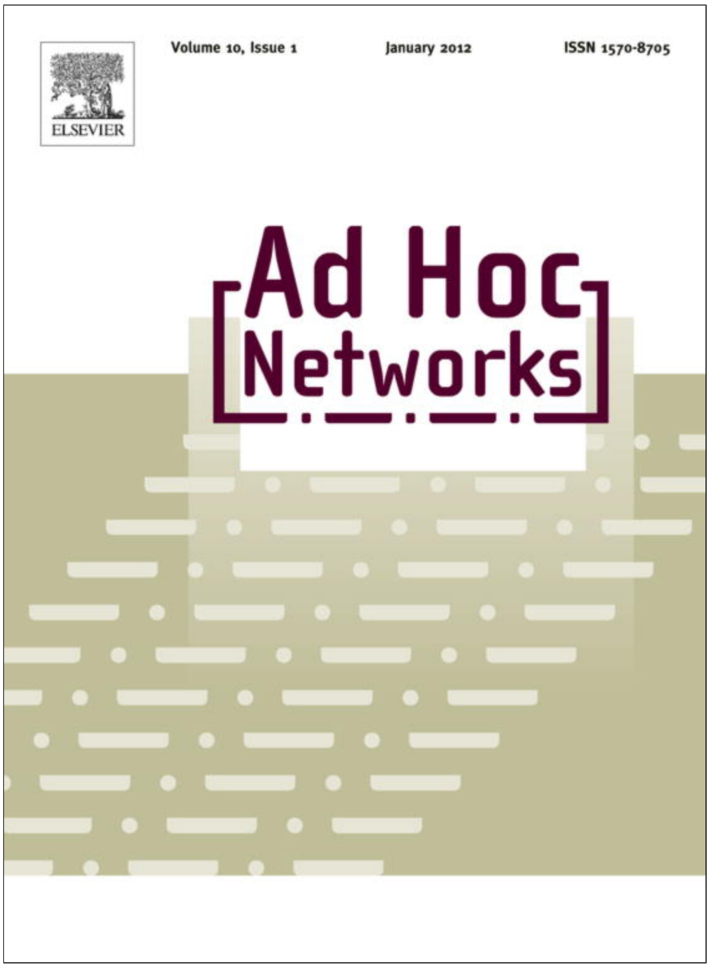

Volume 10, Issue 1
January 2012
ISSN 1570-8705
ELSEVIER
Ad Hoc
Networks

Ad Hoc Networks 10 (2012) 62–74

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Fault-tolerant monitor placement for out-of-band wireless sensor network monitoring

Xian Chen [a,*], Yoo-Ah Kim [c,1], Bing Wang [a], Wei Wei [b], Zhijie Jerry Shi [a], Yuan Song [a]

[a] Computer Science & Engineering Department, University of Connecticut, United States
[b] Department of Computer Science, University of Massachusetts, Amherst, United States
[c] National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, Bethesda, Maryland, United States

A R T I C L E I N F O

Article history:
Received 9 March 2010
Received in revised form 31 December 2010
Accepted 31 May 2011
Available online 15 June 2011

Keywords:
Monitoring
Health monitoring
Wireless sensor network
Fault tolerant monitoring
Out-of-band

A B S T R A C T

Monitoring a sensor network to quickly detect faults is important for maintaining the health of the network. Out-of-band monitoring, i.e., deploying dedicated monitors and transmitting monitoring traffic using a separate channel, does not require instrumenting sensor nodes, and hence is flexible (can be added on top of any application) and energy conserving (not consuming resources of the sensor nodes). In this paper, we study fault-tolerant out-of-band monitoring for wireless sensor networks. Our goal is to place a minimum number of monitors in a sensor network so that all sensor nodes are monitored by k distinct monitors, and each monitor serves no more than w sensor nodes. We prove that this problem is NP-hard. For small-scale network, we formulate the problem as an Integer Linear Programming (ILP) problem, and obtain the optimal solution. For large-scale network, the ILP is not applicable, and we propose two algorithms to solve it. The first one is a $\ln(kn)$ approximation algorithm, where n is the number of sensor nodes. The second is a simple heuristic scheme that has much shorter running time. We evaluate our algorithms using extensive simulation. In small-scale networks, the latter two algorithms provide results close to the optimal solution from the ILP for relatively dense networks. In large-scale networks, the performance of these two algorithms are similar, and for relatively dense networks, the number of monitors required by both algorithms is close to a lower bound.

1. Introduction

A deployed sensor network may suffer from many faults due to environmental impacts, hardware defects, and software bugs [22,30,27]. These faults can cause high loss rates, long transmission delays or even network disconnection, and hence severely affects the normal operations of the network. Therefore, effective architecture and techniques are needed to monitor the health of the network and quickly detect such faults.

Existing studies use two approaches for sensor network monitoring: "in-band" and "out-of-band" monitoring. In-band monitoring uses sensor nodes to monitor themselves and their neighbors, and transmits the monitoring traffic inside the sensor network, sharing the network bandwidth with sensed data [33,34,11,26,19,21,28,14,15]. Out-of-band monitoring uses dedicated monitoring nodes, and transmits monitoring traffic out-of-band, using a separate channel that does not interfere with the transmission of sensed data. A monitoring node can be attached to a sensor node to directly monitor or control the attached sensor node [7,23]; or placed inside the sensor network to monitor nearby sensor nodes [6,20,32].

Out-of-band monitoring has several advantages over in-band monitoring: (1) it does not require instrumenting sensor nodes, and hence requires no change to the

* Corresponding author.
E-mail addresses: Xian.Chen@engr.uconn.edu (X. Chen), kimy3@ncbi.nlm.nih.gov (Y.-A. Kim), bing@engr.uconn.edu (B. Wang), weiwei@cs.umass.edu (W. Wei), zshi@engr.uconn.edu (Z.J. Shi), Yuan.Song@engr.uconn.edu (Y. Song).

[1] This work was done while the author was with the University of Connecticut.

1570-8705/$ - see front matter Crown Copyright © 2011 Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2011.05.008

application; (2) it does not consume scarce resources (e.g., CPU, memory, storage) of the sensor nodes; (3) it uses a separate channel for the monitoring traffic that does not interfere with the transmission of the sensed data; (4) it provides a convenient way to monitor per-hop delays without the need of clock synchronization [32]; and (5) simple inexpensive mote-class monitors can provide satisfactory monitoring performance [5], and hence they can be used for cost-effective out-of-band monitoring.

In this paper, we consider placing dedicated mote-class monitors that are distributed inside a sensor network for out-of-band monitoring (as in [32,6,20]). Compared to attaching monitors to sensor nodes (as in [7,23]), this type of out-of-band monitoring requires less monitors since each monitor can serve multiple nearby sensor nodes, and furthermore, it is easier to tolerate faults at the monitors since we may use multiple monitors to monitor a sensor node. We aim to answer the following question: *how to place the monitors so that each sensor node be monitored by k ($k \geqslant 1$) monitors, each monitor serves at most w sensor nodes, and the number of monitors is minimized?* The value of k determines the extent of tolerance to failures at the monitors. The restriction that a monitor can monitor at most w sensor nodes is to take account of the limited capability of the monitors. The objective of minimizing the number of monitors is to minimize deployment cost. Henceforth, we refer to the above monitoring problem as *k-monitoring problem*.

We prove that the k-monitoring problem is NP-hard even for $k = 1$ and unlimited w (see Section 4). Since the number of candidate monitor positions is infinite (a monitor can be placed anywhere in the sensor network), we first propose a pre-processing algorithm that finds a finite yet sufficient number of candidate positions. After that, we propose three algorithms to place monitors. The first one is an ILP based algorithm. It provides optimal solution, but is only applicable to small-scale problems. For large-scale problems, we develop a Max-Flow based approximation algorithm that has a guaranteed approximation ratio of $\ln(kn)$, where n is the number of sensor nodes in the network, and a Max-Degree based heuristic algorithm that has shorter running times. We evaluate the performance of our algorithms using extensive simulation. In small-scale networks, the Max-Flow and Max-Degree algorithms provide results close to the optimal solution from ILP for relatively dense networks. In large-scale networks, the performance of these two algorithms are similar, and for relatively dense networks, the number of monitors required by both algorithms is close to a lower bound.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 presents the problem setting. Section 4 presents the three algorithms and their analysis. Section 5 evaluates the performance of these algorithms using extensive simulation. Finally, Section 6 concludes the paper.

2. Related work

Many studies on sensor network monitoring use an active in-band approach: the monitoring is performed by instrumenting the sensor nodes to monitor themselves and their neighbors, and the monitoring traffic is transmitted inside the sensor network [33,34,11,26,19,21,28,14,15]. Several recent studies adopt a passive out-of-band approach. DSN (Deployment Support Network) [7] and SRCP (Simple Remote Control Protocol) [23] use separate battery-powered control processors and radio for out-of-band monitoring/management of sensor networks. The monitoring nodes are attached to sensor nodes, and hence the number of monitors equals to the number of sensor nodes. Some studies place dedicated monitors inside a sensor network, detached from sensor nodes [32,6,20]. Therefore, each monitor can monitor multiple nearby sensor nodes and the number of monitors is typically less than the number of sensor nodes. The studies of [6,20] use monitors for code debugging and perform monitor repairing. In our preliminary study [32], we propose a monitoring architecture that monitors per-hop delays and detects abnormal delays using dedicated monitors without the need of clock synchronization. We further study how to place monitors under this architecture so that every link is monitored by a single monitor. In this paper, we extend the monitor placement problem in [32] to k-monitoring problem, $k \geqslant 1$. We focus on node monitoring (i.e., each node is monitored by k distinct monitors), and our proposed algorithms are significantly more general than those in [32]. Our algorithms can be easily extended to link monitoring as shown in [32].

Passive monitoring through dedicated monitors has been used in other types of wireless networks. For instance, it has been successfully used in wireless LANs for network management and characterization (e.g. [1,31,12,16,4,24]). We are, however, not aware of any in-depth study on how to place monitors inside a wireless network to achieve the optimization goals as in our study (e.g., minimizing the number of monitors). Although our algorithms are developed in the context of wireless sensor networks, they can be applied to other types of wireless networks, e.g., wireless LANs.

Our approximation algorithm uses a Max-Flow formulation and is inspired by the algorithms in [2]. However, our work differs from [2] in several important aspects. The study of [2] determines how to choose centers from a set of nodes in a network (each center serves a group of nodes). The set of nodes (and hence the candidate centers) are given beforehand. In our problem, monitors can be placed at any point in the sensor network (and hence there are infinite many candidate locations). Furthermore, we consider fault-tolerant monitoring by requiring a node to be monitored by $k \geqslant 1$ monitors, while [2] only considers the case where $k = 1$.

One problem closely related to ours is the point coverage problem [3,29], which considers a number of discrete points and multiple types of sensors, with the goal of finding a selection of sensors and a subset of points to place the sensors so that each point is covered by a certain number of sensors, and the total cost of the sensors is minimized. This problem differs from our problem in that the sensors are placed at a subset of given points, and it does not impose the constraint on the maximum number of points that a sensor can monitor.

Last, placing passive monitors has been studied in wired networks (e.g., [25]). Monitoring in wireless networks differs from that in wired networks in that a monitor needs to be placed in the transmission range of a wireless node to be able to overhear/monitor that node; such a requirement is not necessary in a wired network.

3. Problem setting

Consider a sensor network deployed in a two-dimensional area. The network consists of n sensor nodes, denoted as $V = \{v_1, \ldots, v_n\}$. We place dedicated monitors inside the network to monitor the liveness of the sensor nodes. Each monitor has two wireless network interfaces using non-interfering channels, one for overhearing packets from nearby sensor nodes, and the other for communicating between the monitors and transmitting alerts to a central server [6,20,7,23,32]. In other words, the first interface is only used for passive listening. The other interface uses low-power, long-range, and low-bandwidth radio and is only used infrequently to transmit packets. We next describe the formulation in detail.

Our goal is to place a set of monitors, M, inside the sensor network so that each sensor node is monitored by at least k monitors, each monitor serves at most w sensor nodes, and the number of monitors is minimized. The parameter k is used to determine the extent of fault tolerance. The parameter w specifies the *workload* constraint: a monitor can monitor at most w sensor nodes due to the limited capability (we assume that the monitors are simple embedded devices for large-scale deployment). If a monitor is in the neighborhood of more than w nodes, it processes the packets overheard from at most w nodes; the rest of the overheard packets will be discarded.

Another constraint in placing monitors is that the placed monitors need to form a connected graph through the non-interfering channel (so that the monitors can communicate with each other, and transmit alters to a central server). We assume the non-interfering channel uses a long-range radio (much longer than that used by the sensor nodes, e.g., as described in [23]), and hence this constraint can be easily satisfied (our results in Section 5 show that this is indeed the case).

For each sensor node $v \in V$, let $\varphi(v)$ be the set of monitors that serves v, referred to as the *assignment* for v. Then $|\varphi(v)| = k$, $\forall v \in V$ and $\cup_{v \in V} \varphi(v) = M$. Fig. 1 shows an example of node monitoring with $k = 2$ and $w = 4$, where seven sensor nodes $v_1, \ldots, v_7$ are monitored by four monitors, $m_1, \ldots, m_4$. A monitor is connected via dotted lines to a sensor node if it serves that sensor node. As shown in the figure, the assignment is that m_3 and m_4 each monitors four sensor nodes $v_2, \ldots, v_5$; m_1 and m_2 each monitors three sensor nodes v_1, v_6, and v_7.

A naive method to place the monitors is dividing area into grids and placing one monitor in the center of each grid. This method is, however, not cost-effective. For example, consider a 500 m × 500 m area that has 100 sensor nodes, and the transmission range of each sensor node is 80 m. In each row, the naive method needs five monitors, and it needs a total of 25 monitors to achieve 1-monitoring. A

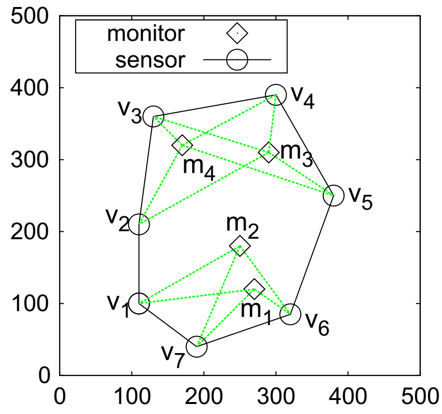


Fig. 1. Illustration of k-monitoring, $k = 2$, $w = 4$.

more carefully designed algorithm needs much less monitors (e.g. the algorithms that we develop only need half as many monitors, see Section 5). The difference is even more dramatic for larger testbeds and $k > 1$.

4. Algorithms for *k*-monitoring

The k-monitoring problem formulated in Section 3 is NP-hard even for $k = 1$ and without the workload constraint, the proof is found in Appendix A. Since the number of candidate monitor locations is infinite, we first pre-process the input to determine a finite number of candidate monitor locations. We then present three algorithms to place monitors at the candidate locations, an ILP based algorithm, an approximation algorithm based on a Max-Flow formulation, and a simple heuristic algorithm. We also analyze the performance of the latter two algorithms.

4.1. Determining candidate monitor locations

For ease of exposition, we first consider the case where a sensor node has regular radio range. We then consider the more realistic case where the radio range is irregular [35]. In both cases, let l_i denote the location (i.e., the coordinate) of sensor node v_i. Let R_i and r_i denote the *coverage region* and *transmission range* of v_i respectively. Considering the reception region characteristics [17], we assume any node in the coverage region, R_i, can hear the transmission of v_i. When the radio range is regular, R_i is a circular area centered at l_i with the radius of r_i. Otherwise, we assume that R_i is a polygon and the average distance from l_i to the vertices of the polygon is r_i.[2]

The algorithms for determining the candidate monitor locations for regular and irregular radio range are similar. In both algorithms, let L denote the set of candidate locations. It is initially empty. The algorithms add candidate locations to L by considering all pairs of nodes in the sensor network.

Regular radio range. The algorithm assuming regular radio range is presented in Algorithm 1. For any pair of

[2] When the radio range is composed of multiple disjoint regions, we can consider these disjoint regions as multiple polygons. It is straightforward to extend our algorithm for determining candidate monitor locations to this scenario.

nodes, v_i and v_j, it adds candidate locations to the set of candidate location set, L, by considering the following four cases. If $R_i \subseteq R_j$ (Fig. 2a), it adds the location of v_i, l_i, as a candidate location (we may use any location in R_i as a candidate location; for simplicity, we use l_i). Similarly, if $R_j \subseteq R_i$ (Fig. 2b), it adds the location of v_j, l_j, as a candidate location. If neither of the above two conditions holds, and $R_i \cap R_j \neq \emptyset$ (Fig. 2c), it adds the two points where the boundaries of R_i and R_j intersect into the candidate location set. Last, if none of the above conditions holds, i.e., $R_i \cap R_j = \emptyset$ (see Fig. 2d), it adds the locations of v_i and v_j into the candidate location set.

Algorithm 1. Determine Candidate Monitor Locations (Regular Radio Range)

```
L = ∅
for ∀v_i, v_j ∈ V, v_j ≠ v_i do
  if R_i ⊆ R_j then
    L = L ∪ {l_i}
  els if R_j ⊂ R_i then
    L = L ∪ {l_j}
  els if R_i ∩ R_j ≠ ∅ then
    L = L ∪ {p_1, p_2}, p_1 and p_2 are the two points
      where the boundaries of R_i and R_j intersect
  else
    L = L ∪ {l_i, l_j}
  end if
end for
Return L
```

We next show that the above approach of determining candidate locations is sufficient, as stated in the following theorem.

Theorem 1. *For any optimal solution M^*, there is a corresponding subset $M \subseteq L$ s.t. $|M^*| = |M|$ and M covers all sensor nodes.*

Proof. We prove the theorem by showing that $\forall m \in M^*$, we can find a location $l \in L$ so that the set of nodes monitored by m can be monitored by a monitor located at l. Without loss of generality, suppose the set of sensor nodes monitored by m is $\{v_1, \ldots, v_x\}$, $x \geqslant 1$. We consider the following two cases.

- Case 1 ($x = 1$). That is, m monitors a single sensor node, v_1. If v_1 is an isolated node (i.e., its coverage region does not intersect with that of any other sensor node), by Algorithm 1, $l_1 \in L$ and a monitor located at l_1 can monitor v_1. Otherwise, there exists another node, v_i, $i \neq 1$, such that the coverage region of v_1 intersects with R_i, then by Algorithm 1, depending on the relationship of R_1 and R_i, we have l_1, l_i, or the two intersection points of the boundaries of R_1 and R_i are in L, and a monitor located at any of these points can monitor v_1.
- Case 2 ($x > 1$). Then m must be in the intersection region of $R_1, \ldots, R_x$. Let B denote the boundary of this intersection region. If there exist two indices, i and j, such that one intersection point of R_i and R_j is on B, $i,j = 1, \ldots, x$, $i \neq j$, then this intersection point can monitor $\{v_1, \ldots, v_x\}$. If the above condition does not hold, i.e., for any i, j, $i,j = 1, \ldots, x$, $i \neq j$, the intersection points of R_i and R_j are not on B, then there must exists one sensor node v_i such at $R_i \subset R_j$, $\forall j = 1, \ldots, x, j \neq i$. In this case, by Algorithm 1, $l_i \in L$, and a monitor located at l_i can monitor $\{v_1, \ldots, v_x\}$.

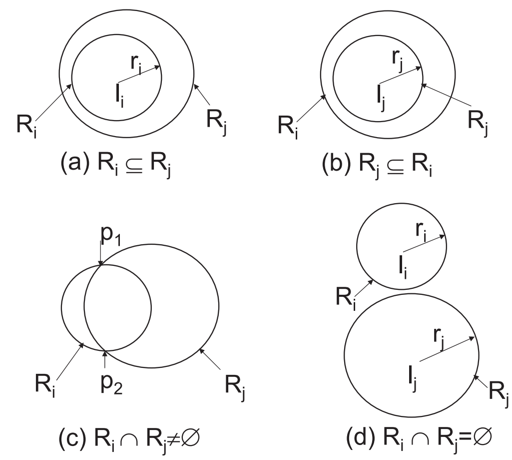


Fig. 2. Illustration of determining candidate monitor locations (regular radio range).

Irregular radio range. In this case, our algorithm for determining candidate monitor locations is similar to Algorithm 1. The only difference is that since R_i and R_j are polygons, when they intersect, they may intersect at multiple points (more than two) or infinite number of points (i.e., their intersection forms an edge). For the former case, we include the multiple points into L; for the latter case, we include the two end points of the edge into L. Therefore, the total number of candidate monitor positions is finite. We can again show that the above algorithm is sufficient; the proof is similar to that for Theorem 1 and is omitted in the interest of space.

4.2. ILP based algorithm

For a given set of candidate monitors, we can formulate the monitor placement problem as an ILP problem, and solve it directly to obtain the optimal solution. This approach is, however, only applicable to small-scale networks; for large-scale networks, it does not provide a solution in a reasonable amount of time. We next describe the ILP formulation.

Let C denote a candidate monitor set. It contains k monitors at each candidate location (determined by Algorithm 1) since each sensor node needs to be monitored by k monitors, and in the worst case, a sensor node needs k monitors at a candidate location to be k-monitored. Recall that V denotes the set of sensor nodes to be monitored. Suppose m is a candidate monitor in C, and v is a sensor node in V. Let V_m denote the set of sensors that can be heard by m. Let C_v represent the set of candidate monitors that can hear the transmission of sensor node v. Let x_m and $y_{m,v}$ be 0–1 variables. In particular, $x_m = 1$ when

$$\text{minimize:} \quad \sum_{m \in C} x_m \tag{1}$$

$$\text{subject to:} \quad \sum_{v \in V_m} y_{m,v} \geq x_m, \forall m \in C \tag{2}$$

$$\sum_{v \in V_m} y_{m,v} \leq |V_m| x_m, \forall m \in C \tag{3}$$

$$\sum_{v \in V} y_{m,v} \leq w, \forall m \in C \tag{4}$$

$$\sum_{m \in C_v} y_{m,v} = k, \forall v \in V \tag{5}$$

$$y_{m,v} = 0, \forall v \in V, m \notin C_v \tag{6}$$

$$x_m \in \{0,1\} \tag{7}$$

$$y_{m,v} \in \{0,1\} \tag{8}$$

Fig. 3. ILP problem formulation.

monitor m monitors at least one sensor node, and $x_m = 0$ otherwise. We have $y_{m,v} = 1$ when monitor m monitors sensor node v, and $y_{m,v} = 0$ otherwise.

The ILP formulation is described in Fig. 3. The objective function is to minimize the number of required monitors. Constraints (2) and (3) regulate the relationship between x_m and $y_{m,v}$. Constraint (4) specifies that the capacity of each monitor is w. Constraint (5) specifies that each sensor node should be monitored by k monitors. Constraint (6) states that monitor m cannot monitor any sensor node v if m cannot hear from v. Last, constraints (7) and (8) state x_m and $y_{m,v}$ are 0–1 variables.

After solving the ILP problem, we can directly obtain an optimal solution to the k-monitoring problem as $M = \{m | x_m = 1\}$ and $\varphi(v) = \{m | y_{m,v} = 1\}$, $\forall v \in V$.

4.3. Max-Flow based approximation algorithm

Inspired by [2], we also develop a Max-Flow based approximation algorithm, referred to as *Max-Flow k-monitoring algorithm*. In the following, we first illustrate why Max-Flow formulation is useful for solving the k-monitoring problem, and then describe the algorithm in detail. We then analyze the performance of this algorithm in Section 4.3.1; and present refinement to the algorithm in Section 4.3.2.

Let C denote a candidate monitor set. As in the ILP-based algorithm, C contains k monitors at each candidate location. We construct a Max-Flow graph as follows. First, we construct a bipartite graph. The two disjoint sets in the bipartite graph represent the candidate monitor set, C, and the set of sensor nodes, V, respectively. A node $m \in C$ is connected to a node $v \in V$ if m is in the coverage region of v (i.e., m can overhear the transmission of v); the capacity of edge (m, v) is 1. We further add a super source and a super sink. The super source is connected to each candidate monitor with the capacity of w. Each sensor node is connected to the super sink with the capacity of k. Fig. 4 illustrates the Max-Flow graph thus constructed. In the Max-Flow graph, the capacity between the super source and a candidate monitor, w, limits that a monitor serves at most w sensor nodes; an edge from a monitor to a sensor node specifies that the monitor can only serve the sensor node if it can overhear the sensor node; and the capacity, k, from a sensor node to the super sink specifies that the sensor node can be monitored by at most k monitors. Let f denote the maximum integral flow of this graph. Then it is easy to see that all the sensor nodes are k-monitored if and only if $f = k|V|$. Furthermore, the assignment for each sensor node can be easily obtained from the Max-Flow solution: if the amount of flow from monitor m to node v is positive, i.e., $f(m, v) > 0$, we assign m to monitor v. In the following, we refer to a Max-Flow graph thus constructed as $G(C,V,E,w,1,k)$, where the first two elements represent the candidate monitor set and the set of sensor nodes to be monitored, respectively; the third element represents the set of edges that connects candidate monitors and sensor nodes; the last three elements represent the capacity of an edge from a super source to a monitor, the capacity of an edge from a monitor to a sensor node, and the capacity of an edge from a sensor node to the super sink, respectively.

Algorithm 2. Max-Flow k-monitoring

1: Place k monitors at each candidate location to construct a candidate monitor set M_c
2: $M = \emptyset$
3: $\varphi(v_i) = \emptyset$, $\forall v_i \in V$
4: $E = \{(m, v_i)|$, $m \in M_c$, $v_i \in V$, m can monitor $v_i\}$
5: **repeat**
6: $\quad M_c = M_c \backslash M$

7: **for** $\forall m \in M_c$ **do**
8: $C = M \cup \{m\}$
9: Construct Max-Flow graph $G(C, V, E, w, 1, k)$
10: Let f_m denote the maximum integral flow of G
11: **end for**
12: $m = \arg\max_{m \in M_c} f_m$
13: $M = M \cup \{m\}$
14: **until** all nodes are k-monitored
15: **for** $\forall m \in M$, $\forall v_i \in V$ **do**
16: **if** $f_m(m, v_i) > 0$ **then**
17: $\varphi(v_i) = \varphi(v_i) \cup \{m\}$
18: **end if**
19: **end for**
20: Return (M, φ)

The main idea of the Max-Flow k-monitoring algorithm is as follows. Initially, the monitor set, M, is empty, and again the candidate monitor set contains k monitors at each candidate location (determined by Algorithm 1). The algorithm runs in iterations, and in each iteration, it selects one monitor from the candidate monitor set and adds it into M. The monitor that is selected is the one that provides the maximum flow (determined by solving Max-Flow problems formulated earlier). The iteration continues until all the sensor nodes are k-monitored.

Algorithm 2 describes the Max-Flow k-monitoring algorithm. Line 1 places k monitors at each candidate location to construct a candidate monitor set, M_c. Line 2 initializes the monitor set, M, to be an empty set. Line 3 initializes the assignment to each sensor node to be an empty set. Lines 4 adds a set of edges, E, between candidate monitors and sensor nodes: it adds an edge (m, v_i) when $m \in M_c$ is in the transmission range of $v_i \in V$. The algorithm runs in iterations. In each iteration (lines 6–13), it selects a monitor $m \in M_c$, $m \notin M$, so that $M \cup \{m\}$ produces the maximum integral flow in the Max-Flow graph $G(M \cup \{m\}, V, E, w, 1, k)$, and adds m into the monitor set. When multiple candidate monitors provide the same max flow, ties are broken by choosing the candidate monitor whose minimum distance to the monitors in M is the largest. The rationale for breaking ties in this way is to spread monitors inside the network (monitors that are close-by are more likely to suffer from faults simultaneously, e.g., faults caused by environmental impacts). This process of adding one monitor into M continues until all sensor nodes are k-monitored. Last, lines 15–19 record the assignment for each sensor node.

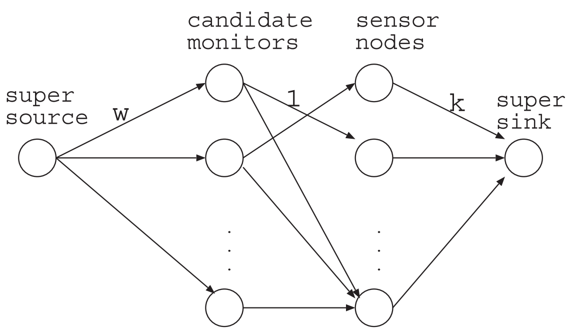


Fig. 4. Illustration of the Max-Flow formulation.

Algorithm 2 can be extended to accommodate the constraint that the placed monitors form a connected graph by imposing an additional request that m be connected to the current monitor network M. Namely, in Algorithm 2, line 12, we pick m from the set of nodes that are connected to M.

4.3.1. Analysis

We can show that the solution of the Max-Flow k-monitoring algorithm is no worse that $\ln(kn)$ times the optimal solution as stated in the following theorem. The proof is found in B.

Theorem 2. *The Max-Flow k-monitoring algorithm (Algorithm 2) is a ln(kn) approximation algorithm.*

We can also show that our analysis is tight if the candidate locations for monitors are fixed (instead of allowing monitors to be placed anywhere) even when all the sensors have identical, regular coverage range. A tight example is shown in Fig. 5. Let us assume that $k = 1$ and the capacity of each monitor is unlimited. In the example, we show the coverage range of the sensors as circles. All of the sensors have regular, identical coverage ranges. That is, sensors are placed in the center of circles. Multiple sensors can be placed at the same location. In the example, consider the circles in the upper row, which includes $i + 1$ sets of sensors. Let us index them from 1 to $i + 1$ from the left to the right. The jth set contains 2^{i+1-j} sensors, $j = 1, \ldots, i + 1$. Therefore, there are $\sum_{j=1}^{i+1} 2^{i+1-j} = 2^{i+1} - 1$ sensors in the row. Exactly the same number of sensors are located in the lower row. We are also given a set of monitors. The solid squares and the two blank squares in Fig. 5 represent the candidate monitor locations, and they are in the intersection areas of the sensor coverage regions (i.e., the circles). Specifically, a solid square is in the intersection area of the two leftmost circles (i.e., the upper and lower

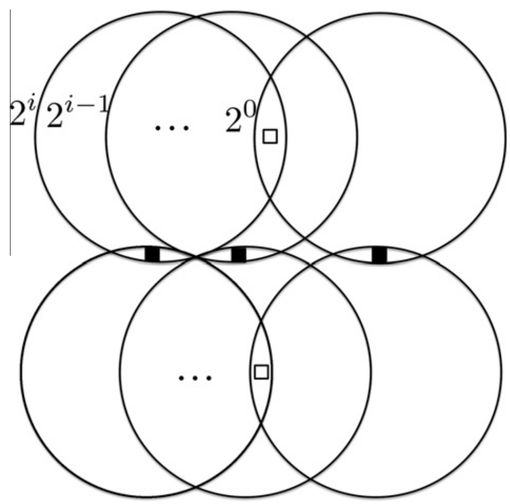


Fig. 5. An example shows that the approximation ratio of Max-Flow algorithm is tight. The solid squares and the two blank squares represent the candidate monitor locations.

circles), which is not included in the second left circles. Therefore, the leftmost solid square covers $2 \times 2^i = 2^{i+1}$ sensors, the second leftmost solid square covers $2 \times 2^{i-1} = 2^i$, and so on. The blank square in each row is in an area covered by all of the sensors in that row, i.e., it can monitor $2^{i+1} - 1$ sensors. Clearly, the optimal solution is to choose the two blank squares. However, the Max-Flow algorithm will end up with choosing the black squares, giving a $\ln(kn)$ approximation.

Note that in our tight example, we assumed that the candidate locations for monitors are given while in our problem setting, we did not restrict the locations of monitors. In fact, our analysis on the approximation bound did not utilize the fact that the monitors can be placed anywhere, and therefore the actual approximation ratio may be better than $\ln(kn)$.

4.3.2. Refinements

We now describe a couple of improvements to the Max-Flow *k*-monitoring algorithm. Even though we were not able to show that they provide a better theoretical bound, we found that the performance were significantly improved with the refinements in the simulation studies.

The first refinement is that we can remove redundant candidate monitors to reduce the running time of this algorithm. We say two candidate monitors m_1 and m_2 are *equivalent* if the set of sensor nodes that they monitor are the same. Let $M(m_i)$ denote the set of candidate monitors that are equivalent to m_i. Let $V(m_i)$ denote the set of sensor nodes that m_i can monitor. Then if $|V(m_i)| \leqslant w$ and $|M(m_i)| > k$, we only need to keep k candidate monitors in $M(m_i)$ (since these k candidate monitors are already sufficient to monitor the sensor nodes in $V(m_i)$). In our simulation (Section 5), we find that the above approach of removing redundant candidate monitors can reduce the number of candidate monitors by 30% in some scenarios. Secondly, we can reduce the storage usage of the Max-Flow *k*-monitoring algorithm by initially placing a single monitor at each candidate location, and then add more candidate monitors when needed. More specifically, we may change Algorithm 2 as follows. Initially, M_c contains a single monitor at each candidate location. Then after adding m (the candidate monitor that provides the max flow) to M, we check whether there exists $v_i \in V$ so that the flow from v_i to m is positive (i.e., $f(m, v_i) > 0$) and v_i is not *k*-monitored. If so, we place a candidate monitor m' at the location of m, and add m' to M_c (since m' may be useful to serve v_i).

4.4. Max-Degree based heuristic algorithm

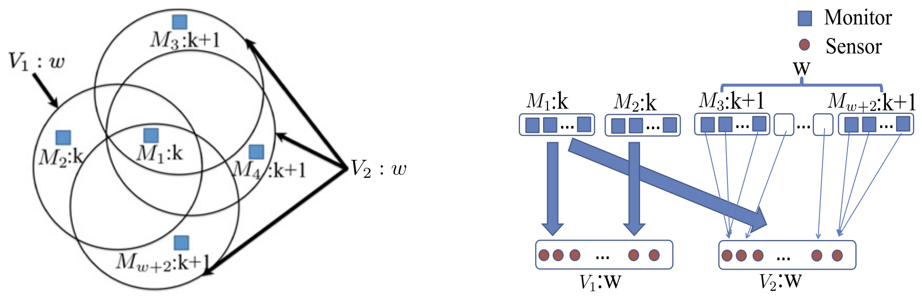
We next describe a simple heuristic algorithm for the *k*-monitoring problem. The outline of the algorithm is as follows. Let C denote a candidate monitor set, and V denote the set of sensor nodes to be monitored. We construct a graph $G(C \cup V, E)$, where $C \cup V$ is the set of vertices, and E is the set of edges. An edge $(m, v) \in E$ if m is in the transmission range of v, $m \in C$ and $v \in V$. The heuristic algorithm runs in iterations. In each iteration, it adds the candidate monitor with the maximum degree into the monitor set. The rationale is that a candidate monitor with a larger degree can serve more sensor nodes, and hence may reduce the number of monitors needed.

We refer to this algorithm as *Max-Degree k-monitoring algorithm*. Algorithm 3 describes this algorithm. Line 1 places k monitors at each candidate location to construct candidate monitor set, M_c. Line 2 initializes the monitor set, M, to be an empty set. Line 3 initializes the assignment to each sensor node to be an empty set. In each iteration (lines 5–15), it first constructs graph $G(M_c \cup V, E)$, where $(m, v) \in E$ if m is in the transmission range of v, $\forall m \in M_c$, $\forall v \in V$. Suppose m has the maximum degree (we break ties in the same way as in the Max-Flow *k*-monitoring algorithm). It adds m to the monitor set, and assign m to monitor a set of sensor nodes that m can overhear, denoted as $N(m)$. If more than w nodes are in $N(m)$, it assigns the w sensor nodes with the lowest degrees to m (the intuition is that sensor nodes with higher degrees may be able to be monitored by other candidate monitors). Line 14 removes m from the candidate monitor set. Line 15 removes all sensor nodes that are *k*-monitored from V. The iteration continues until all nodes are *k*-monitored. Similar to the Max-Flow *k*-monitoring algorithm, Algorithm 3 can be easily extended to sastisfy the constraint that the placed monitors form a connected graph by choosing a monitor m that has the highest degree among those that are connected to at least one node in M on line 6.

The two improvements to the Max-Flow *k*-monitoring algorithm described in Section 4.3.2 are also applicable to the Max-Degree *k*-monitoring algorithm. Namely, we may remove redundant candidate monitors to reduce its running time. Also, we may only place a single monitor at each candidate location initially, and incrementally add candidate monitors when needed. More specifically, at the end of an iteration, if there exists a sensor node that is not *k*-monitored and is monitored by m (the one that has the maximum degree) in the current iteration, we adds a candidate monitor, m', placed at the same location of m, to the candidate monitor set.

Algorithm 3. Max-Degree *k*-monitoring

1: Place k monitors at each candidate location to construct candidate monitor set M_c
2: $M = \emptyset$
3: $\varphi(v_i) = \emptyset, \forall v_i \in V$
4: **repeat**
5: Construct graph $G(M_c \cup V, E)$, $E = \{(m, v_i)|, m \in M_c, v_i \in V, m \text{ can monitor } v_i\}$
6: Suppose that $m \in M_c$ has the maximum degree
7: $M = M \cup \{m\}$
8: $N(m) = \{v_i | v_i \in V, m \in R_i\}$
9: **if** $|N(m)| \leqslant w$ **then**
10: $\varphi(v_i) = \varphi(v_i) \cup \{m\}, \forall v_i \in N(m)$
11: **else**
12: pick w sensor nodes in $N(m)$ that have the lowest degrees and add them to $\varphi(m)$
13: **end if**
14: $M_c = M_c \backslash \{m\}$
15: Remove all sensor nodes that are k-monitored from V
16: **until** all nodes are k-monitored
17: Return (M, φ)



(a) sensor and monitor placement (b) sensor and monitor coverage graph

Fig. 6. An example shows that the approximation ratio of Max-Degree algorithm can be $(w+1)/2$. The squares represent the candidate monitor locations, and the circles represent the sensors.

4.4.1. Analysis

While we do not have an analysis for an upper bound on the approximation ratio of the Max-Degree algorithm in general cases, we observe that this approximation ratio algorithm is also $\ln(kn)$ when $w = \infty$. This is because when $w = \infty$, the monitor which can introduce the maximum flow is equivalent to the one which has the maximum degree. Therefore, the Max-Flow and the Max-Degree give the same results when $w = \infty$.

In general, however, we conjecture that the Max-Degree algorithm gives an approximation ratio worse than $\ln(kn)$, which is also supported by the simulation results (see Section 5). Below we also show that the algorithm can give an approximation ratio of $(w+1)/2$ when $w \neq \infty$ and the candidate locations of the monitors are fixed.

Fig. 6 shows an example that gives a $(w+1)/2$ approximation ratio. In this example, we again assume that all of the sensors have regular, identical coverage ranges. Fig. 6a plots the sensor and monitor placement. This example contains two sets of sensors, V_1,V_2, and each set contains w sensor nodes. The sensors are placed in the center of the circles in Fig. 6a. Specifically, we put all of the w sensors nodes in V_1 at the same location. The squares in Fig. 6a represent the candidate monitor locations. Suppose that we have $w+2$ groups of monitors, $M_1, M_2, \ldots, M_{w+2}$, for which we assume that the capacity of each monitor is w. Furthermore, M_1 and M_2 each contains k monitors, while other groups have $k+1$ monitor nodes. Monitors in M_1 can hear the transmission of sensor nodes in both V_1 and V_2, while monitors in M_2 can hear the nodes in V_1 only. Let $s_1, s_2, \ldots, s_w$ be the sensors in V_2. We assume that monitors in M_{i+2} $(1 \leqslant i \leqslant w)$ can hear the transmission of sensor node s_i in V_2. For easy of understanding, we show the sensor and monitor coverage graph in Fig. 6b. In this example, it is easy to see that the monitors from M_1 and M_2 are the optimal solution, i.e., M_1 monitors sensors in V_2 and M_2 monitors sensors in V_1. However, since the monitors in M_1 have the maximum degree (i.e., $2w$), and the nodes in V_1 have the minimum degree (i.e., $2k$), the Max-Degree algorithm will first choose the monitors in M_1 to monitor the nodes in V_1. After that, for each node in V_2, the Max-Degree algorithm will need to assign k monitors from each of M_i, $i > 2$. Therefore, the Max-Degree algorithm selects $k(1+w)$ monitors, which gives an approximation ratio of $(w+1)/2$.

Note that as with the tight example for Max-Flow algorithm, the example above does not apply to the case when the monitors can be placed anywhere in the network. It is an interesting question whether the algorithm gives a better theoretical bound when there are no restrictions on the locations of the monitors.

5. Performance evaluation

In this section, we evaluate the performance of our k-monitoring algorithms using extensive simulation. We first describe the simulation settings, and then present the simulation results in detail.

We consider two networks, each with n sensor nodes. The first is a small-scale network of size 125 m × 125 m, and $n = 25$. The second is a large-scale network of size 500 m × 500 m, and $n = 100$ or 200. We consider three types of deployments. The first one is uniform random deployment [32,35,13,9,18]. The second is grid uniform deployment [10]. In the small-scale network, we specify the grid size to be 25 m × 25 m, and places one sensor node uniformly randomly in each grid. In the large-scale network, we divide the area into 100 grids, each of 50 m × 50 m, and places $n/100$ sensor nodes uniformly randomly in each grid. Grid uniform deployment provides a more even node distribution than uniform random deployment. The third one is a non-uniform deployment, where the entire region is divided into four sub-regions, the top left and bottom right regions have much higher node density than the other two regions (e.g., when $n = 200$, the two denser regions have 70 sensors while the other two regions have 30 sensors). Furthermore, we also place a region head in the center of each region. The region heads are connected to each other; The nodes in a region are uniformly deployed, and connected to their region head.

We set the workload w of a monitor to 40, 60 or ∞. This is based on our recent measurement study that indicates a mote-class monitor can easily handle 60 packets per second and the traffic in sensor nodes tend to be sparse (a sensor node generates a packet in seconds) [5]. We set k to 1, 2, or 3. The radio range of a sensor node is regular or irregular. Under regular radio range, the coverage region of a sensor node is circular, and all the sensor nodes have the same transmission range, which is varied from 30 to 200 m (corresponding to the transmission range of mote-class sensor nodes). Under irregular radio range, the coverage region is a polygon with 7–16 vertices, and all the sensor nodes have the same average transmission range,[3] which is varied from 30 to 200 m. We assume the range of the non-interfering radio of a monitor is sufficiently long, and for each setting, we obtain the minimum radio range that is required to keep the monitors connected.

The metric that we use is the *number of monitors needed* to achieve k-monitoring. All our simulations run on a server with an Intel Xeon 3.0 GHz CPU. The running time of the Max-Degree based algorithm is a few seconds, while the running time of the Max-Flow based algorithm is less than one minute for 25 nodes, between five to twenty minutes for 100 sensor nodes, and between ten to forty minutes for 200 sensor nodes. The time for solving the instances in the small-scale network using ILP algorithm is between few minutes and three hours; in the large-scale networks, it may take several days to solve one instance.

We only present the results under grid uniform random deployment and irregular radio range; the results under other settings (uniform random deployment, non-uniform deployment, regular radio range) are similar. All the results below are averaged over 10 simulation runs; the confidence intervals are tight and hence omitted.

We first show the results in the small-scale network. In this network, we use the ILP algorithm to obtain the optimal solution in each setting. Both Max-Flow and Max-Degree algorithms give solutions close to the optimal solution, especially in a relatively dense network. In more than 60% settings, the relative difference between our algorithms and the optimal solutions is less than 20%. Fig. 7 plots the average number of monitors required by the Max-Flow, Max-Degree and ILP algorithm, when $n = 25$, $k = 2$, $w = 40$. The average transmission range of sensor nodes varies from 30 to 90 m. We observe that the Max-Flow and Max-Degree algorithms have similar results, both approaching the ILP solutions as the transmission range increases. The other settings exhibit similar trends and are omitted.

We next present the results in the large-scale network. In this network, the ILP algorithm cannot solve the instances in a reasonable time. Therefore, we only evaluate the Max-Flow and Max-Degree algorithms. We compare their performance with a lower bound, $\lceil kn/w \rceil$, the minimum number of monitors required for node monitoring with n sensor nodes and workload of w (the optimal solution cannot be smaller than this lower bound). We observe that Max-Flow based algorithm only slightly outperforms the Max-Degree based algorithm: the maximum relative difference is 20%, and in around 90% of the settings, the relative difference is less than 10%. Therefore, considering both performance and running time, Max-Degree algorithm may be a preferable choice for large networks in

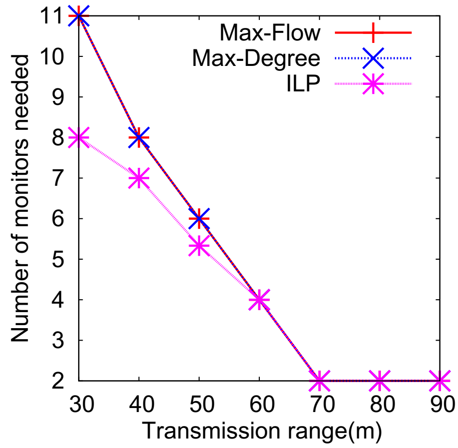


Fig. 7. Small-scale network: Max-Flow, Max-Degree and ILP k-monitoring algorithms under grid uniform deployment, irregular radio range, $n = 25$, $k = 2$, $w = 40$.

practice. We next only present the results of the Max-Flow based algorithm.

Fig. 8 plots the average number of monitors required by the Max-Flow algorithm, when $n = 100$, $w = 40$, and $k = 1, 2$, or 3 (the results for $n = 200$ exhibit similar trends). The average transmission range of the sensor nodes varies from 80 to 200 m (we choose the minimum range of 80 m because the network is disconnected when using a lower value). We observe that for the same transmission range, the number of needed monitors increases linearly with k. For all values of k, as expected, the number of needed monitors decreases as the transmission range increases (i.e., when the coverage regions of more sensor nodes overlap). The decrease is dramatic at the beginning and then less dramatic afterwards. Furthermore, as the transmission increases, the number of required monitors approaches the lower bound, e.g., we need five monitors when the transmission range is larger than 160 m, $k = 1$, while the lower bound yields $\lceil 100/40 \rceil = 3$.

We now investigate the impact of the maximum allowed workload, w, on the performance of the Max-Flow k-monitoring algorithm. Fig. 9 plots the number of required monitors for Max-Flow algorithm when $n = 100$,

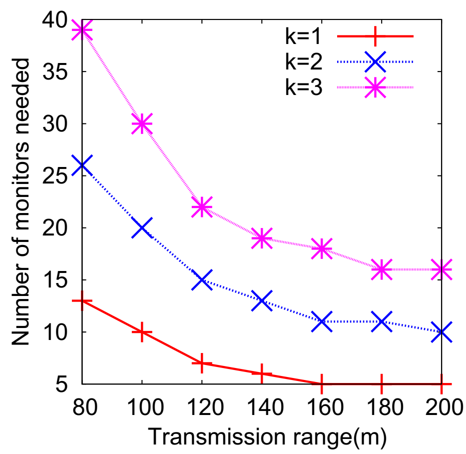


Fig. 8. Large-scale network: Max-Flow k-monitoring algorithm under grid uniform deployment, irregular radio range, $n = 100$, $w = 40$, $k = 1, 2$ or 3.

[3] The average transmission range of a sensor node is the average distance from the vertices of the polygon to the sensor node. For tractability, for a given transmission range r, we generate a polygon so that the distance from a vertex to the sensor node is uniformly distributed in $[0.6r, 1.4r]$, and the average distance from the vertices to the sensor node is in $[0.9r, 1.1r]$ (i.e., we allow the average transmission range of the generated polygon to have a relative error within 10%).

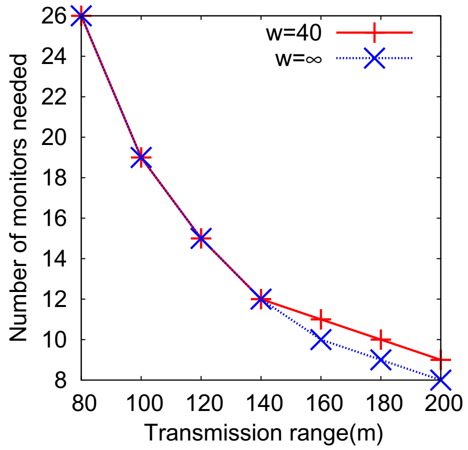


Fig. 9. Large-scale network: Max-Flow *k*-monitoring algorithm under grid uniform deployment, irregular radio range, $n = 100$, $k = 2$, $w = 40$ or ∞ (no limit on w).

$k = 2$, and the workload w is 40 or ∞ (i.e., no limit on the workload). We observe that the number of monitors required when $w = 40$ is similar to that when $w = \infty$, when the transmission range is short; it is slightly worse when the transmission range is large. This is because the number of sensor nodes that have overlapping coverage regions is below 40 when the transmission range is short, and hence allowing larger workload does not help reduce the number of required monitors.

Last, we investigate the requirement on the range of the non-interfering channel (used by the monitors to communicate with each other, see Section 3) so that the placed monitors form a connected graph. As expected, the required range tends to be larger when the number of placed monitors is smaller. In our simulation, the required range in the large-scale network is 60–300 m under grid uniform deployment and 80–380 m under uniform random deployment over all the simulation runs when using these three algorithms. These requirements can be easily satisfied by long-range radios [23].

6. Conclusion

In this paper, we formulated a *k*-monitoring problem for sensor network monitoring. We proved that this problem is NP-hard and proposed three algorithms, ILP, Max-Flow, and Max-Degree *k*-monitoring algorithms. The Max-Flow based algorithm has an approximation ratio of $\ln(kn)$, while the Max-Degree based algorithm uses a simple heuristic and has much shorter running time. We evaluated the performance of these three algorithms using extensive simulation. In small-scale networks, the latter two algorithms provide results close to the optimal solution from the ILP for relatively dense networks. In large-scale networks, the performance of these two algorithms are similar, and for relatively dense networks, the number of monitors required by both algorithms is close to the lower bound.

Acknowledgments

A preliminary version of this paper appeared in Milcom 2009 [32]. This work was partially supported by NSF awards CNS-0821597, CNS-0709005, NSF CAREER awards 0644188 and 0746841, and Qualtech Systems Incorporated.

Appendix A. NP-hard proof

Theorem 3. *The k-monitoring problem is NP-hard even for k = 1 and unlimited w.*

Proof. We prove this theorem by reducing a known NP-hard problem, geometric σ-center problem [8][4], to the *k*-monitoring problem. In geometric σ-center problem, we are given a constant σ, a set of nodes to be served, a set of candidate centers, and the distance from a center to a node (the nodes and centers are in the same plane and the distances satisfy triangular inequality). Let V denote the set of nodes, C denote the set of centers, and $d(v,c)$ denote the distance from v to c, $v \in V$, $c \in C$. The goal is to find a subset of centers, $C' \subseteq C$ and $|C'| = \sigma$, so that $\max_{v \in V}\min_{c \in C'} d(v,c)$ is minimized.

Our reduction is by showing that we can devise an optimal algorithm for σ-center problem from an optimal algorithm for the *k*-monitoring problem. Suppose we have an optimal algorithm, A_k, for the *k*-monitoring problem. We next show that we can devise an optimal algorithm, A_σ, for the σ-center problem. In the σ-center problem, suppose $|V| = n$ and $|C| = m$. We order the distance from a node to a center in non-decreasing order and denote them as $d_1 \leqslant d_2 \cdots \leqslant d_{mn}$. Then for a given d_i, the corresponding *k*-monitoring problem is as follows. The sensor node set is V, the candidate monitor set is C, $k = 1$, w is not limited, and a monitor can serve a sensor node if and only if its distance to the sensor node is within d_i. This instance can be solved using A_k, and let M_i denote the set of monitors in the optimal solution. We devise an algorithm, A_σ, for the σ-center problem as follows. It uses A_k to solve the *k*-monitoring problem by using increasingly larger d_i's. That is, it starts with d_1, and then uses d_2, and so on. The minimum d_i where $|M_i| \leqslant \sigma$ is an optimal solution for the σ-center problem ($C' = M_i, \max_{v \in V}\min_{c \in C'} d(v,c) = d_i$). □

Appendix B. Approximation ratio of the Max-Flow *k*-monitoring algorithm

We now prove that the Max-Flow *k*-monitoring algorithm has an approximation ratio of $\ln(kn)$. Our proof follows that in [2] and takes into account that each node needs to be monitored by k monitors.

Recall that M_c denotes the set of candidate monitors, M denotes the set of monitors when the algorithm terminates. Let M_i denote the set M at the beginning of the i-th iteration, where $M_0 = \emptyset$. Consider the situation at the beginning of iteration i, and let $M = M_i$. Let M^* be an optimal choice of monitors for the given instance and let φ^* be the corresponding optimal assignment. We define T_I and T_O as follows:

[4] This is denoted as geometric *K*-center problem in the literature; we use σ to replace K to avoid confusion with the parameter k in the *k*-monitoring problem.

$T_I = M \cap M^*,$
$T_O = (M_c \setminus M) \cap M^*.$

Since each sensor node needs to be monitored by k monitors, for convenience, we map a sensor node to k virtual sensor nodes; each virtual sensor node needs to be monitored by a single monitor. Let V' denote the set of virtual sensor nodes. Then $|V'| = k|V|$. Given an assignment $\varphi(v)$ for sensor node v, it is straightforward to find the corresponding assignment for v's virtual sensor nodes. In the following, with a slight abuse of notation, we use φ and φ^* to refer to the assignment to virtual sensor nodes. More specifically, if the monitor for virtual sensor node $v \in V'$ is m, we denote it as $\varphi(v) = m$. For a given assignment φ, let $S_\varphi(m)$ denote the set of virtual sensor nodes that m monitors. That is

$$S_\varphi(m) = \{v | v \in V', \varphi(v) = m\}.$$

For assignment φ, let k_i denote the number of v_i's virtual sensor nodes that have been monitored (i.e., $(k - k_i)$ of v_i's virtual sensor nodes are not monitored yet), and let $U(\varphi)$ denote the set of virtual sensor nodes that are not monitored. Then

$$|U(\varphi)| = \sum_{\forall v_i \in V} (k - k_i).$$

We define

$$X(M, \varphi) = |U(\varphi)|,$$
$$X(M) = \min_{\forall \varphi}\{X(M, \varphi)\}.$$

Then $X(M)$ denotes the minimum number of virtual sensor nodes that are not monitored over all feasible assignments for monitor set M. It is clear that $X(M_0) = kn$.

Let OPT denote the set of minimal assignment for M, i.e., the set of assignments that provides $X(M)$:

$$OPT = \{\varphi | X(M, \varphi) = X(M)\}.$$

For any feasible assignment φ, let $hit(\varphi, \varphi^*)$ denote the number of virtual sensor nodes that are monitored by the same monitors in M^* and M, namely,

$$hit(\varphi, \varphi^*) = |\{v \in V' | \varphi(v) = \varphi^*(v)\}|.$$

Let $\hat{\varphi}$ be an assignment in OPT, for which $hit(\hat{\varphi}, \varphi^*)$ is maximal (among all the assignments in OPT). That is,

$$hit(\hat{\varphi}, \varphi^*) = \max\{hit(\varphi, \varphi^*) | \varphi \in OPT\}.$$

Lemma 4. *For any virtual node* $v \in U(\hat{\varphi}), \varphi^*(v) \in T_O$.

Proof. We prove this lemma by contradiction. Assume there exists a virtual node $v \in U(\hat{\varphi})$ such that

$$\varphi^*(v) = m, m \in T_I.$$

Clearly, $|S_{\hat{\varphi}}(m)| = w$, for otherwise we can assign v to m, reducing $X(M)$. Since v is monitored by m in φ^*, and φ^* is feasible, there exists a node $z \in S_{\hat{\varphi}}(m) \setminus S_{\varphi^*}(m)$. Let φ' be an assignment identical to $\hat{\varphi}$, except that $\varphi'(v) = \varphi^*(v)$, and $\varphi'(z)$ is undefined (namely, we take z out of $S_{\varphi'}(m)$). Clearly, this is a feasible assignment. Also note that $v \in U(\hat{\varphi})$ implies that $X(M, \varphi') = X(M, \hat{\varphi}) = X(M)$. Thus, $\varphi' \in OPT$. However, by the way that φ' is defined, $hit(\varphi', \varphi^*) = hit(\hat{\varphi}, \varphi^*) + 1$, contradicting with the assumption that $\hat{\varphi}$ maximizes $hit(\varphi,\varphi^*)$ among all the assignments in OPT. □

Lemma 5. *There exists a monitor* $m \in T_O$ *such that*

$$|S_{\varphi^*}(m) \cap U(\hat{\varphi})| \geqslant \frac{|U(\hat{\varphi})|}{|T_O|}.$$

Proof. By Lemma 4,

$$U(\hat{\varphi}) = \cup_{m \in T_O}(S_{\varphi^*}(m) \cap U(\hat{\varphi})).$$

Since $S_{\varphi^*}(m) \neq S_{\varphi^*}(m')$ for $m \neq m'$ (each virtual sensor node can only be monitored by a single monitor), we have

$$|U(\hat{\varphi})| = \sum_{m \in T_O} |S_{\varphi^*}(m) \cap U(\hat{\varphi})|.$$

Following the pigeonhole principle, at least one of the terms in the summation is of $|U(\hat{\varphi})|/|T_O|$ or more. □

Lemma 6

$$X(M_{i+1}) \leqslant X(M_i)\left(1 - \frac{1}{|M^*|}\right)$$

Proof. Let m' be the monitor whose existence is asserted by Lemma 5. Let $M' = M \cup \{m\}$. Define φ' to be an assignment that is equivalent to $\hat{\varphi}$ except that if $\varphi^*(v) = m'$ and $v \in U(\hat{\varphi})$, then set $\varphi'(v) = m'$. Then every vertex that is now in $S_{\varphi'}(m')$ decreases $X(M)$ by 1 (since every such vertex is unassigned in $\hat{\varphi}$). Furthermore, the feasibility of φ^* assures that m' monitors no more than w virtual sensor nodes in φ', and thus φ' is feasible. Therefore,

$$X(M') \leqslant X(M', \varphi')$$
$$\leqslant X(M, \hat{\varphi}) - |S_{\varphi^*}(m') \cap U(\hat{\varphi})| \leqslant X(M) - |S_{\varphi^*}(m') \cap U(\hat{\varphi})|.$$

By Lemma 5, for m' and M' defined above,

$$X(M') \leqslant X(M) - \frac{|U(\hat{\varphi})|}{|T_O|} = X(M)\left(1 - \frac{1}{|T_O|}\right) \leqslant X(M)\left(1 - \frac{1}{|M^*|}\right).$$

Since the choice of M_{i+1} in the Max-Flow k-monitoring algorithm maximizes the integral flow (i.e., it minimizes the number of unassigned virtual nodes), we have

$$X(M_{i+1}) \leqslant X(M') \leqslant X(M_i)\left(1 - \frac{1}{|M^*|}\right).$$

We now prove that the approximation ratio of the Max-Flow k-monitoring algorithm is $\ln(kn)$.

Proof. From Lemma 6, we have

$$X(M_i) \leqslant X(M_0)\left(1 - \frac{1}{|M^*|}\right)^i = kn\left(1 - \frac{1}{|M^*|}\right)^i.$$

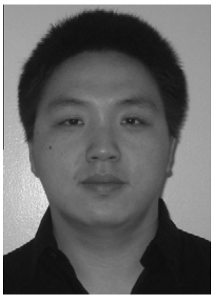
Since $(1 - \frac{1}{x})^x, x \geqslant 0$ is an increasing function that converges to $1/e$ as x approaches ∞, we have

$$X(M_i) \leqslant kne^{-\frac{i}{|M^*|}}.$$

Therefore, after $|M^*|\lceil \ln(kn)\rceil$ iterations, $X(M_i) \leqslant 1$ and hence $X(M_i) = 0$, since $X(M_i)$ is an integer. Since we add a single monitor to the monitor set in each iteration, we end up with at most $||M^*|\lceil \ln(kn)\rceil$ monitors, thus proving this theorem. □

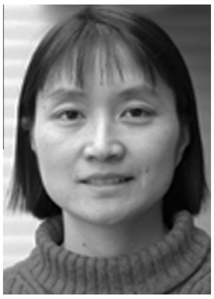
References

[1] A. Adya, V. Bahl, R. Chandra, L. Qiu, Architecture and techniques for diagnosing faults in IEEE 802.11 infrastructure networks, in: Proceedings of ACM MobiCom, September 2004, pp. 30–44.
[2] J. Bar-Ilan, G. Kortzars, D. Peleg, How to allocate network centers, Journal of Algorithms 15 (1993).
[3] K. Chakrabarty, S.S. Iyengar, H. Qi, E. Cho, Grid coverage for surveillance and target location in distributed sensor networks, IEEE Transactions on Computing 51 (12) (2006) 1448–1453.
[4] Y.-C. Cheng, M. Afanasyev, P. Verkaik, P. Benko, J. Chiang, A.C. Snoeren, S. Savage, G.M. Voelker, Automating cross-layer diagnosis of enterprise wireless networks, in: Proceedings of ACM SIGCOMM, Kyoto, Japan, August 2007, pp. 25–36.
[5] J. Cote, B. Wang, W. Zeng, Z. Shi, Capability and fidelity of mote-class wireless sniffers, in: Proceedings of IEEE GLOBECOM, December 2010.
[6] F. Dressler, R. Nebel, A. Awad, Distributed passive monitoring in sensor networks, in: Proceedings of IEEE INFOCOM, 2007.
[7] M. Dyer, J. Beutel, T. Kalt, P. Oehen, L. Thiele, K. Martin, P. Blum, Deployment Support Network – a toolkit for the development of WSNs, in: EWSN, 2007, pp. 195–211.
[8] T. Feder, D. Greene, Optimal algorithms for approximate clustering, in: STOC, 1988, pp. 434–444.
[9] W. Gao, G. Cao, Fine-grained mobility characterization: steady and transient state behaviors, in: MobiHoc '10: Proceedings of the Eleventh ACM International Symposium on Mobile Ad Hoc Networking and Computing, ACM, New York, NY, USA, 2010, pp. 61–70.
[10] Z. Guo, B. Wang, P. Xie, W. Zeng, J.-H. Cui, Efficient error recovery with network coding in underwater sensor networks, Ad Hoc Networrks 7 (4) (2009) 791–802.
[11] C.-F. Hsin, M. Liu, A distributed monitoring mechanism for wireless sensor networks, in: Proceedings of ACM Workshop on Wireless Security (WiSe), 2002, pp. 57–66.
[12] A.P. Jardosh, K.N. Ramachandran, K.C. Almeroth, E.M. Belding-Royer, Understanding congestion in IEEE 802.11b wireless networks, in: Proceedings of ACM SIGCOMM Internet Measurement Conference (IMC), 2005, pp. 279–292.
[13] Y. Keung, B. Li, Q. Zhang, The intrusion detection in mobile sensor network, in: MobiHoc '10: Proceedings of the Eleventh ACM International Symposium on Mobile Ad Hoc Networking and Computing, ACM, New York, NY, USA, 2010, pp. 11–20.
[14] V. Krunic, E. Trumpler, R. Han, NodeMD: diagnosing node-level faults in remote wireless sensor systems, in: MobiSys, June 2007, pp. 43–56.
[15] C. Liu, G. Cao, A multi-poller based energy-efficient monitoring scheme for wireless sensor networks, in: IEEE INFOCOM Mini Conference, April 2009, pp. 2781–2785.
[16] R. Mahajan, M. Rodrig, D. Wetherall, J. Zahorjan, Analyzing the MAC-level behavior of wireless networks in the wild, in: Proceedings of ACM SIGCOMM, Pisa, Italy, September 2006, pp. 75–86.
[17] D. O'Rourke, S. Fedor, C. Brennan, M. Collier, Reception region characterisation using a 2.4 GHz direct sequence spread spectrum radio, in: EmNets '07: Proceedings of the 4th workshop on Embedded networked sensors, ACM, New York, NY, USA, 2007, pp. 68–72.
[18] S. Rallapalli, L. Qiu, Y. Zhang, Y.-C. Chen, Exploiting temporal stability and low-rank structure for localization in mobile networks, in: MobiCom '10: Proceedings of the sixteenth annual international conference on Mobile computing and networking, ACM, New York, NY, USA, 2010, pp. 161–172.
[19] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, D. Estrin, Sympathy for the sensor network debugger, in: SenSys, November 2005, pp. 255–267.
[20] M. Ringwald, K. Romer, A. Vitaletti, Passive inspection of sensor networks, in: DCOSS, 2007, pp. 205–222.
[21] S. Rost, H. Balakrishnan, Memento: a health monitoring system for wireless sensor networks, in: IEEE SECON, 2006, pp. 575–584.
[22] T. Schmid, H. Dubois-Ferriére, M. Vetterli, Sensorscope: experiences with a wireless building monitoring, in: Proceedings of Workshop on Real-World Wireless Sensor Networks, 2005, pp. 332–343.
[23] N. Sharma, J. Gummeson, D. Irwin, P. Shenoy, SRCP: simple remote control for perpetual high-power sensor networks, in: EWSN, 2009, pp. 358–374.
[24] A. Sheth, C. Doerr, D. Grunwald, R. Han, D.C. Sicker, MOJO: a distributed physical layer anomaly detection system for 802.11 WLANs, in: Proceedings of ACM MobiSys, 2006, pp. 191–204.
[25] K. Suh, Y. Guo, J. Kurose, D. Towsley, Locating network monitors: complexity, heuristics, and coverage, Elsevier Journal of Computer Communications (2005) 1564–1577.
[26] G. Tolle, D. Culler, Design of an application-cooperative management system for wireless sensor networks, in: EWSN, January 2005, pp. 121–132.
[27] G. Tolle, J. Polastre, R. Szewczyk, N. Turner, K. Tu, P. Buonadonna, S. Burgess, D. Gay, W. Hong, T. Dawson, D. Culler, A macroscope in the redwoods, in: SenSys, November 2005, pp. 51–63.
[28] M. Wachs, J.I. Choi, J.W. Lee, K. Srinivasan, Z. Chen, M. Jain, P. Levis, Visibility: a new metric for protocol design, in: SenSys, November 2007, pp. 73–86.
[29] J. Wang, N. Zhong, Efficient point coverage in wireless sensor networks, Journal of Combinatorial Optimization 11 (2006) 291–304.
[30] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, M. Welsh, Monitoring volcanic eruptions with a wireless sensor network, in: Proceedings of European Workshop on Sensor Networks (EWSN), January 2005, pp. 108–120.
[31] J. Yeo, M. Youssef, A. Agrawala, A framework for wireless LAN monitoring and its applications, in: Procedings of ACM Workshop on Wireless Security (WiSe), 2004, pp. 70–79.
[32] W. Zeng, X. Chen, Y.-A. Kim, Z. Bu, W. Wei, B. Wang, Z.J. Shi, Delay monitoring for wireless sensor networks: An architecture using air sniffers, in: IEEE Military Communications Conference (Milcom), Boston, MA, October 2009, pp. 1–8.
[33] J. Zhao, R. Govindan, D. Estrin, Residual energy scans for monitoring wireless sensor networks, in: WCNC, March 2002, pp. 17–21.
[34] J. Zhao, R. Govindan, D. Estrin, Computing aggregates for monitoring wireless sensor networks, in: SNPA, May 2003, pp. 139–148.
[35] G. Zhou, T. He, S. Krishnamurthy, J.A. Stankovic, Models and solutions for radio irregularity in wireless sensor networks, ACM Transactions on Sensor Networks 2 (2) (2006) 221–262.

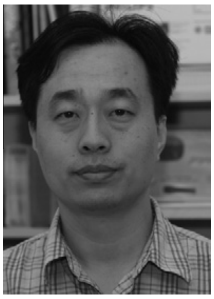


Xian Chen received his B.S. degree in Applied Mathematics from Beijing University of Aeronautics and Astronautics, China in 2003, and M.S. degree in Computer Science from Beijing JiaoTong University, China in 2006. He is currently a Ph.D. candidate in Computer Science Department at the University of Connecticut. His research interests are in the areas of computer networks, fault diagnosis and performance modeling.

Yoo-Ah Kim received her Ph.D. degree in Computer Science from the University of Maryland in 2005. She is currently a research fellow in National Center for Biotechnology Information (NCBI) at National Institute of Health. Before joining to NCBI, she was an assistant professor in the Department of Computer Science and Engineering at the University of Connecticut, Storrs, Connecticut. Her research interests include design and analysis of algorithms, bioinformatics, and parallel and networked systems.



Bing Wang received her B.S. degree in Computer Science from Nanjing University of Science and Technology, China in 1994, and M.S. degree in Computer Engineering from Institute of Computing Technology, Chinese Academy of Sciences in 1997. She then received M.S. degrees in Computer Science and Applied Mathematics, and a Ph.D. in Computer Science from the University of Massachusetts, Amherst in 2000, 2004, and 2005 respectively. Afterwards, she joined the Computer Science and Engineering Department at the University of Connecticut as an assistant professor. Her research interests are in Computer Networks, Multimedia, and Distributed Systems. She received NSF CAREER award in 2008.



Wei Wei received his B.S. degree in Applied Mathematics from Beijing University, China in 1992, and M.S. degree in Statistics from Texas A&M University in 2000. He then received M.S. degrees in Computer Science and Applied Mathematics, and a Ph.D. in Computer Science from the University of Massachusetts, Amherst in 2004, 2004, and 2006 respectively. He is currently a senior postdoc in the Department of Computer Science at the University of Massachusetts, Amherst. His research interests are in the areas of computer networks, distributed embedded systems and performance modeling.

Zhijie Jerry Shi is currently an Associate Professor of Computer Science and Engineering at the University of Connecticut. He received his Ph.D. degree from Princeton University in 2004 and his M.S. and B.S. degrees from Tsinghua University, China, in 1996 and 1992, respectively. He is a member of IEEE and ACM. Dr. Shi received US National Science Foundation CAREER award in 2006. His current research interests include underwater sensor networks, sensor network security, hardware mechanisms for secure and reliable computing, side channel attacks and countermeasures, and primitives for cipher designs.

Yuan Song received his B.S. and M.S. degrees both in Communications Engineering from the Xidian University in 2005 and 2008, respectively. He is currently a Ph.D. candidate in the Department of Computer Science and Engineering, University of Connecticut. His research interests include wireless communication, network security and mobile computing.