General Maximal Lifetime Sensor-Target Surveillance Problem and Its Solution

Hai Liu, *Member*, *IEEE*, Xiaowen Chu, *Member*, *IEEE*, Yiu-Wing Leung, *Senior Member*, *IEEE*, Xiaohua Jia, *Senior Member*, *IEEE*, and Peng-Jun Wan, *Member*, *IEEE*

Abstract—We address a new and general maximal lifetime problem in sensor-target surveillance. We assume that each sensor can watch at most k targets ($k \geq 1$) and each target should be watched by h sensors ($h \geq 1$) at any time. The problem is to schedule sensors to watch targets and forward the sensed data to a base station such that the lifetime of the surveillance network is maximized. This general problem includes the existing ones as its special cases ($k = 1$ and $h = 1$ in [12] and $k = 1$ and $h \geq 2$ in [13]). It is also important in practice because some sensors can monitor multiple or all targets within their surveillance ranges and multisensor fusion (i.e., watching a target by multiple sensors) gives better surveillance results. The problem involves several subproblems and one of them is a new matching problem called (k, h)-*matching*. The (k, h)-matching problem is a generalized version of the classic bipartite matching problem (when $k = h = 1$, (k, h)-matching becomes bipartite matching). We design an efficient *(k, h)-matching algorithm* to solve the (k, h)-matching problem and then solve the general maximal lifetime problem. As a byproduct of this study, the (k, h)-matching problem and the proposed (k, h)-matching algorithm can potentially be applied to other problems in computer science and operations research.

Index Terms—Wireless sensor networks, maximal lifetime, scheduling, matching, routing.

1 Introduction

Sensor-target surveillance is one of the promising applications of wireless sensor networks. Typically a sensor-target surveillance network consists of sensor nodes (sensors for short), targets, and a base station in a surveillance region. Sensors are used to monitor the targets and collect and transmit the sensed data to the base station. In many applications, sensors are used to monitor some given targets located at fixed positions (e.g., monitor the temperature of some machines in a factory, monitor the chemical composition around some cargo containers which carry dangerous chemicals in long journey of shipment, etc.).

Sensors are usually powered by batteries with limited energy reserve. Therefore, it is desirable to prolong or maximize the lifetime of the sensor-target surveillance network, where the lifetime is the duration that the surveillance network can be properly operated (i.e., all targets can be watched and all data can be transmitted). Once the lifetime expires, certain sensors are depleted of energy and consequently certain target(s) cannot be watched or some data cannot be forwarded to the base station.

To maximize the lifetime of the sensor networks, there are three solution approaches: 1) energy efficient routing [1], [6], [8], [9], [10] by which data are efficiently gathered and routed to the base station, 2) on/off scheduling [3], [4], [19] by which the sensors are scheduled to take on or off modes in order to reduce energy consumption while keeping the surveillance network functioning, and 3) integrated routing and on/off scheduling [12], [13] which combines the first two approaches. Among the above three approaches, the integrated routing and on/off scheduling approach is promising because it takes two inter-dependent issues (i.e., routing and on/off scheduling) into account. Two recent studies [12], [13] adopted this approach and solved the resulting maximal lifetime problems for two respective models: 1) a sensor can watch one target at a time and a target should be watched by one sensor at any time [12], and 2) a sensor can watch one target at a time and a target should be watched by h sensors ($h \geq 2$) at any time [13].

In this paper, we study a general maximal lifetime sensor-target surveillance (MLSTS) problem. In this problem, each sensor can watch at most k targets ($k \geq 1$) and each target should be watched by h sensors at any time ($h \geq 1$). This general problem is practically important for two reasons. First, some sensors can watch multiple or all targets within their surveillance ranges. The following are two examples:

Example 1. In industry, a sensor equipped with k pairs of infrared light emitting diodes (IR-LEDs) can be used to monitor the speed of k rotating machines [7]. In this example, each sensor can watch multiple targets (rotating machines) within its surveillance range.

Example 2. Suppose some cargo containers carry dangerous chemicals and they are being shipped in a long journey or are temporarily placed at a port. To detect the possible leakage, sensors can be used to sample the air and detect these chemicals. In this example, each sensor can watch all targets (containers) within its surveillance range.

Second, it is desirable to watch a target by multiple sensors because this can give better information accuracy,

- *H. Liu, X. Chu, and Y.-W. Leung are with the Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Hong Kong SAR. E-mail: {hliu, chxw, ywleung}@comp.hkbu.edu.hk.*
- *X. Jia is with the Department of Computer Science, City University of Hong Kong, Kowloon Tong, Hong Kong SAR. E-mail: jia@cs.cityu.edu.hk.*
- *P.-J. Wan is with the Department of Computer Science, Illinois Institute of Technology, 10 W. 31st Street, Chicago, IL 60616. E-mail: wan@cs.iit.edu.*

Manuscript received 5 May 2010; revised 6 Nov. 2010; accepted 18 Nov. 2010; published online 19 Jan. 2011.
Recommended for acceptance by A. Nayak.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2010-05-0269.
Digital Object Identifier no. 10.1109/TPDS.2011.42.

better noise suppression, and better robustness to sensor failure. In fact, this is a well-established field in signal processing called multisensor fusion [14]. The general problem can take the above practical factors into account. On the other hand, the general problem includes the existing problems as its special cases (i.e., $k = 1$ and $h = 1$ in [12], and $k = 1$ and $h \geq 2$ in [13]).

Our contributions include the following:

- *General MLSTS problem*: We address a new and general MLSTS problem which is practically important (see the above discussion) and novel in its generality (it includes the existing problems [12], [13] as its special cases).
- *New matching problem and algorithm*: The general MLSTS problem can be decomposed into several sub-problems and one of them is a new matching problem called (k, h)-matching. The (k, h)-matching problem is a generalized version of the classic bipartite matching problem [5] (when $k = h = 1$, (k, h)-matching becomes bipartite matching). Given a bipartite graph in which the vertices are partitioned into two disjoint sets (e.g., these sets contain the sensors and the targets, respectively), the (k, h)-matching problem is to match the vertices in one set to the vertices in another set such that: 1) each vertex in the first set is matched to k vertices in the second set, and 2) each vertex in the second set is matched to h vertices in the first set. We design an efficient (k, h)-matching algorithm for this matching problem. As a byproduct of this study, the (k, h)-matching problem and algorithm can potentially be applied to other problems in computer science and operations research.
- *Solution to general MLSTS problem*: We solve the general MLSTS problem and apply the proposed (k, h)-matching algorithm in a key step. Our solution involves a conjecture, and we theoretically prove its special case and empirically verify its general case through over 6,000,000 randomly generated problem instances.

The rest of the paper is organized as follows: in Section 2, we survey the relevant existing results. In Section 3, we formulate the general MLSTS problem. In Section 4, we define the (k, h)-matching problem which is one of the subproblems of the general MLSTS problem. We design an efficient algorithm to solve this matching problem. In Section 5, we solve the MLSTS problem. In Section 6, we present a numeric example for illustration and simulation results for performance evaluation. We conclude our work in Section 7.

2 Related Work

In the literature, three major approaches have been proposed for maximizing the lifetime of wireless sensor networks: 1) energy efficient routing [1], [6], [8], [9], [10], 2) on/off scheduling [3], [4], [19], and 3) integrated routing and on/off scheduling [12] [13]. We survey these existing approaches in Section 1 in the supplemental material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.42.

TABLE 1
Notations

n	number of sensors
m	number of targets
s_0	base station
S	set of sensors: $S = \{s_1, s_2, \ldots, s_n\}$
T	set of targets: $T = \{t_1, t_2, \ldots, t_m\}$
$S(j)$	set of sensors that can watch target t_j
$T(i)$	set of targets that are within the surveillance range of sensor s_i
$N(i)$	set of neighboring sensors of sensor s_i
E_i	initial energy reserve of sensor s_i
φ_{ij}	energy for transmitting one unit of data from sensor s_i to s_j
R	data rate of sensors while watching targets (i.e., the amount of surveillance data generated per unit time)
e_S, e_T, e_R	energy for sensing, transmitting, receiving one unit of data respectively (e_S, e_T and e_R can take different values)

3 Formulation of the MLSTS Problem

In this section, we formulate the general MLSTS problem. We define the notations in Table 1. Sensors, targets, and the base station can be placed anywhere in the surveillance area and their positions are fixed after placement. Their location information can be obtained via the improved centroid localization algorithm [2]. Sensors are time synchronized where synchronization can be achieved by using a time synchronization scheme, e.g., [17], during system initialization. Each sensor can watch at most $k (k \geq 1)$ targets within its surveillance range. For the special case that a sensor can watch all targets within its surveillance range, we set $k = \max\{k_1, k_2, \ldots, k_n\}$ where k_i targets are within the surveillance range of sensor s_i (an alternative is to set $k = m$ but this involves larger computation complexity as $m \geq \max\{k_1, k_2, \ldots, k_n\}$). Each target must be watched by $h (h \geq 1)$ sensors at any time. Each sensor has an initial energy reserve, a fixed surveillance range, and a transmission range which could be fixed or adjustable. If the transmission range is fixed, φ_{ij} can be represented as $\varphi_{ij} = \lambda e_T$, where λ is a constant. If the transmission range is adjustable, the largest possible transmission range depends on the physical features of the sensors (e.g., battery power and antenna type). The source sensor adjusts its transmission range to exactly reach the destination sensor in order to save energy. The energy for transmitting one data unit from sensor s_i to s_j is equal to $\varphi_{ij} = e_T(d_{ij})^a$, where d_{ij} is the euclidean distance between sensors s_i and s_j and α is the signal decline factor.

The general MLSTS problem is to schedule the sensors to watch the targets and route the sensed data to the base station, such that the lifetime of the surveillance network is maximized. The lifetime is the duration that the surveillance network can be operated until at least one of the following events occurs: 1) there exists at least one target, say t_j, such that the number of sensors watching this target is less than h because some sensors are depleted of energy, and 2) the sensed data cannot be forwarded to the base station because the network becomes disconnected due to depletion of energy.

Each sensor is either in the *active* mode (in which it senses and/or forwards data) or in the *sleep* mode (in which it neither senses nor forwards data). The operation time of the surveillance network is divided into a sequence of sessions. In each session, some sensors are in the active mode to watch the targets and forward the sensed data to the base station while other sensors are in the sleep mode. Each sensor remains in the same mode within a session. In a session, a sensor-target surveillance schedule specifies the sensors that should be active and the target(s) that each of these sensors should watch in this session, and a sensor-target surveillance tree specifies how the sensed data are routed to the base station in this session. The MLSTS problem is to determine the sensor-target surveillance schedules and trees for all the necessary sessions in order to maximize the lifetime. Once the surveillance schedules are determined, the number of sessions is fixed throughout the surveillance lifetime.

At the system initialization stage, the base station gathers location information of all sensors by using a data gathering algorithm, e.g. [10]. The base station determines the surveillance schedules and trees using the method described in Section 5, and then disseminates them to all sensors by using a broadcasting algorithm, e.g., [11]. When the system starts operation, all sensors follow the surveillance schedules to watch targets and send/receive data by using a collision-free medium access control protocol, e.g., [15].

4 (k, h)-Matching Problem and Algorithm

In this section, we formulate and solve a new matching problem called (k, h)-matching. It is a generalized version of the classic bipartite matching problem [5] (when $k = h = 1$, (k, h)-matching becomes bipartite matching). The (k, h)-matching problem arises as a subproblem of the general MLSTS problem (the details will be given in Section 5). It can potentially be applied to other problems in computer science and operations research.

4.1 (k, h)-Matching Problem

In the following, we define the (k, h)-matching problem.

Definition 1 (Loose (k, h)-matching). *Let $G(S \cup T, E)$ be a bipartite graph where E is the set of edges of this graph and S and T are two disjoint sets of the vertices, $|S| \geq h, |T| \geq k$, and $k|S| \geq h|T|$. A loose (k, h)-matching is defined to be a set of edges $\psi (\psi \subseteq E)$ such that*

1. *Each vertex in S is an endpoint of at most k edges in ψ.*
2. *Each vertex in T is an endpoint of at most h edges in ψ.*

The loose (k, h)-matching means that each vertex in S is matched to at most k vertices in T and each vertex in T is matched to at most h vertices in S. We further define (k, h)-matching as follows:

Definition 2 ((k, h)-matching). *A loose (k, h)-matching is said to be a (k, h)-matching if it contains $k|S|$ edges and $k|S| = h|T|$.*

The (k, h)-matching means that each vertex in S is matched to exactly k vertices in T and each vertex in T is matched to exactly h vertices in S. The (k, h)-matching problem is defined as follows:

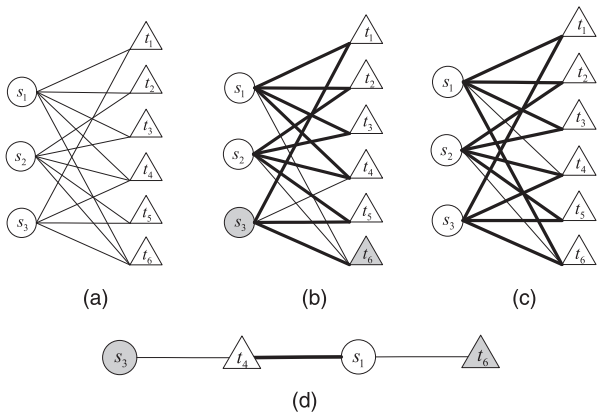


Fig. 1. Compute a (k, h)-matching. (a) A bipartite graph. (b) A loose (4, 2)-matching. (c) A (4, 2)-matching. (d) An augment path.

Definition 3 ((k, h)-matching problem). *Given a bipartite graph $G(S \cup T, E)$, the (k, h)-matching problem is to determine a (k, h)-matching if it exists.*

4.2 (k, h)-Matching Algorithm

To the best of our knowledge, the (k, h)-matching problem has not been studied in the literature. For the special case in which $k = h = 1$, the problem reduces to the classic bipartite matching problem [5]. For the general case in which k and h are any positive integers, the (k, h)-matching problem is more challenging because there are much more choices in the matching process and it is challenging to efficiently select a choice that can fulfill the matching criteria. For example, when $n = m = 20$, bipartite matching is to match every vertex to 1 of the 20 vertices and this involves 20 possible choices, but (10, 10)-matching is to match every vertex to 10 of the 20 vertices and this involves 184,756 possible choices.

In this section, we propose a novel (k, h)-matching algorithm which determines a (k, h)-matching in a given bipartite graph as long as this matching exists. Before describing the details of this algorithm, we need the following definitions.

Definition 4 (k-unsaturated). *A vertex in any vertex set is said to be k-unsaturated if the number of edges associated with the vertex in the loose (k, h)-matching is less than k.*

Definition 5 (k-saturated). *A vertex in any vertex set is said to be k-saturated if the number of edges associated with the vertex in the loose (k, h)-matching is exactly k.*

Definition 6 (Augment path). *with respect to a loose (k, h)-matching. An augment path in the bipartite graph is a path having the following properties: 1) it starts with a k-unsaturated vertex in S and ends with an h-unsaturated vertex in T; 2) if an edge does not belong to the loose (k, h)-matching, the adjacent edges in this path belong to the loose (k, h)-matching; and 3) if an edge belongs to the loose (k, h)-matching, the adjacent edges in this path do not belong to the loose (k, h)-matching.*

Take the example in Fig. 1, 1a is a given bipartite graph $G(S \cup T, E)$, where $S = \{s_1, s_2, s_3\}$ and $T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$. Bold edges in Fig. 1b form a loose (4, 2)-matching in which each vertex in S matches at most four vertices in T and

each vertex in T matches at most two vertices in S. s_1 and s_2 are 4-saturated and t_1, t_2, t_3, t_4, and t_5 are 2-saturated. Gray node s_3 is 4-unsaturated and gray node t_6 is 2-unsaturated. Bold edges in Fig. 1c form a (4, 2)-matching in which all vertices in S are 4-saturated and all vertices in T are 2-saturated. With respect to the loose (4, 2)-matching in Fig. 1b, 1d shows an augment path $s_3 \rightarrow t_4 \rightarrow s_1 \rightarrow t_6$ which starts with 4-unsaturated s_3 and ends with 2-unsaturated t_6. Edges (s_3, t_4) and (s_1, t_6) do not belong to the loose (4, 2)-matching while edge (t_4, s_1) belongs to the loose (4, 2)-matching.

In the following, we propose a novel (k, h)-matching algorithm to compute a (k, h)-matching in bipartite graphs. Our (k, h)-matching algorithm is inspired by the augment path technique for perfect matching. Let ψ denote a loose (k, h)-matching, (s_i, t_j) denote an edge from S to T, and (t_j, s_i) denote an edge from T to S. Although these edges have no direction in the graph, this notation helps describe our algorithm. The algorithm starts with any loose (k, h)-matching in the bipartite graph. Each time, it arbitrarily selects a k-unsaturated vertex of S and tries to find an augment path starting from the vertex. With respect to a loose (k, h)-matching, if an edge is in the matching, this edge is called *matching edge*; otherwise, it is called *nonmatching edge*. According to Definition 6, an augment path always has one more nonmatching edge than matching edges. We change the original matching edges to nonmatching edges and change the original nonmatching edges to matching edges. As a result, the number of matching edges increases by one. We keep on finding augment paths for any k-unsaturated vertex of S and increasing the size of ψ. The process is repeated until ψ becomes a (k, h)-matching. The (k, h)-matching algorithm is presented as follows:

(*k*, *h*)-Matching Algorithm
Input: a bipartite graph $G=(S \cup T, E)$, k, h ($k|S| = h|T|$).
Output: a (k, h)-matching Ψ.

```
1: Begin
2:   choose an arbitrary edge of E and add it to Ψ;
3:   while there exist k-unsaturated vertices of S do
4:   {
5:     select the first k-unsaturated s in S and label s
       with *;
6:     for t_j ∈ T(s) and (s, t_j) ∉ Ψ
7:       label t_j with s;
8:     while newly labeled t_j is h-saturated do
9:     { // an augment path is not found yet
10:      for all newly labeled t_j and unlabeled s_i ∈ S(t_j)
11:        if (s_i, t_j) ∈ Ψ
12:          label s_i with t_j;
13:      for all newly labeled s_i and unlabeled t_j ∈ T(s_i)
14:        if (s_i, t_j) ∉ Ψ
15:          label t_j with s_i;
16:    }// an augment path is found
17:    remove matching-edges of the augment path
       from Ψ;
18:    add non-matching-edges of the augment path
       to Ψ;
19:  }
20: End
```

In the algorithm, $S(t_j)$ denotes the set of vertices in S which are connected to t_j and $T(s_i)$ denotes the set of vertices in T which are connected to s_i. We present an example in Section 2 in the supplemental material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.42, to illustrate the (k, h)-matching algorithm. The following theorem states the correctness of the algorithm.

Theorem 1. *The (k, h)-Matching algorithm can find a (k, h)-matching if this (k, h)-matching exists in the bipartite graph.*

Theorem 2. *The (k, h)-Matching algorithm has a time complexity of* $O(k|S|(|S| + |T|))$.

Proofs of Theorem 1 and Theorem 2 can be found in Section 3 in the supplemental material, which can be found on the Computer Society Digital Library.

In this section, we propose a (k, h)-matching algorithm to solve the (k, h)-matching problem. The (k, h)-matching problem and its solution will be used in next section in determining the sensor-target surveillance schedule.

5 Solution to the MLSTS Problem

The framework for solving the general MLSTS problem is as follows: 1) determine the maximal lifetime, 2) determine a sensor-target surveillance schedule for each session, and 3) determine a sensor-target surveillance tree for each session. Although this framework is similar to the one studied in [12] and [13], the new and major challenge is the surveillance scheduling problem in step 2 and the main contribution of our solution is to solve this scheduling problem.

5.1 Maximizing Lifetime

Let L be the lifetime of the surveillance network, x_{ij} be the total time that sensor s_i watches target t_j, and f_{ij} be the amount of data transmitted from s_i to s_j. The problem of maximizing the lifetime can be formulated as the following:

$$\begin{aligned} &\text{Maximize } L \\ &\text{Subject to} \\ &\sum_{t_j \in T(i)} x_{ij} \le kL \qquad \forall s_i \in S; \end{aligned} \tag{1}$$

$$\sum_{s_i \in S(j)} x_{ij} = hL \quad \forall t_j \in T; \tag{2}$$

$$e_S R \sum_{t_j \in T(i)} x_{ij} + \sum_{s_j \in N(i) \cup \{s_0\}} \varphi_{ij} f_{ij} + e_R \sum_{s_j \in N(i)} f_{ji} \le E_i \quad \forall s_i \in S; \tag{3}$$

$$R \sum_{t_j \in T(i)} x_{ij} + \sum_{s_j \in N(i)} f_{ji} = \sum_{s_j \in N(i) \cup \{s_0\}} f_{ij} \quad \forall s_i \in S; \tag{4}$$

$$0 \le x_{ij} \le L \quad \forall s_i \in S, t_j \in T; \tag{5}$$

$$f_{ij} \ge 0 \quad \forall s_i \in S, s_j \in S \cup \{s_0\}. \tag{6}$$

The objective function is the lifetime of the surveillance network. Constraint (1) specifies that the total time that sensor s_i can watch targets should not exceed k times the network lifetime because each sensor can watch at most k targets at a time. Constraint (2) specifies that the total time that target t_j is under surveillance is equal to h times the

network lifetime because each target should be watched by h sensors throughout the lifetime. At this stage, constraint (2) need not ensure that each target is watched by h sensors at any time (this requirement will be met in Section 5.2 in which the surveillance schedules are determined). Constraint (3) specifies that the total energy consumed by each sensor should not exceed its initial energy reserve, where the total energy consumption of each sensor consists of three components: 1) the energy for watching targets, 2) the energy for transmitting data where φ_{ij} is equal to λe_T based on the fixed transmission range model or $e_T(d_{ij})^\alpha$ based on the adjustable transmission range model, and 3) the energy for receiving data. Since the energy consumption in the sleep mode is several orders of magnitude smaller than that in the active mode [16], our model only considers the energy consumption in the active mode (i.e., sensing, sending, and receiving data). The fourth constraint ensures flow conservation (i.e., for each sensor, the total amount of data sensed and received should be equal to the amount of data transmitted). All data flows will eventually be absorbed by the base station.

The above linear programming formulation is still applicable if sensors have irregular surveillance ranges, different surveillance ranges, and transmission ranges. Specifically, given the shape of the surveillance area, the surveillance range, and the transmission range of each sensor, we can determine $T(i)$, $S(j)$, and $N(i)$ for the sensor and apply the linear programming formulation.

The above linear programming problem involves the variables x_{ij} ($1 \le i \le n$ and $1 \le i \le m$) and f_{ij} ($1 \le i \le n$ and $0 \le j \le n$). It can be solved in polynomial time by any existing linear programming methods to yield the optimal $x_{ij}$and f_{ij} and L. The optimal x_{ij} specifies the total time that sensor s_i should watch target t_j while the optimal f_{ij} specifies the total amount of data transmitted from sensor s_i to sensor s_j or the base station. To realize these optimal x_{ij} and f_{ij} in order to achieve the maximal lifetime, it is necessary to determine a *sensor-target surveillance schedule* (specifying the sensors that should be active and the targets that these sensors should watch) and a *sensor-target surveillance tree* (specifying how the sensed data are routed to the base station) for each session. We will determine the schedules in Section 5.2 and the trees in Section 5.3.

5.2 Determining Sensor-Target Surveillance Schedules

In this section, we determine the sensor-target surveillance schedules to realize the optimal x_{ij} determined in Section 5.1 to achieve the maximal lifetime. The number of sessions, which is equal to the number of surveillance schedules, is also determined. We will prove in Theorem 4 that the number of sessions is upper bounded by n^2mk.

We let $X_{n\times m} = (x_{ij})_{n\times m}$ be a ***workload matrix,*** where x_{ij} is found by solving the linear programming problem in Section 5.1. The workload matrix specifies the total length of time that a sensor should watch a target. We let $P_s(c_s)$ be a ***schedule matrix*** of the sth session, where c_s is the time period of this session. $P_s(c_s)$ represents the schedule of the sth session and it has the following properties:

1. $P_s(c_s)$ is a matrix with n rows and m columns where the element in the ith row and the jth column gives the duration that sensor i is scheduled to watch target j in this session.
2. There are at most k nonzero elements in each row because each sensor can watch at most k targets at a time.
3. There are exactly h nonzero elements in each column because each target should be watched by h sensors.
4. All nonzero elements have the same value c_i (i.e., the duration of the ith session).

To determine the sensor-target surveillance schedules for all sessions, we decompose the workload matrix into a sequence of schedule matrices such that $X_{n\times m} = \sum_{i=1}^{g} P_i(c_i)$, where g is the total number of sessions. We remind that there are n sensors and each sensor can watch at most k targets while there are m targets and each target should be watched by h sensors at any time. Therefore, we have $kn \ge hm$ (otherwise, there is no feasible solution). In the following sections, we first consider the case of $kn = hm$ (i.e., all sensors must be active until the end of the surveillance operation) and then extend the results to the case of $kn > hm$.

5.2.1 The Case of $kn = hm$

To decompose the workload matrix into a sequence of schedule matrices, we transform this problem into the (k, h)-matching problem which has been defined and solved in Section 4. Specifically, we represent the workload matrix $X_{n\times m}$ as a bipartite graph $G(S \cup T, E)$, where one set includes the sensors $S = \{s_1, s_2, \ldots, s_n\}$ and the other set includes the targets $T = \{t_1, t_2, \ldots, t_m\}$. For each nonzero element x_{ij} in $X_{n\times m}$, there is an edge from s_i to t_j and the weight of this edge is x_{ij}. For the schedule matrix of session i, each row has exactly k nonzero elements (i.e., each sensor should watch k targets in this session) and each column has exactly h nonzero elements (i.e., each target should be watched by h sensors in this session). In other words, one sensor matches k distinct targets and one target matches h distinct sensors in the bipartite graph. Thus, the problem of finding a schedule matrix is equivalent to finding the (k, h)-matching in the bipartite graph.

We use the (k, h)-Matching algorithm, presented in Section 4, to determine a (k, h)-matching in the bipartite graph. Once a (k, h)-matching is computed in G, the next step is to determine the corresponding schedule matrix and its surveillance duration. Before proceeding, we need certain features of the workload matrix. Let R_i and C_j be the sum of elements in row i and column j of the workload matrix $X_{n\times m}$, respectively. According to (1) and (2) of the linear programming problem, we have:

$$C_j = hL, \quad j = 1, 2, \ldots, m, \tag{7}$$

$$R_i = kL, \quad i = 1, 2, \ldots, n. \tag{8}$$

In addition, the sum of all rows is equal to the sum of all columns (i.e., $\sum_{i=1}^{n} R_i = \sum_{j=1}^{m} C_j = mhL$). Since $kn = hm$ (i.e., the case considered in this section), we have:

$$\sum_{i=1}^{n} R_i = nkL. \tag{9}$$

Combining (8) and (9), we have:

$$R_i = kL, \quad i = 1, 2, \ldots, n. \tag{10}$$

From (5), (7), and (10), we observe the following three important features:

- **Feature 1.** Every element is not greater than L.
- **Feature 2.** The sum of elements in each column is equal to hL.
- **Feature 3.** The sum of elements in each row is equal to kL.

As discussed at the beginning of Section 5.2, all elements in a schedule matrix are either zero or c_i (i.e., the time duration of session i). Clearly, for any edge (s_i, t_j) in the (k, h)-matching, the corresponding element in row i and column j of the schedule matrix should be nonzero. Elements in remaining positions are zero. c_i should be carefully determined such that the above three features of the workload matrix still hold after deducting the schedule matrix, which guarantees the successful decomposition of the workload matrix. In particular, c_i is as large as possible (for reducing the number of on/off operations) while the resulting schedule and sensor-to-target match are valid. After deducting the schedule matrix from the workload matrix, the sum of elements in each row is updated to $k(L - c_i)$ while the sum of elements in each column is updated to $h(L - c_i)$. So, Features 2 and 3 still hold. To ensure that Feature 1 holds, any element in the resulting workload matrix should not be greater than $L - c_i$. Let a_i denote the smallest weight of edges in the (k, h)-matching found in session i, and b_i denote the largest value of the remaining elements in the workload matrix (these elements are not in the positions which correspond to the matching). We set c_i to $min(a_i, L - b_i)$ which ensures that any element in the resulting workload matrix is not greater than $L - c_i$ and thus Feature 1 holds. The algorithm for determining the schedule matrix is given below.

Following the example in Fig. 1 where $n = 3$, $m = 6$, $k = 4$, and $h = 2$, after solving the linear programming problem, we get the following workload matrix ($L = 1$):

$$\begin{bmatrix} 1 & 0.5 & 1 & 0.5 & 0 & 1 \\ 0 & 1 & 0.5 & 1 & 1 & 0.5 \\ 1 & 0 & 0 & 0.5 & 1 & 0.5 \end{bmatrix}.$$

The above workload matrix, which has 14 nonzero entries, is represented as a bipartite graph $G(S \cup T, E)$ with 14 edges as follows: S is a set of sensors $\{s_1, s_2, s_3\}$, T is a set of targets $\{t_1, t_2, t_3, t_4, t_5, t_6\}$, and there is an edge from sensor s_i to target t_j for each nonzero element x_{ij} in the workload matrix and hence E contains 14 edges (see Fig. 1a). The (k, h)-Matching Algorithm is executed to compute a $(4, 2)$-matching which has 12 edges (these 12 edges are shown in bold in Fig. 1c). We can see that the smallest weight of edges in the matching is 0.5 and the largest weight of edges not in the matching is 0 (i.e., $a_i = 0.5$ and $b_i = 0$). Since $L - b_i = 1$, we set $c_i = a_i = 0.5$. The computed (k, h)-matching defines the following schedule matrix with 12 nonzero entries for this session:

$$\begin{bmatrix} 0.5 & 0.5 & 0.5 & 0 & 0 & 0.5 \\ 0 & 0.5 & 0.5 & 0.5 & 0.5 & 0 \\ 0.5 & 0 & 0 & 0.5 & 0.5 & 0.5 \end{bmatrix}.$$

Schedule-Matrix Algorithm
Input: a workload matrix $X_{n\times m}$, k, h, a (k, h)-matching Ψ determined by the (k, h)-Matching Algorithm.
Output: a schedule matrix $P_i(c_i)$.

```
1: Begin
2:     construct a bipartite graph G from X_{n×m};
3:     a_i = the smallest weight of edges in Ψ;
4:     b_i = the largest weight of edges in G and not in Ψ;
5:     if a_i ≤ L−b_i    c_i = a_i;
6:     else              c_i = L−b_i;
7:     construct P_i(c_i) where elements in the corresponding
       positions of Ψ are c_i and elements in remaining
       positions are 0;
8:     output P_i(c_i);
9: End
```

The workload matrix can be decomposed as follows: we first transform the workload matrix into a bipartite graph. We use the (k, h)-Matching Algorithm to compute a (k, h)-matching in the bipartite graph. We use the Schedule-Matrix Algorithm to determine its corresponding schedule matrix in this session. Duration of the session is deducted from the weight of the kn edges of the (k, h)-matching. For the edges whose weights become zero, they are removed from the bipartite graph. We continue to compute a (k, h)-matching in the updated bipartite graph. This operation is repeated until there is no (k, h)-matching in the bipartite graph.

The following theorem states that there exists a (k, h)-matching in every round of decomposition if the three features of the workload matrix hold for $h = 1$. In the last round of decomposition, all remaining edges form exactly a (k, h)-matching and are completely removed from the bipartite graph.

Theorem 3. *If the three features of a workload matrix hold and $h = 1$, the workload matrix can be expressed as $P_1(c_1) + P_2(c_2) + \cdots + P_g(c_g)$, where each schedule matrix $P_i(c_i)(i = 1, 2, \ldots, g)$ corresponds to a (k, 1)-matching.*

The Proof of Theorem 3 can be found in Section 3 in the supplemental material, which can be found on the Computer Society Digital Library.

Since k and h are symmetric, Theorem 3 also holds for $k = 1$ and $h \geq 1$. We extend Theorem 3 to the general case of k, $h \geq 1$ and propose the following conjecture.

Conjecture. If the three features of a workload matrix hold, the workload matrix can be expressed as $P_1(c_1) + P_2(c_2) + \cdots + P_g(c_g)$, where every schedule matrix $P_i(c_i)(i = 1, 2, \ldots, g)$ corresponds to a (k, h)-matching.

Verification. We have proved a special case ($k = h = 1$) of the conjecture in Theorem 3. We have also verified the general case of the conjecture through over 6,000,000 randomly generated problem instances. Details of the verification can be found in Section 4 in the supplemental material, which can be found on the Computer Society Digital Library. Readers can download our simulation program from [18] to repeat our experiments or conduct further experiments for verification.

Theorem 4. *The total number of sessions g is upper bounded by n^2mk. If $h = 1$, g is upper bounded by nm.*

The Proof of Theorem 4 can be found in Section 3 in the supplemental material, which can be found on the Computer Society Digital Library.

Similarly, g is also upper bounded by nm for $k = 1$ and $h \geq 1$ due to symmetry of k and h.

5.2.2 The Case of $kn > hm$

In this section, we determine the schedule matrices for the case of $kn > hm$. Our main ideas are as follows:

- We first transform the case of $kn > hm$ to the case of $kn = hm$ by adding some dummy sensors and dummy targets into the network. The effect is to add some dummy rows and columns to the workload matrix $X_{n\times m}$, such that the sum of elements in each column is equal to hL and the sum of elements in each row is equal to kL. Note that the dummy sensors and dummy targets are added only for the purpose of computing. Sensors would not consume any energy for watching dummy targets, and dummy sensors would only watch dummy targets. Therefore, after adding dummy sensors and dummy targets, both the maximal lifetime and the sensor-target surveillance schedules remain unchanged.
- We apply the algorithm in Section 5.2.1 to determine the sensor-target surveillance schedules for the above network with dummy sensors and dummy targets.
- We remove the dummy items from the above schedules, so that we get the schedules for the network with only real sensors and real targets. In other words, we get the schedules for the case of $kn > hm$.

Let $W_{p\times q}$ (where $kp = hq$) denote the resulting workload matrix after adding dummy sensors and dummy targets, and $Z_{p\times(q-m)}$ denote the dummy matrix which is appended to the right hand side of $X_{n\times m}$. $W_{p\times q}$ has the following form:

$$W_{p\times q} = \begin{bmatrix} x_{11}x_{12}\ldots x_{1m}\ldots & z_{11}z_{12}\ldots z_{1(q-m)} \\ x_{21}x_{22}\ldots x_{2m} & z_{21}z_{22}\ldots z_{2(q-m)} \\ \ldots\ldots & \ldots\ldots \\ x_{n1}x_{n2}\ldots x_{nm} & z_{n1}z_{n2}\ldots z_{n(q-m)} \\ 00\quad\ldots\quad 0 & \ldots\ldots \\ \ldots\ldots & \ldots\ldots \\ 00\quad\ldots\quad 0 & z_{p1}z_{p2}\ldots z_{p(q-m)} \end{bmatrix}_{p\times q}. \quad (11)$$

In the bottom-left part of $W_{p\times q}$, elements are equal to 0 because dummy sensors only watch dummy targets. Note that each sensor should watch exactly k targets because $kp = hq$. The number of columns in the dummy matrix should not be less than k, i.e., $q - m \geq k$ (otherwise, no feasible schedule can be found). Thus, p is the smallest positive integer which satisfies $p = \min\{p \in I^+ | p \geq n, (kp) \bmod h = 0, (kp)/h - m \geq 0\}$ and $q = (kp)/h$.

To ensure that the workload matrix $W_{p\times q}$ satisfies the three features (i.e., the sum of elements in each column is equal to hL, the sum of elements in each row is equal to kL, and each element is not greater than L), the dummy matrix $Z_{p\times(q-m)}$should satisfy the following conditions:

$$\sum_{j=1}^{q-m} z_{ij} = kL - R_i \quad \text{for all } i = 1, 2, \ldots, p, \quad (12)$$

$$\sum_{i=1}^{n} z_{ij} = hL \quad \text{for all } j = 1, 2, \ldots, q-m, \quad (13)$$

$$z_{ij} \leq L \text{for all } i = 1, 2, \ldots, p \text{ and } j = 1, 2, \ldots, q-m. \quad (14)$$

In (12), R_i is equal to 0 for $i = n+1, \ldots, p$. Let R_i^- record the sum of the remaining undetermined elements in row i of $Z_{p\times(q-m)}$. Initially, we have $R_i^- = kL - R_i$. We propose a simple algorithm to compute the dummy matrix which satisfies the above conditions. This algorithm assigns values to elements of $Z_{p\times(q-m)}$in bottom-up order, and uniformly assigns values to elements in each row without violating (12), (13), and (14). Specifically, it starts with the last row and assigns the same value $R_p^-/(\mathrm{q}-\mathrm{m})$ to all elements in this row. Once it has processed row p, it processes row $p-1$ and assigns values to the elements in this row in a similar way. It repeats this step until it has assigned values to all elements. The algorithm for computing the dummy matrix is given below.

Fill-Matrix Algorithm
Input: a workload matrix $X_{n\times m}$.
Output: a filled matrix $W_{p\times q}$.

```
1: Begin
2:     p=min{p ∈ I+ | p ≥ n, (kp) mod h = 0, (kp)/h −m ≥ k};
3:     q=(kp)/h;
4:     for i=1:p
5:         R_i^- = kL − R_i ;
6:     for i=p:1
7:         for j=1:q−m
8:             z_ij= R_i^- /(q−m);
9: End
```

Lemma 1. *p and q are upper bounded by $(n + n/h + 2)h$ and $(n + n/h + 2)k$, respectively.*

Theorem 5. *For any given workload matrix $X_{n\times m}$, the Fill-Matrix Algorithm can compute the dummy matrix such that the three features hold in the resulting matrix $W_{p\times q}$. The time complexity of the Fill-Matrix Algorithm is $O(n^2hk)$.*

Proofs of Lemma 1 and Theorem 5 can be found in Section 3 of Supplmental Material, which can be found on the Computer Society Digital Library.

The algorithm for decomposing the workload matrix is given below.

Theorem 6. *The time complexity of the Decompose-Matrix Algorithm is $O(n^5h^3k3(h + k))$.*

The Proof of Theorem 6 can be found in Section 3 of Supplmental Material, which can be found on the Computer Society Digital Library.

In the above time complexity analysis, m has been implicitly considered because m is bounded by nk/h.

Using the proposed algorithms, we can fill any given workload matrix $X_{n\times m}$ to form $W_{p\times q}$ and decompose $W_{p\times q}$ into a sequence of schedule matrices such that:

$$W_{p\times q} = P_1(c_1) + P_2(c_2) + \cdots + P_g(c_g). \quad (15)$$

Let $P_i'(c_i)$ denote the matrix containing the first n rows and the first m columns of $P_i(c_i)$ (i.e., it does not contain the dummy rows and columns). We remove all the dummy rows and dummy columns from all the matrices in (15) and get:

$$X_{n\times m} = P_1'(c_1) + P_2'(c_2) + \cdots + P_g'(c_g). \quad (16)$$

In the above manner, a workload matrix can be decomposed into a sequence of schedule matrices such that we can realize the optimal $x_{ij}(1 \leq i \leq n, 1 \leq j \leq m)$ determined in Section 5.1.

Decompose-Matrix Algorithm
Input: a workload matrix $X_{n\times m}$, k, h.
Output: a sequence of schedule matrices $P_i(c_i)$.

```
1: Begin
2:     if kn > hm
3:         compute W_{p×q} by Fill-Matrix Algorithm;
4:     construct a bipartite graph G from W_{p×q};
5:     while there exist edges in G do
6:     {
7:         compute a (k, h)-matching by (k, h)-Matching
           Algorithm;
8:         compute schedule matrix P_i(c_i) by Schedule-
           Matrix Algorithm;
9:         deduct P_i from W_{p×q} and remove the edges
           whose weight becomes zero;
10:    }
11:    output W_{p×q} =P_1(c_1)+P_2(c_2)+…+P_g(c_g);
12: End
```

5.3 Determining Sensor-Target Surveillance Trees

In this section, we determine a sensor-target surveillance tree for each session to achieve the maximal lifetime. These surveillance trees are determined based on the optimal data flows determined in Section 5.1 and the surveillance schedules determined in Section 5.2. Each surveillance tree has the following properties: 1) its root is the base station, 2) its leaf nodes are the active sensors of this session, and 3) its intermediary nodes are the sensors which forward data for others (hence, the intermediary nodes are active nodes). Sensor s_i $(1 \leq i \leq n)$ follows the optimal data flow f_{ij} (determined in Section 5.1) in order to achieve the maximal lifetime L. Suppose sensor s_i has l downstream nodes, denoted by $s_{d_1}, s_{d_2}, \ldots, s_{d_l}$ (i.e., $f_{id_1}, f_{id_2}, \ldots, f_{id_l}$ are nonzero). Since there is no ordering of data flows $f_{id_1}, f_{id_2}, \ldots, f_{id_l}$, we let sensor s_i pass its outgoing data to s_{d_1} until reaching the flow value f_{id_1}, then to s_{d_2} until reaching the flow value $f_{id_2}, \ldots$, and finally to s_{d_l} until reaching the flow value f_{id_l}. The outgoing data from sensor s_i include its own sensed data as well as the data of other sensors that it forwards to the base station (see the left hand side of (3)). In this manner, the optimal surveillance trees that realize the maximal lifetime L can be determined.

5.4 Overall Solution

The overall solution is shown as a flowchart in Fig. 2.

Theorem 7. *The proposed solution to the MLSTS problem has a time complexity of* $O(\max\{(nm)^{3.5}, n^5h^3k^3(h+k)\})$.

Theorem 8. *The proposed solution to the MLSTS problem has a space complexity of* $O(n^5h^3k^3)$.

Proofs of Theorem 7 and 8 can be found in Section 3 of Supplmental Material, which can be found on the Computer Society Digital Library.

6 Example and Simulations

We present a numerical example in Section 5 of the supplmental material, which can be found on the Computer Society Digital Library to illustrate all the steps involved and conduct extensive simulations in Section 6 of the supplemental material, which can be found on the Computer Society Digital Library to evaluate our solution.

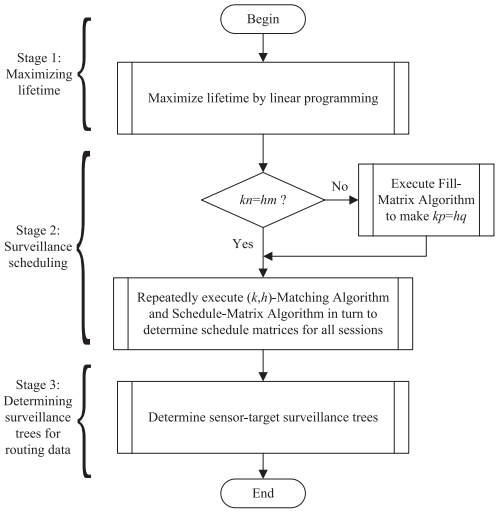


Fig. 2. Steps of the proposed solution to the general MLSTS problem.

Simulation results show that the proposed algorithm can achieve significantly longer lifetime than a greedy algorithm and thus it is worthy to design and implement the sophisticated algorithm for maximizing the lifetime.

7 Conclusions and Future Work

We formulated a general maximal lifetime sensor-target surveillance problem, in which each sensor can watch at most k targets ($k \geq 1$) and each target should be watched by h sensors ($h \geq 1$) at any time. We proposed a new solution to solve MLSTS. In particular, we transformed the scheduling problem involved in MLSTS into a new and general (k,h)-*matching* Problem. We designed an efficient (k,h)-*matching* algorithm for this matching problem. As a byproduct of this study, the (k,h)-*matching* problem and the proposed matching algorithm can potentially be applied to other problems in computer science and operations research.

The present work has two possible extensions. First, the sensors in some applications may have directional sensing ranges (e.g., ultrasonic sensors) and this characteristic gives rise to a new maximal lifetime surveillance problem. Second, the proposed conjecture is an open problem which has mathematical significance in matrix computation and decomposition.

Acknowledgments

This work is supported in part by grant from Research Grants of Hong Kong [Project No. HKBU211009], and NSF China Grants No. 60633020 and No. 60970117. Peng-Jun Wan is supported in part by the US National Science Foundation (NSF) under grants CNS-0831831 and CNS-0916666.

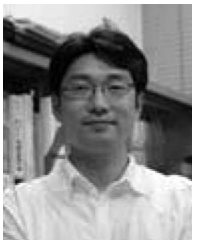
References

[1] A.E. Abdallah, T. Fevens, J. Opatrny, and I. Stojmenovic, "Power-Aware Semi-Beaconless 3D Georouting Algorithm Using Adjustable Transmission Ranges for Wireless Ad Hoc and Sensor Networks," *Ad Hoc Networks*, vol. 8, no. 1, pp. 15-29, 2010.

[2] J. Blumenthal, F. Reichenbach, and D. Timmermann, "Decreasing the Localization Error in Border Areas of Sensor Networks," *Proc. Int'l Conf. Distributed Computing in Sensor Systems (DCOSS '08)*, pp. 47-48, 2008.

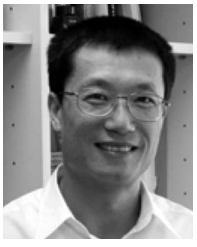
[3] M. Cardei and D.-Z. Du, "Improving Wireless Sensor Network Lifetime through Power Aware Organization," *J. Wireless Networks,* vol. 11, pp. 333-340, 2005.
[4] M. Cardei, M. Thai, Y. Li, and W. Wu, "Energy-Efficient Target Coverage in Wireless Sensor Networks," *Proc. IEEE INFOCOM,* 2005.
[5] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms,* third ed. MIT Press, 2009.
[6] H. Frey, S. Ruehrup, and I. Stojmenovic, "Routing in Wireless Sensor Networks," *Guide to Wireless Ad Hoc Networks,* S. Misra, I. Woungag, and S. Misra, eds., ch. 4, pp. 81-111, Springer-Verlag, 2009.
[7] Infra-Red Sensors, http://www.ikalogic.com/ir_prox_sensors.php, 2011.
[8] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *Proc. MObiCOM,* 2000.
[9] A.M. Kermarrec, G. Tan, "Greedy Geographic Routing in Large-Scale Sensor Networks: A Minimum Network Decomposition Approach," *Proc. 11th ACM Int'l Symp. Mobile Ad Hoc Networking and Computing (MobiHoc '10),* pp. 161-170, 2010.
[10] S. Lindsey, C. Raghavendra, and K.M. Sivalingam, "Data Gathering Algorithms in Sensor Networks Using Energy Metrics," *IEEE Trans. Parallel and Distributed Systems,* vol. 13, no. 9, pp. 924-935, Sept. 2002.
[11] H. Liu, X. Jia, P.-J. Wan, X. Liu, and F. Yao, "A Distributed and Efficient Flooding Scheme Using 1-hop Information in Mobile Ad Hoc Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 18, no. 5, pp. 658-671, May 2007.
[12] H. Liu, X. Jia, P.-J. Wan, C.-W. Yi, S. Makki, and N. Pissinou, "Maximizing Lifetime of Sensor Surveillance Systems," *IEEE/ACM Trans. Networking,* vol. 15, no. 2, pp. 334-345, Apr. 2007.
[13] H. Liu, P.-J. Wan, and X. Jia, "Maximal Lifetime Scheduling for Sensor Surveillance Systems with K Sensors to 1 Target," *IEEE Trans. Parallel and Distributed Systems,* vol. 17, no. 12, pp. 1526-1536, Dec. 2006.
[14] H.B. Mitchell, *Multi-Sensor Data Fusion: An Introduction.* Springer-Verlag, 2007.
[15] V. Rajendran, K. Obraczka, and J.J. Garcia-Luna-Aceves, "Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks," *Proc. First Int'l Conf. Embedded Networked Sensor Systems (SenSys '03),* pp. 181-192, 2003.
[16] A. Savvides, C.C. Han, and M. Srivastava, "Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors," *Proc. MObiCOM,* pp. 166-179, 2001.
[17] O. Simeone, U. Spagnolini, Y. Bar-Ness, and S.H. Strogatz, "Distributed Synchronization in Wireless Networks," *IEEE Signal Processing Magazine,* vol. 25, no. 5, pp. 81-97, Sept. 2008.
[18] Simulation Program, http://www.comp.hkbu.edu.hk/~hliu/code/Sensor-Target-Surveillance/Surveillance.zip, 2011.
[19] T. Yan, T. He, and J.A. Stankovic, "Differentiated Surveillance for Sensor Networks," *Proc. First Int'l Conf. Embedded Networked Sensor Systems (SenSys '03),* pp. 51-62, 2003.

Hai Liu received the BSc and MSc degrees in applied mathematics from South China University of Technology, in 1999 and 2002, respectively. He received the PhD degree in computer science from City University of Hong Kong in 2006. Currently, he is a research assistant professor in the Department of Computer Science, Hong Kong Baptist University. His research interests include wireless networking, mobile computing, and algorithm design and analysis. He is a member of the IEEE.



Xiaowen Chu received the BEng degree in computer science from Tsinghua University, P.R. China, in 1999, and the PhD degree in computer science from the Hong Kong University of Science and Technology in 2003. Currently, he is an assistant professor in the Department of Computer Science, Hong Kong Baptist University. His research interests include distributed and parallel computing and wireless networks. He is a member of the IEEE and the IEEE Computer Society.

Yiu-Wing Leung received the BSc and PhD degrees from the Chinese University of Hong Kong, in 1989 and 1992, respectively. His PhD advisor was Prof. Peter T.S. Yum. Currently, he is a professor in the Department of Computer Science of the Hong Kong Baptist University, Hong Kong. He has been working on two main research areas: 1) networking and multimedia, and 2) sybernetics and system engineering. He has published more than 70 journal papers in these areas. He is a senior member of the IEEE.



Xiaohua Jia received the BSc degree in 1984 and the MEng degree in 1987 from the University of Science and Technology of China, and the DSc degree in information science from the University of Tokyo in 1991. Currently, he is a chair professor in the Department of Computer Science at City University of Hong Kong. His research interests include distributed systems, computer networks, wireless sensor networks, and mobile wireless networks. He is an editor of *IEEE Transaction on Parallel and Distributed Systems* (2006-2009), *Wireless Networks*, *Journal of World Wide Web, Journal of Combinatorial Optimization*, etc. He is the general chair of ACM MobiHoc 2008, TPC cochair of IEEE MASS 2009, area chair of IEEE INFOCOM 2010, TPC cochair of IEEE GlobeCom 2010—Ad Hoc and Sensor Networking Symposium, and panel cochair of IEEE INFOCOM 2011. He is a senior member of the IEEE.

Peng-Jun Wan received the BS degree from Tsinghua University, the MS degree from the Chinese Academy of Science, and the PhD degree from the University of Minnesota. Currently, he is a professor of computer science in the Department of Computer Science, Illinois Institute of Technology, Chicago. His research interests include wireless networks, and algorithm design and analysis. He is a member of the IEEE.