國立交通大學

資訊工程學系

碩 士 論 文

無線感測網路應用於智慧型監視系統及其事件偵測延遲分析

iMouse: An Intelligent Mobile Surveillance Security System by Wireless Sensors and its Detection Delay Analysis

研 究 生：鄭凱陽

指導教授：曾煜棋 教授

中 華 民 國 九 十 四 年 六 月

無線感測網路應用於智慧型監視系統及其事件偵測延遲分析

iMouse: An Intelligent Mobile Surveillance Security System by Wireless Sensors and its Detection Delay Analysis

研 究 生： 鄭凱陽　　Student : Kai-Yang Cheng

指導教授： 曾煜棋　教授　　Advisor : Yu-Chee Tseng

國立交通大學
資訊工程學系
碩士論文

A Thesis
Submitted to Department of Computer and Information Science
College of Electrical Engineering and Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Computer and Information Science

June 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年六月

iMouse: An Intelligent Mobile Surveillance Security System by Wireless Sensors and its Detection Delay Analysis

Student: Kai-Yang Cheng

Advisor: Prof. Yu-Chee Tseng

A Dissertation Submitted to

the Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Chiao-Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science and Information Engineering

Hsinchu, Taiwan

June 2005

無線感測網路應用於智慧型監視系統及其事件偵測延遲分析

學生：鄭凱陽　　　　　　　指導教授：曾煜棋 老師

國立交通大學資訊工程學系 (研究所) 碩士班

摘要

無線感測網路提供一個經濟又方便的方法來進行環境偵測，而將可提供豐富內容的無線感測網路結合至監視系統是一個新的研究方向。在這邊，我們提出了整合無線感測器行動監視系統，該系統包含了大量較便宜但不具行動能力的定點感測器及少量較昂貴但具行動能力的行動感測器。定點感測器是用來進行環境偵測，行動感測器則是用來移動至特定目標點並且採取近一步措施，結合起來就是具移動能力及豐富資訊的監視系統。在這邊採用家庭安全應用來映證我們提出來的系統。在這邊我們也發現影響整個系統效能最大的因素是事件偵測延遲，我們將這個問題制定成任意感測模型或是更精密的多重感測模型。我們將在這兩個模型下分析事件偵測延遲，並且利用幾個模擬結果來映證我們的分析。這個分析也同樣適用於其他無線感測網路應用。

關鍵字: 無線感測網路，監視系統，定點感測器，行動感測器，事件偵測延遲，任意感測模型，多重感測模型。

iMouse: An Intelligent Mobile Surveillance Security System by Wireless Sensors and its Detection Delay Analysis

Student: Kai-Yang Cheng Advisors: Prof. Yu-Chee Tseng

Institute of Computer Science and Information Engineering
National Chiao-Tung University

ABSTRACT

Wireless sensor networks (WSN) provide an inexpensive and convenient way to monitor physical environments. Integrating the context-aware capability of WSN into surveillance systems is an attractive direction. We thus propose an integrated mobile surveillance and wireless sensor (*iMouse*) system, which consists of a large number of inexpensive static sensors and a small number of more expensive mobile sensors. The former is to monitor the environment, while the latter is capable of moving to certain target locations and taking more advanced actions. The iMouse system is a mobile, context-aware surveillance system. We demonstrate our current prototyping for home security applications. One important performance metric of the system is the *event detection latency*. We analyze the latency under an *any-sensor-detection* and a k*-sensor-detection* models, where $k > 1$. The analytical results are also verified by simulations.

Keywords: pervasive computing, robotics, surveillance system, wireless communication, wireless sensor network.

Acknowledgments

My advisor, prof. Yu-Chee Tseng, is the first one I would like to express my gratitude to. With the wonderful research conditions he provided and his attentive instructions, I came to discover the pleasure of research. Then, I would like to thank all HSCC members for their generous advices. Discussing with them benefited me in many ways.

Kai-Yang Cheng at CSIE, NCTU.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recent advances in wireless communications and MEMS technologies have made wireless sensor networks (WSN) possible. A WSN consists of many tiny, low-power devices equipped with sensors, transceivers, and actuators [1]. It provides an inexpensive and convenient way to monitor physical environments. With its context awareness, WSN may enrich human life in many ways. Applications of WSN include surveillance, biological detection, habitat, agriculture, and traffic monitoring [2, 3, 4, 5, 6].

Integrating the context-aware capability of WSN into surveillance system is an attractive direction that deserves investigation. Surveillance systems typically collect a large volume of audio/video information, which requires intensive computation/manpower to analyze. Including the intelligence of WSN can help reduce such overheads and even provide more advanced, context-rich services. For example, in security applications, when something abnormal is detected, in-depth analyses may be conducted to find out the possible sources. In intrusion detection applications, when trespassing is detected, a metal detector may help determine whether the intruder is carrying a weapon or not.

In this work, we propose an integrated mobile surveillance and wireless sensor (iMouse) system. The iMouse system consists of a large number of inexpensive static wireless sensors and a small number of more expensive mobile sensors. The former is to monitor the environment, while the latter is capable of moving to certain target locations (such as potential emergency sites) and taking more advanced actions (such as taking pictures of the emergency scenes and conducting in-depth analyses). The iMouse system is a mobile, context-aware surveillance system. We demonstrate our current prototyping for home security applications. In particular, each mobile sensor has a mini-computer, which is connected to a data collector, a WebCam, and a 802.11 WLAN card, and is mobilized by a Lego car. At normal times, a mobile sensor can collect sensory data and report to an external server. When special events are detected, it can move to the event locations, take snapshots of the scenes, and send pictures to the server through its 802.11 interface. We demonstrate applications of iMouse through a fire emergency example.

In iMouse, when an event or an object appears, the *event detection latency* is an important factor that can affect the responsiveness of the system. This depends on the network deployment and the location where the event appears. We adopt a probabilistic approach to model this problem and analyze the latency under an *any-sensor-detection* and a k-*sensor-detection* models, where k is a predefined integer. Simulation results are presented to verify our analyses. Also, the result is believed to be applicable to general sensor networks.

The rest of this work is organized as follows. Chapter 2 reviews some related work. Chapter 3 discusses detailed design and implementation of our iMouse system. Chapter 4 presents our analyses and simulation results on event detection latency. Conclusions are drawn in Chapter 5.

Chapter 2

Related Work

The objective of this work is to study the feasibility of combining surveillance systems with sensor networks. Most visual surveillance systems deal with the real-time monitoring of persistent and transient objects. The primary goals of these systems are to provide an automatic interpretation of scenes and to understand/predict actions of the observed objects from the information acquired from cameras or CCTV (closed circuit television) [7]. For example, a video-based surveillance network is proposed in [8], where the information collected by each video camera is transmitted by an IEEE 802.11 wireless card. The surveillance issue has also been discussed in the field of robotics [9, 10, 11]. The system is assumed to have a robot with many static cameras installed on locations such as walls. These cameras are used to find obstacles or humans in the field, so that the robot can detour around these obstacles.

On the side of WSN, the event/object tracking issue has been intensively studied [12, 13, 14, 15, 16, 17]. Most works assume that the intrusion objects can emit some signals (such as noise or light), or the objects themselves are phenomenal (such as diffused gas or chemical liquid [16]). However, results reported from a WSN are typically very brief and lack of more in-depth information. This motivates us to study the feasibility of integrating WSN with surveillance systems to support intelligent context-aware surveillance services.

Chapter 3

Design of the iMouse System

3.1 System Architecture

Fig. 3.1 shows the system architecture of the iMouse system. It consists of a static WSN and few mobile sensors. At normal times, the WSN is responsible of collecting and reporting environment information to the mobile sensors. When necessary, the mobile sensors are able to move to the event locations, conduct more advanced analyses of the event scenes, and report the analysis results to the remote sink.

Each sensor of the WSN consists of a data collector and a sensing board. The data collector is used to communicate with other static/mobile sensors. The sensing board is used to collect environment data. In our current prototype, three types of data can be collected, including voice, temperature, and light. Reporting of events is reactive, and an event is defined when the sensory input is higher than a predefined threshold. Different inputs can be combined to define an event. For example, for fire emergency, a combination of light and temperature thresholds can be used. To detect an explosion, a combination of temperature and sound thresholds may be used. For home security, an unusual sound or temperature can be used. More advanced sensors may be added later.

Mobile sensors have five major functionalities: issuing commands to the WSN, gathering data from the WSN, moving to some target areas, taking snapshots, and reporting analyses to the remote sink. Each mobile sensor is empowered by a microprocessor called Stargate [18], which is connected to a data collector, a Lego car, a WebCam, an IEEE 802.11 WLAN card, as shown in Fig. 3.2. The data collector can communicate with static sensors to issue commands or gather data. The Lego car [19], produced by MindStorms, supports mobility. The WebCam is to take photos of the emergency scenes. To eliminate

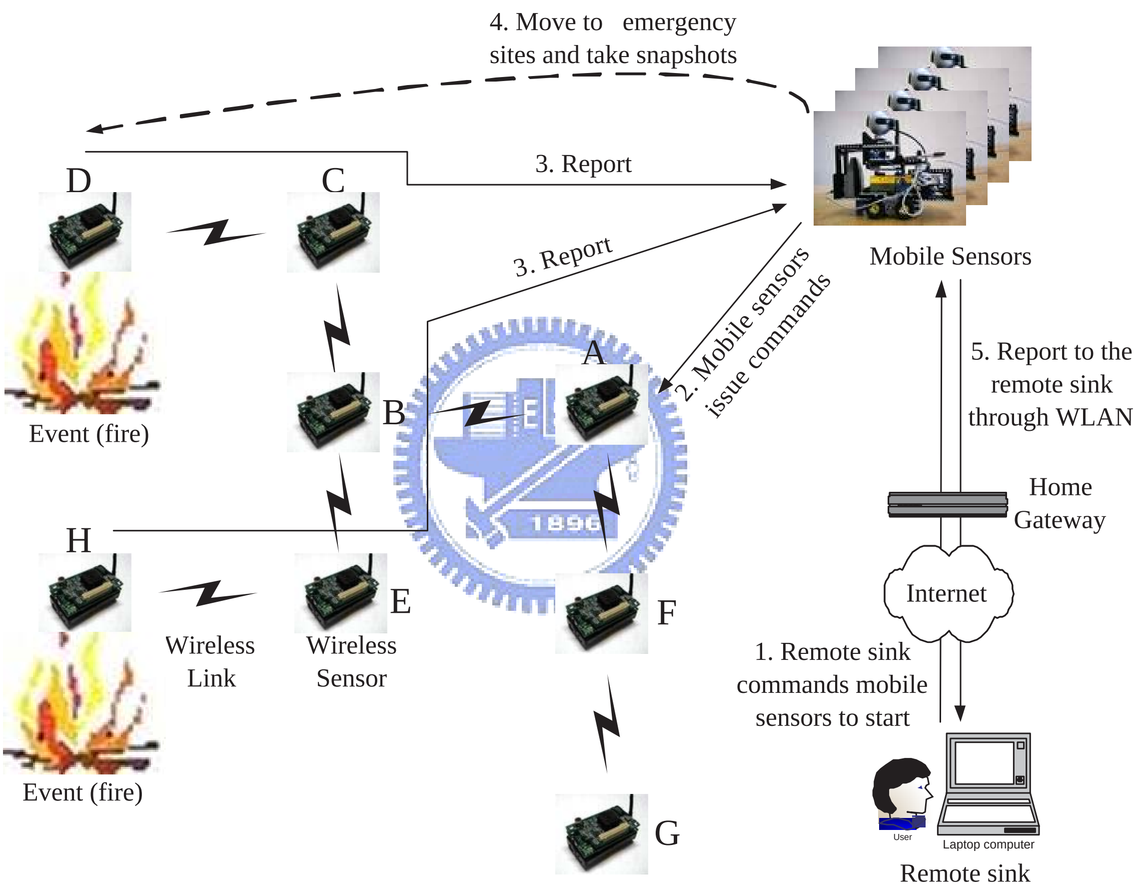


Figure 3.1: System architecture of the iMouse system.

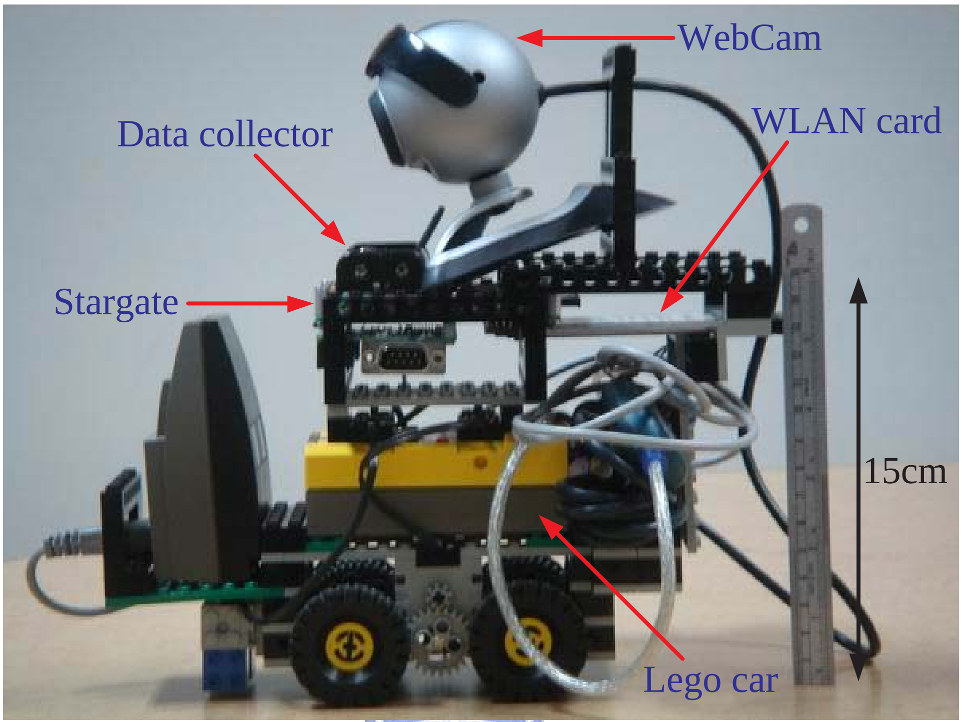
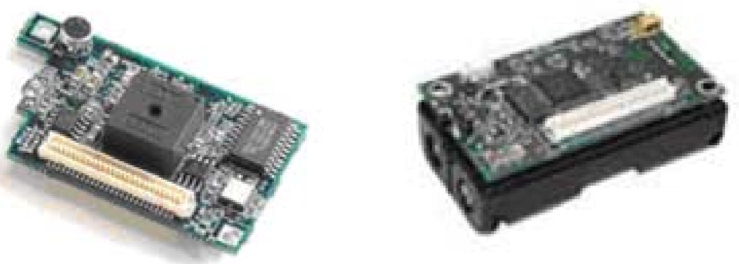


Figure 3.2: The mobile sensor.

the wiring problem, the 802.11 WLAN card is used to talk to the home gateway. The Stargate is the “brain” of a mobile sensor. For example, it decides the visiting sequence of potential emergency sites.

3.2 A Fire Emergency Scenario

Below, we give a fire emergency scenario to demonstrate how the iMouse system works (refer to Fig. 3.1). On receiving the remote sink’s command, the WSN will form a spanning tree to collect environment data. Suppose that sensors D and H reply high temperatures and are thus suspected of fire emergency in their neighborhood. On receiving such notifications, the mobile sensors will coordinate and decide who will be delegated to which sensors via which shortest path. On visiting D and H, the mobile sensor(s) will take snapshots of these sites from different angles. After returning to the home gateway, the mobile sensor(s) will send these snapshots back to the remote sink for further actions.



(a) sensor board (b) data collector

Figure 3.3: The sensor.

Figure 3.4: The Stargate.

3.3 Implementation Details

3.3.1 Hardware Specifications

We use the MOTE-KIT2400-MICAz by CrossBow [20] as sensor nodes (refer to Fig. 3.3). The MICAz is a 2.4 GHz, IEEE 802.15.4-compliant Mote module enabling low-power operations and offering a data rate of 250 kbps with a DSSS radio. The sensing board can offer three kinds of readings: temperature, voice, and light. The Stargate [18], also manufactured by Crossbow, consists of a 32-bit, 400-MHz Intel PXA-255 XScale RISC processor with 64 MB main memory and 32 MB extended flash memory. It also has a daughter board with a RS-232 serial port, a PCMCIA port, a USB port, and a 51-pin extention connector, which can be attached to a MICAz mote. (refer to Fig. 3.4 and

Table 3.1). It drives a WebCam through the USB port, and an 802.11 WLAN card through the PCMCIA slot. The Stargate controls the Lego car [19] via a USB port connected to a Lego tower (9713 IR-TRANSMITTER). The Lego car has an infrared ray receiver in the front and two motors on the bottom (refer to Fig. 3.5). It also has a light sensor, which we use for navigation purpose. This is realized by different colors of the tapes that we stick on the ground. With this mechanism, the Lego car can localize itself in the sensing field.

Table 3.1: The hardware specification of Stagate.

Element	Type
CPU	32-bit, 400 MHz Intel PXA-255 XScale RISC processor
Flash memory	32MB of Intel StrataFlash
Main memory	64MB of SDRAM
Daughter Card	Host USB RS-232 Serial Port via DB-9 Connector
Others	1 Type II CompactFlash Slot 1 PCMCIA slot MICA2 Mote capacity, GPIO/SGPP and other signals via a 51-pin extension connector

An experimental 2×2 grid-like sensing field (Fig. 3.6) is demonstrated. On the ground, golden tapes represent intersections and black tapes represent roads. The origin $(0, 0)$ is at the lower left corner. Four sensors are placed at $(1, 1)$, $(0, 2)$, $(2, 0)$, and $(2, 2)$, respectively. The transmission range is manually set to two units to fit into the relatively small sensing field. A light reading below 800 is to simulate a potential fire emergency.

3.3.2 Protocol Specifications

Each static sensor runs the algorithm in Fig. 3.7. Initially, it waits for commands from mobile sensors or other static sensors. A *tree-construct* command will trigger the sensor to check if its tree parent is null or has expired. If so, it sets the sender as its parent. Then it will re-broadcast the command. To distinguish new from old commands, each *tree-construct* command is assigned a unique sequence number. The goal is to form a spanning tree of the network. Then the sensor turns on its sensing devices. On detecting potential emergencies, sensors will transmit a *status_report* to mobile sensor along the

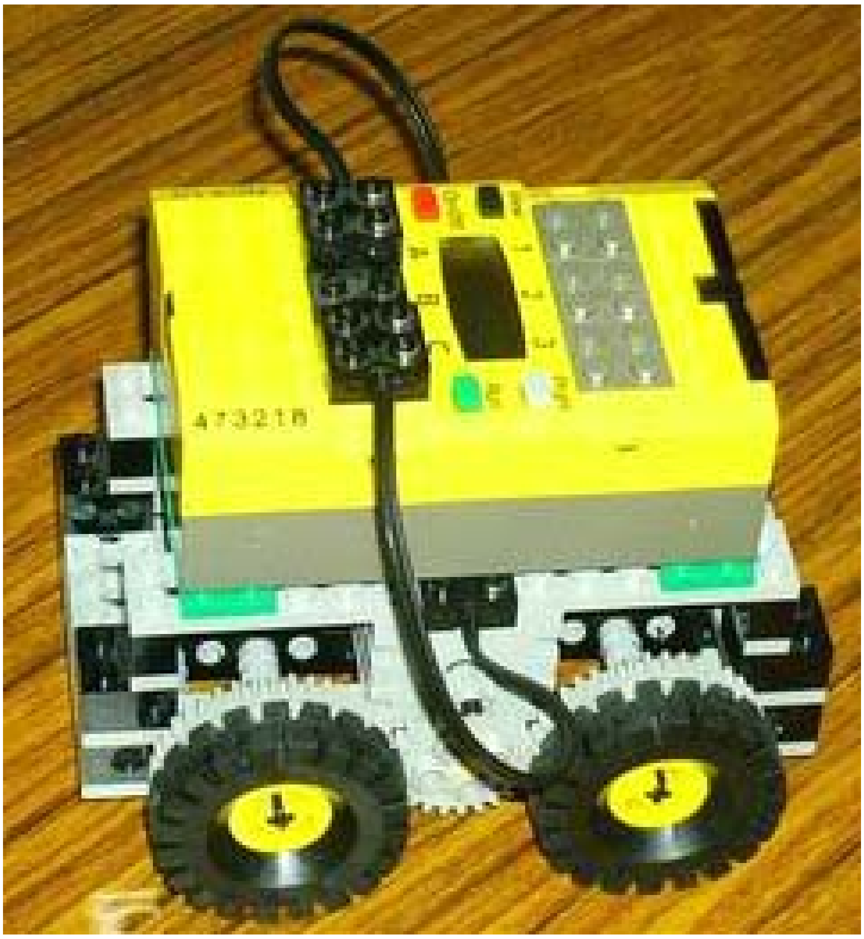


Figure 3.5: The Lego car.

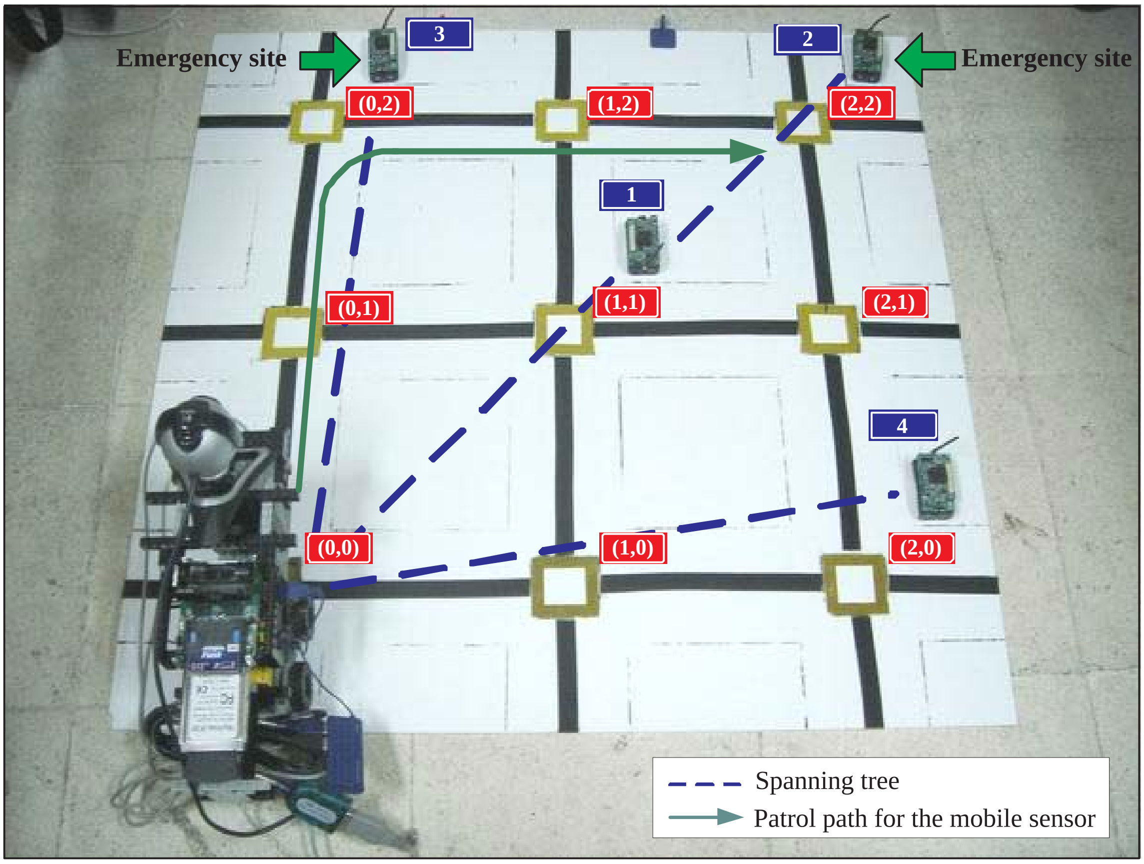


Figure 3.6: A 2×2 grid-like sensing field in our experiment.

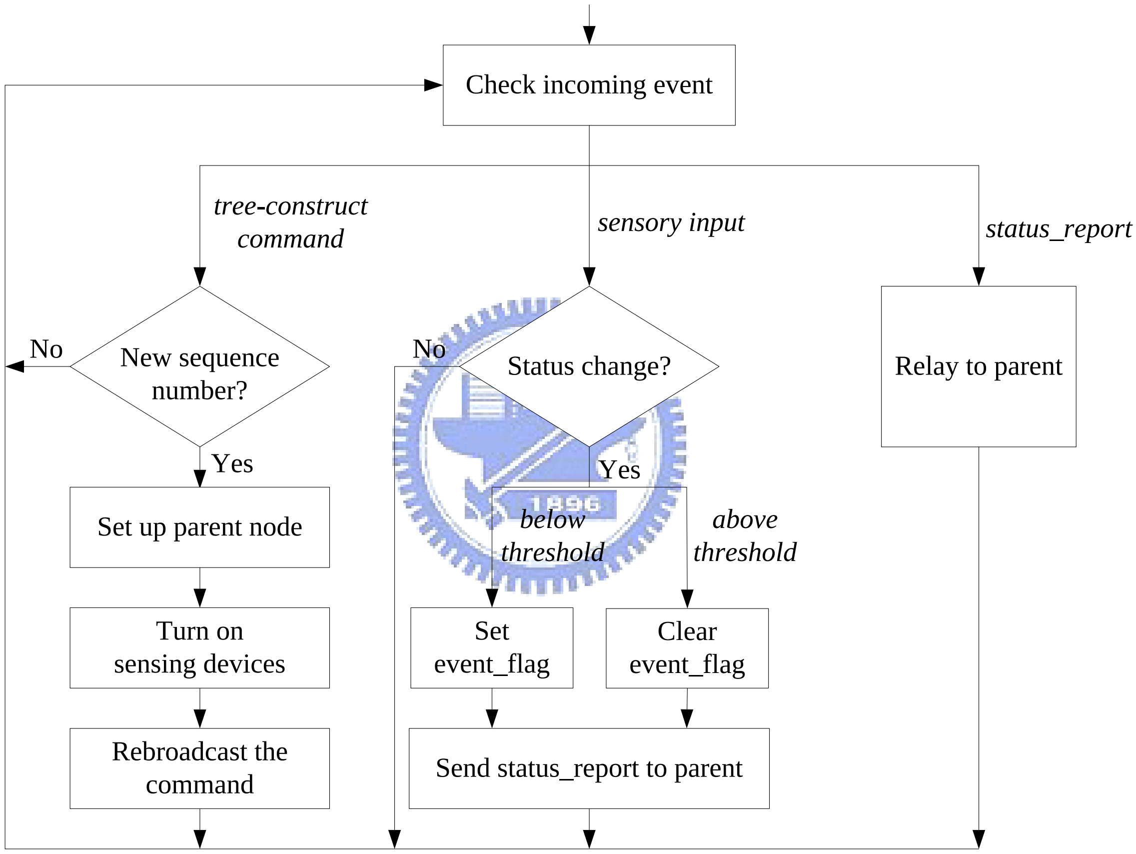


Figure 3.7: The algorithm run by the static sensors.

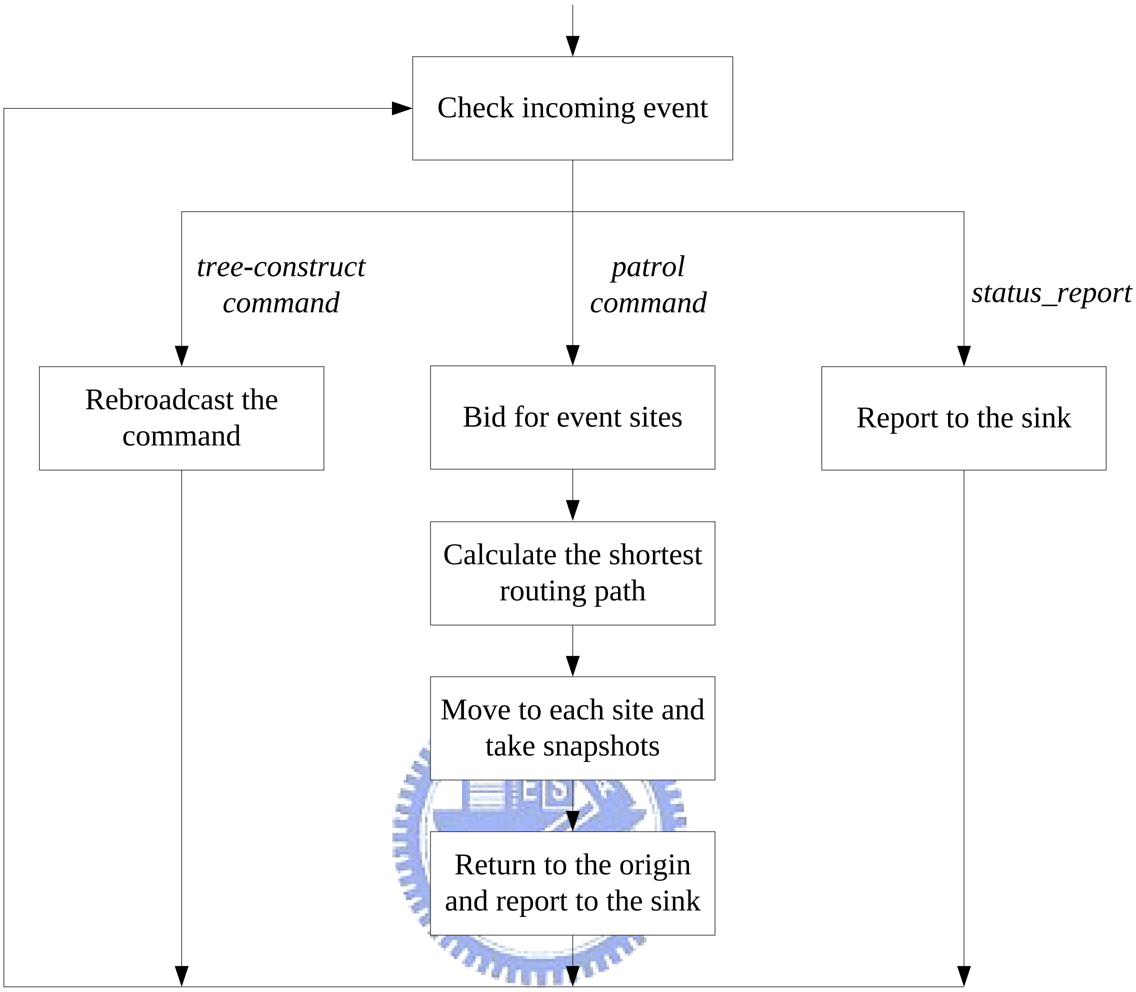


Figure 3.8: The algorithm run by the mobile sensors.

spanning tree. The *event_flag* is to determine if an update of *status_report* is needed.

Each mobile sensor runs the algorithm in Fig. 3.8. If a *tree-construct* command is received from the remote sink, it will broadcast it to the WSN. It then waits for *status_report* from static sensors and forwards the report to the sink. On receiving the sink's *patrol* command, the mobile sensors have to take further actions. The traveling-salesman algorithm APPROX-TSP-TOUR [21] is used to compute their patrolling paths. A spanning tree of all potential emergency sits is formed, and then a heuristic is used to partition the tree into a number of regions, each to be visited by a mobile sensor. For each region, the patrolling path is the preorder tree walk. Photos are saved in the Stargate's flash memory. When moving back to the origin, each mobile sensor forwards photos of all visited sites to the sink.

Two types of packets, *command* and *data*, are defined. The former is initiated by mo-

1 Byte	2 Bytes	2 Bytes	2 Bytes	2 Bytes	2 Bytes	2 Bytes
Type	Original_ID	Source_ID	Dest_ID	Seq_Num	Hop_count	Other

(a) command packet

1 Byte	2 Bytes	2 Bytes	2 Bytes	2 Bytes	2 Bytes
Type	Original_ID	Source_ID	Dest_ID	Seq_Num	Data

(b) data packet

Figure 3.9: Packet formats.

bile sensors and the latter is initiated by static sensors. The formats are shown in Fig. 3.9. The *Original_ID*, *Source_ID*, and *Dest_ID* fields are the initiator, transmitter, and receiver of the packet. The *Seq_Num* field, together with the *Original_ID* field, guarantees the uniqueness of a message. The *Hop_Count* field of the command packet helps establish the spanning tree, and the *Data* field contains the sensing value and status of the originating sensor.

3.3.3 User Interfaces

At the remote sink, we provide an interface to monitor the status of the WSN and to control mobile sensor, as shown in Fig. 3.10. It includes six major components: *Config*, *Command*, *Status*, *Control*, *Monitoring* and *Log* fields. The *Config* area is to input configuration information of the iMouse system, such as mobile sensors' IP addresses, ports, sensors' positions, etc. The Command area is to load the configuration file (such as sensors' positions) to sensors, establish connection from the sink to mobile sensors, issue the tree-construct commands, change the parent of a static sensor [1], calculate the patrolling paths of mobile sensors, disconnect all mobile sensors (so as to reset all environment parameters), add a new sensor in the sensor network, set a sensor's position, remove a sensor from the network, and check the status of a static sensor. The Status area shows the status of a static sensor being queried. The Control area can be used to control the movement of a mobile sensor or ask a mobile sensor to take a snapshot. The Monitoring area shows the network topology of the WSN, and the patrolling paths of mobile sensors. When a sensor detects an event, a fire icon will be shown in the corresponding site. A camera icon will be shown when a snapshot has been taken for the site. Finally, the Log area shows

[1] This is to adjust the topology of the spanning tree.

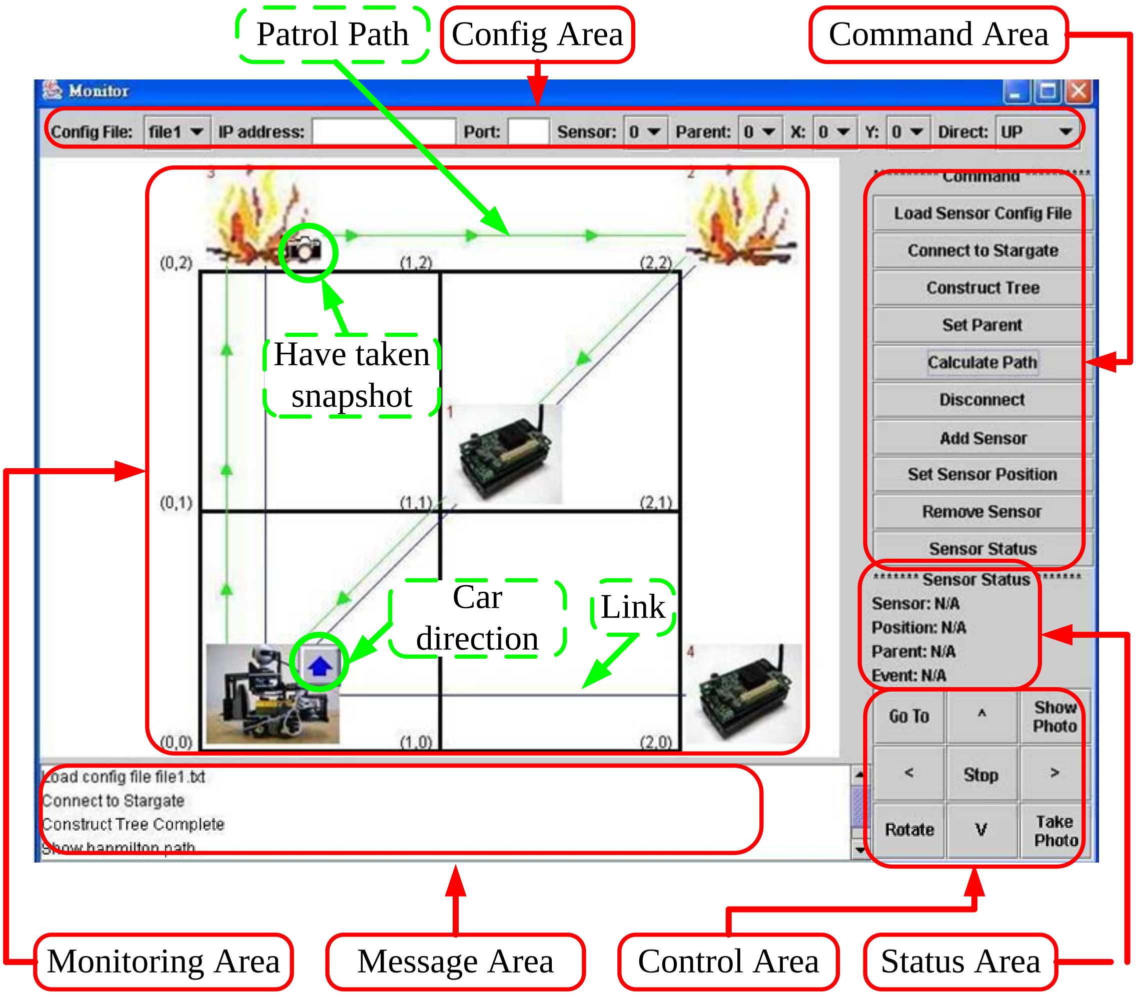


Figure 3.10: User interface at the remote sink.

some status messages. For example, Fig. 3.10 shows the user interface when sensors at coordinates $(2, 2)$ and $(0, 2)$ detect potential fire emergencies.

Chapter 4

Analysis of Event Detection Latency

In this section, we propose a model to analyze the event detection latency of the iMouse system. The results may be applied to general sensor networks, too. We are given a sensing field A, on which there are n homogeneous sensors. Each sensor has a sensing distance of r. Without loss of generality, we assume that these n sensors form a connected network. To simplify the analysis, we assume that the time axis is divided into fixed-length *slots* and the working schedule of each sensor is modeled by *cycles*, where each cycle consists of T slots. Each cycle is divided into an *active phase* and an *idle phase*. The former consists of the first D slots, and the latter the rest of the $T - D$ slots. Sensors only conduct detection jobs in their active phases. However, sensors do not synchronize their clocks, so their cycles are not necessarily aligned. Fig. 4.1 shows an example.

Our goal is to evaluate the detection latency when an event appears in any location in the sensing field. To take errors into account, we assume that in an active slot, a sensor has a probability of p to successfully detect the appearance of the event if the event is within this sensor's sensing range. We consider two detection models in this work:

- **any-sensor-detection model:** To capture the event, the network needs at least one sensor to successfully detect the event.

- **k-sensor-detection model:** To capture the event, the network needs at least k sensors to successfully detect the event, where $k > 1$. (The value of k is application-dependent. For example, for trilateration, at least three sensors are needed.)

Suppose that at time slot 0, an event appears at location (x, y) in the sensing field. Let $M(x, y)$ be the number of sensors whose sensing ranges cover location (x, y). Consider these $M(x, y)$ sensors' first active slots after slot 0. We can classify them into T

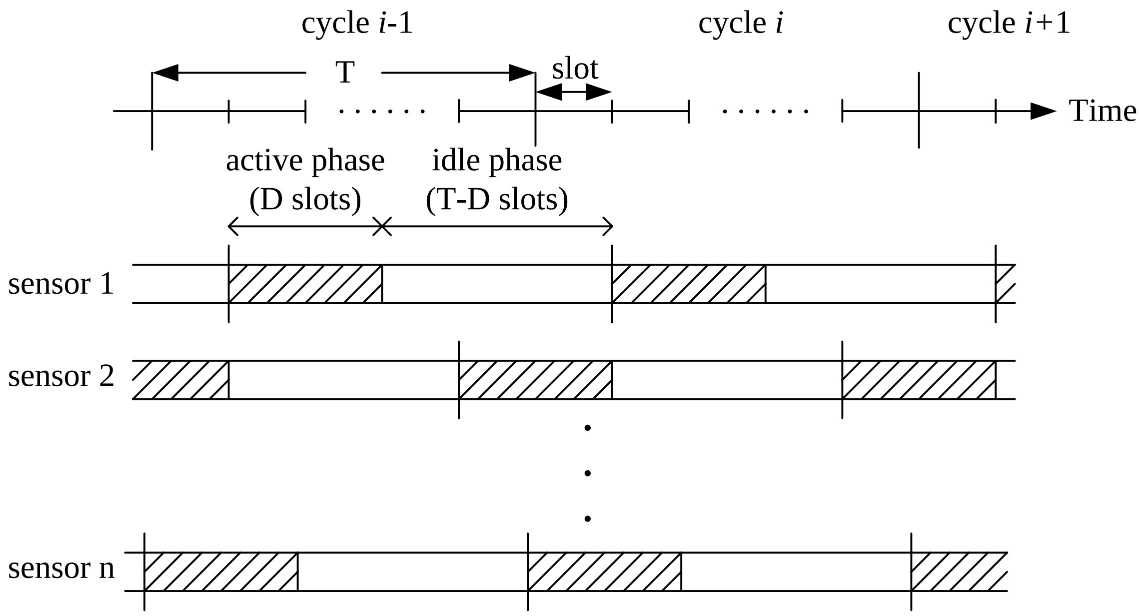


Figure 4.1: Modeling of sensors' active phases and idle phases.

groups such that each group i consists of m_i sensors whose first active slot is at the ith slot, $i = 1, \ldots, T$. Clearly, $\sum_{i=1}^{T} m_i = M(x, y)$. Taking all combinations of m_i's into consideration, the detection latency can be written as

$$Latency(x,y) = \sum_{m_1=0}^{M(x,y)} \sum_{m_2=0}^{M(x,y)-m_1} \sum_{m_3=0}^{M(x,y)-m_1-m_2} \cdots \sum_{m_{T-1}=0}^{M(x,y)-(m_1+m_2+\cdots+m_{T-2})} \left(\frac{M(x,y)!}{m_1! \cdot m_2! \cdot \cdots \cdot m_T!} \times (\frac{1}{T})^{M(x,y)} \right) \times \delta(m_1, m_2, ..., m_T),$$

where the first term is the probability to observe a particular combination (m_1, m_2, $\cdots$, m_T), and the second term $\delta(m_1, m_2, ..., m_T)$ is the expected delay for this particular combination. As the event may appear in any location inside A, the average latency can be written as

$$E_{T,D} = \frac{\Sigma_{\forall(x,y)} Latency(x,y)}{\Sigma_{\forall(x,y)} 1}. \tag{4.1}$$

In Sections 4.1 and 4.2, we will show how to compute $\delta(m_1, m_2, ..., m_T)$ under our two detection models. Section 4.3 presents our simulation result. Table 4.1 summarizes the notations used in this work.

4.1 Any-Sensor-Detection Model

Under this model, the event is considered to be captured by the network, once any sensor successfully detects its existence. Let x_i be the number of active sensors at the ith slot, $i = 1, \ldots, T$. These x_i sensors can be classified into three types: (1) sensors which turn into active at the ith slot, (2) sensors which turn into active between the first and the $(i-1)$th slots, and (3) sensors which turns into active at or before the 0-th slot. Note that $x_i \neq m_i$ unless $D = 1$. This leads to

$$x_i = m_i + \sum_{j=1}^{i-1} m_{i-j} + \sum_{j=0}^{D-i-1} m_{T-j}.$$

We also define x_{aT+b} as the number of active sensors at the ($aT + b$)th slot for any $a \geq 1$. Since cycles repeat every T slots, we have $x_{aT+b} = x_b$.

The probability that there is at least one sensor successfully detecting the event in the first slot is $(1 - (1 - p)^{x_1})$. For $i \geq 2$, the probability that the event is not detected in the first $(i-1)$ slots but is successfully detected in the ith slot is $(1-p)^{x_1+\ldots+x_{i-1}}(1-(1-p)^{x_i})$. Hence, the expected detection latency under the any-sensor-detection model is

$$\delta(m_1, m_2, ..., m_T) = \sum_{a=0}^{\infty} \sum_{b=1}^{T} (aT + b)(1 - p)^{a\times(x_1+\cdots+x_T)+x_1+\cdots+x_{b-1}}(1 - (1 - p)^{x_b}). \tag{4.2}$$

Eq. (4.2) contains an infinite number of expressions. The following theorem shows that it will converge.

Theorem 1. *The expected delay $\delta(m_1, m_2, ..., m_T)$ under the any-sensor-detection model is bounded by*

$$\delta(m_1, m_2, ..., m_T) \leq \frac{T^2}{(1-\alpha)^2},$$

where $\alpha = (1 - p)^{D\times M(x,y)}$.

Proof. Since the sequence $x_1, x_2, ...$ has a period of T, we can rewrite Eq. (4.2) in terms of cycles:

$$
\begin{aligned}
&\sum_{a=0}^{\infty}\sum_{b=1}^{T}(aT+b)(1-p)^{a\times(x_1+\cdots+x_T)+x_1+\cdots+x_{b-1}}(1-(1-p)^{x_b})\\
&\leq \sum_{a=0}^{\infty}\sum_{b=1}^{T}(aT+b)(1-p)^{a\times(x_1+\cdots+x_T)}\\
&\leq \sum_{a=0}^{\infty}\left((1-p)^{a\times D\times M(x,y)}\sum_{b=1}^{T}(a+1)\times T\right)\\
&= \sum_{a=0}^{\infty}\alpha^a(a+1)T^2\\
&= \frac{T^2}{(1-\alpha)^2}. \quad \square
\end{aligned}
$$

4.2 k-Sensors-Detection Model

Under this model, the event is considered to be captured by the network, once there are at least k sensors successfully detecting its occurrence. Since the sequence $x_1, x_2, ...$ has a period of T, the expected latency can be written as

$$\delta(m_1, m_2, ..., m_T) = \sum_{a=0}^{\infty}\sum_{b=1}^{T}(aT+b)\cdot P_{succ}(m_1, m_2, ..., m_T, aT+b), \tag{4.3}$$

where $P_{succ}(m_1, m_2, ..., m_T, aT+b)$ is the probability that there are at least k sensors successfully detecting the event at the $(aT+b)$th slot, but not so before that slot. To find $P_{succ}(m_1, m_2, ..., m_T, aT+b)$, let $N_{already}$ be the number of sensors that already detected the event before the ($aT+b$)th slot, and N_{first} be the number of sensors that detect this event at the ($aT+b$)th slot for the first time. We first categorize sensors according to their behaviors as shown in Fig. 4.2. There are $x_{aT+b} = x_b$ active sensors at the ($aT+b$)th slot, and the rest of $M(x,y) - x_b$ sensors are inactive. The inactive sensors can be further divided into a set of N_1 sensors which have detected this event before the ($aT+b$)th slot, and a set of $M(x,y) - x_b - N_1$ sensors which have not. Similarly, the active sensors can be divided into a set of N_2 sensors which successfully detect this event at this slot, and a set of $x_b - N_2$ sensors which fail to detect this event at this slot. From the latter set, we further identify a set of N_3 sensors which ever succeeded to detect this event before the ($aT+b$)th slot, but fail to detect this event at the current slot.

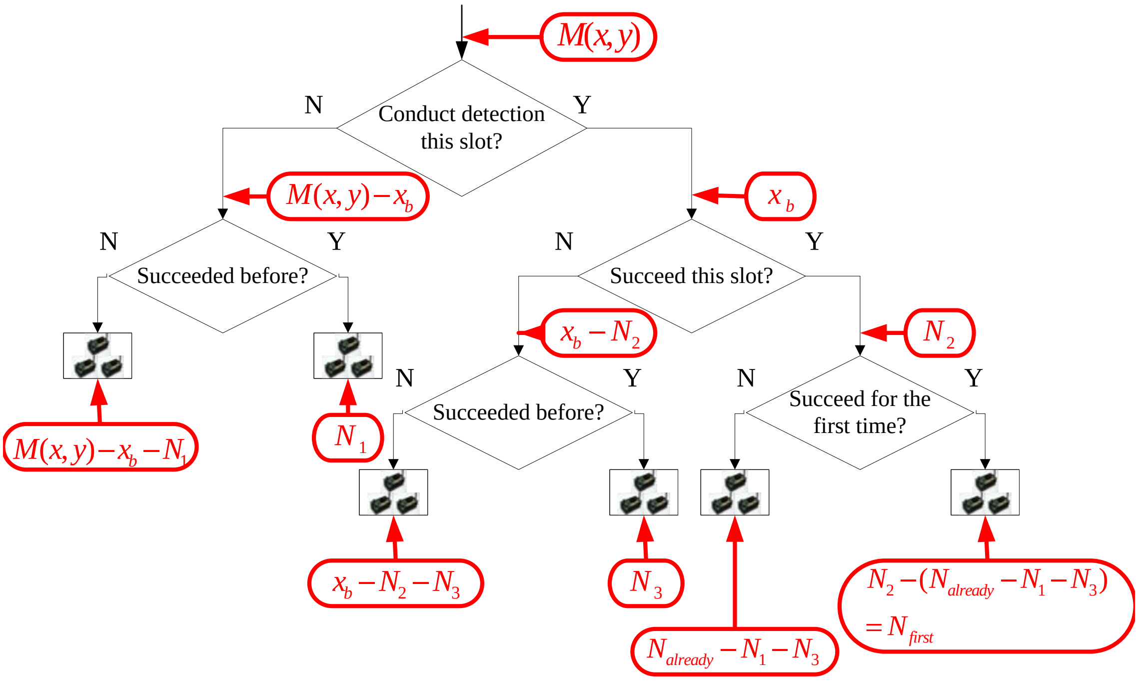


Figure 4.2: Classification of sensors in the $(aT+b)$th slot. Numbers in ovals indicate numbers of sensors.

Based on the above definitions, once the values of $N_{already}, N_1, N_2$, and N_3 are given, the rest of the variables in Fig. 4.2 will all be fixed. Specifically, the number of sensors that successfully detect this event at the current slot but have already detected this event before the $(aT+b)$th slot is $N_{already} - N_1 - N_3$, and the number of sensors that detect this event for the first time at the current slot is $N_{first} = N_2 - (N_{already} - N_1 - N_3)$. In Eq. (4.3), the latency is considered to be aT+ b if $N_{already} < k$ and $N_{first} = (N_1+N_2+N_3) - N_{already} \geq k - N_{already}$. By enumerating all combinations of $N_{already}, N_1, N_2$, and N_3, we can derive that

$$
\begin{aligned}
P_{succ}(m_1, m_2, ..., m_T, aT+b) = \sum_{N_{already}=0}^{k-1} \Big(\sum_{h_1=0}^{N_{already}} Prob[N_1 = h_1] \cdot \Big(\sum_{h_2=0}^{x_b} Prob[N_2 = h_2] \cdot \\
\Big(\sum_{h_3=0}^{N_{already}-h_1} Prob[N_3 = h_3] \cdot Prob[N_{first} \geq k - N_{already}] \Big)\Big)\Big).
\end{aligned} \tag{4.4}
$$

Depending on the value of b, we further expand Eq. (4.4) into three cases.

Case (1): $b < D$**.** Consider the set of $M(x,y) - x_b$ inactive sensors at the current slot.

We divide them into two subsets:

- S_1**:** The set of sensors whose active phases do not cross the boundaries of cycles.
- S_2**:** The set of sensors whose active phases cross the boundaries of cycles.

Clearly, $|S_1| = m_{b+1}+m_{b+2}+...+m_{T-(D-1)}$ and $|S_2| = m_{T-(D-1)+1}+m_{T-(D-1)+2}+ ... + m_{T-(D-b)}$. For example, when $b = 2$, Fig. 4.3 shows the above two subsets in case 1. Recall the definition of N_1. Among these N_1 sensors, let R_1 be the number of sensors belonging to S_1, and R_2 the number of sensors belonging to S_2. Note that $R_1 + R_2 = N_1$. We can expand Eq. (4.4) as follows:

$$P_{succ}(m_1, m_2, ..., m_T, aT + b) = \sum_{N_{already}=0}^{k-1} (\sum_{h_1=0}^{N_{already}} (\sum_{r_1=0}^{h_1} Prob[R_1 = r_1] \cdot Prob[R_2 = h_1 - r_1]) \cdot$$
$$(\sum_{h_2=0}^{x_b} Prob[N_2 = h_2] \cdot (\sum_{h_3=0}^{N_{already}-h_1} Prob[N_3 = h_3] \cdot Prob[N_{first} \geq k - N_{already}]))). \tag{4.5}$$

Given two integers x and y such that $x \geq y$ and a probability value z, let us define

$$Bino(x, y, z) = \binom{x}{y} z^y \cdot (1 - z)^{x-y}. \tag{4.6}$$

The probability functions in Eq. (4.5) are derived as follows:

$$Prob[R_1 = r_1] = Bino(|S_1|, r_1, 1 - (1 - p)^{aD}),$$
$$Prob[R_2 = h_1 - r_1] = \sum_{i=1}^{D-2} \frac{m_{T-(D-1)+i}}{|S_2|} \cdot Bino(|S_2|, h_1 - r_1, 1 - (1 - p)^{aD+i}),$$
$$Prob[N_2 = h_2] = Bino(x_b, h_2, p),$$
$$Prob[N_3 = h_3] = Bino(x_b - h_2, h_3, \sum_{i=0}^{b-1} \frac{m_{b-i}}{x_b}(1 - (1 - p)^{aD+b-i}) + \sum_{i=0}^{D-b-1} \frac{m_{T-i}}{x_b}(1 - (1 - p)^{aD+b})),$$
$$Prob[N_{first} \geq k - N_{already}] = Bino(h_2, N_{first}, \sum_{i=0}^{b-1} \frac{m_{b-i}}{x_b}(1 - p)^{aD+b-i} + \sum_{i=0}^{D-b-1} \frac{m_{T-i}}{x_b}(1 - p)^{aD+b-1}).$$

$Prob[R_1 = r_1]$ is the probability that r_1 sensors in S_1 have ever detected this event before the current slot, where $1 - (1 - p)^{aD}$ is the probability that a sensor has ever detected this event before the current slot. $Prob[R_2 = h_1 - r_1]$ is derived similarly, except that we are concerned about sensors in S_2 and, among these sensors, there is a ratio of $\frac{m_{T-(D-1)+i}}{|S_2|}$ of

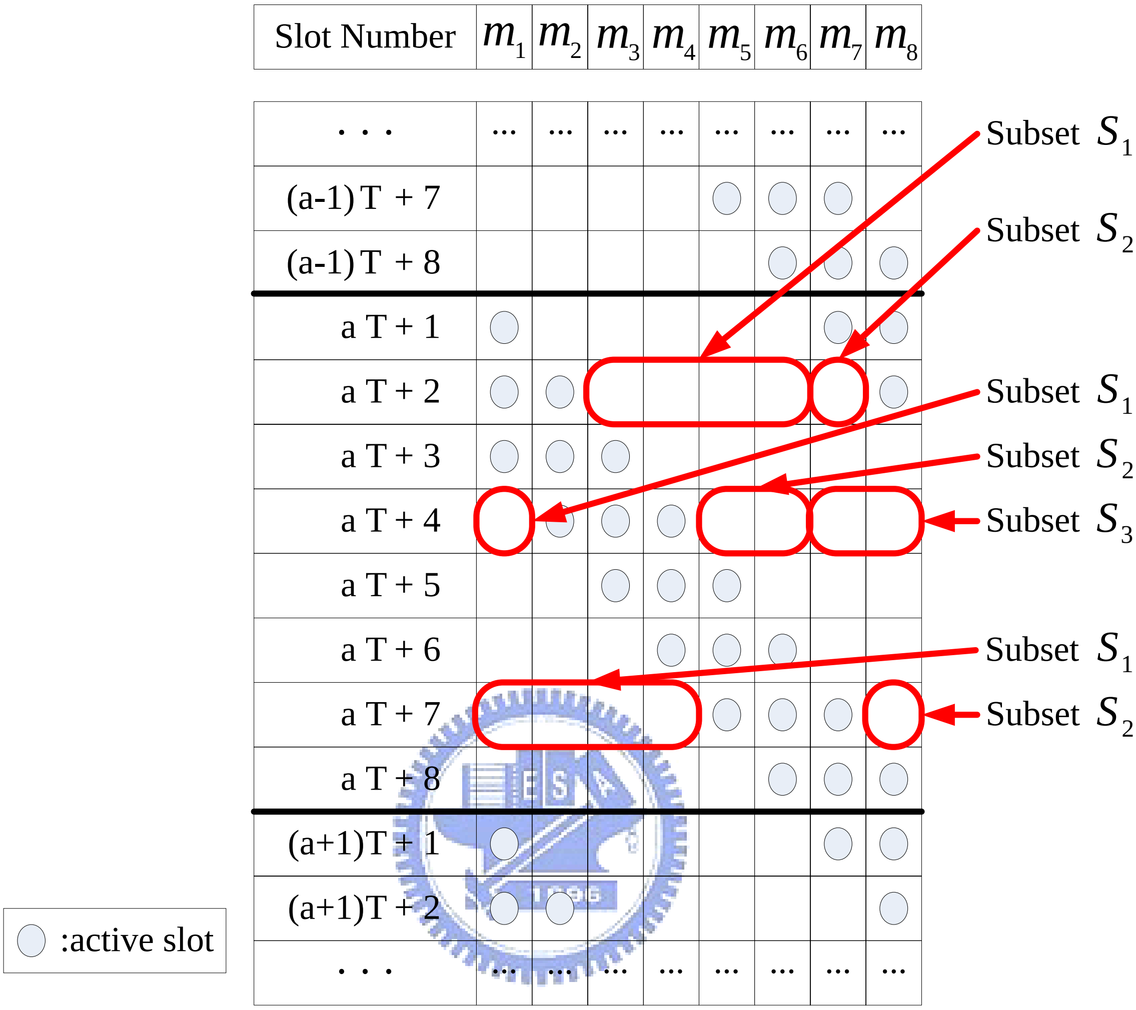


Figure 4.3: Classification of sensors in a network with $T = 8$ and $D = 3$.

sensors which have tried to detect this event for $aD + i$ slots (and we take their average). $Prob[N_2 = h_2]$ is the probability that there are h_2 sensors among x_b sensors successfully detecting the event at the current slot. $Prob[N_3 = h_3]$ is the probability that there are h_3 sensors among $x_b - h_2$ sensors that have ever successfully detected the event before the current slot. The third term in $Bino(\cdot)$ is to take care of those sensors whose active slots do not (the first expression) and do (the second expression) cross the boundaries of cycles, and we take their average. $Prob[N_{first} \geq k - N_{already}]$ is similar to the previous probability except that these sensors succeed for the first time at the current slot.

Case (2): $D \leq b \leq T - D + 1$. In this case, we divide the set of inactive $M(x, y) - x_b$ sensors at the current slot into three subsets according to whether their active slots cross the boundaries of cycles:

- S_1**:** The set of sensors which have finished their active slots at the current cycle and whose active slots do not cross the boundaries of cycles.

- S_2**:** The set of sensors which have not started their active slots at the current cycle and whose active slots do not cross the boundaries of cycles.

- S_3**:** The set of sensors whose active slots cross the boundaries of cycles.

We can obtain that $|S_1| = \sum_{i=1}^{b-D} m_i$, $|S_2| = \sum_{i=b+1}^{T-(D-1)} m_i$, and $|S_3| = \sum_{i=T-(D-1)+1}^{T} m_i$. For example, when $b = 4$, Fig. 4.3 shows these subsets in case 2. We can expand Eq. (4.4) as follows:

$$P_{succ}(m_1, m_2, ..., m_T, aT + b) = \sum_{N_{already}=0}^{k-1} (\sum_{h_1=0}^{N_{already}} (\sum_{r_1=0}^{h_1} \sum_{r_2=0}^{h_1-r_1} Prob[R_1 = r_1] \cdot$$
$$Prob[R_2 = r_2] \cdot Prob[R_3 = h_1 - r_1 - r_2]) \cdot (\sum_{h_2=0}^{x_b} Prob[N_2 = h_2] \cdot$$
$$(\sum_{h_3=0}^{N_{already}-h_1} Prob[N_3 = h_3] \cdot Prob[N_{first} \geq k - N_{already}]))), \tag{4.7}$$

where

$$Prob[R_1 = r_1] = Bino(|S_1|, r_1, 1 - (1 - p)^{(a+1)D}),$$
$$Prob[R_2 = r_2] = Bino(|S_2|, r_2, 1 - (1 - p)^{aD}),$$
$$Prob[R_3 = r_3] = \sum_{i=0}^{D-2} \frac{m_{T-(D-1)+1+i}}{|S_3|} Bino(|S_3|, r_3, 1 - (1 - p)^{aD+i+1}),$$
$$Prob[N_2 = h_2] = Bino(x_b, h_2, p),$$
$$Prob[N_3 = h_3] = Bino(x_b - h_2, h_3, \sum_{i=0}^{D-1} \frac{m_{b-i}}{x_b}(1 - (1 - p)^{aD+i})), \text{ and}$$
$$Prob[N_{first} \geq k - N_{already}] = Bino(h_2, N_{first}, \sum_{i=0}^{D-1} \frac{m_{b-i}}{x_b}(1 - p)^{aD+i}).$$

Case (3): $b > T - D + 1$**.** In this case, we divide the set of inactive $M(x, y) - x_b$ sensors at the current slot into two subsets according to whether their active slots cross the boundaries of cycles:

- S_1**:** The set of sensors whose active slots do not cross the boundaries of cycles.

- S_2**:** The set of sensors whose active slots cross the boundaries of cycles.

We have $S_1 = \sum_{i=1}^{b-D} m_i$ and $S_2 = \sum_{i=b+1}^{T} m_i$. Fig. 4.3 gives an example when $b = 7$. We derive Eq. (4.4) as follows:

$$P_{succ}(m_1, m_2, ..., m_T, aT+b) = \sum_{N_{already}=0}^{k-1} (\sum_{h_1=0}^{N_{already}} (\sum_{r_1=0}^{h_1} Prob[R_1 = r_1] \cdot Prob[R_2 = h_1 - r_1]) \cdot (\sum_{h_2=0}^{x_b} Prob[N_2 = h_2] \cdot (\sum_{h_3=0}^{N_{already}-h_1} Prob[N_3 = h_3] \cdot Prob[N_{first} \geq k - N_{already}]))), \tag{4.8}$$

where

$$Prob[R_1 = r_1] = Bino(|S_1|, r_1, 1-(1-p)^{(a+1)D}),$$

$$Prob[R_2 = r_2] = \sum_{i=1}^{D-2} \frac{m_{T-(D-1)+1+i}}{|S_2|} Bino(|S_2|, r_2, 1-(1-p)^{aD+i+1}),$$

$$Prob[N_2 = h_2] = Bino(x_b, h_2, p),$$

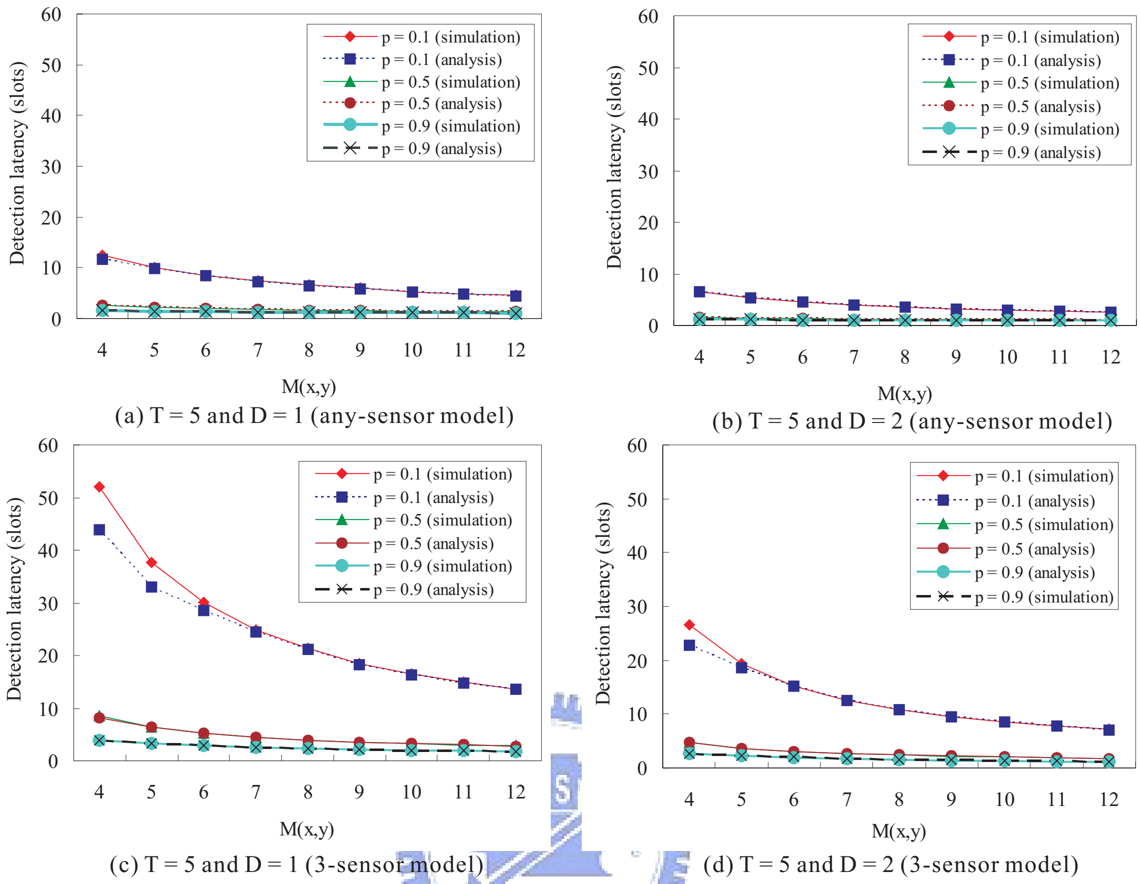
$$Prob[N_3 = h_3] = Bino(x_b - h_2, h_3, \sum_{i=0}^{T-b} \frac{m_{T-(D-1)-i}}{x_b}(1-(1-p)^{aD+i+1}) + \sum_{i=0}^{b-T+(D-1)-1} \frac{m_{T-(D-1)+1+i}}{x_b}(1-(1-p)^{aD+i+1})), \text{ and}$$

$$Prob[N_{first} \geq k - N_{already}] = Bino(h_2, N_{first}, \sum_{i=0}^{T-b} \frac{m_{T-(D-1)-i}}{x_b}(1-p)^{aD+i+1} + \sum_{i=0}^{b-T+(D-1)-1} \frac{m_{T-(D-1)+1+i}}{x_b}(1-p)^{aD+i+1}).$$

Finally, by replacing $P_{succ}(m_1, m_2, ..., m_T, aT+b)$ in Eq. (4.3) with one of the above three cases, we can obtain the expected latency $\delta(m_1, m_2, ..., m_T)$ under the k-sensor-detection model.

4.3 Simulation Results

We have developed a simulator to verify our analytical results. A sensing field of size 10×10 is simulated, on which 50 sensors are deployed randomly. Each sensor has a sensing distance of 3 units. Events may appear in any location in the sensing field.



(a) T = 5 and D = 1 (any-sensor model)
(b) T = 5 and D = 2 (any-sensor model)
(c) T = 5 and D = 1 (3-sensor model)
(d) T = 5 and D = 2 (3-sensor model)

Figure 4.4: The detection latencies under different values of $M(x, y)$.

Given a network configuration, we evaluate the detection latency by both Eq. (4.1) and by simulation. For simulation, at least 1000 experiments are repeated, and we take their average.

Fig. 4.4 shows the detection latencies under different values of $M(x, y)$ and detection probability p. The simulation results coincide well with the analytical results, except when $p = 0.1$ and $M(x, y) \leq 5$ under the 3-sensor-detection model. This is because our analysis assumes a larger-scale network. It can be observed that a larger $M(x, y)$, which implies a higher network density, can help reduce the detection latency. A larger detection probability p, which reflects the sensibility of sensors, can also reduce the detection latency. The result can be used to determine how sensors should be arranged at the deployment stage. Fig. 4.5 shows the detection latencies under different values of $M(x, y)$, D, and p when T is set to 16. We can observe that the latency drops significantly when $D \leq 3$. The result can be used to decide the length of a sensor's active phase to reduce both detection latency and energy consumption of a WSN.

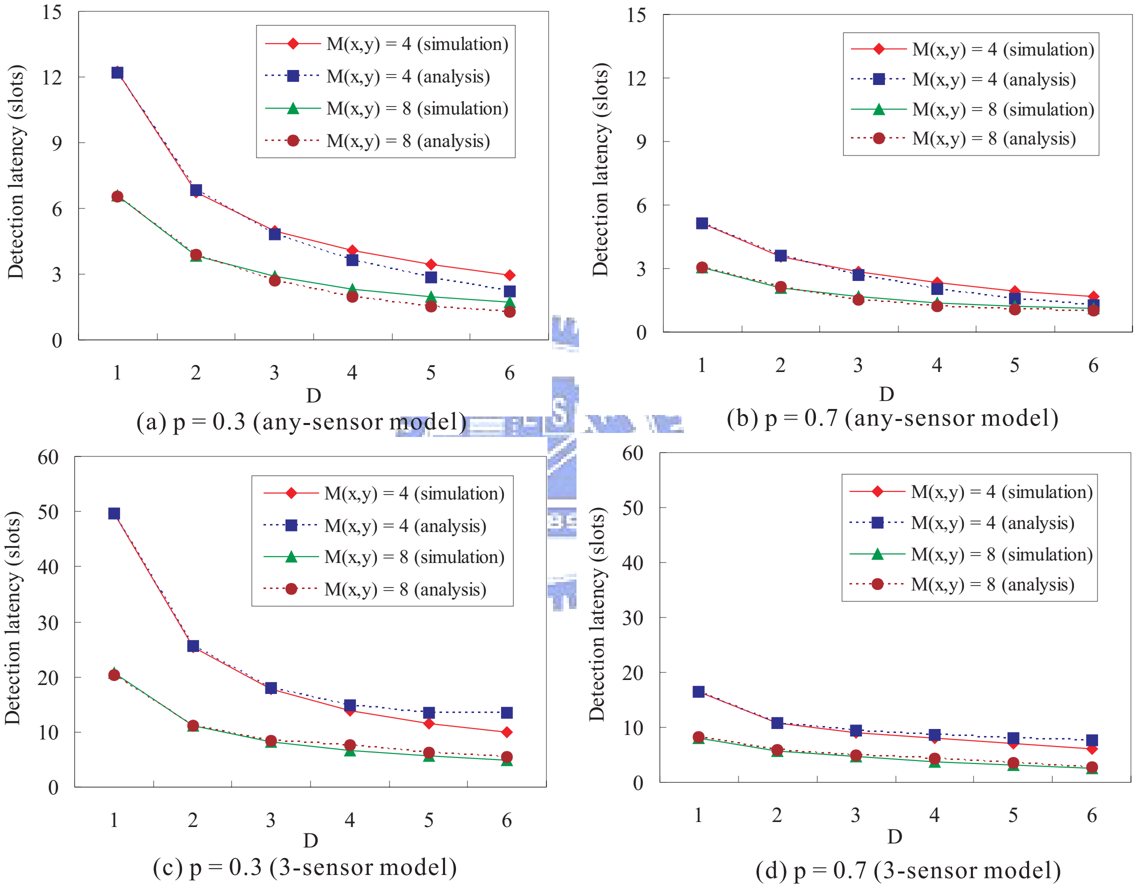


Figure 4.5: The detection latencies under different values of D ($T = 16$).

Table 4.1: Summary of notations.

Notations	Definition
n	number of sensors in the sensing field
k	minimum number of sensors required to successfully detect the event
T	number of slots in a cycle
D	number of slots that a sensor continues detecting the event
p	probability that a sensor successfully detects the event during a slot
$M(x, y)$	number of sensors that can detect the event when the event occurs in (x, y)
m_i	number of sensors in $M(x, y)$ that conduct detect in the ith slot after the event occurs
x_i	number of sensors detect the event in the ith slot of a cycle
$P_{succ}(m_1, m_2, ..., m_T, aT + b)$	probability that there are at least k sensors which succeed to detect the event
$N_{already}$	number of sensors that have ever succeeded to detect the event before the $(aT + b)$th slot
N_{first}	number of sensors that first succeed to detect the event at the $(aT + b)$th slot
N_1	number of sensors that have ever succeeded to detect the event before but do not detect at current slot
N_2	number of sensors that succeed to detect the event at the $(aT + b)$th slot
N_3	number of sensors that have ever succeeded to detect the event before but fail at current slot
S_i	number of sensors in the subset i
R_i	number of sensors that have succeeded to detect the event in the subset i

Chapter 5

Conclusions

The proposed iMouse system combines two areas, wireless sensor network technology and surveillance system, to support intelligent mobile surveillance services. On one hand, the mobile sensors can help improve the weakness of traditional wireless sensor networks that they only provide vague environmental information of the sensing field by including some mobile cameras to conduct in-depth analysis of the sensing field. On the other hand, the wireless sensor network provides context awareness and intelligence to the surveillance system. Therefore, the weakness of traditional “dumb” surveillance system is greatly improved because the real critical images/video sections can be retrieved and proactively sent to the users.

We believe that the prototype that we have demonstrated still can be improved in several ways. First, the grid-like patrolling paths of mobile sensors should be further improved. Second, the coordination among mobile sensors, especially when they are on-the-road, can be exploited. Third, how to utilize mobile sensors to improve the network topology deserves further investigation.

Bibliography

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A survey on sensor networks,” *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless sensor networks: a survey,” *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[3] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, “Wireless sensor networks for habitat monitoring,” in *ACM International Workshop on Wireless Sensor Networks and Applications*, 2002, pp. 88–97.

[4] J. Burrell, T. Brooke, and R. Beckwith, “Vineyard computing: sensor networks in agricultural production,” *IEEE Pervasive Computing*, vol. 3, no. 1, pp. 38–45, 2004.

[5] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, “Next century challenges: scalable coordination in sensor networks,” in *ACM/IEEE International Conference on Mobile Computing and Networking*, 1999, pp. 263–270.

[6] J. M. Kahn, R. H. Katz, and K. S. J. Pister, “Next century challenges: mobile networking for “Smart Dust”,” in *ACM/IEEE International Conference on Mobile Computing and Networking*, 1999, pp. 271–278.

[7] M. Valera and S. A. Velastin, “Intelligent distributed surveillance systems: a review,” *IEE Proceedings - Vision, Image and Signal Processing*, vol. 152, no. 2, pp. 192–204, 2005.

[8] W. C. Feng, E. Kaiser, W. C. Feng, and M. L. Baillif, “Panoptes: scalable low-power video sensor networking technologies,” *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 1, no. 2, pp. 151–167, 2005.

[9] J. H. Lee and H. Hashimoto, “Controlling mobile robots in distributed intelligent sensor network,” *IEEE Transaction on Industrial Electronics*, vol. 50, no. 5, pp. 890–902, 2003.

[10] B. J. Lee, J. H. Lee, and G. T. Park, “The resource sharing architecture of mobile robots in the home network environment using Jini,” in *International Symposium on Robotics*, 2001, pp. 233–238.

[11] A. Hoover and B. D. Olsen, “Sensor network perception for mobile robotics,” in *IEEE International Conference on Robotics and Automation*, 2000, pp. 83–88.

[12] H. T. Kung and D. Vlah, “Efficient location tracking using sensor networks,” in *Wireless Communications and Networking*, 2003, pp. 1954–1961.

[13] J. Liu, M. Chu, J. Liu, J. Reich, and F. Zhao, “Distributed state representation for tracking problems in sensor networks,” in *Information Processing in Sensor Networks*, 2004, pp. 234–242.

[14] W. Zhang and G. Cao, “DCTC: dynamic convoy tree-based collaboration for target tracking in sensor networks,” *IEEE Transactions on Wireless Communications*, vol. 3, no. 5, pp. 1689–1701, 2004.

[15] C. Gui and P. Mohapatra, “Power conservation and quality of surveillance in target tracking sensor networks,” in *ACM/IEEE International Conference on Mobile Computing and Networking*, 2004, pp. 129–143.

[16] X. Ji, H. Zha, J. J. Metzner, and G. Kesidis, “Dynamic cluster structure for object detection and tracking in wireless ad-hoc sensor networks,” in *IEEE International Conference on Communications*, 2004, pp. 3807–3811.

[17] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus, “Tracking a moving object with a binary sensor network,” in *International Conference on Embedded Networked Sensor Systems*, 2003, pp. 150–161.

[18] Crossbow , “SPB400 - Stargate Gateway,” http://www.xbow.com, 2004.

[19] MINDSTORM, “Robotics Invention System,” http://mindstorms.lego.com/eng/default.asp, 2004.

[20] Crossbow, “MOTE-KIT2400 - MICAz Developer’s Kit,” http://www.xbow.com, 2004.

[21] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, “Introduction to Algorithms Second Edition,” The MIT Press, 2001.

Curriculum Vita

Kai-Yang Cheng (kycheng@csie.nctu.edu.tw) received his B.S. degree in Computer Science from the National Tsing Hua University, Taiwan, in 2003. His research interests include wireless network and wireless sensor network.