system lifetime.

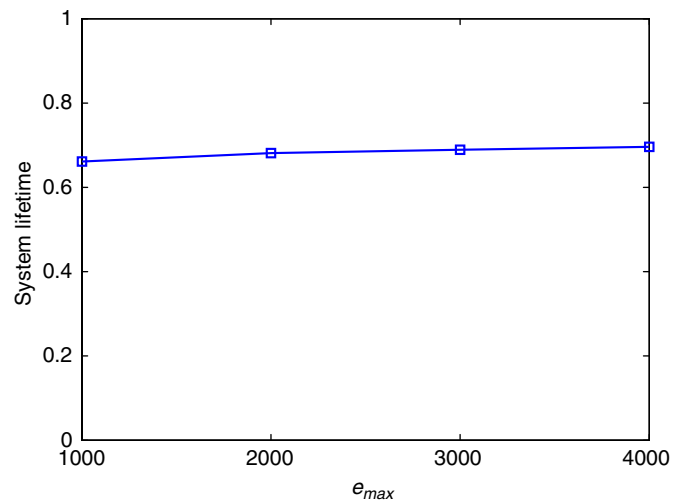


Fig. 12. The impact of e_{max} on system lifetime. The y-axis has been normalized to the optimal system lifetime.

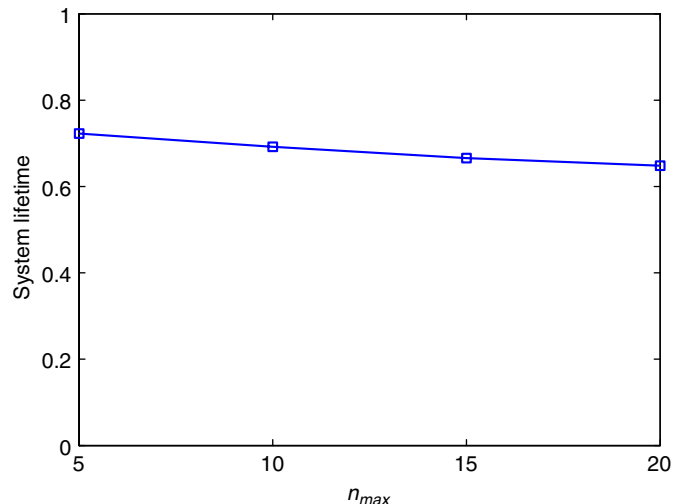


Fig. 13. The impact of n_{max} on system lifetime. The y-axis has been normalized to the optimal system lifetime.

The results show that the performance of the heuristic is more sensitive to the number of nodes, the number of source nodes, and communication radius than to e_{max} and n_{max} . Figs. 9 and 11 show that the system lifetime reduces as the number of nodes or the number of source nodes increases. The system lifetime improves when the communication radius increases, as shown in Fig. 10. On the other hand, the system lifetime did not change too much when we varied e_{max} and n_{max} , as shown in Figs. 12 and 13. This indicates that the topology of the system has more impact on the system lifetime than the properties of the individual nodes do. This observation implies that, given a set of nodes with pre-determined properties (for example, available energy), the system performance can be maximized by carefully designing the topology of the deployment. This leads to the interesting problem of sensor localization with respect to lifetime maximization.

8. Conclusion

We considered a class of data gathering applications where the event is sensed by a subset of the sensors, each of which generates a specified number of data packets during each round, and all the data packets need to be transferred to the base station. The objective was to maximize the system lifetime in terms of the number of rounds the system can operate, under the energy constraints of the sensors. We reduced the above problem to a restricted flow problem and proposed a strongly polynomial time algorithm. Our algorithm is guaranteed to find an integer valued solution that specifies the number of data packets to be transferred between any two neighboring sensors for each round. We also experimentally studied the performance of a distributed shortest path heuristic. The heuristic cannot guarantee optimality but has good average case performance. Our future direction is to develop a distributed algorithm for the MDG problem that has better provable performance than the shortest path heuristic. Another direction is to consider the utilization of data aggregation [7] to further reduce energy consumption. We are also interested in the system synthesis problem that searches for the optimal deployment of the nodes (with the objective to maximize the system lifetime). We also plan to consider the maximization of lifetime under other system models such as the dynamic models proposed in [12].

References

- [1] M. Agarwal, J.H. Cho, L.X. Gao, J. Wu, Energy efficient broadcast in wireless ad hoc networks with hitch-hiking, *IEEE Infocom*, 2004.
- [2] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1993.
- [3] I.F. Akyildiz, W. Su, Y. Sankarasubramanian, E. Cырci, *Wireless sensor networks: a survey*, *Comput. Networks* 38 (4) (2002) 393–422.
- [4] T.H. Cormen, C.E. Leiserson, R.L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1992.
- [5] W. Ding, S.S. Iyengar, R. Kannan, W. Ruml, Energy equivalence routing in wireless sensor networks, *J. Microcomput. Appl.* (2004), special issue on wireless sensor networks.
- [6] C. Efthymiou, S. Nikolettseas, J. Rolim, Energy balanced data propagation in wireless sensor networks, in: 18th Internat. Parallel and Distributed Processing Symposium (IPDPS'04), April 2004.
- [7] M. Enachescu, A. Goel, R. Govindan, R. Motwani, Scale free aggregation in sensor networks, in: Workshop on Algorithmic Aspects of Wireless Sensor Networks (AlgoSensors '04), July 2004.
- [8] F. Gaertner, A survey of self-stabilizing spanning-tree construction algorithms, Technical Report IC/2003/38, School of Computer and Communication Sciences, Swiss Federal Institute of Technology (EPFL), 2003.
- [9] W.R. Heinzelman, A. Chandrakasan, H. Balakrishnan, Energy efficient communication protocol for wireless micro-sensor networks, in: *Proceedings of IEEE Hawaii International Conference on System Sciences*, 2000.
- [10] K. Kalpakis, K. Dasgupta, P. Namjoshi, Maximum lifetime data gathering and aggregation in wireless sensor networks, *IEEE Networks '02 Conference*, 2002.
- [11] R. Kannan, S.S. Iyengar, Game-theoretic models for reliable path-length and energy-constrained routing with data aggregation in wireless sensor networks, *IEEE J. Selected Areas of Comm.* 22 (6) (2004).
- [12] P. Leone, J. Rolim, Towards a dynamical model for wireless sensor networks, *ALGOSENSORS 2004*, July 2004.
- [13] F. Ordonez, B. Krishnamachari, Optimal information extraction in energy-limited wireless sensor networks, *IEEE J. Selected Areas in Comm.* 22 (6) (2004).
- [14] N. Sadagopan, B. Krishnamachari, Maximizing data extraction in energy-limited sensor networks, *IEEE Infocom 2004*, 2004.
- [15] A. Sinha, A. Chandrakasan, Dynamic power management in wireless sensor networks, *IEEE Design Test Comput.* 18 (2001) 62–74.
- [16] Y. Zou, K. Chakrabarty, Target localization based on energy considerations in distributed sensor networks, *First IEEE International Workshop on Sensor Network Protocols and Application*, May 2003.

Bo Hong is currently an assistant professor in the Electrical and Computer Engineering Department at Drexel University. He received his Ph.D. degree in Electrical Engineering from the University of Southern California in 2005 and his M.Eng. degree from Tsinghua University, Beijing in 2000. His research interests include parallel and distributed processing, modeling and optimization of high performance computing systems and applications, and distributed computing in networked sensor systems.

Viktor K. Prasanna received his B.S. in Electronics Engineering from the Bangalore University and his M.S. from the School of Automation, Indian Institute of Science. He obtained his Ph.D. in Computer Science from the Pennsylvania State University in 1983. Currently, he is a Professor in the Department of Electrical Engineering as well as in the Department of Computer Science at the University of Southern California, Los Angeles. He is also an associate member of the Center for Applied Mathematical Sciences (CAMS) at USC. He served as the Division Director for the Computer Engineering Division during 1994–1998. His research interests include parallel and distributed systems, embedded systems, configurable architectures and high performance computing. Dr. Prasanna has published extensively and consulted for industries in the above areas. He has served on the organizing committees of several international meetings in VLSI computations, parallel computation, and high performance computing. He is the Steering Cochair of the International Parallel and Distributed Processing Symposium [merged IEEE International Parallel Processing Symposium (IPPS) and the Symposium on Parallel and Distributed Processing (SPDP)] and is the Steering Chair of the International Conference on High Performance Computing (HiPC). He serves on the editorial boards of the *Journal of Parallel and Distributed Computing* and the *Proceedings of the IEEE*. He is the Editor-in-Chief of the *IEEE Transactions on Computers*. He was the founding Chair of the IEEE Computer Society Technical Committee on Parallel Processing. He is a Fellow of the IEEE.