國立交通大學

資訊工程系

博士論文

行動無線感測網路下的佈建、派遣、與封包排程之議題研究

The Deployment, Dispatch, and Packet-scheduling Issues of Mobile Wireless Sensor Networks

研 究 生：王友群

指導教授：曾煜棋　教授

中華民國九十五年十月

行動無線感測網路下的佈建、派遣、與封包排程之議題研究
The Deployment, Dispatch, and Packet-scheduling Issues of Mobile Wireless Sensor Networks

研 究 生：王友群 Student：You-Chiun Wang

指導教授：曾煜棋 Advisor：Yu-Chee Tseng

國 立 交 通 大 學
資 訊 工 程 系
博 士 論 文

A Dissertation
Submitted to Department of Computer Science
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy
in

Computer Science

October 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年十月

行動無線感測網路下的佈建、
派遣、與封包排程之議題研究

學生:王友群　　　　　　　　　　　　　　　　指導教授:曾煜棋 博士

國立交通大學資訊工程學系博士班

摘　　　要

無線感測網路已成為近年來一門新興的科技，能夠改善並豐富我們的日常生活。無線感測網路是由眾多無線裝置所組成，其中每個裝置皆具有從環境收集資訊、並且互相溝通的能力。在本論文中，我們將探討行動無線感測網路下的佈建、派遣、與封包排程三大議題，其中所謂的行動無線感測網路指的是網路中部份或是全部的節點具有移動的能力。特別來講，佈建的議題主要是探討給予一個佈署區域，如何決定最少數量的感測器以及其佈建位置，使得這個區域內任何一點都可以被感測器所涵蓋到、且形成的網路具有連結性；派遣的議題則是研究如何有效率安排行動感測器拜訪特定的位置去執行某些任務，使得這些行動感測器所剩有的電量能夠儘量被保存；當網路建構完畢後、或是行動感測器到達目的地時，封包排程的議題則是討論如何管理來至感測器數據的回報，使得重要的即時訊息不會被延遲、且其他非即時訊息的傳輸仍然被保障。

針對佈建的議題，我們首先提出一套通用的佈建方案，而這套方案容許佈署區域的形狀任意、且其中可能包含障礙物；我們所提出的佈建方案亦允許感測器的通訊距離和感測距離可以隨意，而這點是被以前的文獻所忽略的。我們的佈建方案先是根據佈署區域的狀況以及感測器的通訊距離與感測距離之關係來計算放置最少數量的感測器之位置，然後我們再派遣感測器去這些位置，其中派遣的方式必需滿足某些特定的電源消耗之目標函數；藉由這個方式，我們的佈建方案將可以改善之前文獻的限制並對佈建之議題研究更加的完善。

在本論文中，我們進一步研究如何佈建一個具有多重涵蓋性的感測網路，而這個多重涵蓋性則是許多無線感測網路的應用或協定所必要的假設。針對這個佈建問題，我們亦提出了一套通用的方案能夠允許任意關係的感測器通訊距離與感測距離；我們所提出的佈建方案可以使用較少的感測器，此外，我們也提出了兩套分散式的感測器派遣方法來協助網路的佈建。

針對派遣的議題，我們提出了一個感測器派遣方法可以有效安排行動感測器拜訪不同的事件發生地點，我們所提出的派遣方法可以允許不同數量的行動感測器以

及事件地點，這套派遣方法不但可以均衡行動感測器的移動距離，並且能夠儘量保存它們的電量；藉由這個方式，我們將能在行動感測器有限電量的限制下，儘量延長行動感測器的使用期限。

針對封包排程的議題，我們提出兩套無線封包公平排程演算法：TD-FQ與MR-FQ。TD-FQ在排程封包時考量到資料流的特性，它藉由給予即時性資料流較高的優先權以減緩其延遲時間，然而它仍保障全體資料流傳輸的公平性。MR-FQ則考量一個較複雜的無線傳輸環境，其中感測器可以根據目前傳輸頻道的狀況來調整其傳輸速率；MR-FQ可以根據資料流目前的傳輸頻道狀況以及其滯延程度來調整它的傳輸速率；藉由這個方式，MR-FQ可以同時保障資料流傳輸的公平性以及提升整體的系統效能。

在本論文中，我們亦實作了一套稱為iMouse系統的行動感測器平台，這套系統結合了無線感測網路的環境感知能力以及監控系統，以便取得環境中重要資訊並立即通報人們，因此iMouse系統可以改善傳統視訊監控系統的分析負擔；在本論文中，我們將展示iMouse系統在家庭/辦公室安全使用上的應用。

關鍵詞：連結性、涵蓋性、派遣、公平佇列、行動計算、行動感測器、網路佈建、封包公平排程、服務品質管理、監控應用、拓樸控制、無線感測網路。

The Deployment, Dispatch, and Packet-scheduling Issues of Mobile Wireless Sensor Networks

Student: You-Chiun Wang Advisor: Dr. Yu-Chee Tseng

Department of Computer Science
National Chiao Tung University

ABSTRACT

Wireless sensor networks have become one emerging technology that greatly enrich our life. Such a network consists of many tiny, wireless devices that can gather information from the environment and communicate with each other. In this dissertation, we will study the *deployment*, *dispatch*, and *packet-scheduling* issues of a mobile wireless sensor network, in which some or all nodes in the network have a mobile capability. In particular, the deployment issue discusses how to determine the minimum number of sensors and their locations to be placed in the region of interest so that every point in the region can be covered by sensors and the network is connected. The dispatch issue addresses how to efficiently schedule mobile sensors to reach certain locations to perform some missions so that their energies can be conserved as much as possible. After the network is constructed or mobile sensors arrive at their destinations, the packet-scheduling issue considers how to manage the messages reported from sensors so that the delays of important real-time messages can be bounded while other non-real-time messages will not be starved.

For the deployment issue, we first propose a general deployment solution that allows the deployed region to be arbitrary-shaped and possibly contain obstacles. Our solution also allows an arbitrary relationship of sensors' communication distances r_c and their sensing distances r_s, which is ignored by previous works. Our solution first computes the positions to place the least number of sensors according to the condition of deployed region and the relationship of r_c and r_s. Then we dispatch sensors to these locations under certain constraints of energy consumptions. In this way, our solution can relax the limitations of previous works and is more complete to the deployment problem.

In this dissertation, we further investigate how to deploy a sensor network for multi-level coverage, which is an essential assumption required by many applications and protocols in wireless sensor networks. For this deployment problem, we also propose a general solution in which the relationship of r_c and r_s can be arbitrary. Our solution can use fewer sensors compared with other schemes. In addition, we also propose two distributed dispatch schemes to help deploy sensors.

For the dispatch issue, we propose an efficient dispatch method for mobile sensors to visit event locations in a hybrid sensor network. Our dispatch method is general in which the numbers of event locations and mobile sensors can be arbitrary. Our dispatch method can balance the moving distances of mobile sensors while preserve their energies as much as possible during each round of dispatch. In this way, we can maximize the system time for mobile sensors to perform their missions with their limited energies.

For the packet-scheduling issue, we propose two wireless packet fair scheduling algorithms, *Traffic-Dependent wireless Fair Queuing (TD-FQ)* and *Multi-Rate wireless Fair Queuing (MR-FQ)*. TD-FQ takes traffic types of flows into account when scheduling packets. It gives a higher priority for real-time flows to alleviate their queuing delays, but still guarantees the fairness among all flows. MR-FQ considers a more complicated multi-rate environment in which sensors can adopt different modulation techniques to transmit their packets under different channel conditions. MR-FQ adjusts a flow's transmission rate according to the flow's channel condition and its lagging degree, so that both fairness and system performance can be taken care of.

In this dissertation, we also implement a mobile sensor platform, called the *integrated mobile surveillance and wireless sensor (iMouse) system*. The iMouse system integrates the context-aware capability of wireless sensor network into surveillance system so that the real critical information in the environment can be retrieved and immediately send to users. In this way, the overheads of traditional visual surveillance systems can be reduced. We demonstrate the iMouse system with a home/office security scenario in this dissertation.

Keywords: connectivity, coverage, dispatch, fair queuing, mobile computing, mobile sensors, network deployment, packet fair scheduling, QoS management, surveillance applications, topology control, wireless sensor networks.

誌　　謝

誠摯感謝我的指導教授曾煜棋老師這幾年來的細心指導，以及家人在精神上的支持與鼓勵，讓我能夠順利完成這篇論文；另外，也由衷感謝彭文志老師在dispatch這部份的支援以及寶貴的意見；同時也很感謝我的口試委員王國禎教授、余孝先副所長、吳世琳教授、金仲達教授、陳健教授、以及馮明惠主任對於這篇論文的建議；最後，我要感謝HSCC實驗室中每個曾經共事過的成員，讓我在研究的生涯中得以互相切磋並且渡過許多快樂的時光。

要感謝的人實在太多了，在此真誠地感謝我所擁有的一切！

王友群 於交通大學，2006年。

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivations

Recently, the remarkable advances in both embedded micro-sensing MEMS[1] and wireless communication technologies have promoted the development of *wireless sensor networks*. A wireless sensor network is composed of many tiny, low-power devices that integrate sensing units, transceivers, actuators, and possible mobilizers with limited on-board processing and wireless communication capabilities [3]. These devices are deployed in a region of interest to gather information from the environment, which will be reported to nearby data aggregators or a remote sink. In the past, sensors are connected with wire-lines [12, 57, 70, 74]. Nowadays, by combing with the novel *ad hoc networking* technology to assist in inter-sensor communications [88, 93, 107], the installing and configuring of a sensor network become more flexible. In the recent years, a large amount of research activities and studies have focused on wireless sensor networks, from investigating physical (PHY) and media access control (MAC) layers [104, 109, 122, 124] to designing routing and transport protocols [20, 78, 91, 125]. Many applications and scenarios have also been practiced by wireless sensor networks, such as surveillance, biological detection, and traffic, pollution, habitat, and civil infrastructure monitoring [4, 21, 49, 77, 113].

Among these various researches of wireless sensor networks, *sensor deployment* is one of the most important issues because it decides the network's construction cost and ability to monitor the environment. A good sensor deployment should consider both *sensing coverage* and *network connectivity* [79, 114, 128]. In particular, given a region of interest to be deployed with sensors, the sensing coverage requires that every point inside the region can be monitored by sensors, while the network connectivity requires that the network is not partitioned in terms of sensors' communication capability. Note

[1]Micro-Electro-Mechanical Systems

that coverage is affected by sensors' sensitivity, while connectivity is decided by sensors' communication ranges. In addition, to reduce the cost to construct the network, the number of sensors used should be as few as possible.

In the literature, several computational geometric problems [58, 108, 121] also address the similar issue, but their solutions cannot be applied to the sensor deployment problem because of their assumptions or objectives. Therefore, a large amount of schemes have been proposed to address the sensor deployment issue. For example, references [19, 29, 71] model the sensing field as grid points and discuss how to place sensors on some of these grid points to satisfy certain coverage requirements. The work [62] suggests to deploy sensors strip by strip to ensure coverage and connectivity of the network. In [111], a deployment scheme to construct a sensor network for multi-level coverage is proposed. However, most existing works consider only the ideal case in which they deploy sensors on a simple 2D plane without boundaries and obstacles. Thus, their solutions cannot be applied to the regions containing obstacles, such as the areas in buildings or on bridges. Besides, these works do not address the relationship between communication distances and sensing distances of sensors, which limits their applications.

Several studies consider that sensors have a mobile capability. They assume that sensors are randomly dropped in the sensing field and propose different strategies to make these mobile sensors "self-deploy" to form a network. For example, references [50, 118, 132] discuss how to move sensors to enhance coverage of the sensing field by using the Voronoi diagram or attractive/repulsive forces between sensors. In [119, 123], the sensing field is partitioned into grids, and sensors will be moved from high-density grids to low-density ones so that the number of sensors in each grid can be balanced. As can be seen, the objective of these works is to achieve a more uniform coverage of the sensing field. However, they may not guarantee to satisfy the coverage and connectivity requirements of the deployed network. In addition, they neither address the relationship of sensors' communication distances and their sensing distances.

Therefore, in this dissertation we consider a more general deployment problem in a *mobile wireless sensor network*. In particular, we allow the sensing field to be arbitrary-shaped and possibly contain arbitrary-shaped obstacles. Besides, the relationship of sensors' communication distances and their sensing distances can be also arbitrary. We address two related sub-problems: *sensor placement* and *sensor dispatch*. The sensor placement problem asks how to place the least number of sensors in the region of interest to achieve desire coverage and connectivity properties. The sensor dispatch problem assumes that sensors are mobilized and the goal is to delegate sensors to move to the designated locations inside the region according to the solutions of the placement problem such that certain objective functions can be satisfied.

In the literature, the design of mobile robots have been widely discussed in the field of robotics [37, 55, 67, 68]. Recent studies [61, 101, 103] have also reported their design and implementation of *mobile sensors*. Such mobile platforms are controlled by embedded computers and mounted with sensors. With these platforms, we can dispatch sensors to move to some locations to perform certain missions. This motivates us to further investigate the dispatch problem of mobile sensors. In particular, we consider a hybrid sensor network consisting of static and mobile sensors. The static sensors are deployed in the region of interest to monitor the environment, while the mobile sensors are dispatched to the event locations reported by static sensors to perform more advanced actions (such as conducting more in-depth sensing of events). Since mobile sensors also use small batteries for their operations, one important research issue is to conserve their energies when dispatching these mobile sensors.

In the literature, some studies also address to move sensors for different purposes. For example, the work in [9] suggests to move some nodes to enhance the network connectivity. The studies [14, 15] discuss how to move sensors to the event locations while still maintaining complete coverage of the sensing field. The works [33, 131] consider to add several mobile sensors to help improve the network topology of an existing static sensor network. In [117], a navigation scheme is proposed to guide mobile sensors to the event locations. As can be seen, none of these works consider to save energies of mobile sensors when dispatching them.

Therefore, in this dissertation we further investigate how to efficiently dispatch mobile sensors so that their lifetimes can be prolonged. Specifically, we consider how to assign mobile sensors to visit event locations so that during each round of dispatch, the moving distances of mobile sensors can be balanced while their energies can be preserved as much as possible. In this way, we can maximize the lifetimes of mobile sensors to perform their missions under the constraints of their limited energies.

After constructing the network, sensors will periodically report their sensing data or send real-time notifications to the data aggregators once they detect important events. Besides, mobile sensors arriving at event locations will also report their analyzed data. Based on the types and emergency of these reports, we can classify them into *real-time flows* and *non-real-time flows*. For example, events reported from sensors or analyzed data sent by mobile sensors are classified into real-time flows because events may disappear soon later, while periodical sensing reports from sensors will be classified into non-real-time flows. Among these flows, if we let sensors compete to transmit their reports, real-time flows may miss their delay constraints and thus important messages will expire. On the contrary, if we allow real-time flows always to preempt non-real-time flows, the latter will be starved. Therefore, this motivates us to investigate how to manage these messages reported from sensors so that the delays of real-time flows can be bounded while the fairness among flows' transmissions can be guaranteed.

In wire-line networks, many packet fair scheduling algorithms [43, 44, 92, 127] have been proposed to bound delays and guarantee fairness of packet transmissions. However, wireless channels are characterized by the following features that distinguish themselves from wire-line networks: (1) serious bursty errors, (2) location-dependent errors, and (3) multi-rate communication capability. Bursty errors may break continuous services of a flow, while location-dependent errors may allow error-free flows to receive more services than they deserve, thus violating the fairness and bounded-delay requirements. A wireless channel may provide different transmission rates to different terminals depending on channel qualities. Due to these reasons, existing wire-line solutions may not be suitable for the wireless networks [10, 16]. Therefore, many wireless fair scheduling algorithms have been proposed to address the features (1) and (2) of wireless networks. For example, in IWFQ [75], each packet is associated with a *finish tag* and the scheduler always serves the error-free packet with the smallest finish tag. In CIF-Q [86], fairness is achieved by transferring the services allocated to error flows to those error-free flows, and then compensating these error flows later according to their weights. In SBFA [95], a fraction of bandwidth is reserved particularly to compensate those error flows. Unfortunately, most of these works do not consider the traffic types of flows so that the delay constraints of real-time flows may not be satisfied. Besides, feature (3) of wireless networks is not well addressed. In particular, these works assume that a wireless channel is either in a *good* state or a *bad* state. Transmissions in a good state will succeed, but fail in a bad state. In fact, the situation is not so pessimistic because different modulation techniques can be used to adapt to different channel conditions. In this way, the system performance can be greatly improved.

Therefore, in this dissertation we propose wireless packet fair scheduling algorithms for data aggregators to manage the messages reported from sensors. In particular, our scheduling algorithms should consider the traffic types of flows in a way that the queuing delays of real-time flows can be alleviated while non-real-time flows will not be starved. In addition, the proposed scheduling algorithms should utilize the multi-rate communication feature of wireless channels so that the overall system performance can be improved.

1.2 Contributions of the Dissertation

In this dissertation, we study the *deployment*, *dispatch*, and *packet-scheduling* issues of a mobile wireless sensor network. For the deployment issue, we first consider a general sensor deployment problem in a sensing field possibly with obstacles, and then discuss how to deploy a sensor network for multi-level coverage by mobile sensors. For the dispatch issue, we investigate how to efficiently dispatch mobile sensors to visit event locations so that their lifetimes can be prolonged. For the packet-scheduling issue, we propose two wireless packet fair scheduling algorithms for data aggregators to manage the messages from sensors. Finally, we also implement a mobile sensor platform used for a surveillance application. The contributions of this dissertation are detailed as follows:

- We have considered a more general sensor deployment problem in this dissertation. Specifically, given an arbitrary-shaped sensing field possibly with arbitrary-shaped obstacles, we are asked to deploy the least number of sensors in the field to achieve both sensing coverage and network connectivity. The solution must be applied for any relationship between the communication distance r_c and the sensing distance r_s of sensors. In this dissertation, we have addressed two related sub-problems: *sensor placement* and *sensor dispatch*. The former asks how to place the least number of sensors in a field to achieve sensing coverage and network connectivity, while the latter asks how to determine from a set of mobile sensors a subset of sensors to be moved to an area of interest with certain objective functions such that the coverage and connectivity properties can be satisfied. Our solution to the placement problem allows an arbitrary-shaped sensing field possibly with arbitrary-shaped obstacles and an arbitrary relationship of r_c and r_s, and thus significant relaxes the limitations of existing results. Our approach first partitions the sensing field into smaller sub-regions according to the obstacles and boundaries of the field. In each sub-region, we arrange sensors row by row such that each row guarantees continuous coverage and connectivity and that adjacent rows ensure continuous coverage. Finally, columns of sensors are added to ensure connectivity between rows. Simulations results have shown that our approach requires fewer sensors to ensure complete coverage of the sensing field and connectivity of the network as compared with other schemes. In addition, we have also discussed how to adjust the aforementioned placement solution when a *probabilistic sensing model* is adopted, where the detection probability of a sensor decays with the distance from the sensor to the object. For the dispatch problem, our solutions include a centralized one and a distributed one. The centralized solution is based on adopting the former placement results and converting the dispatch problem to the *maximum-weight maximum-matching problem* with the objective of minimizing the total energy consumption to move sensors or maximizing the average remaining energy of sensors after the movement. Designed in a similar way, the distributed solution allows sensors to determine their moving directions in an autonomous manner. In particular, sensors can select their destinations according to the objective function and then compete to move to these locations.

- We have further investigated the *k-coverage sensor deployment problem* to ensure multi-level (k) coverage of the sensing field in this dissertation. We also deal with this problem by two related sub-problems: *k-coverage sensor placement problem* and *distributed sensor dispatch problem*. The k-coverage placement problem asks how to determine the minimum number of sensors required and their locations in the sensing field to guarantee that the field is k-covered and that the deployed network is connected. Given that there are sufficient sensors and these sensors are randomly dropped in the sensing field, the distributed dispatch problem asks how to determine the schedule of sensors' movements to the designated locations according to the result computed by the placement solutions such that the total energy consumption due to movement can be minimized. For the k-coverage problem, we allow an arbitrary relationship between sensors' communication distance and their sensing distance, thus relaxing the limitations of existing results. We have proposed two methods to the k-coverage placement problem. The first method adopts an intuitive duplication idea in which we first determine a good 1-coverage placement scheme and then duplicate k sensors on each location computed by the placement scheme. The second method is based on a more complicated interpolating idea and thus can save the number of sensors required. For the dispatch problem, we have proposed two distributed schemes. The first scheme assumes that sensors have the knowledge of all target locations in the sensing field and these sensors can then compete with each other for moving toward their closest locations. The second scheme relaxes the above assumption by making sensors deriving the target locations on their own, according to several known locations and the patterns in our placement methods. Therefore, the server only needs to generate several seed locations in the beginning, and then sensors can construct the remaining part of the network in a distributed manner.

- We have designed an efficient dispatch method for mobile sensors to prolong their lifetimes. In particular, we consider a hybrid sensor network consisting of static and mobile sensors, where the former is deployed to detect events, while the latter equipped with more resources such as sensing capability and computation power is dispatched to the event locations to conduct more advanced analysis. In this dissertation, we investigate how to efficiently dispatch mobile sensors to visit these event locations with the purpose of maximizing the *system lifetime*, which is defined as the time duration until there are some event locations that cannot be reached by any mobile sensor due to lack of energy. We have pointed out that simply maximizing the total remaining energy of mobile sensors in each one-round dispatch cannot guarantee to maximize the system lifetime since some mobile sensors may early exhaust their energy and thus burden other still alive ones, which results in shortening the system lifetime. Based on the aforementioned observation, we have proposed an efficient dispatch method that takes the *load-balance* issue into consideration when scheduling mobile sensors to visit event locations. Our dispatch method is general in which the numbers of event locations and mobile sensors can be arbitrary. When the number of event locations is no larger than that of mobile sensors, we transform the dispatch problem to a maximum matching problem in a weighted bipartite graph. However, instead of finding a matching with a maximum edge weight, we use a *preference list* and a *bound* to select the matching, where the former helps assign an event location with a suitable mobile sensor, while the latter avoids selecting edges with extreme weights so that loads among mobile sensors can be balanced. When the number of event locations is larger than that of mobile sensors, we have developed efficient clustering schemes to group event locations into clusters, whose number is equal to the number of mobile sensors, so that the above matching approach can be adopted. When a mobile sensor is assigned with a cluster, it can use the traveling-salesman approximate algorithm to reach all event locations in that cluster. Simulation results have shown that by exploring the load balancing of mobile sensors, our proposed dispatch method can prolong the system lifetime.

- To help data aggregators manage the messages reported from sensors, we have developed two packet fair scheduling algorithms for a *single-rate wireless environment* and a *multi-rate wireless environment*. In a single-rate environment, sensors transmit their packets in a fixed rate and the wireless channels may switch between a good state and a bad state. Packet transmissions in a good state are assumed to be able to succeed but to fail in a bad state. With the above assumptions, we have proposed a *Traffic-Dependent wireless Fair Queuing (TD-FQ)* algorithm that take traffic types of flows into account when scheduling packets. TD-FQ benefits real-time flows by giving them higher priorities over non-real-time flows. In this way, the queuing delays of real-time flows can be alleviated. However, TD-FQ still guarantees the fairness among flows so that non-real-time flows will not be starved under the scheduling policy of TD-FQ. In this dissertation, we have further considered a multi-rate environment in which sensors can adopt different modulation techniques to transmit their packets under different channel conditions. Specifically, sensors can transmit their packets using a higher rate when the channel is good but they can still use a lower rate for transmission when the channel becomes bad. With this assumption, we have proposed a *Multi-Rate wireless Fair Queuing (MR-FQ)* algorithm. MR-FQ addresses both the *service fairness* and *time fairness* issues arisen from such a multi-rate environment. MR-FQ adjusts a flow's transmission rate according to the flow's channel condition and its lagging degree. In particular, a flow is allowed to transmit using a lower rate only if it is lagging to a certain degree. Besides, the more serious a flow is lagging, the lower rate the flow is allowed to use. Such differentiation can take care of both service fairness and time fairness. In this way, MR-FQ not only maintains the fairness properties and bounded delays of flows, but also improves the overall system performance. In this dissertation, we have analytically derived the fairness properties and delay bounds of the proposed TD-FQ and MR-FQ algorithms. Simulation results have also been presented to verify their effectiveness.

- We have designed and implemented a mobile sensor platform, called the *integrated mobile surveillance and wireless sensor (iMouse) system* in the dissertation. The objective of this iMouse system is to study the feasibility of integrating the context-aware capability of mobile wireless sensor networks into surveillance systems. The iMouse system consists of a large number of static sensors and a small number of more powerful mobile sensors. The former is used to monitor the environment while the latter can move to event locations to conduct more in-depth analyses. In the iMouse system, each mobile sensor is equipped with a processing platform, a Lego car, a Mote, a WebCam, and an IEEE 802.11 WLAN card, so that they can move to the event locations, exchange messages with other sensors, take snapshots of event scenes, and transmit pictures and analyzed data to the remote server. In this way, the iMouse system combines mobile wireless sensor networks with the surveillance systems and thus the overheads of traditional visual surveillance systems can be reduced because the real critical information can be retrieved and proactively sent to the users. The iMouse system is thus a mobile, context-aware surveillance system. We have demonstrated our current prototyping system for home/office security applications.

1.3 Organization of the Dissertation

The rest of this dissertation is organized as follows. In Chapter 2, we surveys the studies in the literature that address the deployment, dispatch, and packet-scheduling issues. Besides, we also survey several works related to the design of mobile sensor platforms. In Chapter 3, we solve the sensor deployment problem in a sensing field with boundaries and obstacles. In Chapter 4, we discuss how to deploy a sensor network for multi-level coverage by mobile sensors. In Chapter 5, we investigate how to efficiently dispatch mobile sensors in a hybrid sensor network. In Chapter 6, we propose the two wireless packet fair scheduling algorithms, TD-FQ and MR-FQ, for data aggregators. In Chapter 7, we report the design and prototyping experience of our mobile sensor platform, the iMouse system. In Chapter 8, we give the conclusions and future directions of this dissertation.

Chapter 2

Preliminaries

In this chapter, we first survey some studies in the literature that address the issues of deployment, dispatch, and packet-scheduling, followed by the works that address the design of mobile sensor platforms.

2.1 Deployment Issue

In this section, we discuss the related works addressing the deployment issue. We first study three relevant computational geometric problems, and then survey several works that discuss how to place sensors to form a network. Finally, we survey some works that discuss how to self-deploy a network by mobile sensors.

2.1.1 Related Computation Geometric Problems

Art Gallery Problem

The *art gallery problem* [58, 102] was first proposed by *Victor Klee* in 1973. This problem asks what is the minimum number of guards that would be needed at the corners of an art gallery to guarantee that every point in the gallery is always in sight of at least one guard, assuming that a guard can watch a point as long as a line-of-sight exists. It has been proven that at least $\lfloor \frac{n}{3} \rfloor$ guards are needed if the gallery can be modeled by a *simple* polygon[1] with n vertices on a 2D plane [22, 39]. This can be done by triangulating the polygon P and then painting the vertices of P with three different colors in such a way that each triangle has a vertex of each color. Since there are n vertices, there is one color c that is used for at most $\lfloor \frac{n}{3} \rfloor$ vertices. Then we can place guards on these vertices painted

[1]A simple polygon is a polygon with the constraint that non-consecutive edges do not intersect.

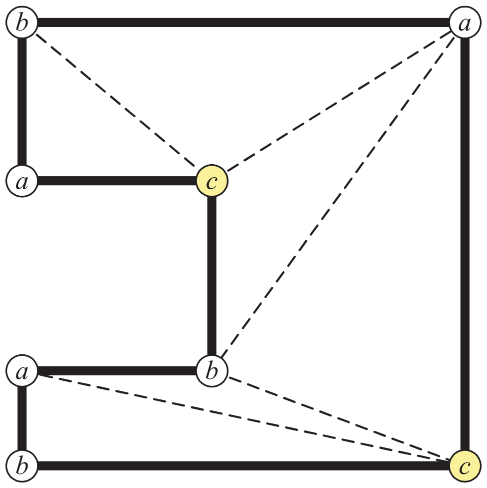


Figure 2.1: An example of triangulating a polygon and painting its vertices. Guards will be placed in the vertices marked with c.

with color c to guarantee that every point inside P is watched by at least one guard. Fig. 2.1 gives an example, where we can place two guards on vertices marked with c.

In the literature, there are many variations of the art gallery problem. For example, the studies [7, 90] consider how to use the minimum number of *mobile guards* to monitor a polygon, where a mobile guard is allowed to patrol either along edges of the polygon or along straight lines wholly contained within the polygon. In [52, 98], the issue of how to place guards in a polygon with holes is addressed. Although both the art gallery problem and the sensor deployment problem ask how to place the least number of guards/sensors to monitor a region, the solutions to the art gallery problem cannot be applied to the sensor deployment problem because of the following two reasons. First, guards in the art gallery problem are assumed to have an infinite viewpoint unless there is an obstacle. On the contrary, the monitoring ranges of sensors in fact are limited by their sensing distances. Second, the sensor deployment problem needs to consider the connectivity between sensors while this issue is ignored in the art gallery problem.

Circle Packing Problem

The *circle packing problem* [34, 108] asks how to make an arrangement of circles inside a given boundary such that no two circles overlap and some (or all) of them are mutually tangent. For example, Fig. 2.2 shows two possible packing ways of equal circles in a plane. In the literature, there are myriad of variations of the circle packing problem. References [18, 112] discuss how to make the densest packing of equal circles in a square. Different shapes of the bounded space have also been studied, for example, packing n equal circles in a circular region [40, 42, 46, 81], or packing circles in an equilateral triangle [45, 83]. The issue of packing different sizes of circles has also been addressed in [41].

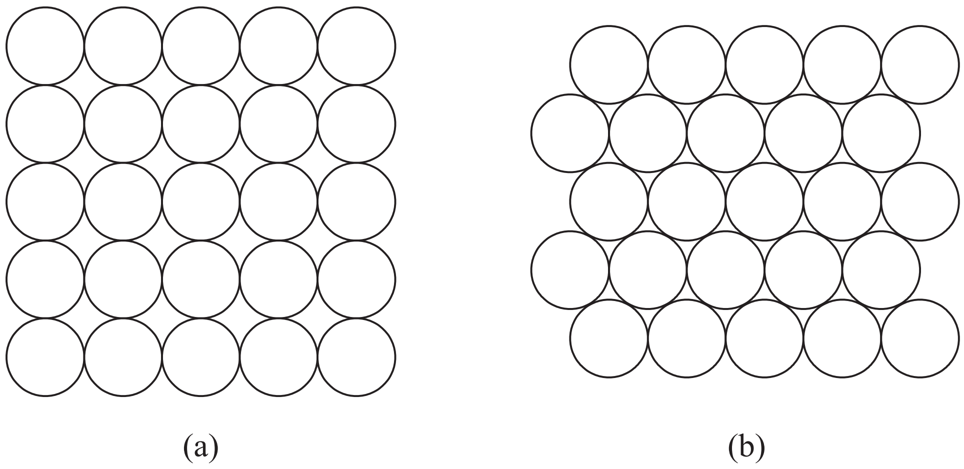


Figure 2.2: Two possible methods to pack equal circles in a 2D plane: (a) the square packing method and (b) the hexagonal packing method.

The circle packing problem can be applied to several applications such as estimation the size of a wire bundle in electric systems [110]. However, it cannot solve the sensor deployment problem because of their different objectives. In particular, the objective of circle packing problem is to find out the densest arrangement of circles in a region while the sensor deployment problem attempts to use the fewest number of sensors to cover a region. In addition, the arrangement of circles in the circle packing problem will leave a lot of uncovered holes in the region, which violates the coverage requirement of the sensor deployment problem.

Circle Covering Problem

The third relevant problem in computational geometry is the *circle covering problem* [121]. This problem asks how to arrange equal circles in a polygon such that this polygon can be fully covered by these circles. Given the number of circles, the objective is to minimize the radius of a circle. This issue has been widely studied for covering a unit square in the works [51, 82, 89]. Reference [51] discusses how to optimally arrange with less than or equal to five circles and seven circles. For example, the optimal coverings of five and seven circles are given in Fig. 2.3. The solutions have been extended to six, eight, and eleven circles in [82]. Table 2.1 summarizes the minimum radius r_{min} to cover a unit square with $n \leq 30$ equal circles. The details of arrangements can be found in [89].

The circle covering problem has also been addressed in different shapes of polygons such as equilateral triangles [80]. However, it is very limited to apply these solutions to the sensor deployment problem because the sensing distance of sensors has to be restricted to certain constants.

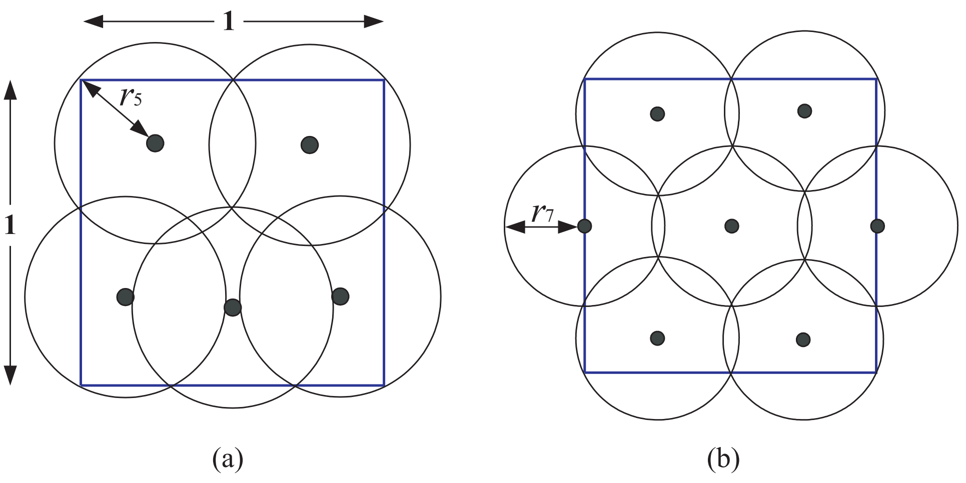


Figure 2.3: Two examples of optimal coverings with n equal circles in a unit square: (a) $n = 5$, where the minimum radius r_5 is approximate to 0.32616 and (b) $n = 7$, where the minimum radius r_7 is approximate to 0.27429.

n	r_{min}	n	r_{min}	n	r_{min}
1	$0.707106781\cdots$	11	$0.212516016\cdots$	21	$0.148953789\cdots$
2	$0.559016994\cdots$	12	$0.202275889\cdots$	22	$0.143693177\cdots$
3	$0.503891109\cdots$	13	$0.194312371\cdots$	23	$0.141244822\cdots$
4	$0.353553390\cdots$	14	$0.185510547\cdots$	24	$0.138302883\cdots$
5	$0.326160584\cdots$	15	$0.179661759\cdots$	25	$0.133548706\cdots$
6	$0.298727062\cdots$	16	$0.169427051\cdots$	26	$0.131746875\cdots$
7	$0.274291885\cdots$	17	$0.165680929\cdots$	27	$0.128633534\cdots$
8	$0.260300105\cdots$	18	$0.160639663\cdots$	28	$0.127317553\cdots$
9	$0.230636927\cdots$	19	$0.157841981\cdots$	29	$0.125553507\cdots$
10	$0.218233512\cdots$	20	$0.152246811\cdots$	30	$0.122036881\cdots$

Table 2.1: The minimum radius r_{min} to cover a unit square by n equal circles reported in [89].

2.1.2 Placements of Wireless Sensor Networks

Since the solutions to the traditional computational geometric problems cannot be directly applied to the sensor deployment problem, many researchers have proposed their methods to solve the sensor deployment problem by determining the locations to place sensors.

In the literature, several studies [19, 29, 71] consider to model the sensing field as *grid points* and then determine the placement of sensors on some of these grid points to satisfy certain coverage requirements. Reference [19] considers that there are two types of sensors with different costs and sensing ranges to be deployed. The objective is to find an assignment of sensors to grid points such that every grid point is covered by at least $k \geq 1$ sensors and the total cost of the sensors can be minimized. For example, Fig. 2.4 illustrates the minimum-cost sensor placements for a 4×4 sensing field with different values of k, where a type-A sensor has a cost of \$150 and one unit of sensing range and a type-B sensor has a cost of \$200 and two units of sensing range. The work [19] then solves this problem by using a linear programming. In [29], sensors are also placed at grid points. It is assumed that a sensor at grid point i can detect a target at grid point j with a probability $p_{ij} = e^{-\alpha|\overline{ij}|}$, where the parameter α is used to model the quality of the sensor and the rate at which its detection probability diminishes with distance. Clearly, $p_{ij} = 1$ if $i = j$ and $p_{ij} = p_{ji}$ because of symmetry. However, $p_{ij} = 0$ if there is an obstacle in the line of sight from grid point i to grid point j. Based on the above conditions, the objective of the sensor placement is to determine the minimum number of sensors and their locations such that every grid point is covered with a minimum confidence level. To achieve this goal, the work [29] proposes a greedy algorithm that determines the best placement of sensors one by one, where each time the placement of the sensor can help reduce the most miss probabilities of grid points. Such iteration will be repeated until either a present upper limit on the number of sensors is reached, or sufficient coverage of the grid points is achieved. The work in [71] discusses how to place sensors in a grid-based sensing field to achieve complete coverage and discrimination. In [71], a *power vector* is defined for each grid point i to indicate the set of sensors that can cover i. The sensing field is called *completely covered* if any grid point in the sensor field can be covered by at least one sensor. In a completely covered sensing field, if every grid point can be identified by a unique power vector, the sensing field is said to be *completely discriminated*. Fig. 2.5 gives an example, where the power vector of grid point 8 is $(0, 1, 0, 0, 1, 0)$ corresponding to whether the sensors 2, 5, 6, 10, 11, and 14 cover the grid point 8. Note that a completely discriminated sensing field may not be constructed because of resource limitations. Thus, the work [71] formulates such sensor placement problem as a combinatorial optimization problem for minimizing the maximum *distance error*, which is defined as the Euclidean distance between two indistinguishable[2] grid points, when the complete discrimination is impossible.

The aforementioned schemes limit sensors to be placed on predefined grid points to achieve desired coverage. Several studies relax such limitation and propose non-grid placements of sensors. The work in [62] assumes that sensors have a sensing/communication distance of r. Sensors are then placed strip by strip, where a *strip* is a string of sensors placed along a line such that the distance between any two adjacent sensors is r. Clearly, the sensors in a strip will form a connected component. To cover a 2D plane, two adjacent

[2]Two grid points are called *indistinguishable* if their power vectors are the same.

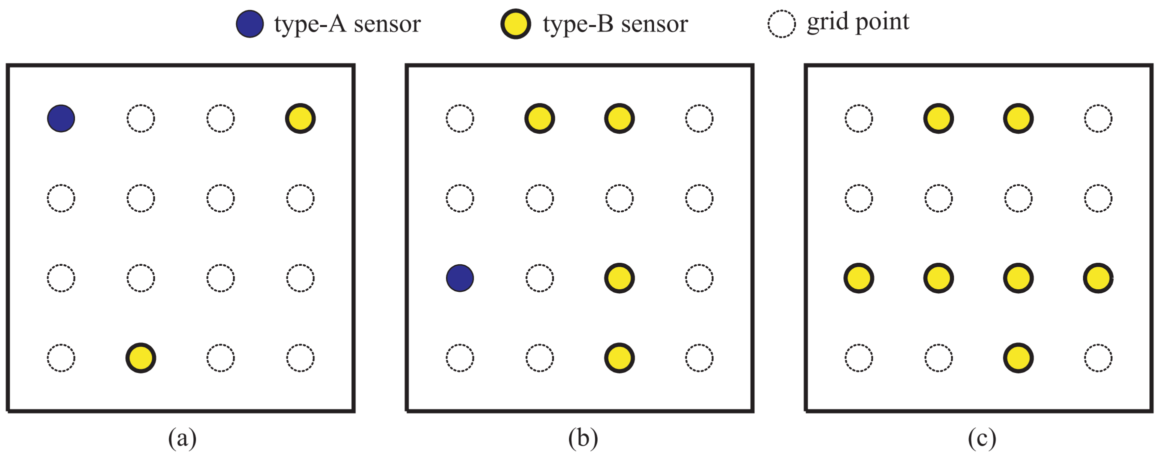


Figure 2.4: The minimum-cost sensor placements in the sensing field with a 4×4 grid-size under different values of coverage level k: (a) $k = 1$ and cost $=$ \$550, (b) $k = 2$ and cost $=$ \$950, and (c) $k = 3$ and cost $=$ \$1400.

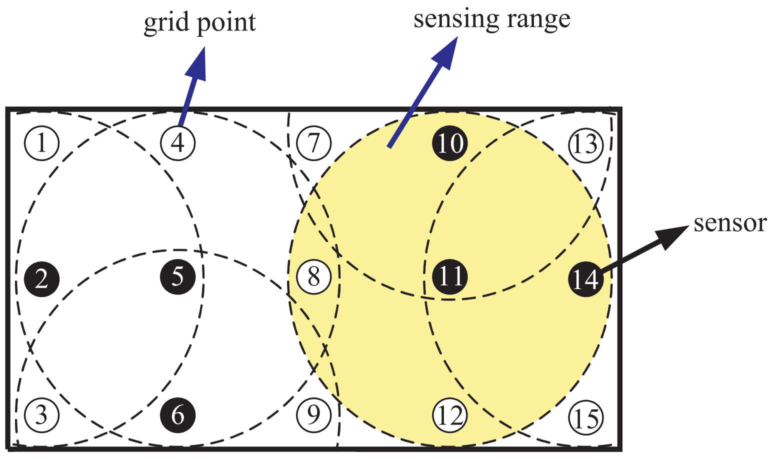


Figure 2.5: A complete covered and discriminated sensing field. There are six sensors placed on the grid points 2, 5, 6, 10, 11, and 14.

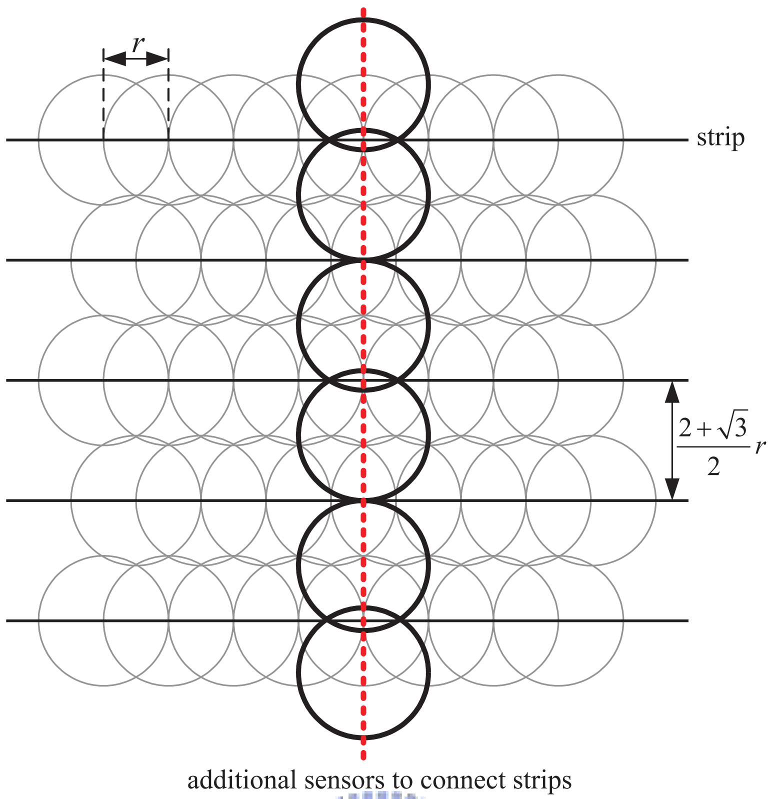


Figure 2.6: The sensor placement with a strip-by-strip fashion proposed in [62].

strip is separated by a distance of $\frac{2+\sqrt{3}}{2}r$ and shifted by a distance of $\frac{r}{2}$. Note that additional sensors should be placed between two adjacent strips to guarantee the connectivity of the network. Fig. 2.6 presents an example.

Reference [111] proposes a sensor placement strategy that can result in multi-level coverage, where every point in the sensing field can be covered by at least k sensors and k is a given parameter. It is assumed that the communication distance is no less than twice of the sensing distance, so that the coverage of the sensing field can also guarantee the connectivity of the network [128]. In [111], placements with $k \leq 3$ are addressed and the traditional hexagon placement is adopted. When $k = 1$, each sensor is separated by a distance of $\sqrt{3}r_s$, where r_s is the sensing distance. When $k = 2$, two 1-covered placements are combined to form a 2-covered placement. When $k = 3$, the distance between two adjacent sensors is shrunk to r to generate 3-coverage overlaps. A placement with $k > 3$ can be generated by combing these three basic placements. Fig. 2.7 shows such placements when $k \leq 3$.

2.1.3 Self-deployments with Mobile Sensors

Instead of deciding where to place sensors, some works consider to make sensors deploy by themselves. These works assume that sensors have mobile capability and can obtain

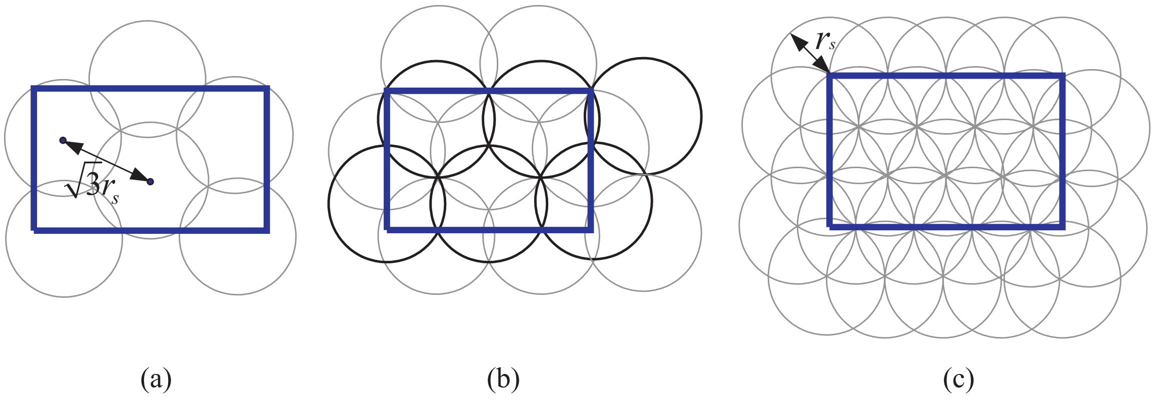


Figure 2.7: The k-covered placement scheme proposed in [111]: (a) $k = 1$, (b) $k = 2$, and (c) $k = 3$.

their physical locations by the global positioning system (GPS) [53] or other schemes [13]. They consider to randomly deploy sensors initially and then to move sensors with different strategies to enhance the coverage of the sensing field.

In [132], the concept of *virtual force* is introduced to move sensors. Each sensor s_i is assumed to be exerted by three kinds of forces: the attractive (positive) force $\overrightarrow{F_{iA}}$ by the areas of preferential coverage, the repulsive (negative) force $\overrightarrow{F_{iR}}$ by all obstacles, and the force $\overrightarrow{F_{ij}}$ between sensors s_i and s_j. Therefore, the total force $\overrightarrow{F_i}$ on sensor s_i can be expressed as

$$\overrightarrow{F_i} = \overrightarrow{F_{iA}} + \overrightarrow{F_{iR}} + \sum_{j=1, j\neq i}^{n} \overrightarrow{F_{ij}},$$

where n is the number of sensors. The force $\overrightarrow{F_{ij}}$ is expressed in polar coordinate notation (r, θ), where r is the magnitude and θ is the orientation of the vector $\overrightarrow{F_{ij}}$. Thus, $\overrightarrow{F_{ij}}$ can be derived as

$$\overrightarrow{F_{ij}} = \begin{cases} (w_A(d_{ij} - d_{th}), \theta_{ij}), & \text{if } d_{ij} > d_{th} \\ 0, & \text{if } d_{ij} = d_{th} \\ (w_R(\frac{1}{d_{ij}}), \theta_{ij} + \pi), & \text{otherwise} \end{cases},$$

where d_{ij} is the Euclidean distance between sensors s_i and s_j, d_{th} is a threshold distance, θ_{ij} is the orientation (angle) of a line segment from s_i to s_j, and w_A/w_R is a measure of the attractive/repulsive force. Note that the threshold d_{th} controls how close sensors get to each other. Fig. 2.8 shows an example, where there are four sensors and d_{th} is set to d_{12}. In Fig. 2.8, s_2 exerts no force on s_1, s_3 exerts an attractive force $\overrightarrow{F_{13}}$ on s_1, and s_4 exerts a repulsive force $\overrightarrow{F_{14}}$ on s_1 because $d_{12} = d_{th}$, $d_{13} > d_{th}$, and $d_{14} < d_{th}$, respectively. The sensor s_1 is thus moved by the compound force $\overrightarrow{F_1}$.

The idea of repulsive force is also adopted in the work [50], where each sensor is treated as an electron and will be repulsed by other sensors. The force corresponding to higher

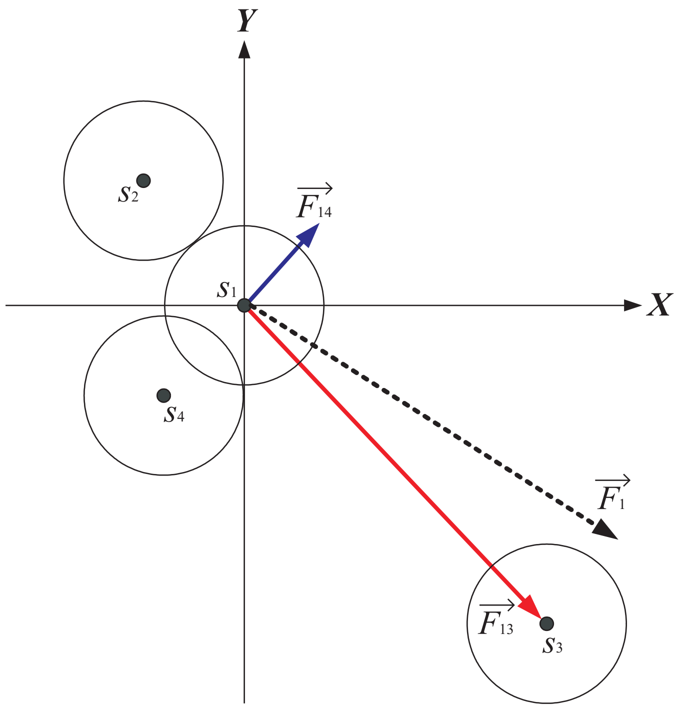


Figure 2.8: An example of virtual forces with four sensors.

local density is greater than the force corresponding to lower local density, and the force from a node that is closer is greater than that from a node that is farther. In [50], a force function $f(\cdot)$ is defined to satisfy the following conditions:

- $f(d_{ij}) \geq f(d_{ik})$ if $d_{ij} \leq d_{ik}$, where d_{ij} is the distance between sensors i and j.
- $f(0^+) = f_{max}$, where f_{max} is the maximum force.
- $f(d) = 0$ if d is larger than the communication distance of a sensor.

Sensors will be moved step by step. In each step k, the force on sensor i by another sensor j in its communication range is calculated to be a repulsive force as

$$f_k^{i,j} = \frac{D_k^i}{\mu^2}(r_c|p_k^i - p_k^j|)\frac{p_k^j - p_k^i}{|p_k^j - p_k^i|},$$

where D_k^i is the local density of sensor i at step k, μ is the expected density (after the final deployment), r_c is the communication distance of a sensor, and p_k^i is the location of sensor i at step k. When a sensor moves less than a threshold distance for a specified time period, it is considered to have reached the stable status and thus stops moving.

Reference [118] uses the Voronoi diagram to find out potential *coverage holes* and then moves sensors to cover these holes. Given a set of nodes on a 2D plane, the *Voronoi diagram* [6, 32] is formed from perpendicular bisectors of lines that connect two neighboring nodes, as shown in Fig. 2.9. The Voronoi diagram can represent the proximity information

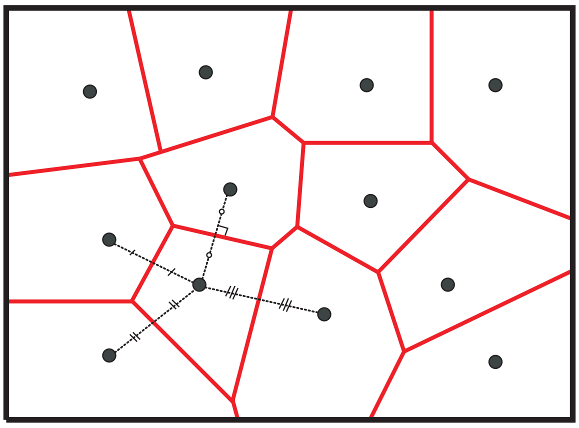


Figure 2.9: The Voronoi diagram.

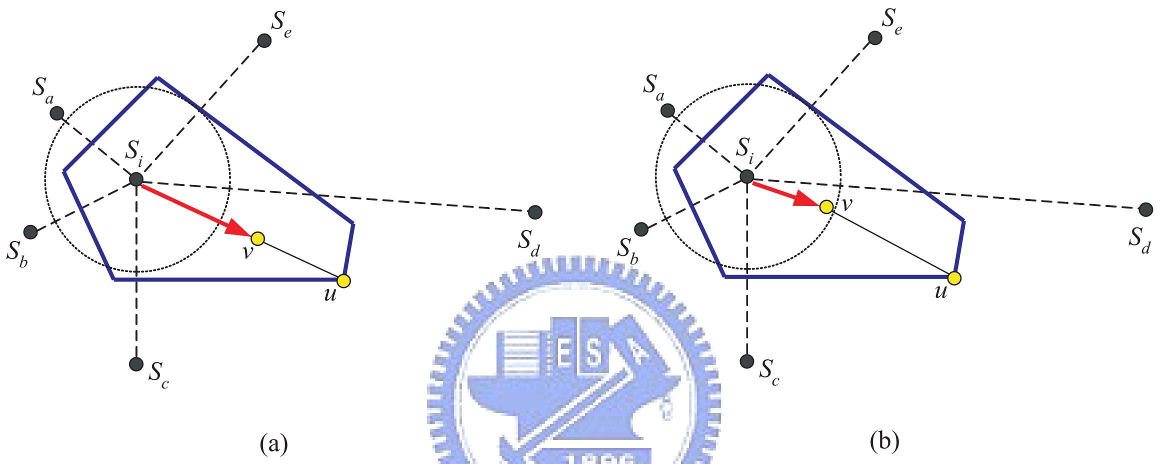


Figure 2.10: Two examples of (a) the VOR strategy and (b) the Minimax strategy.

about a set of geometric nodes. Every point in a Voronoi polygon is closer to the node in this polygon than to any other node. In [118], after initial deployment, each sensor will calculate the Voronoi polygon with the location information of its neighbors. Then three strategies, naming *vector-based* (VEC), *Voronoi-based* (VOR), and *Minimax* algorithms, are proposed to move sensors. The VEC strategy also adopts the idea of virtual force. Two sensors s_i and s_j will be pushed to move a distance of $(d_{avg} - d_{ij})/2$ away from each other, where d_{avg} is the average distance between two sensors when the sensors are evenly distributed in the sensing field. However, if a sensor can completely cover its Voronoi polygon, it should not be moved. In the VOR strategy, if a sensor detects the existence of a coverage hole, it will move toward its farthest Voronoi vertex. Fig. 2.10(a) gives an example, where point u is the farthest Voronoi vertex of sensor s_i. s_i will move to the point v, where $|\overline{uv}|$ is equal to the sensing distance of a sensor. In the Minimax strategy, a sensor s_i will choose the point v inside its Voronoi polygon whose distance to the farthest Voronoi vertex u is minimized to move, as shown in Fig. 2.10(b).

Some studies use a grid structure to perform sensor relocation. They partition the sensing field into grids and then move sensors from high-density grids to low-density grids. In [119], the *grid-quorum* and *cascaded movement* schemes are proposed to find the redundant sensors and to relocate sensors to the target locations, respectively. In the grid-quorum scheme, the sensing field is divided into grids and each grid elects a grid-head to maintain the grid's information. The grid-head of a high-density grid will send an advertisement message in its row, while the grid-head of a low-density grid will send a request message to its column. Due to the grid structure, there will be a grid that can receive both the advertisement and request messages. Fig. 2.11 presents an example, where the grid $(1, 2)$ has redundant sensors and the grid $(3, 0)$ needs extra sensors. The grid $(3, 2)$ will receive both advertisement and request messages from these two grids. By adopting this scheme, only $O(\sqrt{N})$ messages are needed to be exchanged, where N is the number of grids. After identifying redundant sensors and requesting grids, sensors are moved using the cascaded movement rather than the direct movement to prevent these moving sensors from wasting too much energy, as shown in Fig. 2.12. In the cascaded movement scheme, each sensor s_i is associated with a tolerable delay T_i, during which its successor must move to its original location. Thus, a sensor s_j can become a successor of s_i if

$$\frac{d_{ji}}{\nu} - (t_i - t_j) \leq T_i,$$

where ν is the moving speed of a sensor and t_i is the departure time of sensor s_i. In [119], the cascading sensor nodes will be selected to minimize the different between the total energy consumption and the minimum remaining energy of moving sensors.

Reference [123] also partitions the sensing field into grids. The objective is to balance the number of sensors in each grid. This can be done by two rounds of balancing, one for each grid and one for each column. Fig. 2.13 shows an example.

2.2 Dispatch Issue

In this section, we survey several works that address the dispatch issue of mobile wireless sensor networks. These works consider to move sensors to improve the network topology or to perform other missions. The work in [9] discuss how to move mobile nodes to result in a stronger network. Specifically, the objective is to form a biconnected network such that the total moving distance of nodes can be minimized. A network is called *biconnected* if it is not partitioned after removing any of its nodes. Each such node is referred to as a *cutvertex*. Fig. 2.14 shows an example, where node c is a cutvertex and we can form a biconnected network by moving node a. Based on this observation, the work [9] proposes a *block movement* algorithm to remove all the cutvertices from the network by moving

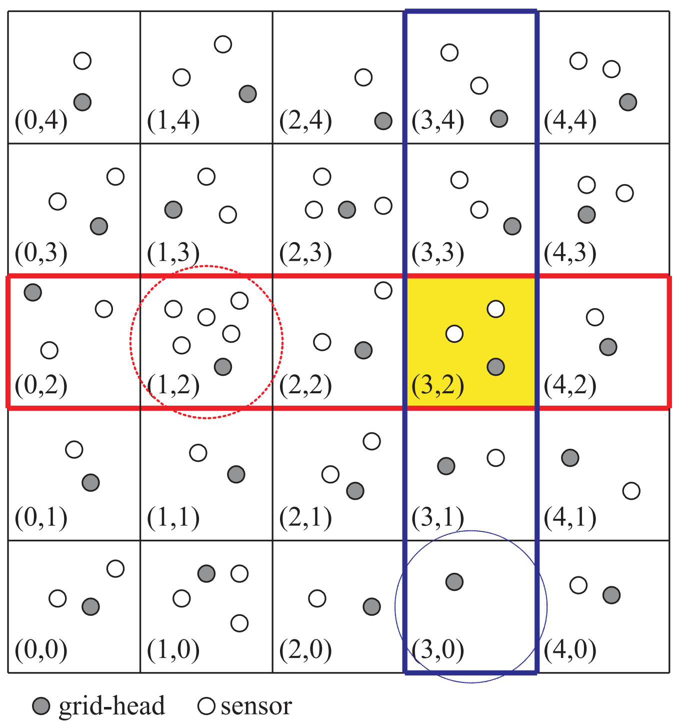


Figure 2.11: An examples of the grid-quorum.

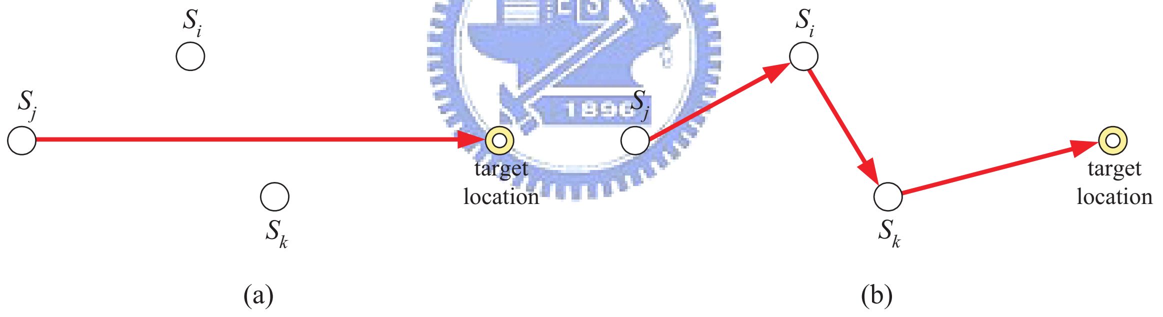


Figure 2.12: Examples of (a) the direct movement and (b) the cascade movement.

3	7	5	18	32
3	5	14	7	1
5	3	9	20	8
17	11	6	32	9
10	27	23	5	20

(a)

13	13	13	13	13
6	6	6	6	6
9	9	9	9	9
15	15	15	15	15
17	17	17	17	17

(b)

12	12	12	12	12
12	12	12	12	12
12	12	12	12	12
12	12	12	12	12
12	12	12	12	12

(c)

Figure 2.13: An ideal example of balancing the numbers of sensors in each grid: (a) the initial case, (b) balancing the numbers of sensors in each row, and (c) balancing the numbers of sensors in each column.

certain of nodes to new locations. In particular, given a graph G that describes the network topology, the biconnected components (also called *blocks*) of G are first identified along with its cutvertices, and then the graph G can be translated to a *block tree* [30]. Note that a block can have between 0 and n nodes, where n is the total number of nodes in the network. If two cutvertices are directly connected by an edge, the corresponding block contains no nodes. In the block tree, we can select the block with the maximum number of nodes as its root. Fig. 2.15(a) and (b) give an example, where five blocks (including the empty block B_3) and two cutvertices C_1 and C_2 are identified. The block B_1 is selected as the root of the block tree and the blocks B_2, B_4, and B_5 are the leaf nodes. The block movement algorithm will execute the following iteration until the graph becomes biconnected:

- Move the leaf block toward the nearest node in its parent block, by the distance that exactly one new edge appears.
- If the parent block contains no node, we move the leaf block to the upstream cutvertex of its parent block.

Fig. 2.15(c) illustrates an example, where the block B_2 will move toward the node v of the block B_1 and the blocks B_4 and B_5 will move toward the cutvertex C_2 because their parent block B_3 is empty. The final network topology is shown in Fig. 2.15(d).

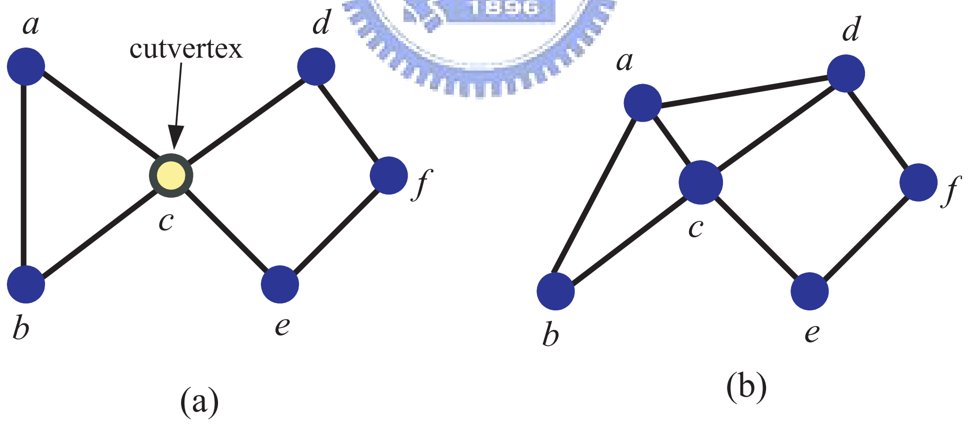


Figure 2.14: Achieving a biconnected network by node movement: (a) the initial network topology and (b) the network becomes biconnected after moving node a.

The works in [33, 131] consider to add mobile sensors into an existing stationary sensor network to adjust the network topology so that the coverage and connectivity of the original network can be improved. In [33], there are M mobile sensors added to improve the network topology. For these M mobile sensors, λM of them are used to increase the sensing coverage while the rest $(1-\lambda)M$ of them are used to improve the routing and connectivity, where $0 < \lambda < 1$. To increase the sensing coverage, two schemes

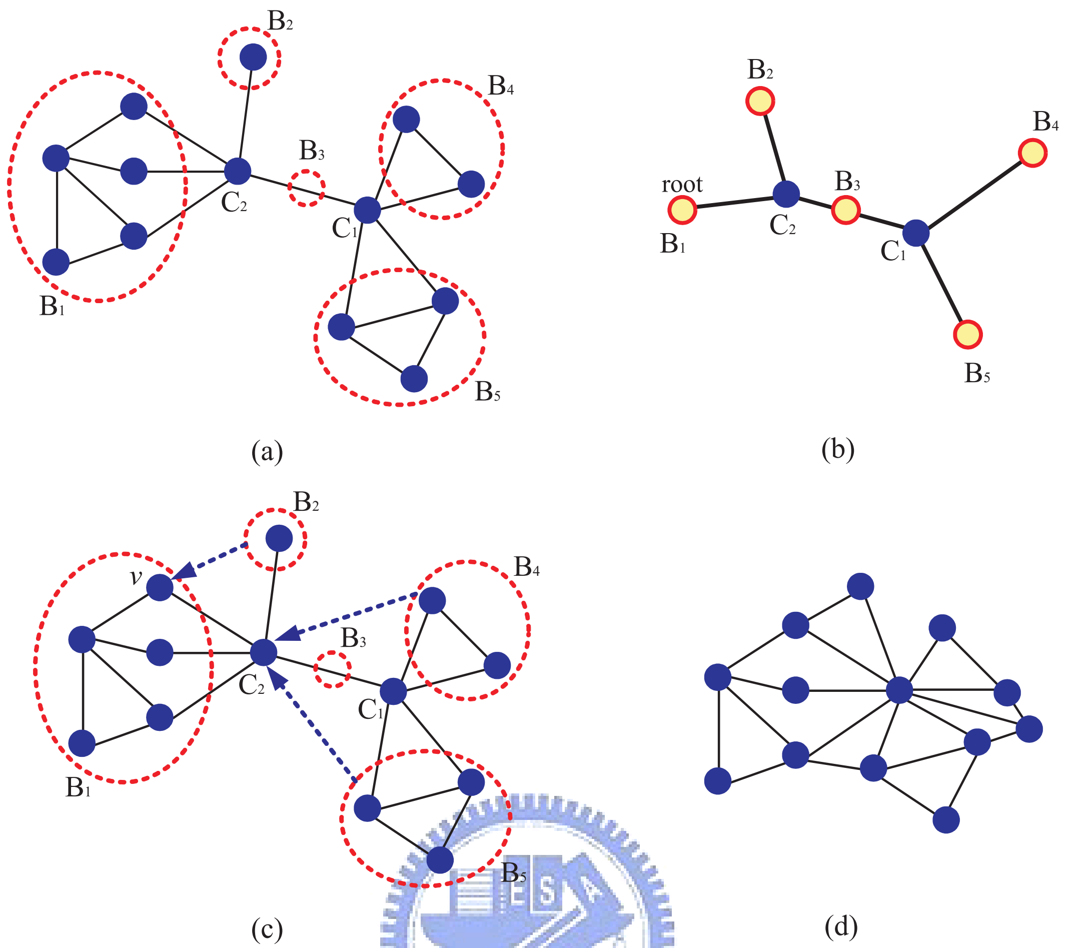


Figure 2.15: An example of the block movement algorithm: (a) decomposition of a network into cutvertices and biconnected components, (b) the corresponding block tree, (c) the movement of blocks, and (d) the final network topology after block movement.

are proposed. The first scheme is to partition the sensing field into small equal-sized *cells*. The sink then counts the number of stationary sensors in each cell. Cells with the number of sensors less than the average are considered as *under-covered* and the mobile sensors will be dispatched to these under-covered cells. Another scheme is to place a mobile sensor on the center point of the line segment connecting the pair of sensors with the longest distance. This procedure will continue until all λM mobile sensors are dispatched. After dispatching λM mobile sensors, the remaining mobile sensors are used to improve the routing and connectivity of the network. In particular, when the energy of a sensor s is under a certain threshold, it will check if its neighbors can help cover the area after the sensor s exhausts its energy. If most of the area cannot be covered, the sensor s will broadcast a Help message to search for mobile sensors. A mobile sensor receives such message will move to s's location to perform packet relay and sensing task. However, it is possible that the network is partitioned into several isolated components due to sensors' failure. In this case, the mobile sensors have to move *actively* to connect these components. In particular, if the sink does not receive any message from a certain area for a long time, it will ask nearby mobile sensors to roam to that area for maintaining connectivity of the network.

The work in [131] considers that randomly-deployed stationary sensors are not distributed evenly so that these sensors may form several isolated groups, each called an *island*. These islands cannot connect with each other so that we have to add mobile sensors to connect them, as shown in Fig. 2.16. Thus, a three-step algorithm is proposed in [131] to dispatch mobile sensors to fix the partitioned network. The first step is to search islands by grouping stationary sensors. The second step is to calculate the least number of mobile sensors $M_{A,B}$ needed to connect two islands A and B:

$$M_{A,B} = \lceil \frac{d_{A,B}}{r_c} - 1 \rceil,$$

where $d_{A,B} = min\{distance(s,t)\}, s \in A, t \in B$ is the shortest distance between two islands A and B. The last step uses a dynamic programming to find the optimal set of islands to be connected. Specifically, let C_G be the coverage of an island G and $W(G,m)$ is the optimal island set starts from island G, using m mobile sensors. We can obtain that

$$W(G,m) = max\{C_{G \cup H} + W(G \cup H, m - M_{G,H})\},$$

where H is an island to be connected with G. The above equation can be divided down and thus solved by the dynamic programming. Note that for each island G, if the left x mobile sensors cannot allow it to connect any other island, we will let $W(G,x) = 0$. After identifying the optimal set of islands, mobile sensors will be placed along the lines that connect these islands and the concept of virtual force is adopted to make these mobile sensors to achieve maximum coverage. Fig. 2.16 gives an example.

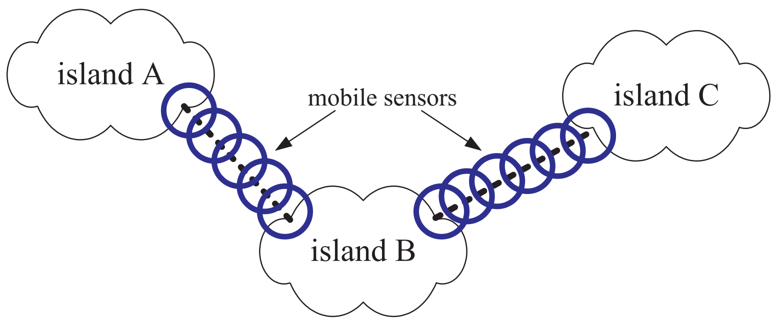


Figure 2.16: The isolated islands are connected by mobile sensors.

Some studies consider to move sensors close to the event locations so that there can be more sensors (possibly with stronger capabilities) used to detect the events. References [14, 15] consider a sensor network consisting of only mobile sensors and suggest to move more sensors close to the event locations while still maintaining the network coverage. Two event-based movement schemes are proposed to aggregate sensors near the event locations. The first one allows each sensor to react to an event by moving according to a function $f(d)$, where d is the distance between the sensor and the event. Note that this function should satisfy the following three criteria:

$$\begin{aligned} & 0 \leq f(d) \leq d, \quad \forall d, \\ & f(\infty) = 0, \text{ and} \\ & f(d_1) - f(d_2) < d_1 - d_2, \quad \forall d_1 > d_2. \end{aligned}$$

In [15], $f(d)$ is suggested to be set to the form as $\alpha d^{\beta} e^{-\gamma d}$ for values of parameters α, β, and γ such that $\alpha e^{-\gamma d}(\beta d^{\beta-1} - \gamma d^{\beta}) > 1$ for all possible d. The second scheme maintains a small amount of event history to analyze the distribution of events. Each sensor calculates the cumulative distribution (CDF) of the scaled event distribution, which describes what fraction of the sensors should be to the left of each point. The sensor then finds the point in the scaled CDF corresponding to its initial position, and moves to that location. In order to maintain the network coverage, the Voronoi diagram is constructed to determine whether a sensor can move or not. Specifically, each sensor calculates the motion of every other sensor and uses this information to compute its Voronoi polygon after each movement. If any part of the sensor's Voronoi polygon is farther away than its sensing distance, it knows that no other sensor is closer to this point, and it should not move away from that point.

The work in [117] considers a hybrid sensor network comprising of static and mobile sensors, where the former is used to monitor the environment and to guide mobile sensors to event locations, while the latter is equipped with more resources such as sensing capability and computation power and can move to event locations for providing advanced detection. When the static sensors detect an event, they will elect a leader [35, 94] to broadcast a weight request packet (WREQ) to the network. A mobile sensor receives such WREQ packet will reply its weight, which is calculated as follows:

$$\text{weight} = \frac{\text{Voronoi\_area} \times \text{distance}}{\text{energy}},$$

where Voronoi_area is used to measure the impact (i.e., the size of coverage hole) when this mobile sensor leaves. After receiving the replies from multiple mobile sensors, the leader will select the one with the minimum weight and then broadcasts an advertisement packet (ADV) to build up the navigation field for guiding the mobile sensor. In particular, the leader sets the highest credit value C_1 for itself and inserts C_1 into the ADV packet. Other static sensors receiving such ADV packet will set their credit values as C_2, where $C_2 < C_1$. Then only those static sensors that have relayed the reply message from the mobile sensor will broadcast an ADV packet containing C_2. Other static sensors receiving such ADV packet set their credit values as C_3, where $C_3 < C_2$. This procedure will be repeated until the ADV packet reaches the mobile sensor. Then the mobile sensor will move toward the static sensor with higher credit values. Fig. 2.17 illustrates an example.

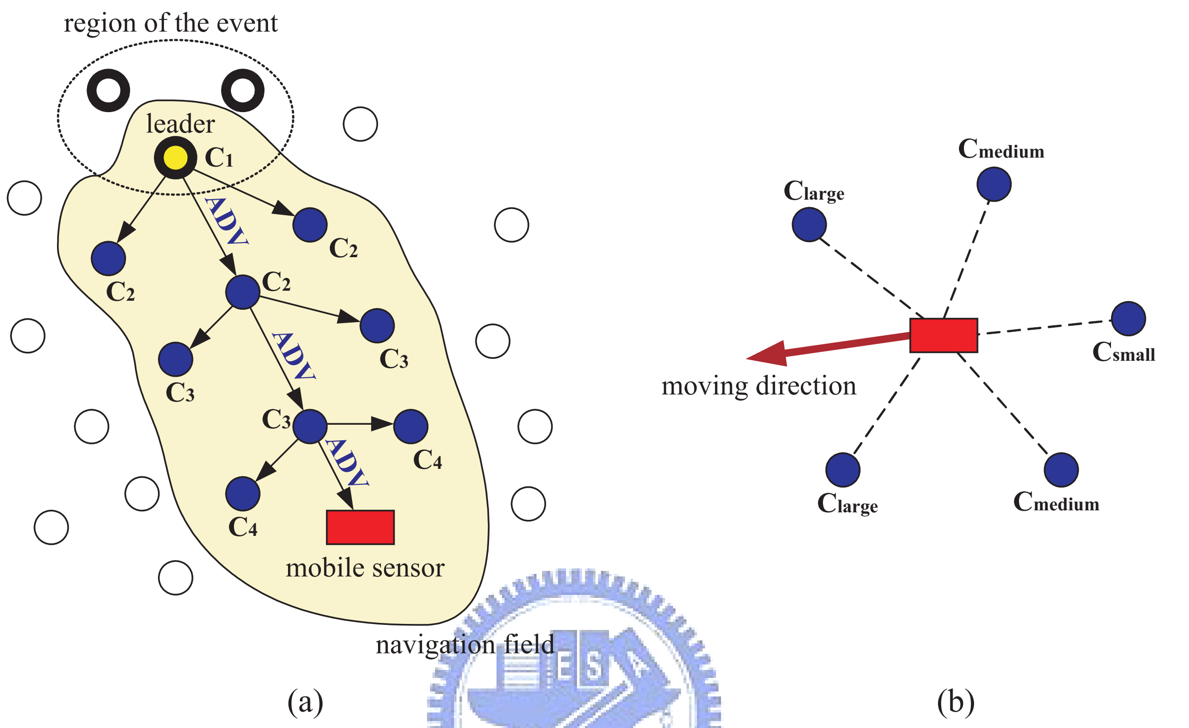


Figure 2.17: Navigation of the mobile sensor: (a) building up the navigation field and (b) calculating the moving distance of the mobile sensor.

2.3 Packet-scheduling Issue

In this section, we survey several packet fair scheduling algorithms designed for wireless networks, which are classified into four categories. The first category of algorithms adopt the rules of wire-line scheduling algorithms, but they will swap the channel access between those backlogged flows that encounter channel errors and those backlogged flows that do not. The second category of algorithms provide explicit mechanisms to compensate those flows who receive less services due to channel errors. The third category of algorithms dynamically adjust a flow's weight according to its channel's and queue's conditions. The scheduling algorithms in the last category consider some application-specific issues, such as traffic types of flows and handoff between base stations.

2.3.1 Algorithms with Error-free Reference Models

The scheduling algorithms in this category define an *ideal fair service model* that assumes no channel errors. This error-free service model usually adopts the rules of a wire-line scheduling algorithm, and it can provide a reference for channel allocation. Specifically, the scheduler can know how much service that a flow should receive in an ideal case (i.e., no channel errors occur), and then compensate those flows that receive less services because of channel errors. In this section, we discuss two representative algorithms, *IWFQ (Idealized Wireless Fair Queuing)* [75] and *CIF-Q (Channel-condition Independent Fair Queuing)* [86].

Idealized Wireless Fair Queuing (IWFQ)

IWFQ adopts a wire-line scheduling algorithm *WFQ (Weighted Fair Queueing)* [127] as its reference error-free model to help determine the service sequence of packet flows. A flow is said to be *leading*, *lagging*, or *in sync* if its queue size in the real error-prone system is smaller than, larger than, or the same as the queue size in the reference error-free system, respectively. In IWFQ, each packet p_n^i of a flow i is associated with a *start tag* s_n^i and a *finish tag* f_n^i, which are computed followed the rules in WFQ:

$$s_n^i = \max\{v(t_n^i), f_{n-1}^i\}, \tag{2.1}$$

$$f_n^i = s_n^i + \frac{L_n^i}{w_i}, \tag{2.2}$$

where t_n^i is the physical arrival time of the packet p_n^i, $v(t_n^i)$ is the system virtual time at time t_n^i defined in IWFQ, f_{n-1}^i is the finish tag of the previous packet, L_n^i is the packet size of p_n^i, and w_i is the weight of flow i.

In IWFQ, the scheduler always selects the packet with the smallest finish tag for service. However, when the selected packet cannot be transmitted due to channel errors, the scheduler will try to select the packet with the next smallest finish tag for service. This process will continue until the scheduler finds a packet that can be transmitted. Since the finish tags of packets will never be changed, a flow that loses its services due to errors may accumulate many packets with small finish tags after exiting from errors. In order to prevent these packets from starving other flows, IWFQ sets two bounds:

- *Bounds on lag*: The total lag kept by all flows is bounded by a constant of B bits. A lagging flow i with a weight w_i is allowed to compensate no more than b_i bit, where

$$b_i = B \times \frac{w_i}{\sum_{k \in F} w_k},$$

and F is the set of all flows. In particular, for any lagging flow i, the total length of those packets with finish tags less than the current system virtual time is bounded by b_i bits. All other such packets will be deleted.

- *Bounds on lead*: A leading flow is allowed to keep its lead by a maximum length of l_i bits. Specifically, for each leading flow i, if the start tag s^i_{HOL} of the head-of-line packet is greater than the current system virtual time $v(t)$ by more than l_i/w_i, then its start and finish tags will be updated as:

$$\begin{aligned} s^i_{HOL} &= v(t) + \frac{l_i}{w_i}, \\ f^i_{HOL} &= s^i_{HOL} + \frac{L^i_{HOL}}{w_i}, \end{aligned}$$

  where L^i_{HOL} is the length of the head-of-line packet. The tags of other packets in flow i will be also updated by Eqs. (2.1) and (2.2) accordingly.

The IWFQ scheduler checks every queue after it transmits a packet. If the scheduler finds that a queue violates the above bounding principles, it adjusts that queue following the aforementioned two rules to guarantee the delay bound and throughput of flows.

Channel-condition Independent Fair Queuing (CIF-Q)

In CIF-Q, four principles that a wireless packet scheduling algorithm should obey are proposed:

1. *Delay and throughput guarantees*: Delay bound and throughput for error-free flows should be guaranteed and should not be affected by other flows that encounter channel errors.

2. *Long-term fairness*: After a flow exits from channel errors, as long as it has enough service demands, it can get back all of its “lost” services during channel errors.

3. *Short-term fairness*: The difference between the normalized services received by any two error-free flows that are continuously backlogged and are in the same state (leading, lagging, or satisfied) during any time interval should be bounded.

4. *Graceful degradation*: A leading backlogged flow should be guaranteed to receive at least a minimum fraction of its services when it is error-free.

The proposed CIF-Q algorithm is developed according to the aforementioned principles. CIF-Q adopts the *start-time fair queuing (SFQ)* [44] as its reference error-free system to determine the service order. It maintains a *virtual time* v_i for each flow i to keep track of the normalized services that the flow receives in the reference error-free system, and a parameter lag_i is used to record the difference between the services that are actually received by flow i in the real error-prone system and the services that are expected to be received by flow i in the reference error-free system. When a flow i is selected and transmits a packet with length l_p, its v_i is then increased by l_p/w_i. However, when the selected flow i cannot transmit packets due to channel errors, the scheduler will choose another flow j to serve. At this time, v_i is still increased by l_p/w_i, but lag_i and lag_j are increased and decreased by l_p, respectively. Therefore, according to lag_i, we can determine the state of a flow i. In particular, flow i is called *leading* if $lag_i < 0$, called *lagging* if $lag_i > 0$, and called *satisfied* otherwise. Note that $\sum_{i \in A} lag_i$ should be always zero since CIF-Q is work-conserving, where A is the set of all active flows. In addition, to satisfy the graceful degradation principle, an extra parameter α is used to define the minimal fraction of services that can be received by a leading flow. Specifically, a leading flow i is allowed to continue receiving services at an average rate of $\alpha \cdot r_i$ so that it will not be starved.

The scheduling process of CIF-Q is summarized as follows:

- The scheduler always selects a flow i with the smallest v_i for service. The selected flow i cannot transmit if either one of the following cases occurs:
  - Flow i suffers from channel errors.
  - Flow i is leading and has received more than $\alpha \cdot r_i$ amount of services.

  In the latter case, flow i has to give up its transmission opportunity to other lagging flows.

- When a flow i is selected but it cannot transmit packets, the scheduler will choose another flow j to serve. Such services obtained from flow i are called *additional services.*

- When there are additional services available, lagging flows always have a higher priority to receive such services. The additional services are then distributed among lagging flows proportional to their service weights.

- If no lagging flow can receive the additional services provided by an error flow, the scheduler distributes these services among leading and satisfied flows proportional to their weights.

2.3.2 *Algorithms with Explicit Compensation Mechanisms*

This category of algorithms use a server (or several counters) to compensate those flows that receive less services due to channel errors. When a flow suffers from errors and cannot be served, the scheduler will record the lost services of that flow in the server. After this flow recovers from channel errors, the scheduler will compensate this flow with the amount of services recorded in the server. Here we introduce two representative schemes, *SBFA (Server Based Fairness Approach)* [95] and *Havana* [96].

Server Based Fairness Approach (SBFA)

The basic idea of SBFA is to reserve part of network's bandwidth for compensation. This reserved part can be achieved by the *long-term fairness server (LTFS)*. In particular, LTFS is a *virtual data flow* created for the compensation purpose. Although it is a virtual flow, LTFS still shares the bandwidth and is scheduled by the system.

In SBFA, each data flow (except for LTFS) is associated with a *packet queue (PQ)* and a *slot queue (SQ)*. When a packet of flow i arrives, it is inserted into PQ_i. At the same time, an abstract entity called *slot* is created in SQ_i with a tag S_i. In SBFA, any wire-line packet scheduling algorithm can be used, and the scheduling policy is applied to the slot queues. Specifically, when a slot in SQ_i is selected, the head-of-line packet in the corresponding PQ_i is transmitted as long as flow i is error-free. In this case, both the slot in SQ_i and the packet in PQ_i will be removed. However, if flow i suffers from channel errors, only the slot in SQ_i is removed but it will be inserted into LTFS for future compensation.

Here we use an example to demonstrate how the SBFA works, as shown in Fig. 2.18. The scheduling policy used in this example adopts a simple round-robin scheme. Consider that there are two flows i and j in the system. Flow i suffers from channel errors during time $[0, 1]$, but flow j is always error-free. At time 0 (refer to Fig. 2.18(a)), the scheduler turns to serve flow i, but it finds that flow i suffers from channel errors. Thus, the scheduler dequeues a slot from SQ_i and inserts the slot into LTFS. Then the scheduler changes to serve flow j and transmits P_j1 in PQ_j of flow j. Also, a slot in SQ_j is removed (refer to Fig. 2.18(b)). At time 1, by the round-robin policy, the scheduler will turn to serve flow j. This will succeed, so the first entry in both PQ_j and SQ_j will be removed. At time 2, the scheduler will turn to serve LTFS. Since it finds that the head-of-line slot in LTFS has a tag of S_i, the scheduler goes to serve flow i (refer to Fig. 2.18(c)). At this time, only the packet P_i1 in PQ_i is removed. Note that if flow i is still in error state at time 2, the slot S_i will be kept in LTFS and the scheduler will turn to select another error-free flow to serve. In this way, flow i can still be compensated at next time.

In summary, LTFS provides a mechanism to record which flows should be compensated in the future. Note that there can be more than one LTFS in SBFA, and the number of LTFS depends on the requirements of the flows sharing the wireless link. SBFA suggests that it is better to assign flows with similar requirements to the same LTFS.

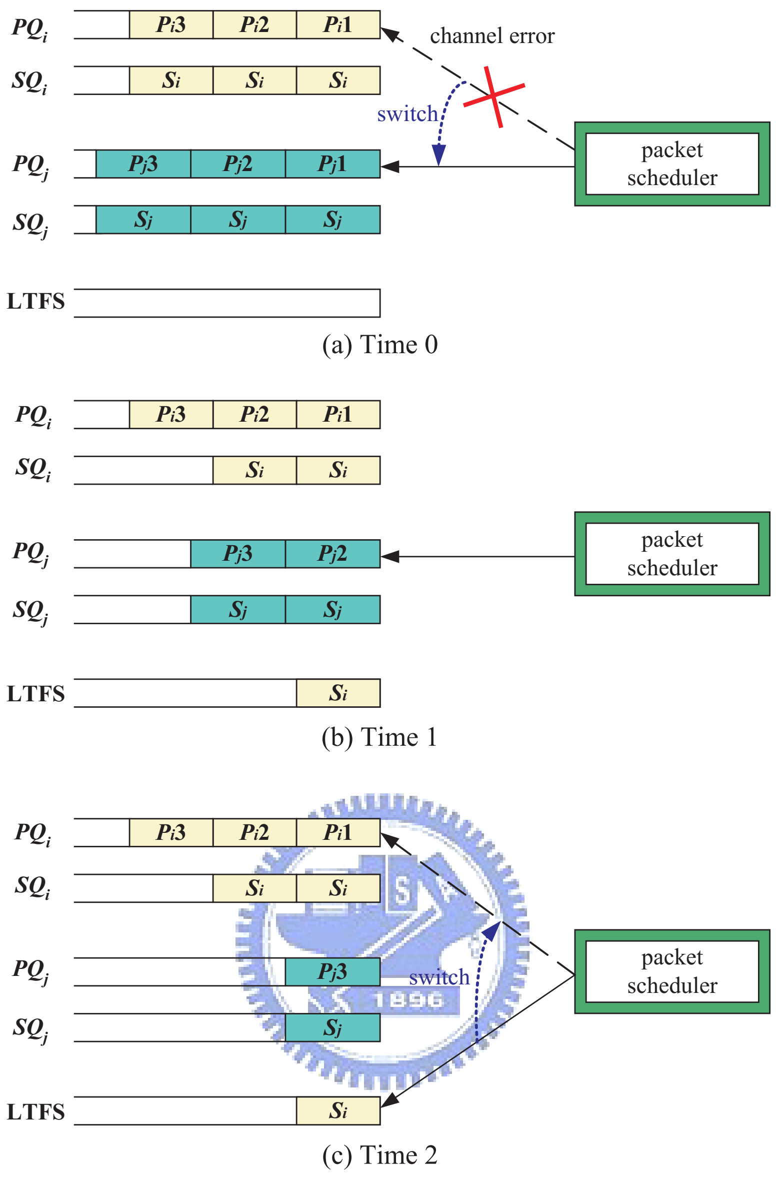


Figure 2.18: An example of SBFA using the round-robin scheduling policy.

Havana

In Havana, a modified version of *Deficit Round Robin (DRR)* [105] scheduler is used to credit and compensate flows in response to potential unfairness experienced by mobile stations due to different channel conditions. In DRR, every flow has its own queue. The scheduler serves queues in a round-robin fashion. During each round, the number of packets served in each queue i is determined by two parameters: *quantum* (Q_i) and *deficit counter* (DC_i). Specifically, Q_i accounts for the quota given to flow i in each round, while DC_i keeps track of the credit already accumulated by flow i. A *round* is the process of visiting each queue once by the scheduler. At the beginning of each round, a credit of Q_i is added to DC_i. When the scheduler turns to serve flow i, the scheduler will transmit the first k packets of queue i such that their total size does not exceed DC_i and k is the largest possible integer. After the transmission, DC_i is deducted by the exact amount of data being transmitted. If the scheduler finds that there is no backlogged packet in flow i, DC_i is reset to zero.

To compensate the flows that lose their services due to channel errors, Havana maintains an extra *compensation counter* (CC_i) to keep track of the amount of lost services of each flow i. If the scheduler defers transmission of an error flow i, the corresponding DC_i is decreased by the Q_i, but CC_i is increased by Q_i. At the beginning of each round, an amount of $\alpha_i \cdot CC_i$ credits is added to DC_i, and CC_i is decreased by the same amount, where $0 < \alpha_i \leq 1$. The value of α_i represents the fraction of the compensation credit actually given to flow i during this round.

Here we give an example to demonstrate how the Havana works, as shown in Fig. 2.19. There are two flows i and j in the system with the parameters $Q_i = 80$, $Q_j = 120$, $\alpha_i = 1/2$, and $\alpha_j = 1/4$. The status of flows (i.e., their queues and parameters) at the beginning of round k is shown in Fig. 2.19(a). The number in each packet represents its size. Assume that flow j is error-free but flow i suffers from channel errors at round k. Then in round k, the transmission of flow i is deferred and that of flow j will proceed. Fig. 2.19(b) shows flows' status at the end of round k. DC_i is decreased by $Q_i(= 80)$ and CC_i is increased by Q_i. DC_j is decreased by the size of the head-of-line packet $(= 100)$ and CC_j is unchanged. At the beginning of round $k + 1$ (refer to Fig. 2.19(c)), flow i's DC_i is increased by the value of $Q_i + \alpha_i \cdot CC_i = 80 + \frac{1}{2} \times 260 = 210$, and CC_i is decreased by $\alpha_i \cdot CC_i = \frac{1}{2} \times 260 = 130$. The similar updates will be also applied to flow j.

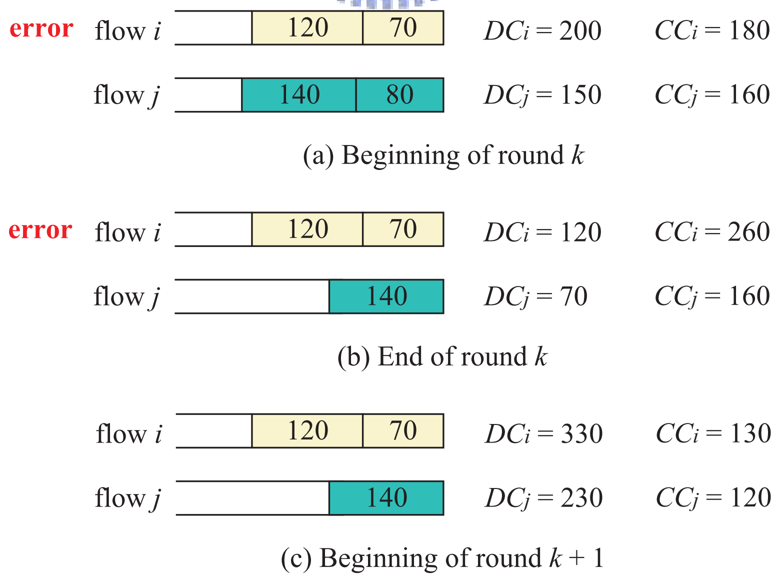


Figure 2.19: An example of the Havana scheduling, where $Q_i = 80$, $Q_j = 120$, $\alpha_i = 1/2$, and $\alpha_j = 1/4$.

Note that to avoid unbounded compensation, upper limits are imposed on deficit counters. In particular, the credit accumulated in DC_i should not exceed an upper bound DC_i^{max} at any time.

2.3.3 Algorithms with Weight Adjustment Mechanisms

Up to now, the scheduling algorithms that we have reviewed all consider that the weights of flows are "static". Specifically, once the weight of a flow is decided, it will not be changed during the whole scheduling process. Several algorithms take a different viewpoint. In particular, they dynamically adjust flows' weights according to different conditions. In this section, we discuss two such algorithms, *Effort-Limit Fair (ELF)* [36] and *Fair Queuing with Adaptive Compensation (AC-FQ)* [120].

Effort-Limit Fair (ELF)

ELF is proposed to extend the wire-line scheduling algorithm WFQ with a dynamic weight-adjustment mechanism. In particular, the ELF scheduler will adjust each flow's weight in response to the error rate of that flow, up to a maximum weight defined by that flow's *power factor*, which is provided by the admission control module (refer to Fig. 2.20). Specifically, assume that there are n flows sharing a wireless channel. Each flow i maintains a weight w_i and a power factor p_i. Also, let flow i experience an error rate e_i, where $0 \leq e_i < 1$. Then ELF defines the adjusted weight a_i of flow i as

$$a_i = \min\left\{\frac{w_i}{1-e_i}, p_i \times w_i\right\}.$$

Intuitively, for an error flow i, the ELF scheduler scales its weight w_i to make up its loss due to channel errors, but we limit the adjustment to a factor p_i. By this dynamic weight adjustment, ELF can control the scheduler's behavior in the presence of channel errors.

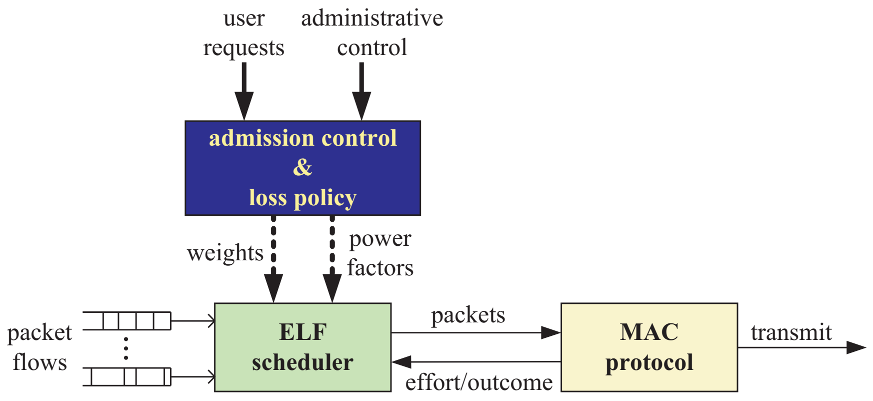


Figure 2.20: The architecture of the ELF scheduler.

Fair Queuing with Adaptive Compensation (AC-FQ)

AC-FQ is proposed based on the observation in [59], which indicates that CIF-Q may cause the services of error-free flows becoming deteriorated. AC-FQ modifies the way how leading flows give up their services for compensating other lagging flows in CIF-Q. The design architecture of AC-FQ is shown in Fig. 2.21. In AC-FQ, when a flow becomes leading, it is divided into two sub-flows: *transmission flow* and *compensation flow*. A leading flow can receive a fraction of services through its transmission flow, and provide part of bandwidth to other lagging flows through its compensation flow. To achieve this, each flow i maintains three different weights w_i, w_i^H, and w_i^L, where $w_i > w_i^H > w_i^L$. When a leading flow i has a longer queue length, the weights of its transmission flow and compensation flow are set to w_i^H and $w_i - w_i^H$, respectively. In this way, flow i can keep more services and its queuing delay can be thus alleviated. Otherwise, the weights of its transmission flow and compensation flow are set to w_i^L and $w_i - w_i^L$, respectively. In this case, flow i has to give up more services to compensate other lagging flows. In summary, AC-FQ alleviates the queuing delays of leading flows by dynamically changing their weights according to their queue lengths.

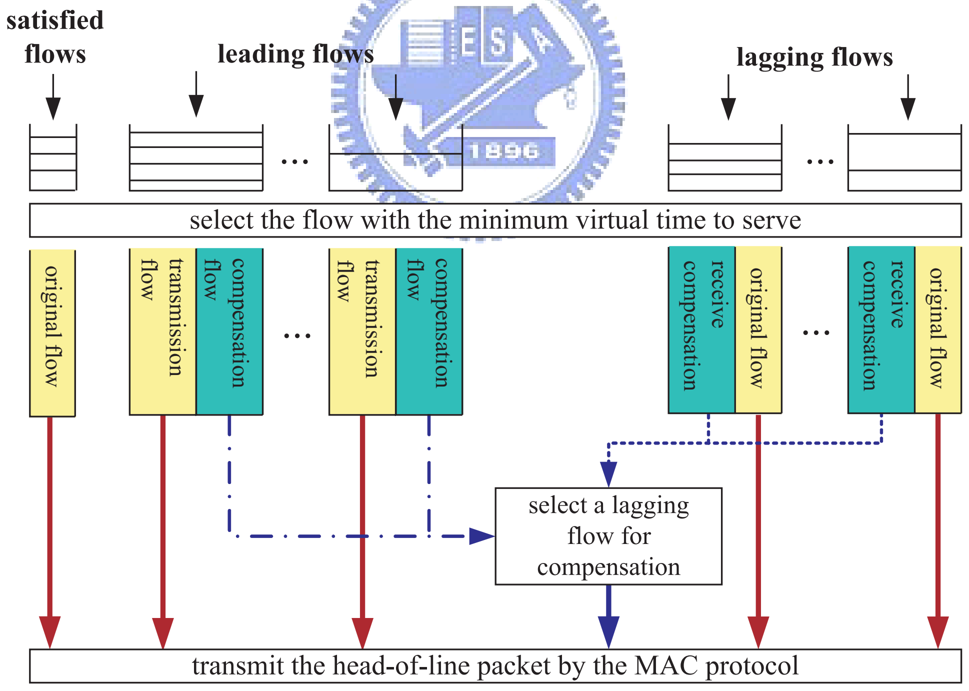


Figure 2.21: The design architecture of AC-FQ.

2.3.4 *Algorithms that Consider Traffic Types of Flows*

An inherent limitation of wire-line scheduling algorithms is that the delay observed by the packets of a flow is tightly coupled with the fraction of the channel given to the flow (i.e., their weights) among all backlogged flows [76]. Thus, those wireless scheduling algorithms that modify wire-line scheduling algorithms without considering traffic types of flows may also suffer from this limitation. Therefore, several scheduling algorithms have been proposed to take traffic types of flows into consideration and thus decouple the delay and bandwidth requirements of flows. Here we discuss two related scheduling algorithms, *Wireless Fair Service (WFS)* [76] and *Handoff Compensation Scheme (HCS)* [69], in this section.

Wireless Fair Service (WFS)

WFS assigns each flow i with a *rate weight* r_i and a *delay weight* d_i, and associates each packet p_k^i with a *start tag* $S(p_k^i)$ and a *finish tag* $F(p_k^i)$:

$$\begin{aligned} S(p_k^i) &= \max\left\{V(A(p_k^i)), S(p_{k-1}^i) + \frac{L_{k-1}^i}{r_i}\right\}, \\ F(p_k^i) &= S(p_k^i) + \frac{L_k^i}{d_i}, \end{aligned}$$

where L_k^i is the size of the kth packet p_k^i of a flow i, $A(p_k^i)$ is the arrival time of p_k^i, and $V(t)$ is the virtual time at time t. Essentially, flow i is drained into the scheduler according to the rate weight r_i, but served according to the delay weight d_i. In addition, at each time t, the WFS scheduler transmits the packet with the minimum finish tag among those packets whose start tags are not greater than $V(t) + \varrho$, where ϱ provides a measure of the number of packets over which the scheduler can locally switch the schedule of packets without disrupting the long-term rate. By varying ϱ, WFS can provide different levels of delay-bandwidth decoupling.

Handoff Compensation Scheme (HCS)

In HCS, a compensation scheme to handle the channel errors and handoff situation is proposed. This compensation scheme is designed based on a *priority swapping* and a *compensation flow*. A flow experiencing channel errors will defer its transmission by swapping the resources to other error-free flows, and this flow will receive additional resources when it recovers from channel errors. A compensation flow is a virtual flow with a pre-assigned weight w_{CS}. It participates in the scheduling process and redistributes its resources to active flows according to the state of flows. In HCs, all flows are classified into four groups: *poorer*, *poor*, *normal*, and *rich*. A flow is said to be *poor* if it receives less services than it expects. When a poor flow transmitting real-time traffic is about to drop packets due to long waiting time, this flow is changed to a *poorer* flow. When there are additional services available, the poorer flows always have the highest priority to receive such services, so the queuing delays of real-time flows can be alleviated. For the handoff flows, HCS uses the compensation flow to give/receive weights to/from handoff flows. Specifically, when a flow i handoffs into a cell, it can be given a weight w_i from the compensation flow if $\alpha \cdot w_{CS} \geq w_i$, where $0 < \alpha \leq 1$. On the contrary, when flow i leaves a cell, it returns its service share to the scheduler by increasing w_{CS} with its weight w_i, so other flows can share these unused services left by flow i.

2.4 *Implementations of Mobile Sensor Platforms*

In this section, we survey some works that discuss the implementations of mobile sensor platforms. One similar issue, *mobile robots*, has been widely discussed in the field of robotics [37, 55, 67, 68]. These works consider to make the robot move automatically in a space, in which there may be static obstacles or humans (i.e., moving obstacles). To achieve this goal, they install cameras on walls or the robot to identify obstacles or humans in the environment, and this information will be used to guide the robot to detour around these obstacles. The visual information obtained from cameras can also help locate the robot. For example, the robot proposed in [68] has four color bars, each painted with three different colors, as shown in Fig. 2.22. By identifying the color bars and the angle θ_{wr}, the system can obtain the current location and direction of the robot and thus can help navigate the robot.

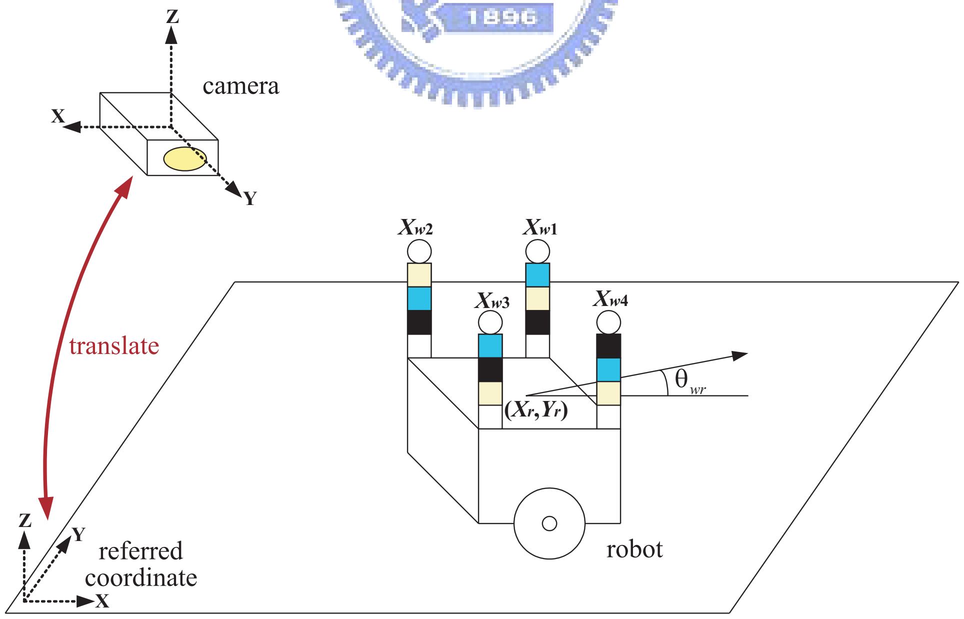


Figure 2.22: The system architecture of mobile robots proposed in [68].

Several studies have proposed their design and implementations of mobile sensors. In [101], Sharp et al. implement the *pursuer-evader problem* [28] by using a hybrid sensor network. In the pursuer-evader problem, there are two mobile nodes, called *pursuer* and *evader*, in the field. The evader node will roam in the field while the pursuer node attempts to intercept the evader node by using the information obtained from the environment. In [101], the pursuer and evader nodes are implemented by ground robots, which are essentially mobile off-road laptops equipped with GPS devices. Each robot runs Linux on a 266 MHz Pentium2 CPU with 128 MB of RAM, and is equipped with an IEEE 802.11 wireless radio, all-terrain off-road tires, a motor-controller subsystem, and a GPS device. In addition, a static sensor network is deployed in the sensing field to obtain the information of the evader node. Such information will be sent to the pursuer node so that it can catch up with the evader node. More details of this system can be found in [101].

Mobile Emulab [61], a robotic wireless and sensor network testbed, is proposed for researchers to evaluate their proposed mobility-related network protocols, applications, and systems. In this testbed, robots carry single-board computers and sensors (so that they can be treated as "mobile sensors") through a fixed indoor field, all running the user's selected software. In real-time, interactively or driven by a script, remote users can position and control these robots. Mobile Emulab is composed of three components, including video cameras, robots (or mobile sensors), and fixed sensors. The cameras are mounted on the ceiling to overlook the sensing field and to track robots. The mobile sensors, which use Acroname Garcia robots [2] as their mobile platforms, carry an XScale-based Stargate [26] small computer running Linux, a MICA2 mote [24], and an IEEE 802.11b WLAN card for computing, communicating, and controlling purposes. Finally, the fixed sensors are used to detect events in the sensing field. The system architecture of Mobile Emulab is illustrated in Fig. 2.23. The user can control robots or query data through the *robot backend*. When a robot motion request is received, the *robotd* subsystem will translate it into low-level motion commands to control the robots. Besides, the *visiond* subsystem will periodically track the positions of robots through the cameras and feedback this information to the robotd subsystem so that it can revise the motions of robots.

Reference [103] considers a hybrid sensor network that consists of both static and mobile sensors, where the mobile sensors can move to the locations of some static sensors to perform certain applications such as node replacement. Unlike the previous work, a mobile sensor is navigated by the received signal strength rather than by the location information or image processing technologies. Specifically, the sink will construct a routing path to each static sensor by flooding a message, as shown in Fig. 2.24. These routing paths are used as navigation paths of mobile sensors. In particular, when the mobile sensor wants to visit a static sensor s, the sensor s will send a packet containing its ID to the sink. Static sensors on the routing path will also add their IDs in the packet so that a navigation path can be constructed. Fig. 2.24 shows an example, where sensor a is the target node and a packet containing IDs *abcde* that indicates the reverse navigation path

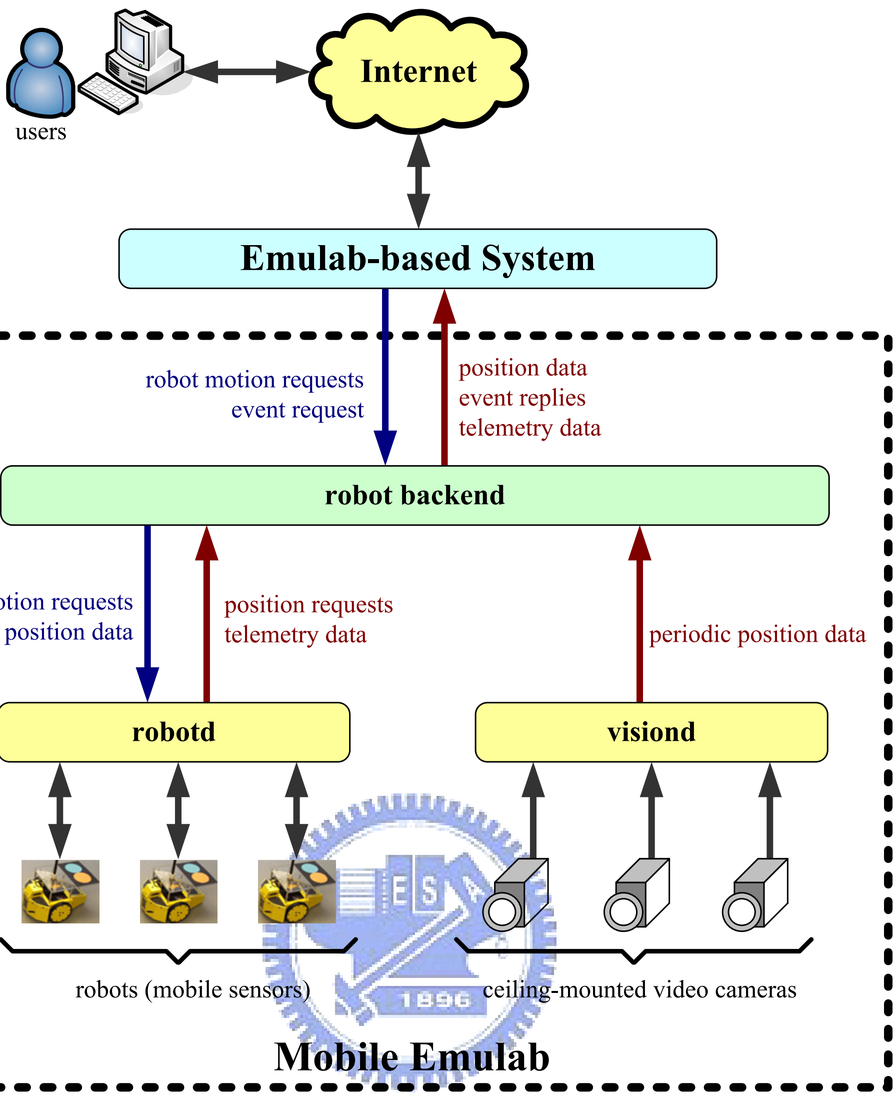


Figure 2.23: The system architecture of Mobile Emulab.

will be sent to the sink. After receiving such packet, the sink will dispatch a mobile sensor to the target node. The mobile sensor will move to the next static sensor by continuously monitoring the signal strength of the beacons sent from the static sensor at the next hop. To approach a specified static sensor, the mobile sensor will go forward and detect the change of received signal strength. If the strength is increasing, it means that this mobiles sensor is approaching a *turning point*, which is defined as the midpoint of a line segment that the mobile sensor can receive the strongest signal strength, as shown in Fig. 2.25(a). Otherwise, the mobile sensor is moving away from the target and it will immediately reverse its current moving direction. When the mobile sensor arrives at the turning point, it knows that the target is either in the right or left side of its moving direction. In this case, the mobile sensor will turn right[3] to seek for the target. By repeating this

[3]If the target is in the left side, the signal strength received by the mobile sensor will become weaker as it moves away from the target. In this case, the mobile sensor will reverse its direction.

procedure, the mobile sensor can eventually arrive to the final destination. Fig. 2.25(b) gives an example to show the moving steps of a mobile sensor.

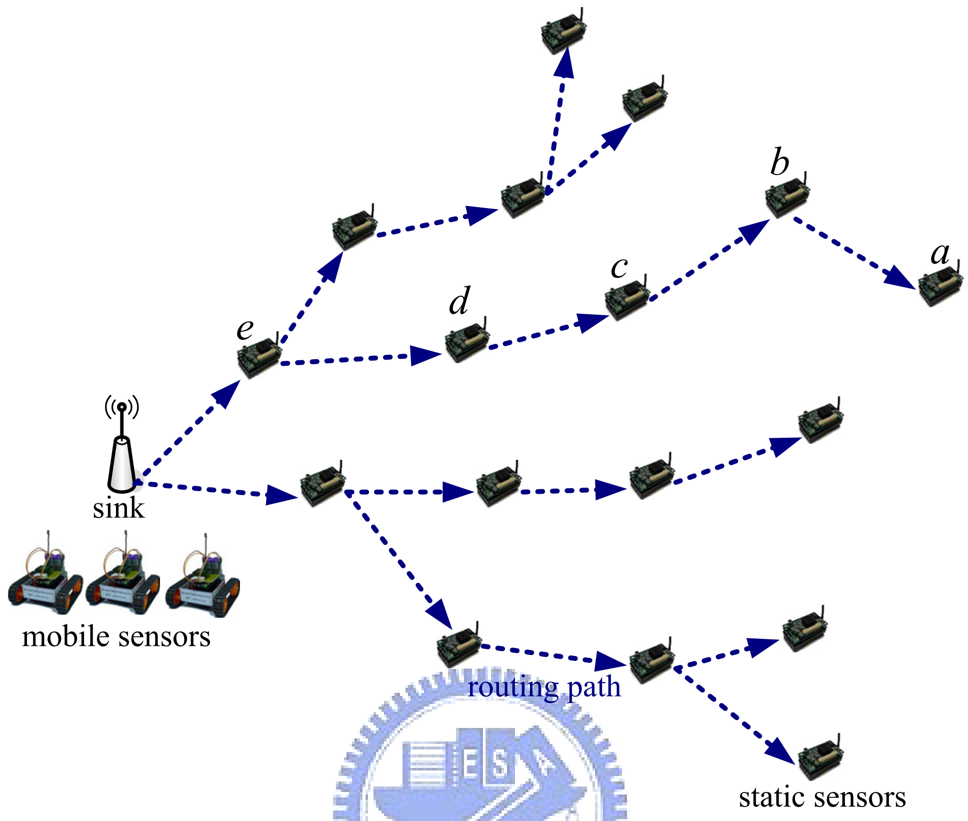
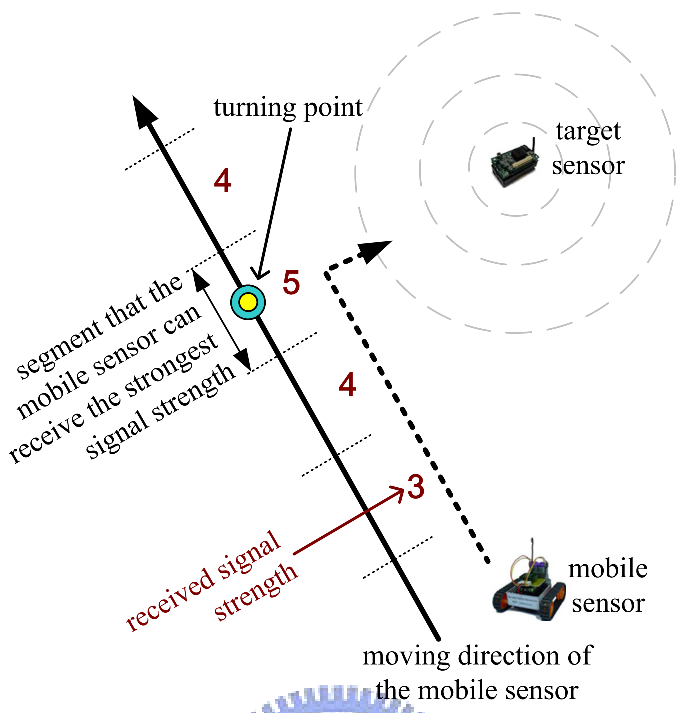
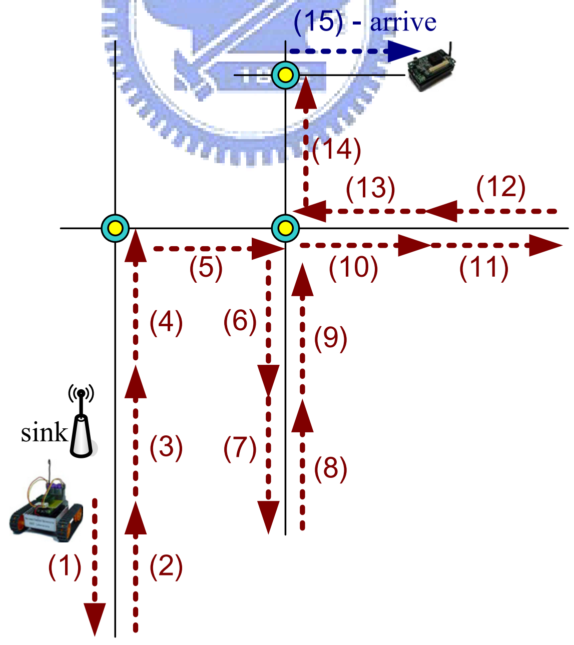


Figure 2.24: Constructing a routing path from the sink to each static sensor.


turning point
target sensor
4
5
4
3
segment that the mobile sensor can receive the strongest signal strength
received signal strength
mobile sensor
moving direction of the mobile sensor

(a)



(b)

Figure 2.25: Navigation of a mobile sensor by the received signal strength: (a) the turning point and (b) the moving steps of a mobile sensor from the sink to a static sensor.

Chapter 3

Deployment of a Wireless Sensor Network for Single-level Coverage

Sensor deployment is a critical issue because it affects the cost and detection capability of a wireless sensor network. In this chapter, we aim at the planned deployment of sensors in the environments such as in buildings or known fields. We address two related problems: *sensor placement* and *sensor dispatch*. The placement problem asks how to place the least number of sensors in a field to achieve desired *coverage* and *connectivity* properties, where coverage is to guarantee that every location in the sensing field is monitored by at least one sensor and connectivity is to ensure that there are sufficient routing paths between sensors. Note that coverage is affected by sensors' sensitivity, while connectivity is decided by sensors' communication ranges. The dispatch problem assumes that sensors are mobilized and the goal is, given a set of mobile sensors and an area of interest inside the sensing field, to choose a subset of sensors to be delegated to the area of interest with certain objective functions such that coverage and connectivity properties are satisfied.

In this chapter, we propose more general solutions to the sensor placement problem than existing results. Our approach allows an arbitrary relationship between a sensor's communication distance and its sensing distance. The sensing field is assumed to be a polygon of any shape in which there may be arbitrary-shaped obstacles. So the results can model an indoor environment. Our approach first partitions the sensing field into smaller sub-regions. In each sub-region, we arrange sensors row by row such that each row guarantees continuous coverage and connectivity and that adjacent rows ensure continuous coverage. Finally, columns of sensors are added to ensure connectivity between rows. The result requires fewer sensors compared to other schemes. For the sensor dispatch problem, we have proposed a centralized and a distributed schemes based on the former placement results. Both schemes attempt to minimize the total energy consumption to move sensors, or to maximize the average remaining energy of those sensors that are moved into the area of interest. The first scheme converts the dispatch problem to the maximum-weight maximum-matching problem, whose optimal solution can be found in polynomial time. With a greedy strategy, the second scheme is distributed in that sensors will select the most suitable locations as their destinations and compete with each other to move to these locations.

3.1 Problem Statement

3.1.1 The Sensor Placement Problem

We are given a sensing field A to be deployed with sensors. Each sensor has a communication distance r_c and a sensing distance r_s. Sensors are homogenous, but we allow an arbitrary relationship of r_c and r_s. The sensing field A is modeled by an arbitrary 2D polygon. Obstacles may exist inside A, which are also modeled by polygons of arbitrary shapes. However, these obstacles do not separate A into non-connected sub-regions. Otherwise, it wouldn't be possible to maintain the network connectivity. With the presence of obstacles, we consider two sensors s_i and s_j as *connected* if $|\overline{s_i s_j}| \leq r_c$ and the line segment $\overline{s_i s_j}$ does not cross any obstacle or boundary of A; otherwise, they are *disconnected*. Fig. 3.1(a) and (b) present two examples. Obstacles may also diminish a sensor's coverage. We define that a point can be monitored by a sensor if it is within a distance of r_s and line-of-sight exists with the existence of obstacles. Fig. 3.1(c) and (d) give two examples. Note that here we adopt the *binary sensing model* [50, 62] of sensors, where a location can be either monitored or not monitored by a sensor. In Section 3.2.4, we will discuss how to adjust our placement solution to adopt to the *probabilistic sensing model* [23, 29, 133], where a location will be monitored by a sensor with some probability function.

Our objective is to place sensors in A to guarantee both *sensing coverage* (in the sense that every point inside A can be monitored by sensors) and *network connectivity* (in the sense that no sensor gets disconnected) using as few sensors as possible. The concepts of coverage and connectivity in an office environment are illustrated in Fig. 3.2. Note that we assume $r_c = r_s$ in this example.

3.1.2 The Sensor Dispatch Problem

We are given a sensing field A, an area of interest I inside A, and a set of mobile sensors S resident in A. The *sensor dispatch problem* asks how to find a subset $S' \subseteq S$ of sensors to be moved to I such that after the deployment, I satisfies our coverage and connectivity

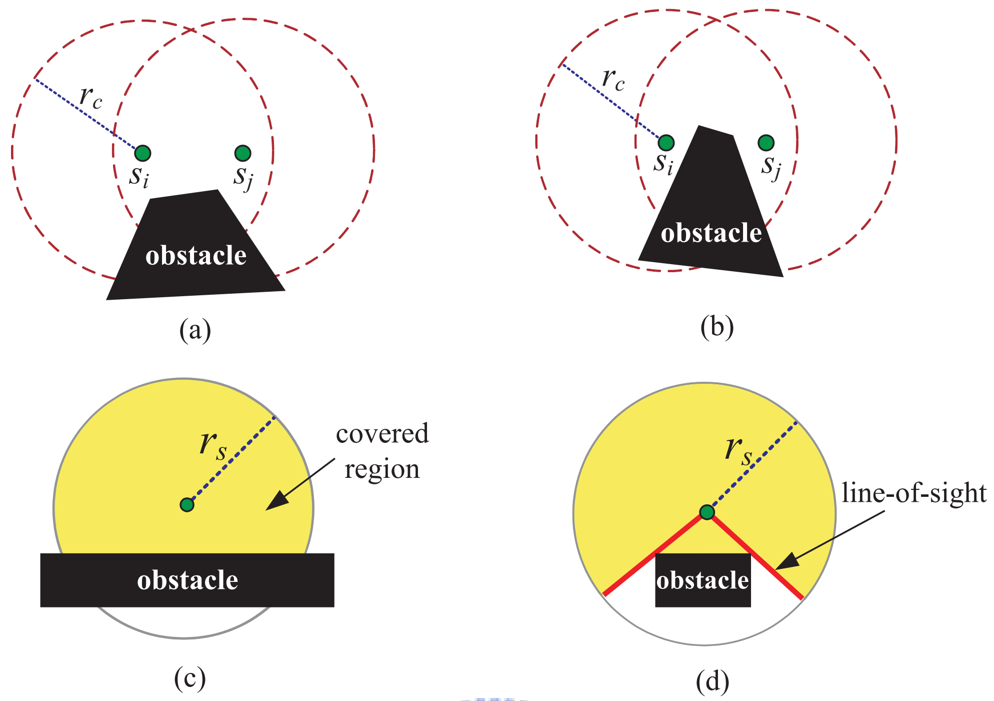


Figure 3.1: Assumptions on connectivity and coverage: (a) s_i and s_j are connected, (b) the obstacle disconnects s_i and s_j, (c) coverage with a large obstacle, and (d) coverage with a small obstacle.

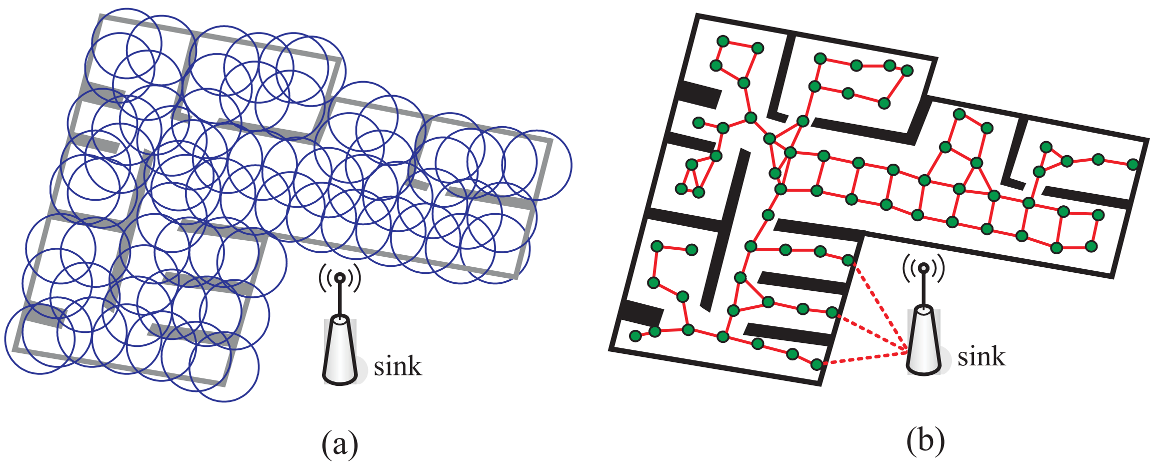


Figure 3.2: An example of sensor deployment in an office environment: (a) sensing coverage and (b) network connectivity.

requirements and the movement cost satisfies some objective function. Here we consider two functions. The first one is to minimize the total energy consumption to move sensors, i.e.,

$$min \sum_{i \in S'} \Delta_m \times d_i, \tag{3.1}$$

where Δ_m is the unit energy cost to move a sensor in one step and d_i is the distance that sensor i has to be moved. The second one is to maximize the average remaining energy of sensors after the movement, i.e.,

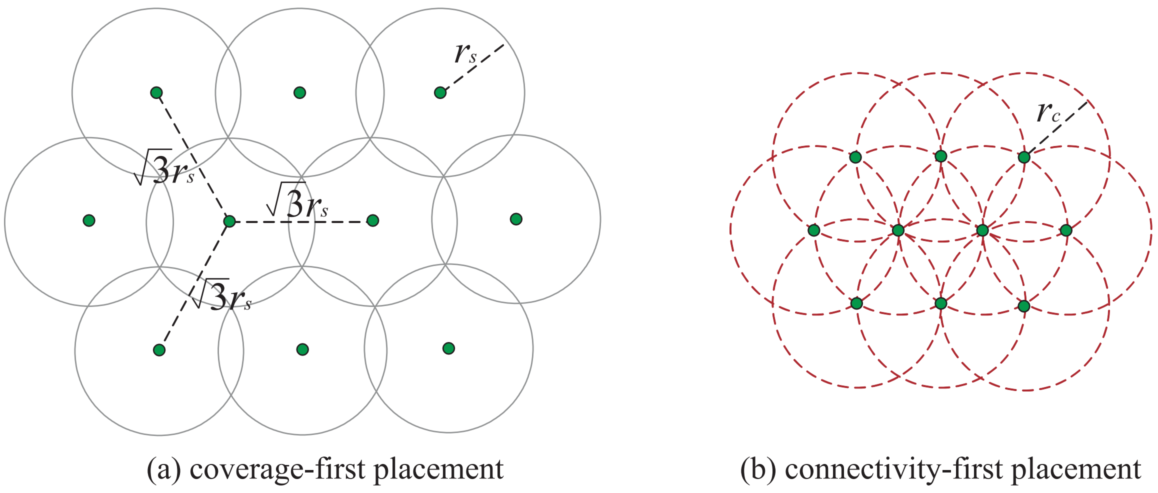
$$max \ \frac{\sum_{i \in S'} (e_i - \Delta_m \times d_i)}{|S'|}, \tag{3.2}$$

where e_i is the initial energy of sensor i. Note that the calculation of d_i should take the existence of obstacles into account.

3.2 Solutions to the Sensor Placement Problem

To start with, we first consider two possible placements. The first one attempts to reduce the number of sensors by minimizing the overlapping coverage. The result would be as shown in Fig. 3.3(a), where neighboring sensors are evenly separated by a distance of $\sqrt{3}r_s$. This scheme is efficient when $r_c \geq \sqrt{3}r_s$ since connectivity is automatically guaranteed. However, when $r_c < \sqrt{3}r_s$, extra sensors need to be added to maintain network connectivity. It is inefficient because the whole sensing field has been covered and these newly-added sensors will not make any contribution to coverage. The second possible placement is to meet the connectivity requirement first. This placement would be as shown in Fig. 3.3(b), where neighboring sensors are evenly separated by a distance of r_c. This scheme is efficient when $r_c \leq \sqrt{3}r_s$ because coverage can be automatically guaranteed. However, when $r_c > \sqrt{3}r_s$, extra sensors need to be added to maintain coverage. It is inefficient because the overlapping coverage could be large.

Our placement has the following features. First, it avoids the dilemma in the above placements by taking both r_c and r_s into account. Second, our solution is more general as it allows an arbitrary shape of sensing field A and possibly obstacles in A. Our scheme works in two steps. First, it partitions A into a number of regions. Regions are classified into *single-row regions* and *multi-row regions*. A single-row region is a belt-like area with width no larger than $\sqrt{3}r_{min}$, where $r_{min} = \min(r_c, r_s)$, so a row of sensors is sufficient to fully cover the region while maintaining connectivity. A multi-row region is perceivably larger and can be covered by several rows of sensors. Fig. 3.4 gives an example, where the sensing field is partitioned into eight single-row regions and six multi-row regions.



r_s
$\sqrt{3}r_s$
$\sqrt{3}r_s$
$\sqrt{3}r_s$
r_c
(a) coverage-first placement
(b) connectivity-first placement

Figure 3.3: Two possible sensor placements: (a) considering the coverage property first and (b) considering the connectivity property first.

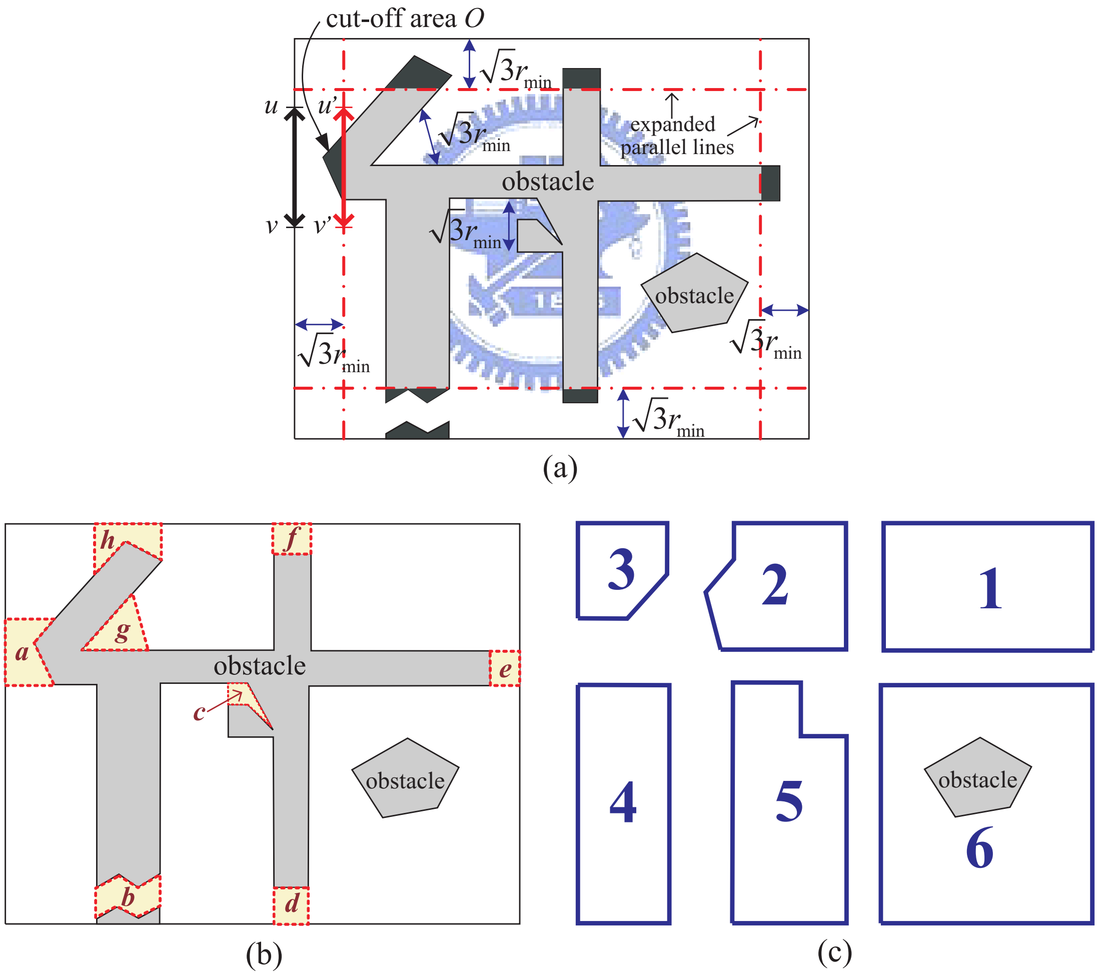


Figure 3.4: Partitioning a sensing field: (a) expanding the boundary of field inwardly by a distance of $\sqrt{3}r_{min}$, (b) the eight single-row regions found by taking projections from obstacles, and (c) the six multi-row regions found by excluding the single-rows regions.

```
algorithm Partition
Input:  $A$: the sensing field
        $B$: the set of boundaries of $A$ and perimeters of obstacles
Output: single-row and multi-row regions
1:  for each $\overline{uv} \in B$          /* find out all single-row regions */
2:      expand a parallel line $\overline{u'v'}$ by a distance of $\sqrt{3}r_{\min}$;
3:      if $\overline{u'v'}$ cuts off a partial region, say, $O$ of an obstacle
4:          then take a projection $P$ from $O$ to $\overline{uv}$;
5:              if $P$ overlaps with an existing single-row region $P'$
6:                  then merge $P$ and $P'$ into one single-row region;
7:              else
8:                  make $P$ a new single-row region;
9:  end for
10: exclude all single-row regions from $A$ and the rest of the regions are multi-row regions;
```

Figure 3.5: The pseudo code of the partition algorithm.

3.2.1 Partitioning the Sensing Field

Fig. 3.5 illustrates our partition algorithm. The idea is to first identify all single-row regions. After excluding single-row regions, the remaining regions are multi-row regions.

To identify single-row regions, we expand the boundaries of A *inward* and perimeters of obstacles *outward* by a distance of $\sqrt{3}r_{min}$. If there is a single-row region between one obstacle and the line segment $\overline{uv}$ of A's boundary, the expanded parallel line $\overline{u'v'}$ must cut off a partial region, say, O of the obstacle or $\overline{A}$ (the area outside A). Then we can take a projection from O to $\overline{uv}$ to obtain the single-row region. Fig. 3.4(a) shows how to find single-row regions for the boundary, where the dotted lines are the expanded parallel lines of A's boundaries. After taking projections, we can obtain six single-row regions a, b, d, e, f, and h in Fig. 3.4(b). Then we can perform the same steps for each obstacle. Note that a single-row region obtained from one obstacle may have overlapping with those obtained earlier (due to different projections). In this case, we can simply merge those with overlappings into one single-row region. This guarantees that our partition algorithm will produce a unique output. Fig. 3.4(b) shows all obtained single-row regions.

The aforementioned step may obtain several single-row regions. Excluding such regions, the remaining areas of A are multi-row regions. An example is given in Fig. 3.4(c). Note that there could be still obstacles inside a multi-row region (e.g., the region 6).

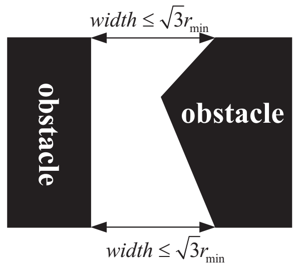
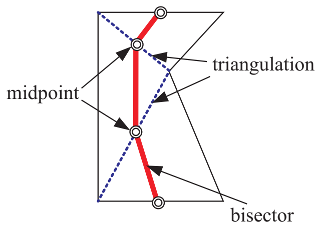
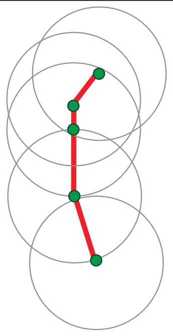
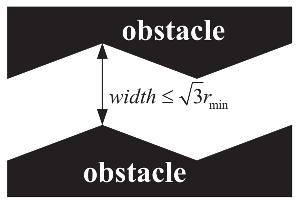
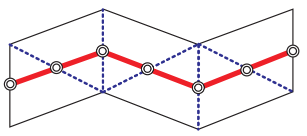
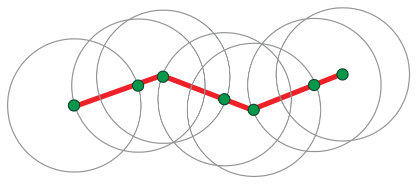
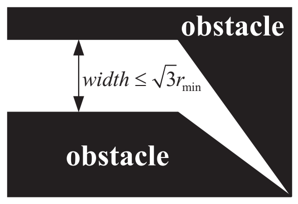
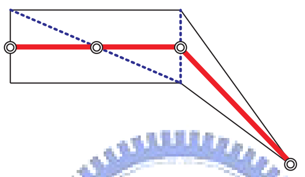
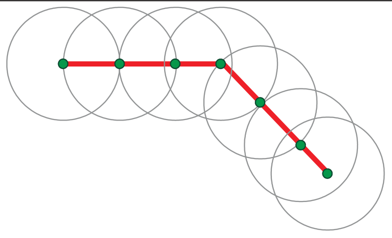
case	single-row regions	bisectors	sensor placements
(a)	 $width \le \sqrt{3}r_{\min}$ obstacle obstacle $width \le \sqrt{3}r_{\min}$		
(b)			
(c)			

Figure 3.6: Three examples of finding bisectors of single-row regions and their corresponding sensor placements when $r_c = r_s$.

3.2.2 Placing Sensors in Single-row Regions

For a single-row region, we can find its bisector and then place a sequence of sensors along the bisector to satisfy both coverage and connectivity. A bisector can be found by doing a triangulation on that region, as shown in Fig. 3.6, and then connecting the midpoints of all dotted lines. Following the bisector, we can place a sequence of sensors each separated by a distance of r_{min} to ensure coverage and connectivity of that region, as shown in Fig. 3.6. Note that we always add an extra sensor at the end of the bisector for ensuring connectivity to neighboring regions.

3.2.3 Placing Sensors in Multi-row Regions

Multiple rows of sensors will be placed in such regions. Next, we first consider how to place sensors in a simple 2D plane without boundaries and obstacles. Then we extend the results to a region with boundaries and obstacles.

A Simple 2D Plane

Given a 2D plane without boundaries and obstacles, we will place sensors row by row. The basic concept is to place a row of sensors that can guarantee continuous coverage and connectivity. Then adjacent rows should guarantee continuous coverage of the whole area. Finally, we may place some columns of sensors between adjacent rows, if necessary, to ensure the connectivity of the whole network. According to the relationship of r_c and r_s, we separate our discussion into two cases.

- **Case of $r_c < \sqrt{3}r_s$:** In this case, sensors on each row will be separated by a distance of r_c to ensure the connectivity of that row. Since $r_c < \sqrt{3}r_s$, each row of sensors can cover a belt-like area whose width is $2 \times \sqrt{r_s^2 - \frac{r_c^2}{4}}$. Adjacent rows will be separated by a distance of $r_s + \sqrt{r_s^2 - \frac{r_c^2}{4}}$ and shifted by a distance of $\frac{r_c}{2}$. With such an arrangement, the coverage of the whole area can be guaranteed. Fig. 3.7(a) – (c) present three possible subcases. Note that in this case, we have to place a column of sensors between two adjacent rows, each separated by a distance no larger than r_c, to connect them. More columns of sensors can be added to strengthen the network connectivity.

- **Case of $r_c \geq \sqrt{3}r_s$:** In this case, the aforementioned method will waste a large amount of sensors since a small r_s will make two adjacent rows too close to each other. Therefore, when $r_c \geq \sqrt{3}r_s$, we suggest to place sensors in a typical hexagon manner such that adjacent sensors are regularly separated by a distance of $\sqrt{3}r_s$. In this way, both coverage and connectivity can be guaranteed.

Multi-row Regions with Boundaries and Obstacles

In the following, we modify the aforementioned solution to place sensors in a region with boundaries and obstacles. Observe that in our solution, sensors are placed with regular patterns. So we can transform it into an incremental approach where sensors are added into the field one by one. In Table 3.1, we list the coordinates of the six neighbors of a sensor. We can decide the location to place the first sensor inside the region. From the first sensor, the six locations that can potentially be added with sensors are then determined. These locations are inserted into a queue Q. We then enter a loop in which each time an entry (x, y) is dequeued from Q. If (x, y) is not outside the region and not inside any obstacle, a sensor will be placed at the location (x, y). Similarly, the six neighboring locations of (x, y) in Table 3.1 will be inserted into Q if they have not been checked before. This process is repeated until Q becomes empty.

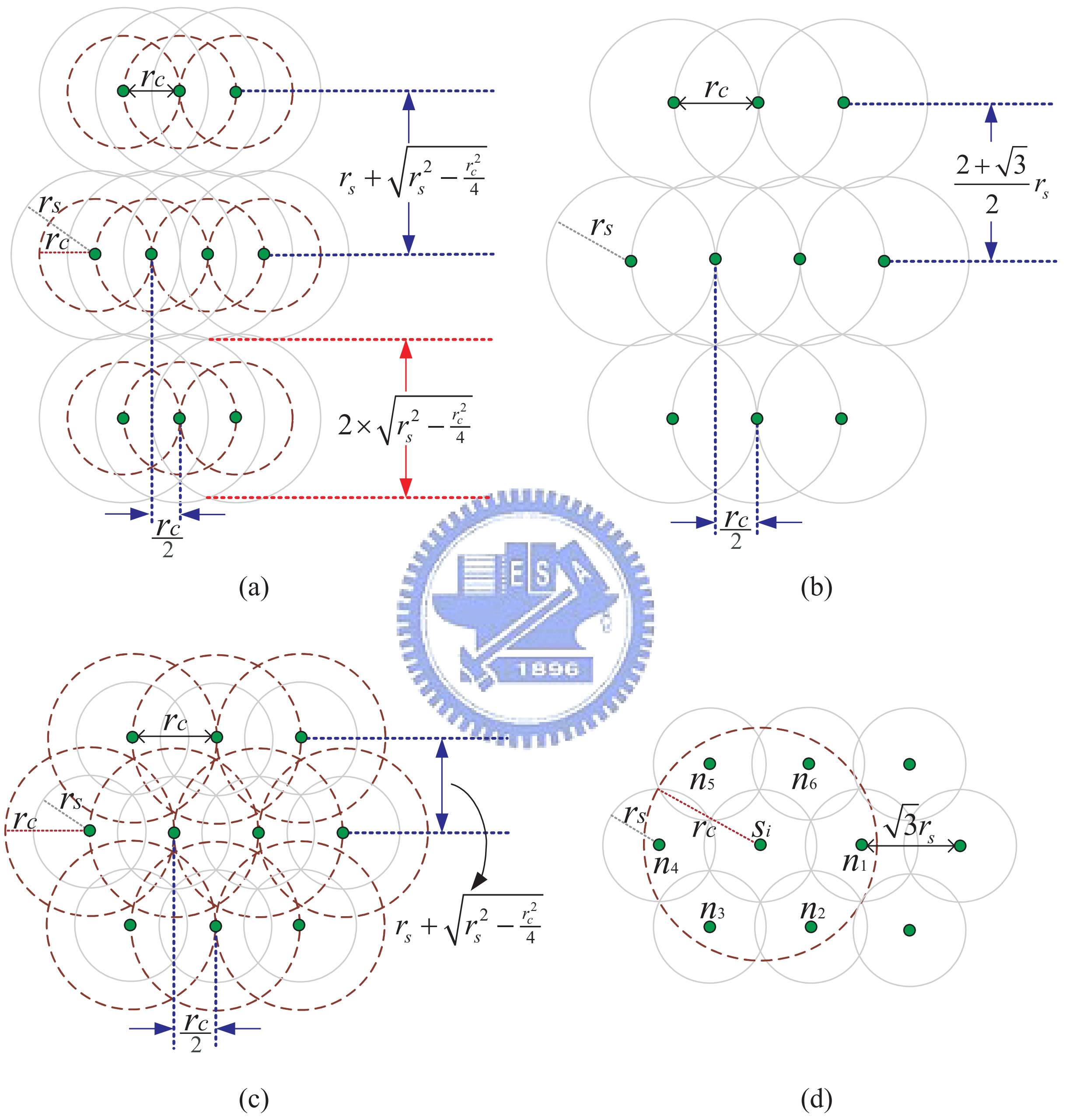


Figure 3.7: Placing sensors in a simple 2D plane: (a) the case of $r_c < r_s$, (b) the case of $r_c = r_s$, (c) the case of $r_s < r_c < \sqrt{3} r_s$, and (d) the case of $r_c \geq \sqrt{3} r_s$.

Table 3.1: Coordinates of the six neighbors of a sensor s_i located at (x, y) in a multi-row region.

neighbor	$r_c < \sqrt{3}r_s$	$r_c \geq \sqrt{3}r_s$
n_1	$(x + r_c, y)$	$(x + \sqrt{3}r_s, y)$
n_2	$(x + \frac{r_c}{2}, y - (r_s + \sqrt{r_s^2 - \frac{r_c^2}{4}}))$	$(x + \frac{\sqrt{3}r_s}{2}, y - \frac{3r_s}{2})$
n_3	$(x - \frac{r_c}{2}, y - (r_s + \sqrt{r_s^2 - \frac{r_c^2}{4}}))$	$(x - \frac{\sqrt{3}r_s}{2}, y - \frac{3r_s}{2})$
n_4	$(x - r_c, y)$	$(x - \sqrt{3}r_s, y)$
n_5	$(x - \frac{r_c}{2}, y + (r_s + \sqrt{r_s^2 - \frac{r_c^2}{4}}))$	$(x - \frac{\sqrt{3}r_s}{2}, y + \frac{3r_s}{2})$
n_6	$(x + \frac{r_c}{2}, y + (r_s + \sqrt{r_s^2 - \frac{r_c^2}{4}}))$	$(x + \frac{\sqrt{3}r_s}{2}, y + \frac{3r_s}{2})$

There are three minor issues left in the above solution. First, some areas close to the region's boundaries or obstacles may be still uncovered. Second, when $r_c < \sqrt{3}r_s$, we have to add some extra sensors between adjacent rows to connect them. Third, connectivity between neighboring regions should also be maintained. Fig. 3.8(a) presents an example. These problems can be solved by sequentially placing sensors along the boundaries of the region and obstacles, as shown in Fig. 3.8(b). There are two cases to be considered. When $r_c < \sqrt{3}r_s$, since the maximum width of the uncovered area does not exceed r_c, sensors should be separated by a distance of r_c. When $r_c \geq \sqrt{3}r_s$, the maximum width of the uncovered area does not exceed $\sqrt{3}r_s$, so sensors should be separated by a distance of $\sqrt{3}r_s$. In this case, the connectivity between these extra sensors and the regularly-placed sensors can be also guaranteed.

Note that we can reduce the number of sensors in the last step by carefully selecting the first sensor's position in each multi-row region. In particular, for each multi-row region, we can place the first sensor near its longest boundary with a distance of δ, where $\delta = \sqrt{r_s^2 - \frac{r_c^2}{4}}$ if $r_c < \sqrt{3}r_s$ and $\delta = \frac{r_s}{2}$ otherwise. This will make the first row of sensors fully cover the longest boundary of the region and thus we do not have to add extra sensors in the last step. In addition, if the distance between a row of sensors and a boundary of the region (or an obstacle) is no larger than δ, we can also skip the last step. For example, some boundaries in Fig. 3.8(b) are not added with extra sensors.

3.2.4 *Adapting to the Probabilistic Sensing Model*

Up to now, our placement solution is based on the assumption of binary sensing model. In some cases, however, the detection probability of a sensor will decay with the distance from the sensor to the object. For example, references [29, 133] suggest that the detection

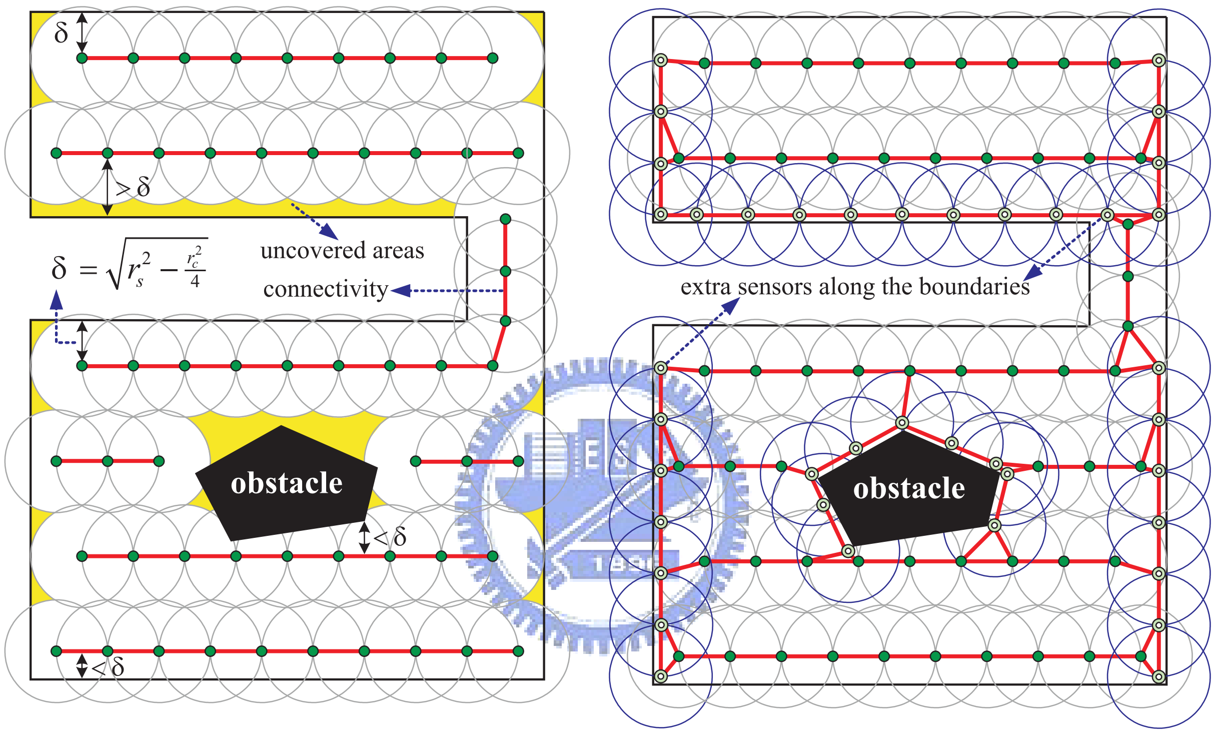

δ
> δ
uncovered areas
δ = $\sqrt{r_s^2 - \frac{r_c^2}{4}}$
connectivity
obstacle
< δ
< δ
extra sensors along the boundaries
obstacle
(a)
(b)

Figure 3.8: Placing sensors along the boundaries and obstacles: (a) uncovered areas along the boundaries and obstacles and (b) placing extra sensors to fill with these uncovered areas and to maintain the network connectivity. Note that this example assumes that $r_c = r_s$.

probability of a location u by a sensor s_i can be modeled by:

$$p_{s_i}^u = \begin{cases} e^{-\alpha d(s_i,u)}, & \text{if } d(s_i,u) \leq r_s \\ 0, & \text{otherwise} \end{cases},$$

where α is a parameter representing the physical characteristics of the sensor and $d(s_i, u)$ is the distance between s_i and u. Thus, when an object located at u is within the sensing ranges of a set $\hat{S}$ of sensors, the detection probability can be evaluated as

$$p(u) = 1 - \prod_{s_i \in \hat{S}} (1 - p_{s_i}^u).$$

It can be observed that in our placement solutions, for any combination of r_c and r_s, there must exist a location which is covered by only one sensor and has a distance of r_s to that sensor. The detection probability for such a location is thus $e^{-\alpha r_s}$. Therefore, our placement solutions can guarantee a detection probability of at least $e^{-\alpha r_s}$ in any location of the sensing field. On the other hand, if we want to guarantee that every point in the sensing field has a detection probability no smaller than a given threshold p_{th}, we can compute a virtual sensing distance r_s' by

$$e^{-\alpha r_s'} = p_{th} \Rightarrow r_s' = -\frac{\ln p_{th}}{\alpha}.$$

According to the above argument, if we replace r_s by r_s' when running our placement solutions, it is guaranteed that every point in the sensing field has a detection probability of at least p_{th}.

3.3 Solutions to the Sensor Dispatch Problem

Given a set of sensors already deployed in A and an area of interest I that has to be monitored intensively, the dispatch problem will be solved by the following steps:

1. Based on our placement results, we first compute the locations to be placed with sensors in I and then select some sensors to be moved to these locations.

2. In order to correctly report sensed data in I to the sink, we need to connect sensors in I and the sink. We then place a row of sensors, each separated by a distance of r_c, from I to the sink.

3. After dispatching sensors in steps 1 and 2, the remaining sensors can be deployed uniformly in the region of $A - I$ to ensure that the coverage of $A - I$ is not reduced too much.

The above step 2 can be achieved easily. Step 3 can be done by applying the solutions using repulsive forces between sensors [50, 132] on $A-I$. As a result, we will only focus on the design of step 1 below. Two solutions are proposed. The centralized solution converts the dispatch problem to the maximum-weight maximum-matching problem, while the distributed solution is based on a greedy strategy.

3.3.1 A Centralized Dispatch Solution

Given a set S of sensors in A and an area of interest I, our solution involves the following five steps:

1. Run the sensor placement algorithm in Section 3.2 on the area I to determine the locations in I to be placed with sensors. Let the set of locations be $L = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$. If $m \leq |S|$, go to step 2; otherwise, we are short of sensors and the algorithm terminates.

2. For each sensor $s_i \in S$, determine the energy cost $c(s_i, (x_j, y_j))$ to move s_i to each location (x_j, y_j), $j = 1...m$. We define $c(s_i, (x_j, y_j)) = \Delta_m \times d(s_i, (x_j, y_j))$, where $d(s_i, (x_j, y_j))$ is the shortest distance from s_i's current position to (x_j, y_j) considering the existence of obstacles.

3. From S and L, we construct a weighted complete bipartite graph $G = (S \cup L, S \times L)$ such that the vertex set contains S (all sensors) and L (all locations to be placed with sensors) and the edge set contains all edges from every element $s_i \in S$ to every element $(x_j, y_j) \in L$. The weight of each edge $(s_i, (x_j, y_j))$ can be defined either as

$$w(s_i, (x_j, y_j)) = -c(s_i, (x_j, y_j)),$$

   if Eq. (3.1) is the objective function, or as

$$w(s_i, (x_j, y_j)) = e_i - c(s_i, (x_j, y_j)),$$

   if Eq. (3.2) is the objective function.

4. Solve the maximum-weight maximum-matching problem on graph G. In particular, we construct a new graph $\hat{G} = (S \cup L \cup \hat{L}, S \times \{L \cup \hat{L}\})$ from G, where $\hat{L}$ is a set of $|S| - |L|$ elements, each called a *virtual location*. The weight of each edge in $\hat{G}$ that also appears in G remains the same as that in G, and the weight of each edge from $s_i \in S$ to $(x_j, y_j) \in \hat{L}$ is set to w_{min}, where

$$w_{min} = \min_{s_i \in S,\ (x_j, y_j) \in L} \{w(s_i, (x_j, y_j))\} - 1.$$

Intuitively, a virtual location is a dummy one. Its purpose is to make the two sets S and $\{L \cup \hat{L}\}$ of the bipartite graph $\hat{G}$ to have equal sizes. This allows us to transform the problem to the *maximum-weight perfect-matching problem* on graph $\hat{G}$, whose purpose is to find a perfect matching M in $\hat{G}$ with the maximum total weights of edges in M. Note that the value of w_{min} is set in such a way that selecting an edge incident to a virtual location has no impact to a solution to the maximum-weight perfect-matching problem.

5. For each edge $(s_i, (x_j, y_j))$ in M such that $(x_j, y_j) \notin \hat{L}$, we move sensor s_i to location (x_j, y_j) via the shortest path. However, if there is any edge $(s_i, (x_j, y_j)) \in M$ such that $(x_j, y_j) \notin \hat{L}$ and $e_i - c(s_i, (x_j, y_j)) \leq 0$, it means that we do not have sufficient energy to move sensors to all locations in L because M is the optimal solution. Thus the algorithm terminates.

Computing the Shortest Distance $d(s_i, (x_j, y_j))$

Our goal is to find the shortest *collision-free* path from s_i's current position to (x_j, y_j), considering the existence of obstacles. Specifically, the movement of s_i should not collide with any obstacle. Several studies have also addressed this issue [27, 73, 130]. Here we propose a modified approach of [73].

Considering its physical size, a sensor s_i can be modeled as a circle with a radius r. Intuitively, s_i has a collision-free motion if its center always keeps at a distance of r or larger away from every obstacle and A's boundaries. This can be done by expanding the perimeters of all obstacles outwardly and A's boundaries inwardly by a distance of r and preventing s_i from moving into these expanded areas. The problem can be translated to one of finding a shortest path from s_i to (x_j, y_j) in a weighted graph $H = (s_i \cup (x_j, y_j) \cup V, E)$, where V contains all vertices v of the polygons representing the expanded areas of obstacles and A's boundaries such that v is not inside other expanded areas, and E contains all edges (u, v) such that $u, v \in \{s_i \cup (x_j, y_j) \cup V\}$ and $\overline{uv}$ does not pass any expanded area of obstacles or A. The weight of $(u, v) \in E$ is length of $\overline{uv}$. Fig. 3.9 gives an example, where double circles are vertices of H. Nodes g and h are not vertices because they are inside obstacles 2's and 3's expanded areas, respectively. Edges (a, c), (a, d), (b, c), and $(b, d) \in E$, but (b, e) and $(b, f) \notin E$ because they pass the expanded area of obstacle 2.

Finding the Maximum-Weight Perfect-Matching M

Recall that given the bipartite graph $\hat{G} = (S \cup L \cup \hat{L}, S \times \{L \cup \hat{L}\})$, the goal is to find a perfect matching M in $\hat{G}$ with the maximum total weights of edges in M. In this section,

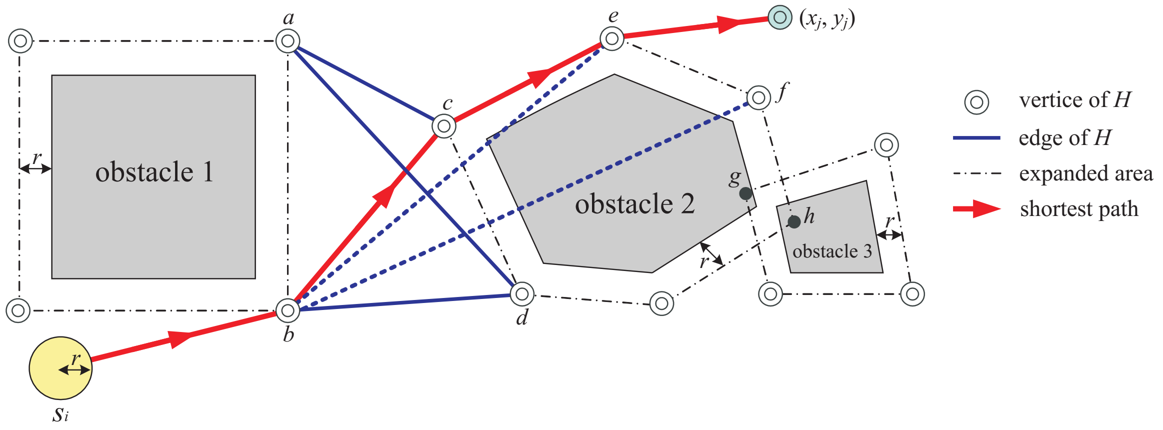


Figure 3.9: Finding a collision-free path from s_i to (x_j, y_j). Note that not all edges of H are shown in the figure.

we discuss how to use the Hungarian method [64] to solve this problem.

Definition 1. *Given $\hat{G} = (S \cup L \cup \hat{L}, S \times \{L \cup \hat{L}\})$, a* ***feasible vertex labeling*** *of $\hat{G}$ is a real-valued function f on $\{S \cup L \cup \hat{L}\}$ such that for all $s_i \in S$ and $(x_j, y_j) \in \{L \cup \hat{L}\}$,*

$$f(s_i) + f((x_j, y_j)) \geq w(s_i, (x_j, y_j)).$$

Definition 2. *Given a feasible vertex labeling of $\hat{G}$, an* ***equality subgraph*** *$\hat{G}_f = (S \cup L \cup \hat{L}, E_f)$ is the subgraph of $\hat{G}$ in which E_f contains all edges $(s_i, (x_j, y_j))$ in $\hat{G}$ such that*

$$f(s_i) + f((x_j, y_j)) = w(s_i, (x_j, y_j)).$$

Theorem 3.1. *Let f be a feasible vertex labeling of $\hat{G}$ and M be a perfect matching of $\hat{G}_f$, then M will be a maximum-weight perfect matching of $\hat{G}$.*

Proof. We show that no other perfect matching M' in $\hat{G}$ can have a total edge weight larger than M.

$$\begin{aligned}
w(M') &= \sum_{(s_i,(x_j,y_j)) \in M'} w(s_i, (x_j, y_j)) && (s_i \in S \text{ and } (x_j, y_j) \in \{L \cup \hat{L}\}) \\
&\leq \sum_{(s_i,(x_j,y_j)) \in M'} f(s_i) + f((x_j, y_j)) && (\because \text{From the definition of feasible vertex labelings}) \\
&= \sum_{(s_i,(x_j,y_j)) \in M} f(s_i) + f((x_j, y_j)) && (\because \text{Total labelings are the same in any matching}) \\
&= \sum_{(s_i,(x_j,y_j)) \in M} w(s_i, (x_j, y_j)) && (\because M \text{ is a perfect matching in } \hat{G}_f) \\
&= w(M),
\end{aligned}$$

so M has the maximum total weights of edges. □

The Hungarian method is based on the observation from Theorem 3.1. It first assigns an arbitrary feasible vertex labeling for the graph $\hat{G}$, and then adjusts the labels of vertices until it can find a perfect matching M in the equality subgraph $\hat{G}_f$. One possible feasible vertex labeling is to set $f((x_j, y_j)) = 0$ for all $(x_j, y_j) \in \{L \cup \hat{L}\}$ and to set $f(s_i)$ to the maximum of the weights of the edges adjacent to s_i for all $s_i \in S$. Specifically,

$$\begin{cases} f((x_j, y_j)) = 0, & \text{for } (x_j, y_j) \in \{L \cup \hat{L}\} \\ f(s_i) = \max\limits_{(x_j,y_j)\in\{L\cup\hat{L}\}} \{w(s_i, (x_j, y_j))\}, & \text{for } s_i \in S \end{cases}.$$

The complete procedure of the Hungarian method is stated as follows:

Step 1: Find a maximum matching M in $\hat{G}_f$. If M is perfect, we find out the solution and the method finishes. Otherwise, there must be an unmatched vertex $s_i \in S$. We then assign two sets $A = \{s_i\}$ and $B = \emptyset$.

Step 2: In the graph $\hat{G}_f$, if $N_{\hat{G}_f}(A) \neq B$, where $N_{\hat{G}_f}(A)$ is the set of vertices in $\{L \cup \hat{L}\}$ that are adjacent to the vertices in A, then go to step 3. Otherwise, we set

$$\alpha = \min_{s_i \in A,\ (x_j,y_j)\in\{L\cup\hat{L}\}-B} \{f(s_i) + f((x_j, y_j)) - w(s_i, (x_j, y_j))\},$$

and construct a new labeling f' for $\hat{G}$ by

$$f'(v) = \begin{cases} f(v) - \alpha & \text{for } v \in A \\ f(v) + \alpha & \text{for } v \in B \\ f(v) & \text{otherwise} \end{cases}.$$

Then we replace f by f', reconstruct the equality subgraph $\hat{G}_{f'}$, and go to step 1. Note that we have to satisfy the conditions of $\alpha > 0$ and $N_{\hat{G}_{f'}}(A) \neq B$; otherwise, we need to reselect another α value that can satisfy the above conditions.

Step 3: Choose a vertex (x_l, y_l) in $N_{\hat{G}_f}(A)$ but not in B. If (x_l, y_l) is matched with $s_k \in S$ in M, then we update $A = A \cup \{s_k\}$ and $B = B \cup \{(x_l, y_l)\}$, and go back to step 2.

Note that each time when we relabeling the graph $\hat{G}$, we may introduce new edges into the new equality subgraph $\hat{G}_f$, until all edges in $\hat{G}$ are included. Therefore, the Hungarian method can always find a perfect matching in $\hat{G}_f$ since $\hat{G}$ is a complete bipartite graph.

Time Complexity Analysis

Next, we analyze the time complexity of our sensor dispatch solution. Let $|S| = n$, $|L| = m$, and k be the number of vertices of the polygons of all obstacles and A. In step

2, there are $O(nm)$ pairs of $(s_i, (x_j, y_j))$. To compute the energy cost of each pair, we construct a graph of $O(k)$ vertices. Finding a shortest path on such graph can use the Dijkstra's algorithm [31], which takes $O(k^2)$ time. So the total time complexity of step 2 is $O(mnk^2)$. The conversion in step 3 takes $O(nm)$ time. In step 4, constructing the graph $\hat{G}$ from G takes $O(n(n-m))$ time since it needs to add $n-m$ vertices and $n(n-m)$ edges. Running the Hungarian method on $\hat{G}$ has a time complexity of $O(n^3)$. Finally, it takes $O(n)$ time in step 5 to check all edges in M. Therefore, the total time complexity is

$$O(mnk^2) + O(nm) + O(n(n-m)) + O(n^3) + O(n) = O(mnk^2 + n^3).$$

3.3.2 A Distributed Dispatch Solution

The aforementioned solution is optimal but centralized. Here we propose a distributed solution based on a greedy strategy. The solution involves the following steps:

1. The sink executes the sensor placement algorithm in Section 3.2 on the area I to obtain a set of locations $L = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ to be occupied by sensors. The sink then broadcasts L to all sensors.

2. On receiving the table L, a sensor will keep a copy of L and mark each location (x_j, y_j) as *unoccupied*, where $j = 1, \ldots, m$.

3. Each sensor s_i then chooses an unoccupied location (x_j, y_j) from L as its destination. The selection of (x_j, y_j) is dependent on our objective function.

   - If Eq. (3.1) is the objective function, s_i will choose the location (x_j, y_j) as its destination such that the moving distance $d(s_i, (x_j, y_j))$ is minimized.
   - If Eq. (3.2) is the objective function, s_i will choose the location (x_j, y_j) as its destination such that after moving to (x_j, y_j), its remaining energy is maximized.

   Sensor s_i then marks (x_j, y_j) as occupied and starts moving to (x_j, y_j).

4. On s_i's way moving toward its destination, it will periodically broadcast the status of its table L, its destination, and its cost to move to that destination. Note that the cost is based on which objective function is used. On sensor s_j receiving s_i's broadcast, the following actions will be taken:

   - For all locations marked as occupied by s_i, s_j will also mark them as occupied.
   - If both s_i and s_j are moving toward the same destination, they will compete by their costs. The one with a lower cost will win and keep moving toward

that destination. The one with a higher cost will give up moving toward that destination and go back to step 3 to reselect a new destination. (Note that if s_j has arrived at its destination, it will have a cost of zero. In this case, s_i will lose in the competition.)

5. Each sensor will repeat the above steps until it reaches its destination or loses to another sensor and finds that all locations in L have been marked as occupied. In the former case, the sensor will execute its monitoring job at the designated location. In the latter case, the sensor will continue to support the remaining steps 2 and 3 mentioned in the beginning of Section 3.3 (to connect I and the sink or to monitor the area $A - I$).

3.4 Experimental Results

In this section, we present some experimental results to verify the effectiveness of our proposed schemes. The evaluation includes two parts. First, we measure the numbers of sensors required by different placement schemes. Second, we estimate the performances of our proposed dispatch schemes.

3.4.1 Effectiveness of the Proposed Placement Schemes

The first experiment evaluates the number of sensors required to be placed in a sensing field. In this experiment, six types of sensing fields with different shapes are considered, as shown in Fig. 3.10. Sensors are assumed to have omnidirectional sensing capability (such as acoustic sensors). The communication distance r_c is set to 10m (which is close to that specified in IEEE 802.15.4 [66] in an indoor environment). To reflect the relationships of $r_c < r_s$, $r_c = r_s$, $r_s < r_c \le \sqrt{3}r_s$, and $r_c > \sqrt{3}r_s$, we set the sensing distance r_s to 12m, 10m, 7m, and 5m, respectively. We compare our result against the *coverage-first* and *connectivity-first* methods discussed in the beginning of Section 3.2.

Fig. 3.11 shows the number of sensors required in different sensing fields. When $r_c \le r_s$, the connectivity-first method uses more sensors because it is dominated by the value of r_c and the overlapping in coverage could be large. On the contrary, when $r_s < r_c < \sqrt{3}r_s$, the coverage-first method uses more sensors because it has to add a large amount of extra sensors to maintain connectivity between originally-placed sensors. When $r_c \ge \sqrt{3}r_s$, the connectivity-first method becomes very inefficient because the overlapping in coverage is very large. Our proposed method requires less sensors because it can adjust the distances between sensors based on the relationship of r_c and r_s. Note that when $r_c \ge \sqrt{3}r_s$, our method works the same as the coverage-first method in each individual region, so they will use the same number of sensors.

3.4.2 Evaluations of the Proposed Dispatch Schemes

The second experiment evaluates different dispatch schemes. The sensing field A is a 900m $\times$ 900m square. The region of interest I is a 300m $\times$ 300m square located at the center of A. Sensors are randomly scattered in the region of $A - I$. With the setting of $(r_c, r_s) = (28, 16), (23.5, 13.45), (21, 12), (19.5, 11.05), (17.5, 10.1), (16.5, 9.45)$, and $(15.5, 8.9)$, we will need 150, 200, 250, 300, 350, 400, and 450 sensors, respectively, to be dispatched to I, according to our sensor placement algorithm. To fairly compare the centralized and the distributed schemes, the number of sensors is intentionally set to the required number of sensors in I. Sensors' initial energies are randomly selected from $[1000, 1500]$ units, and we set moving cost $\Delta_m = 1$ energy unit per meter. For comparison, we also design a *random* method, where we arbitrarily select a sensor to move to each location in a centralized manner. Fig. 3.12 shows the simulation results under different numbers of sensors required in I. From Fig. 3.12(a), we can observe that our centralized method (using Eq. (3.1) as the objective function) consumes the least energy compared to other methods. This is a result of our maximum matching approach. The distributed method consumes more energy than the centralized method since our greedy strategy can make local decisions. The similar result can be observed from Fig. 3.12(b), where the centralized method (using Eq. (3.2) as the objective function) can achieve the highest average remaining energy of sensors in I. From Fig. 3.12(b), we can observe that the average remaining energy decreases as the number of sensors increases when the distributed method is adopted. This is because when the number of sensors increases, the probability that two sensors select the same destination also increases. In this case, sensors have to reselect new destinations, and thus consume more energy. Note that under both objective functions, the random method always consumes the most energy, even though sensors are selected in a centralized manner. This reflects the importance of the dispatch issue since blindly moving sensors will lead to shorten the network lifetime.

3.5 Summary

In this chapter, we have proposed systematical solutions for sensor placement and dispatch. Our solution to the sensor placement problem allows a sensing field of shape as an arbitrary polygon with possible existence of obstacles. Thus, the result can be used for an indoor environment. Our solution also allows an arbitrary relationship of sensors' communication distances and their sensing distances. It is verified that the proposed schemes require fewer sensors to ensure full coverage of the sensing field and connectivity

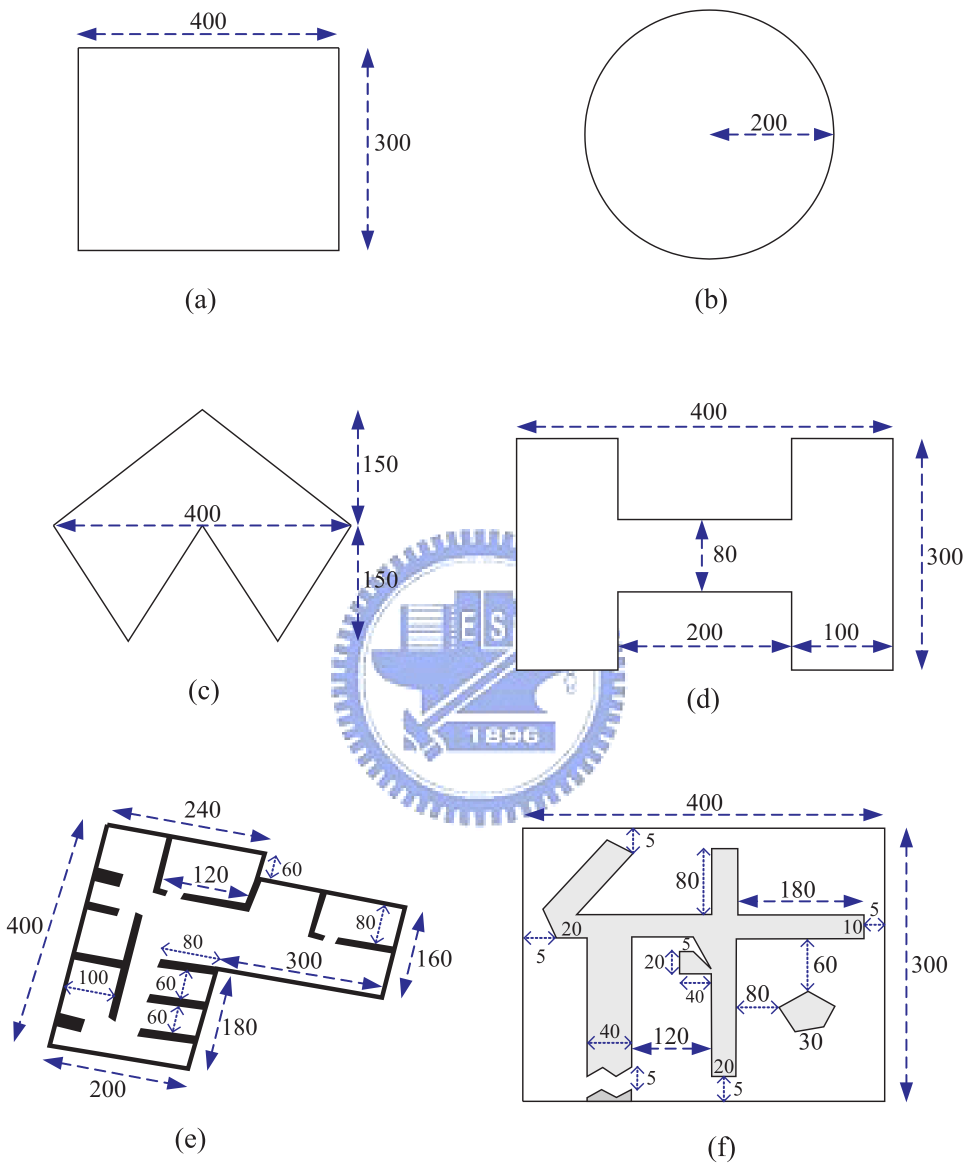


Figure 3.10: Six types of sensing fields used in the simulations: (a) a rectangle, (b) a circle, (c) a non-convex polygon, (d) a H-shaped region, (e) the office example in Fig. 3.2, and (f) the arbitrary-shaped region in Fig. 3.4. Note that the unit of length is in meter.

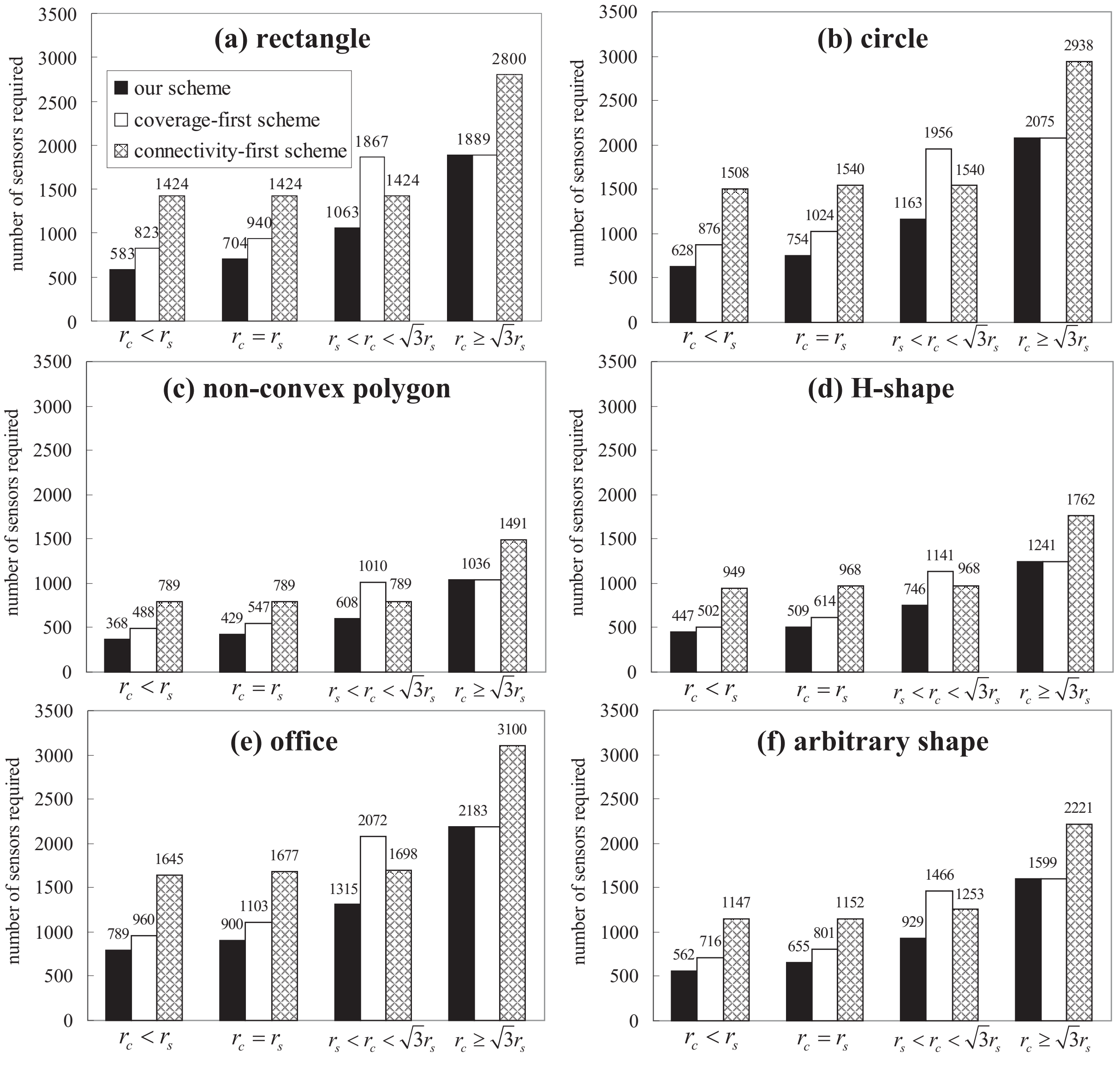
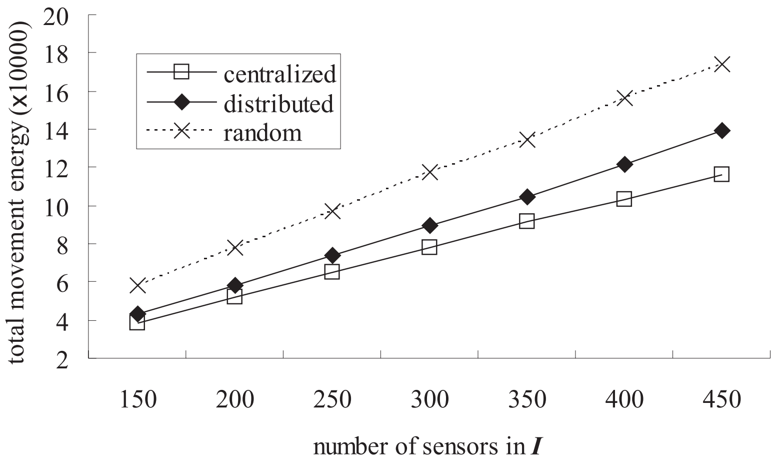
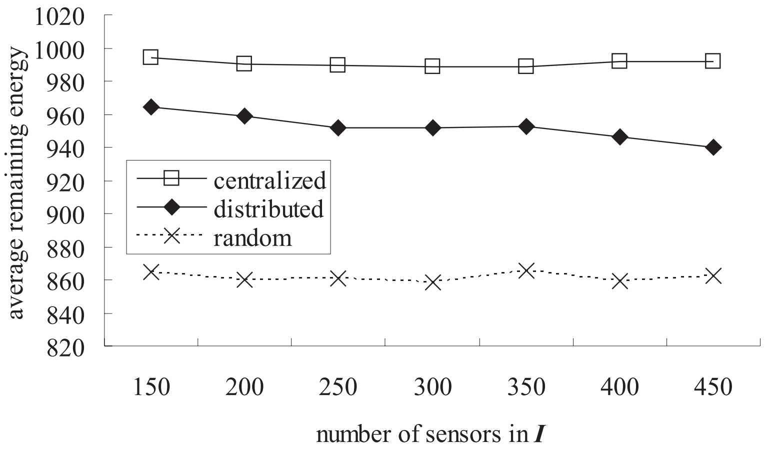


Figure 3.11: Comparison of numbers of sensors required under different sensing fields.



(a)



(b)

Figure 3.12: Comparison of different dispatch methods: (a) the total energy consumption due to movement when using Eq. (3.1) as the objective function, and (b) the average remaining energy of sensors when using Eq. (3.2) as the objective function.

of the network as compared to the coverage-first and connectivity-first schemes in various types of sensing fields. In addition, a new sensor dispatch problem is defined and two energy-efficient dispatch algorithms are presented to move sensors to the target locations determined by our sensor placement scheme.

Chapter 4

Deployment of a Wireless Sensor Network for Multi-level Coverage

This chapter considers the *k-coverage sensor deployment problem* to ensure multi-level (k) coverage of the region of interest. In particular, given a region of interest, we say that the region is *k-covered* if every point in the region can be monitored by at least k sensors, where k is a given parameter. Many applications and protocols may impose the requirement of $k > 1$. For example, positioning protocols using triangulation [85, 87, 100] require at least three sensors (i.e., $k \geq 3$) to monitor each location where an object may appear. Military applications with a strong monitoring requirement may impose that $k \geq 2$ to avoid leaving uncovered holes when some sensors are destroyed. To conduct data fusion and to minimize the impact of sensor failure, some strategies [63, 111] are based on the assumption of $k \geq 3$. Also, to prolong a sensor network's lifetime, sensors are divided into multiple sets, each of which can cover the whole area, to work in shifts [17, 106, 114]. This also requires that $k > 1$.

In this chapter, we consider the sensor deployment problem by assuming that

- multiple-level coverage of the area of interest is required.
- connectivity of sensor nodes (in terms of communications) should be maintained.
- the area of interest may change over time.
- sensor nodes are autonomous and mobile and thus can move to desired locations as instructed by one's deployment strategy.

Our general goals are to reduce the number of sensor nodes and to minimize the energy consumption due to movement.

In this chapter, we deal with the k-coverage sensor deployment problem by two subproblems: *k-coverage sensor placement problem* and *distributed sensor dispatch problem.*

The placement problem asks how to determine the minimum number of sensors required and their locations in I to ensure that I is k-covered and that the network is connected. Assuming that there are sufficient sensors and sensors are arbitrarily placed in the sensing field, the goal of the dispatch problem is to determine the schedule of sensors' movements to the designated locations according to the result computed by the placement strategy such that the total energy consumption for movement is minimized. For the placement problem, we allow an arbitrary relationship between sensors' communication distance and their sensing distance. We propose two solutions to the placement problem. The first one is based on an intuitive duplication idea. The second one is based on a more complicated interpolating idea and thus can save the number of sensors required. For the dispatch problem, we propose two distributed approaches. The first approach assumes that sensors have the knowledge of all target locations in the area of interest; sensors will then compete with each other for moving toward their closest locations. The second approach relaxes the above assumption in a way that sensors can derive other target locations based on several known locations, according to the patterns in our placement strategies. Therefore, the server only needs to generate several seed locations in the beginning, and then sensors can extend their range based on the placement pattern in a distributed manner.

4.1 Problem Statement

We are given a sensing field A, a region of interest I inside A, and a set of mobile sensors S resident in A. Each sensor has a communication distance r_c and a sensing distance r_s. Two sensors can communicate with each other if their distance is no larger than r_c. A point in A is *k-covered* if it is within k sensors' sensing regions, and an area in A is *k-covered* in a similar sense, where k is a given integer. We assume that sensors are homogenous, but the relationship of r_c and r_s can be arbitrary.

Given an integer k, the *k-coverage sensor deployment problem* has two sub-problems: *k-coverage sensor placement problem* and *distributed sensor dispatch problem*. The objective of the placement problem is to determine the minimum number of sensors required and their locations in the area of interest I to ensure that I is k-covered and that the network is connected. Assuming that mobile sensors are arbitrarily placed in A and that there are sufficient sensors, the dispatch problem asks how to move some sensors to designated locations in a distributed manner according to the result computed by the above placement strategy such that the total energy consumption for movement is minimized, i.e.,

$$\min \sum_{i \in S} \Delta_m \times d_i, \tag{4.1}$$

where Δ_m is the energy cost to move a sensor in one unit-distance and d_i is the total distance that sensor i is moved. Clearly, Eq. (4.1) can be treated as minimizing the total moving distance of sensors.

4.2 *k*-Coverage Sensor Placement Schemes

In this section, we propose two solutions to the placement problem. The first one is based on a simple duplication idea. The second one is based on a more complicated interpolating idea.

4.2.1 A Naive Duplicate Scheme

The basic idea of this scheme is to use a good sensor placement scheme to determine the locations of sensors to ensure 1-coverage and connectivity in the region of interest I, and then duplicate k sensors on each designated location. For the 1-coverage placement, we adopt the method mentioned in Section 3.2.3, which suggests to place sensors row by row, where each row of sensors will guarantee continuous coverage and connectivity and adjacent rows will guarantee continuous coverage of the area. Recall that according to the relationship of r_c and r_s, we separate the discussion into two cases, as shown in Fig. 4.1. When $r_c < \sqrt{3}r_s$, sensors on each row are separated by a distance of r_c, so the connectivity of sensors in each row can be guaranteed. Since $r_c < \sqrt{3}r_s$, each row of sensors can cover a belt-like area of width $2 \times \sqrt{r_s^2 - \frac{r_c^2}{4}}$. Adjacent rows will be separated vertically by a distance of $r_s + \sqrt{r_s^2 - \frac{r_c^2}{4}}$ and shifted horizontally by a distance of $\frac{r_c}{2}$. This guarantees the coverage of the whole area. When $r_c \geq \sqrt{3}r_s$, the aforementioned placement will use too many sensors, so a typical hexagon placement in which adjacent sensors are regularly separated by a distance of $\sqrt{3}r_s$ should be adopted.

After determining the 1-coverage placement, we can duplicate k sensors in each location to ensure k-coverage. Note that in the case of $r_c < \sqrt{3}r_s$, since the distance between adjacent rows is larger than r_c, it is necessary to add a column of sensors, each separated by a distance no larger than r_c, to connect adjacent rows.

4.2.2 An Interpolating Placement Scheme

The previous duplicate scheme may result in some sub-regions in I that have coverage levels much higher than k. Intuitively, the following interpolating placement scheme will try to balance the coverage levels of sub-regions. Observe that in Fig. 4.1(a), a large amount of sub-regions in a row are more than 1-covered. So we can "reuse" these sub-regions when generating a multi-level coverage placement. Based on this observation, the

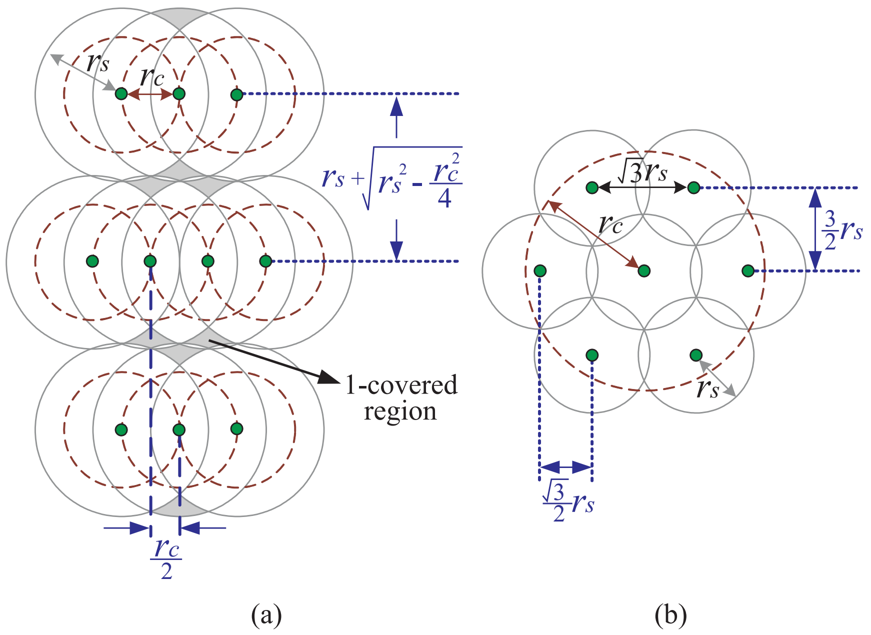


Figure 4.1: A 1-coverage sensor placement scheme: (a) the case of $r_c < \sqrt{3}r_s$ and (b) the case of $r_c \geq \sqrt{3}r_s$.

interpolating placement scheme will first find out those insufficiently covered sub-regions and then place the least number of sensors to cover these regions. Note that these newly-added sensors should remain connected with the formerly placed sensors. According to the relationship of r_c and r_s, we separate the discussion into three cases.

Case (1): $r_c \leq \frac{\sqrt{3}}{2}r_s$. From Fig. 4.1(a), we can observe that the insufficiently covered sub-regions (i.e., only 1-coverage) are located between adjacent rows (marked by gray). If we add an extra row of sensors between each pair of adjacent rows in Fig. 4.1(a), as shown in Fig. 4.2, the coverage level of the sensing field will directly become three. Here each extra row is placed above the previous row by a distance of r_s, and neighboring sensors in each extra row are still separated by a distance of r_c. To summarize, the duplicate scheme uses $3x$ rows of sensors to ensure 3-coverage of a belt-like area of width $(x-1)r_s + (x+1)\cdot\sqrt{r_s^2 - \frac{r_c^2}{4}}$, while this interpolating scheme uses $2x+1$ rows of sensors to ensure 3-coverage of the same region.

In general, for $k > 3$, we can apply $\lfloor \frac{k}{3} \rfloor$ times of the above 3-coverage placement and apply $(k \mod 3)$ times of the 1-coverage placement to achieve k-coverage of I. Therefore, while the duplicate placement requires kx rows of sensors to cover a region, this interpolating placement only requires $\left(\lfloor \frac{k}{3} \rfloor(2x+1) + (k \mod 3)\cdot x\right)$ rows of sensors.

Note that in this case, since $r_c \leq \frac{\sqrt{3}}{2}r_s$, the distance between two adjacent rows may be larger than r_c. Thus, we have to add a column of sensors between two adjacent rows, each separated by a distance no larger than r_c, to connect them.

Case (2): $\frac{\sqrt{3}}{2}r_s < r_c \leq \frac{2+\sqrt{3}}{3}r_s$. In this case, if the desired k is 2, we can directly

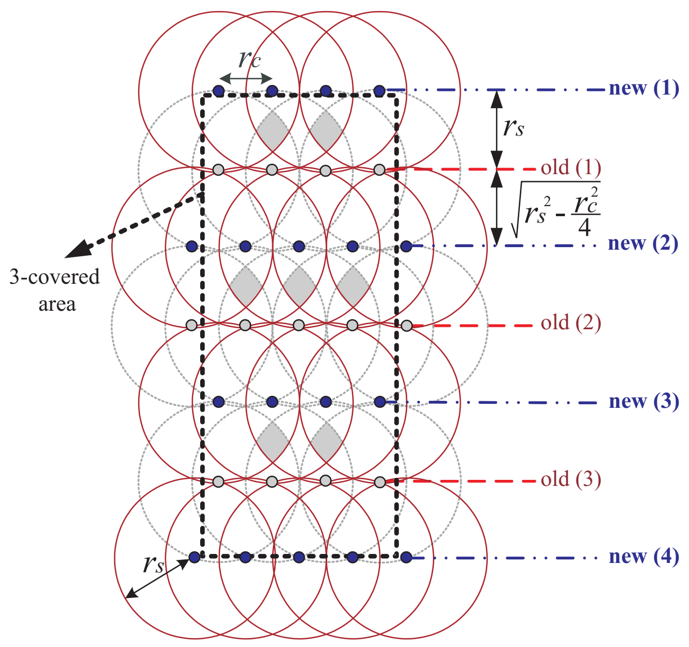


Figure 4.2: The interpolating placement scheme for the case of $r_c \leq \frac{\sqrt{3}}{2} r_s$.

apply the same placement as in case 1. The result is as shown in Fig. 4.3(a). However, because the sensing distance is relative smaller (as opposed to case 1) in the placement in Fig. 4.3(a), there are some sub-regions that are only 2-covered, but not 3-covered (marked by gray in Fig. 4.3(a)). Therefore, if the desired k is 3, we need to add one extra row of sensors between each old row i and new row i, as shown in Fig. 4.3(b), marked as new'. Note that the extra rows are shifted horizontally by a distance of $\frac{r_c}{2}$ from the previous rows and neighboring sensors are separated regularly by a distance of $2r_c$. To summarize, the duplicate scheme uses $3x$ rows of sensors to ensure 3-coverage of a belt-like area of width $(x-1)r_s + (x+1) \cdot \sqrt{r_s^2 - \frac{r_c^2}{4}}$, while this interpolating scheme uses $2.5x + 1$ rows of sensors to ensure 3-coverage of the same region (the third addition of rows only needs half of sensors compared with earlier ones).

In general, for $k > 3$, we can also apply $\lfloor \frac{k}{3} \rfloor$ times of the above 3-coverage placement and apply $(k \mod 3)$ times of the 1-coverage placement to achieve k-coverage of I. Therefore, while the duplicate placement requires kx rows of sensors to cover a region, this interpolating placement only requires $\left(\lfloor \frac{k}{3} \rfloor (2.5x + 1) + (k \mod 3) \cdot x \right)$ rows of sensors.

Note that with this placement, sensors on new' rows can communicate with their neighbors in the adjacent old and new rows, as shown in Fig. 4.4. In particular, since

$$d = \sqrt{\left(\frac{r_s}{2}\right)^2 + \left(\frac{r_c}{2}\right)^2} < \frac{1}{2}\sqrt{\left(\frac{2r_c}{\sqrt{3}}\right)^2 + r_c^2} < r_c,$$

the sensor s_n on a new' row can communicate with its four neighbors s_a, s_b, s_c, and s_d in the adjacent rows.

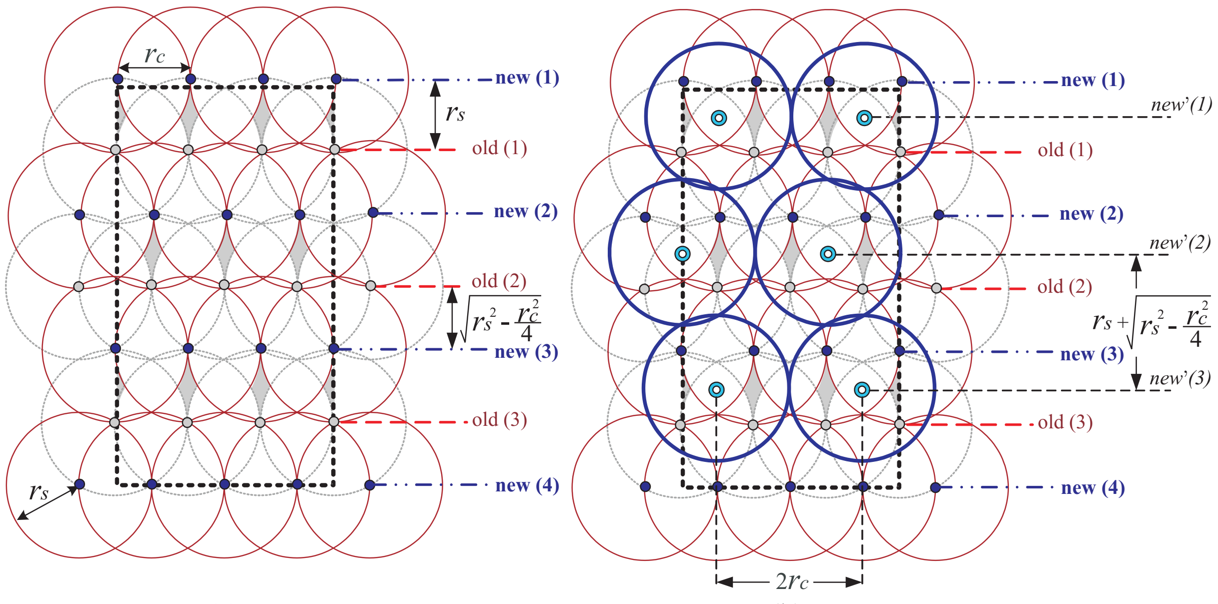


Figure 4.3: The interpolating placement scheme for the case of $\frac{\sqrt{3}}{2}r_s < r_c \leq \frac{2+\sqrt{3}}{3}r_s$: (a) the placement for $k = 2$ and (b) the placement for $k = 3$.

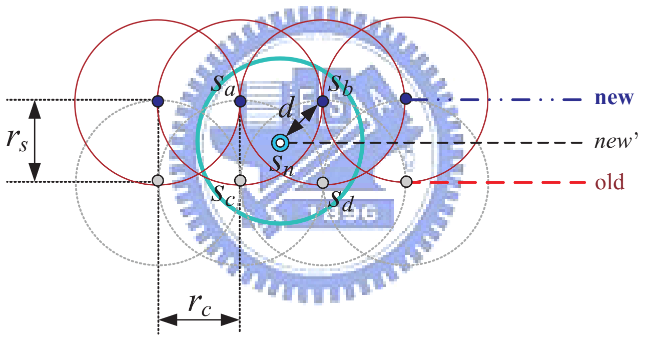


Figure 4.4: An example to show that the connectivity between a new' row and its adjacent rows is guaranteed.

Case (3): $r_c > \frac{2+\sqrt{3}}{3}r_s$. In this case, this interpolating placement will not save sensors compared with the duplicate placement, so we adopt the duplicate scheme in this case.

4.3 *Distributed Sensor Dispatch Schemes*

After determining the locations to be placed with sensors, the next issue is how to move existing sensors in the sensing field A to the designated locations in I such that the energy consumption due to movement is minimized. In this section, we assume that sensors are mobile and thus autonomous solutions are desired. Two distributed dispatch schemes are then proposed.

4.3.1 A Competition-based Dispatch Scheme

We assume that there is an external server, which will compute the locations in I to be deployed with sensors and broadcast the designated locations to all mobile sensors. On receiving the notification, sensors will compete with each other to move to these locations. This scheme involves the following rules:

Rule 1: The server first computes the set $L = \{(x_1, y_1, n_1), (x_2, y_2, n_2), \cdots, (x_m, y_m, n_m)\}$ according to the placement scheme in Section 4.2, where each element (x_j, y_j, n_j), $j = 1..m$, means that n_j sensors need to be deployed on location (x_j, y_j). The server then broadcasts L to all sensors.

Rule 2: On receiving L for the first time, each sensor s_i constructs an array $OCC[1..m]$ such that each entry $OCC[j] = \{(s_{j_1}, d_{j_1}), (s_{j_2}, d_{j_2}), \ldots, (s_{j_\alpha}, d_{j_\alpha})\}$, $\alpha \leq n_j$, contains the set of sensors that have already moved into, or are still on their ways moving toward, location (x_j, y_j) and their corresponding distances to (x_j, y_j). Specifically, entry $(s_{j_\beta}, d_{j_\beta})$, $\beta = 1..\alpha$, means that sensor s_{j_β} has chosen to cover location (x_j, y_j) and its current estimated distance to (x_j, y_j) is d_{j_β}. When $d_{j_\beta} = 0$, it means that sensor s_{j_β} has already arrived at (x_j, y_j). Initially, $OCC[j] = \emptyset$ for all $j = 1..m$. To simplify the presentation, we say that a location (x_j, y_j) is *covered* if a sufficient number n_j of sensors have committed to move toward (x_j, y_j) (i.e., $|OCC[j]| = n_j$); otherwise, (x_j, y_j) is *uncovered*. A sensor s_i is *engaged* if it has chosen to move to, or already moved into, any location in L; otherwise, it is *free* or *terminated*. The initial state of each sensor is free. A free sensor will try to become engaged and move toward a destination. When it finds that there is no location that it can cover, it will enter the terminated state. Fig. 4.5 illustrates the state transition diagram of a sensor.

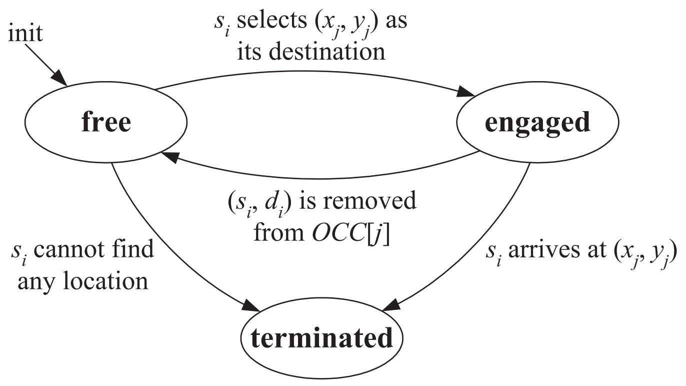

init
s_i selects (x_j, y_j) as its destination
free
engaged
(s_i, d_i) is removed from $OCC[j]$
s_i cannot find any location
s_i arrives at (x_j, y_j)
terminated

Figure 4.5: The state transition diagram of each sensor s_i in the competition-based dispatch scheme.

Rule 3: When the state of a sensor s_i is free, it will select a location in L to be its destination as follows:

- The first priority is to consider uncovered locations. Specifically, if there is a location (x_j, y_j) such that $|OCC[j]| < n_j$, then (x_j, y_j) will be considered first. If multiple locations are qualified, then the (x_j, y_j) such that $d(s_i, (x_j, y_j))$ is minimized will be selected, where $d(s_i, (x_j, y_j))$ is the distance between s_i's current position to (x_j, y_j). In this case, s_i will add an entry $(s_i, d(s_i, (x_j, y_j)))$ in its $OCC[j]$ and enter the engaged state.
- If all locations are already covered, then s_i will select a location (x_j, y_j) such that there exists an entry $(s_k, d_k) \in OCC[j]$ and $d(s_i, (x_j, y_j)) < d_k$. If multiple locations are qualified, then the (x_j, y_j) such that $d(s_i, (x_j, y_j)) - d_k$ is maximized will be selected. In this case, s_i will replace the entry $(s_k, d_k) \in OCC[j]$ by a new entry $(s_i, d(s_i, (x_j, y_j)))$ in $OCC[j]$ and enter the engaged state.

If s_i becomes engaged, it will begin moving toward that location. Otherwise, s_i will enter the terminated state since it does not need to cover any location.

Rule 4: For maintenance purpose, each sensor s_i will periodically perform the following two actions:

- Broadcasting its current status to its direct neighbors, including its ID, its $OCC[1..m]$ array, and its current location and state.
- Updating its $OCC[1..m]$ array as follows. For each $(s_{j_\beta}, d_{j_\beta}) \in OCC[j]$, $j = 1..m$, decrease d_{j_β} by the expected moving distance of s_{j_β} during the past period of time, until $d_{j_\beta} = 0$.

The above actions can be controlled by setting two timers $T_{broadcast}$ and T_{update_OCC}. Also note that the update of the $OCC[1..m]$ array is based on the assumption that sensors all move in the same constant speed (if this assumption is not valid, then d_{j_β} is only an estimated distance to (x_j, y_j) and it is not hard to make such an extension).

Rule 5: When a sensor s_i receives an update message from a sensor s_k, two actions will be taken.

- First, it has to update its $OCC[1..m]$ array as follows. Let us denote by $OCC_i[1..m]$ and $OCC_k[1..m]$ the arrays of s_i and s_k, respectively. Specifically, for each $j = 1..m$, we will calculate the union:

$$U_j = OCC_k[j] \cup OCC_i[j].$$

If $|U_j| \leq n_j$, we will replace $OCC_i[j]$ by U_j. Otherwise, it means that there are too many mobile sensors that are scheduled to cover (x_j, y_j), in which case we will truncate those entries in U_j that have longer moving distances, until the size $|U_j| = n_j$. Then we will replace $OCC_i[j]$ by the truncated U_j. The above merge of two sets may lead to a special case that s_i was in the original $OCC_i[j]$, but is not in the new $OCC_i[j]$ (which means that s_i has been replaced by some other sensors with a shorter distance to (x_j, y_j)). If so, s_i should change its state from engaged to free and execute Rule 3.

- After the above merge, if s_i remains engaged, say, with (x_j, y_j) as its destination, then we will do the following optimization. We will check if

$$d(s_i, (x_l, y_l)) + d(s_k, (x_j, y_j)) < d(s_i, (x_j, y_j)) + d(s_k, (x_l, y_l)),$$

where (x_l, y_l) is the destination of s_k. If so, it means that the total moving distance of s_i and s_k can be reduced if we exchange their destinations. If so, s_i will communicate with s_k for this trade (which is not hand to realize, so we omit the details). Once the trade is confirmed, s_i will replace the entries (s_i, d_i) and (s_k, d_k) in $OCC_i[j]$ and $OCC_j[l]$ by the new entries $(s_k, d(s_k, (x_j, y_j)))$ and $(s_i, d(s_i, (x_l, y_l)))$, respectively.

In the above steps, if any entry in $OCC_i[1..m]$ array has been changed, s_i will broadcast the content to its direct neighbors.

Rule 6: When a sensor s_i is in the engaged state, it will keep moving toward (x_j, y_j). However, in case that s_i is removed from its current $OCC_i[j]$ set as specified in Rule 5, it will stop moving and change its state to free.

Rule 7: When an engaged sensor s_i arrives at its destination (x_j, y_j), it will change its state to terminated and begin its monitoring job at the designated location. Meanwhile, it will still execute the maintenance actions in Rule 4, until the server commands it not to do. Since the server will eventually see that I is k-covered (by receiving sensing reports from sensors), it can notify all sensors to exit from this dispatch algorithm.

To verify the correctness of this competition-based scheme, we have to show that every location (x_j, y_j) in L will eventually be covered by n_j sensors. Rule 6 guarantees that an engaged sensor s_i will eventually arrive at the location (x_j, y_j) if the entry (s_i, d_i) remains in s_i's $OCC[j]$. If (s_i, d_i) is removed during s_i's movement toward (x_j, y_j) (by Rule 5), then it means that either another sensor s_k trades (x_l, y_l) with s_i or s_i loses the competition. In the former case, locations (x_j, y_j) and (x_l, y_l) are covered by s_k and s_i, respectively. In the latter case, it means that (x_j, y_j) has been committed by more than n_j sensors, so it is safe to remove (x_j, y_j). In this case, s_i has to execute Rule 3 to reselect a destination. If s_i finds that $|OCC[j]| = n_j$ for all $j = 1..m$, then every location in L has been committed by sufficient sensors. Thus all locations will be eventually covered by n_j sensors. So the competition-based dispatch scheme can work correctly when there are sufficient sensors.

4.3.2 A Pattern-based Dispatch Scheme

The aforementioned competition-based scheme requires that each sensor have the knowledge of all target locations in the region of interest. In this section, we propose a pattern-based scheme that allows sensors to derive their target locations based on some known locations, thus relaxing the above assumption.

Observe that our placement schemes in Section 4.2 actually deploy sensors with some regular patterns. In the duplicate placement scheme, sensors will be placed in a hexagon-like fashion. Thus, each sensor at location (x, y) can derive its potential six neighbors' positions according to Table 3.1. When the interpolating placement scheme is adopted, the pattern will be changed according to the relationship of r_c and r_s. There are three cases to be discussed:

- $r_c \le \frac{\sqrt{3}}{2} r_s$. Recall Fig. 4.2. There are two patterns 1 and 2, which will repeat in each new row and old row, as shown in Fig. 4.6 (a). Thus, a sensor s_i located at (x, y) can derive its five neighbors' positions according to its pattern. Moreover, s_i can also derive the patterns of its neighbors depending on its own pattern (indicated by the numbers inside circles in Fig. 4.6(a)).

- $\frac{\sqrt{3}}{2} r_s < r_c \le \frac{2+\sqrt{3}}{3} r_s$. In this case, if the desired coverage level k is 2, we can directly apply the patterns in the previous case. However, when $k \ge 3$, there is an extra row (marked as new') between each old and new rows in Fig. 4.3(b). This will result in four deployment patterns, as shown in Fig. 4.6(b), depending on a sensor's location and its neighbors' locations. Thus, a sensor s_i located at (x, y) can derive its neighbors' positions according to its pattern. Moreover, s_i can also derive the patterns of its neighbors depending on its own pattern (indicated by the numbers inside circles in Fig. 4.6(b)). Note that we do not derive the pattern for sensors at the extra new' rows (although this is feasible, deriving these patterns will complicate the problem a lot). That's why sensors marked by double circles are not assigned with any pattern number.

- $r_c > \frac{2+\sqrt{3}}{3} r_s$. In this case, since the duplicate placement scheme is adopted, a sensor can compute its neighbors' positions according to Table 3.1.

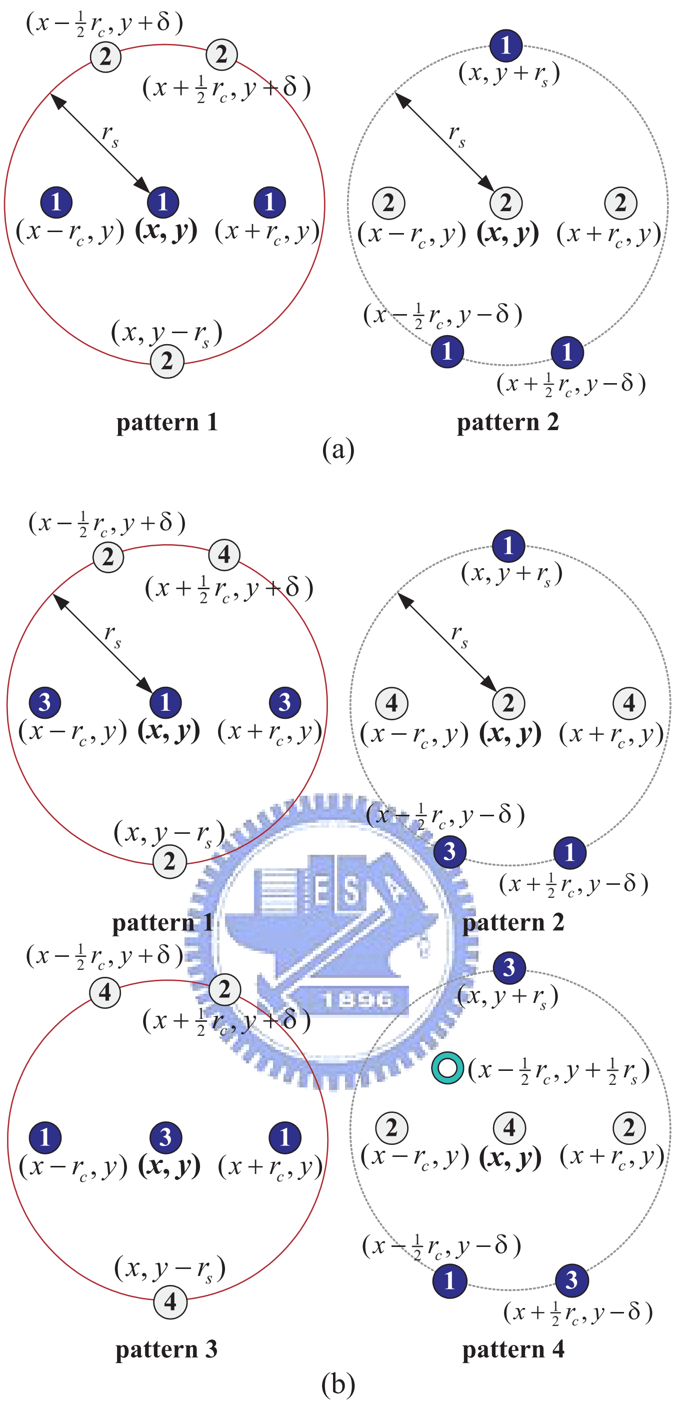

$(x-\frac{1}{2}r_c, y+\delta)$
$(x+\frac{1}{2}r_c, y+\delta)$
r_s
$(x-r_c, y)$
(x, y)
$(x+r_c, y)$
$(x, y-r_s)$
pattern 1
$(x, y+r_s)$
$(x-\frac{1}{2}r_c, y-\delta)$
$(x+\frac{1}{2}r_c, y-\delta)$
pattern 2
(a)
pattern 3
$(x-\frac{1}{2}r_c, y+\frac{1}{2}r_s)$
pattern 4
(b)

Figure 4.6: The patterns in the interpolating placement, where $\delta = \sqrt{r_s^2 - \frac{r_c^2}{4}}$: (a) the case of $r_c \le \frac{\sqrt{3}}{2} r_s$ and (b) the case of $\frac{\sqrt{3}}{2} r_s < r_c \le \frac{2+\sqrt{3}}{3} r_s$.

To summarize, the above observations allow a sensor to derive its direct neighbors as well as the patterns to be used by them. This property allows us to expand from a partial deployment to a full deployment of sensors in I.

Based on the aforementioned observations, the pattern-based dispatch scheme works as follows. The server first computes a set of seed locations $L' = \{(x_1, y_1, n_1, p_1), (x_2, y_2, n_2, p_2), \cdots, (x_\alpha, y_\alpha, n_\alpha, p_\alpha)\}$, which is a partial list of locations to be deployed with sensors, where

p_j is the pattern used by the sensor at location (x_j, y_j) (intuitively, L' can be considered as a subset of L). Then the server broadcasts L' to all sensors. Note that these seed locations in L' should be sparsely distributed in the region of interest. On receiving L', each sensor executes the competition-based scheme to compete for their closest locations in L'. However, Rules 3 and 7 in the competition-bases scheme should be modified as follows:

- **Revised Rule 3:** When a free sensor s_i cannot find any available location from its current $OCC[\cdot]$ array, it will compute some new locations based on the known locations in $OCC[\cdot]$. Then s_i will re-execute Rule 3 and try to find a new destination. However, if s_i cannot derive any new location from its current L' (which means that $L' = L$), then s_i will enter the terminated state since it does not need to cover any location.

- **Revised Rule 7:** When an engaged sensor s_i arrives at its destination, it will derive some new locations from its current L' and add the corresponding new entries in its $OCC[\cdot]$ array.

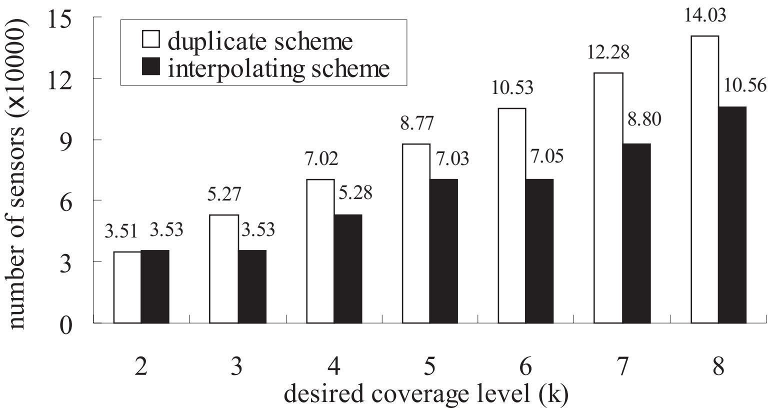
Since a sensor can either derive new locations by itself (according to new Rule 3) or learn new locations from other sensors (by Rule 4), the complete information of L can thus be propagated throughout the network. Thus, the pattern-based scheme can work correctly when there are sufficient sensors.

4.4 Experimental Results

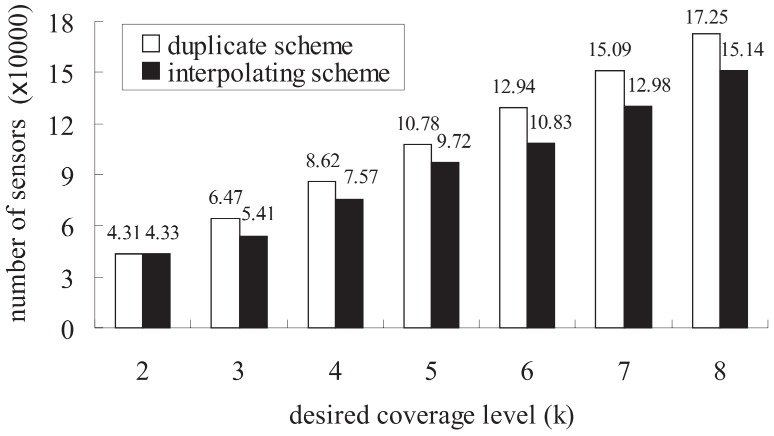
In this section, we present some experimental results to verify the effectiveness of our proposed schemes. The evaluation includes three parts. First, we measure the numbers of sensors required by different placement schemes. Second, we estimate the performances of our dispatch schemes. Finally, we will study the effect of seed locations on the pattern-based dispatch scheme.

4.4.1 Evaluations of the Proposed Placement Schemes

The first experiment measures the number of sensors required by different placement schemes. We design a region of interest I as a 1000×1000 square region to be deployed with sensors. To observe the effects under different relationships of r_c and r_s, we set the values of (r_c, r_s) as $(5, 6)$ and $(5, 5)$, which correspond to the case of $r_c \leq \frac{\sqrt{3}}{2} r_s$ and $\frac{\sqrt{3}}{2} r_s < r_c \leq \frac{2+\sqrt{3}}{3} r_s$, respectively. Fig. 4.7 illustrates the numbers of sensors required by the duplicate and interpolating schemes when the desired coverage level k increases from two to eight. When $k = 2$, the interpolating scheme requires slightly more sensors compared with the duplicate scheme, because the former needs to add an extra row of sensors to ensure 2-coverage of I's boundary. However, when $k \geq 3$, the interpolating scheme can save approximately 20% $\sim$ 33% and 10% $\sim$ 16% sensors as opposed to the duplicate scheme in the case of $r_c \leq \frac{\sqrt{3}}{2} r_s$ and $\frac{\sqrt{3}}{2} r_s < r_c \leq \frac{2+\sqrt{3}}{3} r_s$, respectively.



(a)



(b)

Figure 4.7: Comparison on numbers of sensors required by the duplicate and interpolating placement schemes: (a) the case of $r_c \leq \frac{\sqrt{3}}{2} r_s$ and (b) the case of $\frac{\sqrt{3}}{2} r_s < r_c \leq \frac{2+\sqrt{3}}{3} r_s$.

We also compare our placement schemes with other k-coverage placement scheme proposed in [111], namely the *hexagon-like scheme* (refer to Fig. 2.7). To satisfy the assumption in [111] (as discussed in Section 2.1.2), we set $r_c = 5$ and $r_s = 2.5$. Fig. 4.8 shows the numbers of sensors required by the duplicate scheme and the hexagon-like scheme. We can observe that the proposed duplicate scheme requires fewer sensors compared with the hexagon-like scheme proposed in [111]. Note that in this case, since $r_c > \frac{2+\sqrt{3}}{3} r_s$, the interpolating scheme works the same as the duplicate scheme, so we omit its performance.

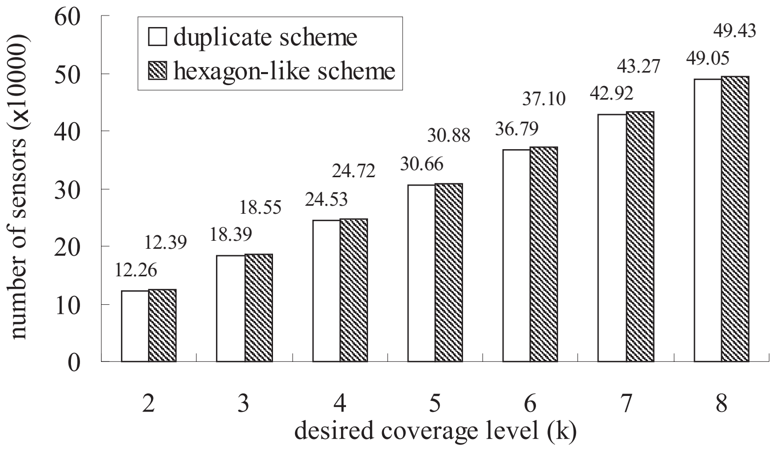
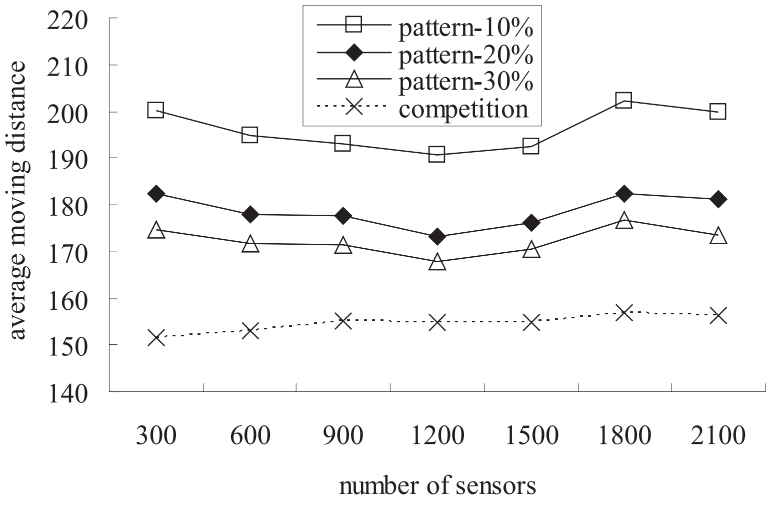


Figure 4.8: Comparison on numbers of sensors required by the duplicate and hexagon-like placement schemes, where $r_c \geq 2r_s$.

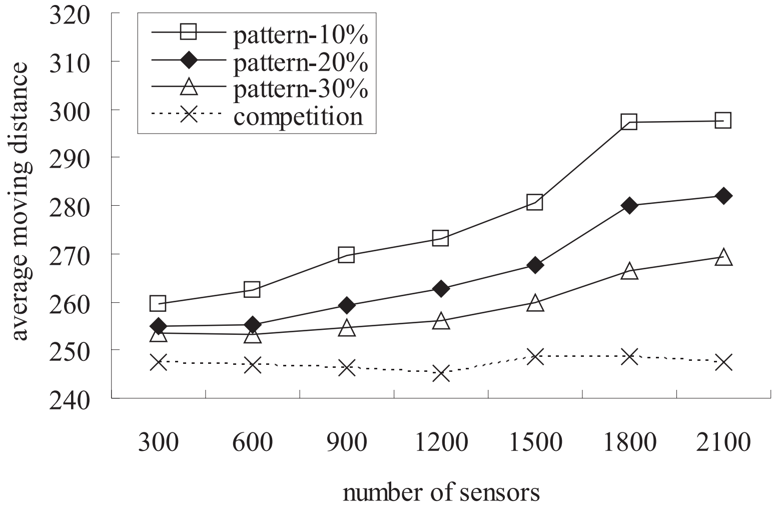
4.4.2 Performances of the Proposed Dispatch Schemes

The second experiment estimates the average moving distances of sensors when the competition-based and pattern-based dispatch schemes are adopted. We design a sensing field A as a 600×600 square region. The region of interest I is a 300×300 square region located at the center of A. Two scenarios, namely *hollow* and *concentrated*, are considered. In the hollow scenario, sensors are randomly placed inside the region of $A - I$, while in the concentrated scenario, sensors are randomly placed inside a 150×600 rectangle region located at the right-hand side of $A - I$. With the setting of $(r_c, r_s) = (34.7, 20.0)$, $(24.1, 13.9)$, $(19.3, 11.1)$, $(16.7, 9.62)$, $(14.9, 8.6)$, $(13.4, 7.71)$, and $(12.5, 7.16)$, we can obtain 100, 200, 300, 400, 500, 600, and 700 locations to be placed with sensors inside I, respectively, according to the interpolating placement scheme (in case 3). We set the desired coverage level $k = 3$, so that there will be 300, 600, 900, 1200, 1500, 1800, and 2100 sensors needed to be dispatched to I. The moving speed of each sensor is set to one unit-distance per second. The two timers $T_{broadcast}$ and T_{update_OCC} in both dispatch schemes are set to five seconds. In the pattern-based scheme, the server randomly selects 10%, 20%, and 30% target locations in I as the seed locations to be broadcasted for all sensors.

Fig. 4.9 presents the average moving distances of sensors under both scenarios. The competition-based scheme has a shorter average moving distance compared with the pattern-based scheme because sensors have the full knowledge of target locations inside I. However, the average moving distance of the patter-based scheme can decrease as we increase the number of seed locations. In this case, sensors can have more choices when selecting their destinations, and thus the number of competitions can be reduced. From Fig. 4.9, we can observe that the average moving distance is almost irrelative to the number of sensors in the competition-based scheme. This is because sensors have already known every target locations inside I. Thus, a sensor losing the competition can immediately find another uncovered location as its new destination. However, the average moving distance of the pattern-based scheme in the concentrated scenario will increase as the number of sensors increases. Since sensors are initially placed on the right-hand side of region $A - I$, most of them will select the right-most seed locations inside I as their destinations. This will cause a large number of competitions and thus most sensors have to reselect their destinations many times. So each sensor will have a longer moving distance. This situation will become worse as the number of sensors becomes larger.



(a)



(b)

Figure 4.9: Comparison on average moving distances of sensors under different scenarios: (a) the hollow scenario and (b) the concentrated scenario.

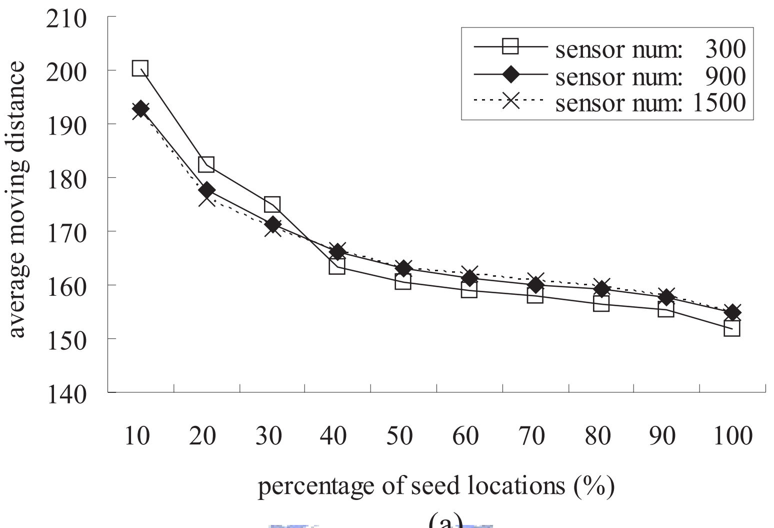
4.4.3 Effect of Seed Locations on the Pattern-based Scheme

In this section, with the same simulation setup, we evaluate the effect of seed locations on the average moving distance of sensors in the pattern-based dispatch scheme.

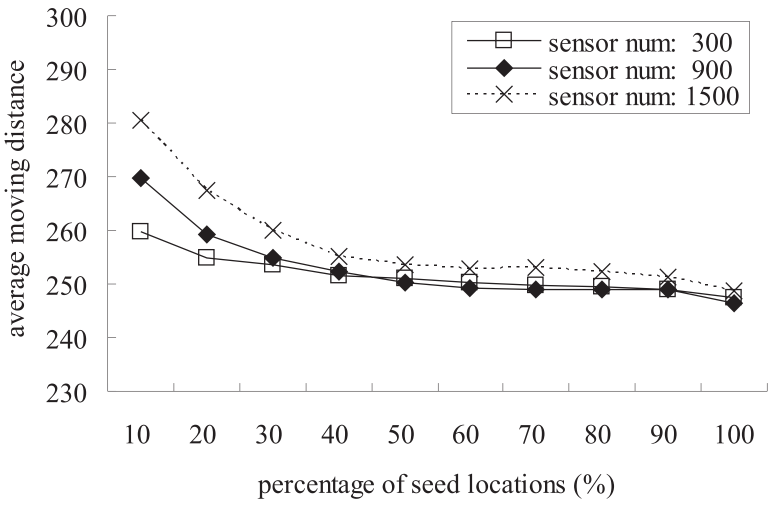
Simulation results are shown in Fig. 4.10. As can be seen, the average moving distance of sensors in the pattern-based scheme can be significantly reduced when we increase the number of seed locations. However, when there are more than 40% target locations selected as the seed locations, the improvement becomes quite limited in both scenarios. Therefore, the optimal value of seed locations in the pattern-based scheme is suggested to be 40% of target locations inside I. In this case, both the average moving distance of sensors and the number of seed locations broadcasted by the server can be minimized. Note that when the percentage of seed locations is 100%, the pattern-based scheme will perform as better as the competition-based scheme since all target locations are known by every sensor in the beginning.

4.5 Summary

In this work, we have proposed systematical solutions to the k-coverage sensor placement and dispatch problems. Our solutions to the placement problem allows an arbitrary relationship of sensors' communication distance and their sensing distance. It is verified that the interpolating scheme requires fewer sensors to ensure k-coverage of the sensing field and connectivity of the network as compared with the duplicate scheme. Our solutions to the dispatch problem are based on a competitive nature of a distributed network. Simulation results shows that the competition-based scheme works better than the pattern-based scheme, but the latter can significantly improve its performance by selecting 40% target locations as the seed locations.



(a)



(b)

Figure 4.10: Effect of seed locations on the average moving distance of sensors in the pattern-based dispatch scheme: (a) the hollow scenario and (b) the concentrated scenario.

Chapter 5

Dispatch of Mobile Sensors with Energy-efficient Consideration

A wireless sensor network is composed of many small devices used to monitor the environment. Due to their small sizes, these sensor nodes are usually simple [3] and may roughly describe the events occurred in the environment. Take a security application as an example. We may deploy a large amount of sensors that can detect noise in a region to check if somebody intrudes this region. However, these sensors can only report that something happens in the region when they have detected unusual sound. They cannot tell you what intrudes this region. In addition, some background noises such as winds will make these sensors generate false alarms. To solve these problems, we may use more sophisticated (and thus more expensive) sensors such as cameras to recognize the intruding object. However, it is unlikely to mount cameras on most sensor nodes because of their large number. Alternatively, a better way is to mount these sophisticated sensors on few mobile platforms, and then dispatch these *mobile sensors* to move to event locations to conduct more advanced analyses.

In this chapter, we thus consider a hybrid sensor network consisting of static and mobile sensors. The former is deployed in the sensing field to monitor the environment, while the latter is equipped with more resources such as computation power and sensing capability and can be dispatched to the event locations to conduct more in-depth analysis. Because mobile sensors use small batteries for their operations, one critical issue is to conserve the energy of mobile sensors. In particular, the energy cost due to movement is the dominated factor of energy consumption of mobile sensors. Thus, we focus on investigating how to efficiently dispatch mobile sensors to move to event locations such that the *system lifetime*, which is defined as the time period until some event locations cannot be reached by any mobile sensor due to lack of energy, can be maximized.

To solve this problem, one intuitive solution is to maximize the total remaining energy of mobile sensors in each one-round dispatch. Unfortunately, this method may cause some mobile sensors early to exhaust their energies, which results in shortening the system lifetime. In particular, we consider an example in Fig. 5.1, where there are two mobile sensors s_a and s_b located at l_1 and l_2, respectively. Both mobile sensors initially have an energy of 1000 units. Fig. 5.1(a) shows the energy consumption to visit each location, and we assume that a mobile sensor has to spend an energy of five units to conduct the sensing and communication jobs in the event location. Consider that there are two events occurring at locations l_3 and l_4 (respectively, l_1 and l_2) during each odd (respectively, even) round. Fig. 5.1(b) illustrates the execution of the aforementioned method. In order to maximize the total remaining energy, s_a and s_b are scheduled to move between the pair of locations (l_1, l_3) and (l_2, l_4), respectively. This will result in a minimum cost of 196 units during each round. However, after eight rounds, s_a exhausts its energy. In this case, s_b has to move to both locations l_4 and l_3 and thus remains 196 units of energy in the ninth round. Finally, in the tenth round, no mobile sensor can reach the event location l_2, so that the system lifetime is totally nine rounds. From Fig. 5.1(b), we can observe that maximizing the total remaining energy of mobile sensors during each one-round dispatch could make some mobile sensors early to exhaust their energy and thus burden other still alive ones. Alternatively, we can "balance" the loads of mobile sensors when dispatching them. Fig. 5.1(c) gives an example, where s_a and s_b are scheduled to move between the pair of locations (l_1, l_4) and (l_2, l_3), respectively. Although this load-balancing method spends more energy (i.e., 200 units) during each round, it can extend the system lifetime to ten rounds. From Fig. 5.1, we can conclude that simply maximizing the remaining energy of mobile sensors during each one-round dispatch cannot guarantee to maximize the system lifetime since unbalanced loads of mobile sensors will cause some mobile sensors fast to exhaust their energy, and thus greatly increase the loads of other still alive ones. Such a chain-reaction will make the system lifetime become shorter.

Based on the aforementioned observation, we thus propose an efficient dispatch method that takes the *load-balance* issue into consideration when scheduling mobile sensors to visit event locations. Our dispatch method is general in which the numbers of event locations and mobile sensors can be arbitrary. When the number of event locations is smaller than or equal to that of mobile sensors, we convert this dispatch problem to the problem of finding a maximum matching in a weighted bipartite graph. However, instead of finding a matching with a maximum edge weight, we adopt a *preference list* [1] and a *bound* to select the matching, where the former helps assign an event location with a suitable mobile sensor, while the latter prevents the matching from including those edges with extreme weights so that the loads of mobile sensors can be balanced. When the number of event

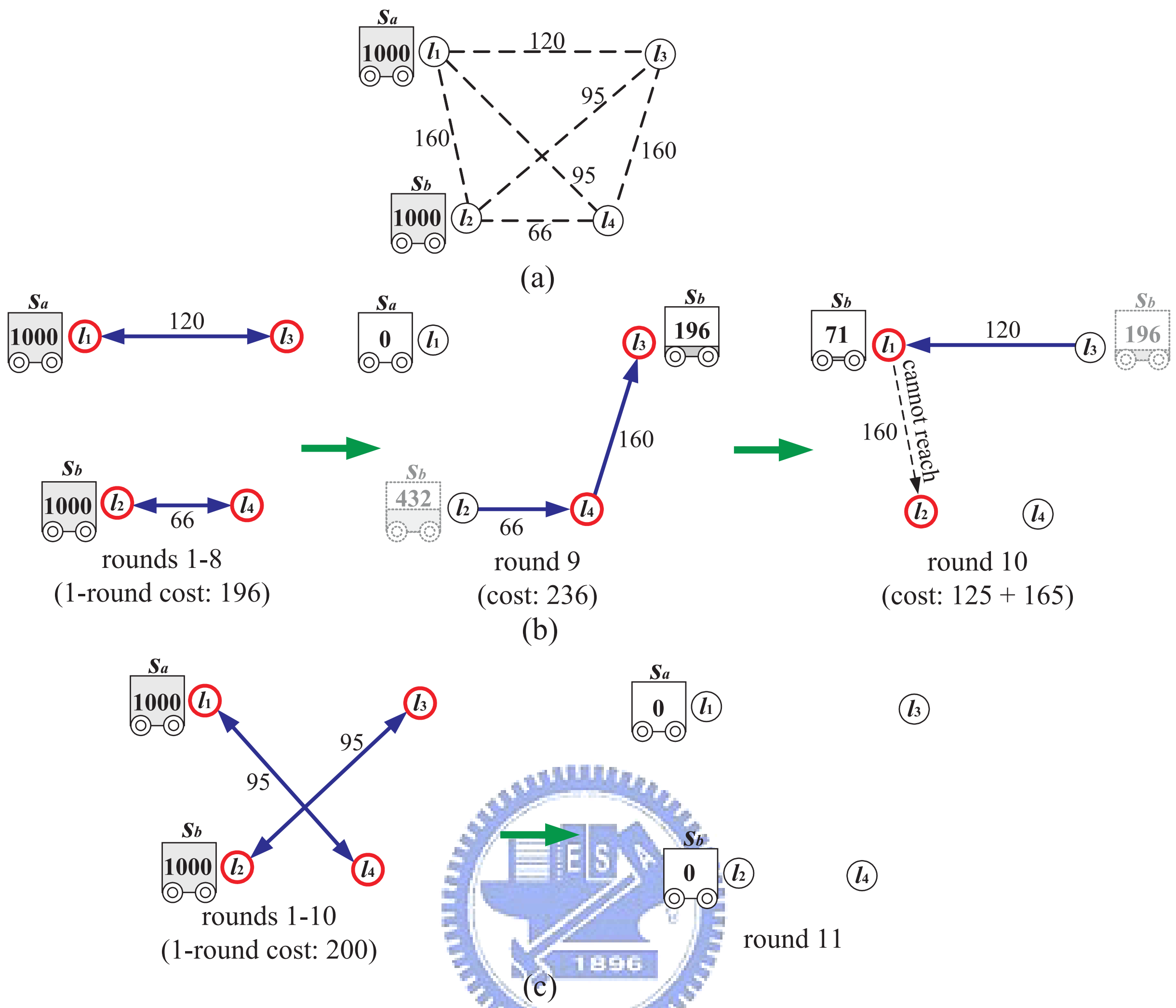


Figure 5.1: Comparison of different dispatch methods: (a) the energy consumption for a mobile sensor to move to each location, (b) the dispatch method by maximizing the total remaining energy during each one-round dispatch, and (c) the load-balancing method. Note that each mobile sensor has to spend an energy of five units to conduct the sensing and communication jobs after it arrives at the event location.

locations is larger than that of mobile sensors, we first group event locations into clusters, where the number of cluster is the same as that of mobile sensors, and then adopt the aforementioned matching approach to assign mobile sensors to visit these clusters. After a mobile sensor arrives at the assigned cluster, it can adopt the approximate solution of traveling-salesman problem to reach all event locations in that cluster.

5.1 *Problem Statement*

We are given a hybrid sensor network that consists of *static* and *mobile sensors.* Static sensors are assumed to fully cover the region of interest and form a connected network, so that they can completely monitor the region. When there are events reported from static sensors, a set of n mobile sensors $S = \{s_1, s_2, \ldots, s_n\}$ can be dispatched to the *event locations* to provide sensing results of higher quality. We assume that sensors can know their current locations, which are achieved by the global positioning system (GPS) [53] or other localization techniques [13, 56].

The mobile sensor dispatch problem is stated as follows. We consider that there is a set of event locations $L = \{l_1, l_2, \ldots, l_m\}$ reported from static sensors. Each location is to be visited by a mobile sensor. We allow an arbitrary relationship between the numbers of event locations (m) and mobile sensors (n). The objective of this dispatch problem is to calculate a *dispatch schedule* DS_i for each mobile sensor s_i such that every location in L can be visited by one mobile sensor exactly once. Each schedule DS_i is a sequence of event locations, and the jth location to be visited in the schedule is denoted as $DS_i[j]$. Let e_i be the current energy of a mobile sensor s_i and $c(DS_i)$ be the energy required to finish s_i's visit schedule, we can obtain

$$c(DS_i) = e_{move} \times \left(d(s_i, DS_i[1]) + \sum_{j=1}^{|DS_i|-1} d(DS_i[j], DS_i[j+1]) \right) + e_{job} \times |DS_i|,$$

where e_{move} is the energy cost for a mobile sensor to move one-unit distance, $d(\cdot,\cdot)$ denotes the Euclidean distance between two locations, and e_{job} is the energy cost for a mobile sensor to conduct its jobs (such as sensing and communication actions) when it arrives at the event location. Obviously, the schedule needs to satisfy the condition of $e_i \geq c(DS_i)$.

To efficiently dispatch mobile sensors, we attempt to maximize the total remaining energy of mobile sensors after they finish their dispatch schedules, i.e.,

$$\max \sum_{s_i \in S} (e_i - c(DS_i)). \tag{5.1}$$

In addition, to balance the loads of mobile sensors, we also have to minimize the standard deviations of sensors' energy consumptions.

Note that the aforementioned modeling takes only single round of sensors' dispatch schedules into consideration. In general, multiple rounds of schedules need to be determined. In particular, we have to handle those events being detected over a fixed amount of time, and the objective is to prolong the lifetimes of mobile sensors to cover the maximum number of rounds, where the length of a round depends on users' real-time constraints. Because event locations are unexpected, we only focus on the solution of each round.

Table 5.1 summarizes the notations used in this chapter.

Table 5.1: Summary of notations used in Chapter 5.

notations	definition
L	set of event locations reported in each round; $\|L\| = m$
S	set of mobile sensors; $\|S\| = n$
DS_i	the dispatch schedule of a mobile sensor s_i
$c(DS_i)$	energy consumption of s_i to perform DS_i
e_{move}	energy cost to move a sensor one-unit distance
e_{job}	energy cost for a mobile sensor to conduct its jobs at the event location
e_i	current energy of a mobile sensor s_i
PL_j	the preference list of an event location l_j
B_j	the bound of an event location l_j
Δ_B	increasing level of the bounds
$\hat{c}_k$	a cluster of event locations

5.2 The Mobile Sensor Dispatch (MSD) Method

Our dispatch solution depends on the numbers of mobile sensors and event locations. When $|S| \geq |L|$, we convert the dispatch problem to the problem of finding a maximum matching in a weighted bipartite graph. When $|S| < |L|$, we group the event locations in L into $|S|$ clusters so that each mobile sensor will need to visit one cluster of event locations. Then the maximum matching approach in the case of $|S| \geq |L|$ can be applied again.

5.2.1 Case of $|S| \geq |L|$

We first construct a weighted complete bipartite graph $G = (S \cup L, S \times L)$, where the vertex set contains all mobile sensors and all event locations while the edge set includes all edges (s_i, l_j) from each $s_i \in S$ to each $l_j \in L$. The weight of each edge (s_i, l_j) is defined as $w(s_i, l_j) = e_{move} \times d(s_i, l_j)$, which is the energy cost to move s_i to l_j. Then the problem can be formulated as one of finding a matching P in G such that the following objectives can be satisfied:

- The number of matches (i.e., non-adjacent edges) in P is maximal.
- Mobile sensors can remain the maximum energy after they are dispatched to event locations according to the matches in P.
- The standard deviation of edge weights of P is as small as possible.

Note that under this formulation, the first objective is a must, while the other two objectives are desirable, but the result does not necessarily have the minimum values among all possible matchings.

Below, we present our solution to find the matching P.

1. For each $l_j \in L$, we associate with it a preference list PL_j, which ranks each $s_i \in S$ by the corresponding edge weight $w(s_i, l_j)$ in an increasing order. When the weights are equal, sensors' IDs can be used to break the tie.

2. To achieve the third objective (i.e., minimizing the standard deviation of edge weights of P), we adopt a *bound* B_j for each $l_j \in L$ to limit the mobile sensors that l_j can match with. Initially, the value of each bound B_j is set to the average of the minimum weights of all edges incident to each event location, i.e.,

$$B_1 = B_2 = \cdots = B_m = \frac{1}{m} \sum_{j=1}^{m} \min_{\forall i, (s_i, l_j) \in S \times L} \{w(s_i, l_j)\}.$$

   With bound B_j, location l_j only considers a mobile sensor s_i as a *candidate* if $w(s_i, l_j) \le B_j$.

3. Construct a queue Q which contains all event locations in L.

4. Dequeue an event location, say l_j, from Q. We then select s_i from the candidate mobile sensors in PL_j such that after moving to l_j, s_i can remain the maximum energy, i.e., $e_i - w(s_i, l_j)$ can be maximized. We then check if s_i can be matched with l_j according to the following rules:

   (i) If s_i is unmatched yet, we can include the pair (s_i, l_j) into P.

   (ii) Otherwise, s_i must have been matched with another location, say l_k. In this case, l_j will compete with l_k for s_i by their bounds B_j and B_k. In particular, l_j can win the competition if one of the following conditions is satisfied:

   - $B_j > B_k$. In this case, since there is a higher risk that the standard deviation of P will be increased, we will prefer matching s_i with l_j to matching s_i with l_k.
   - $B_j = B_k$ and $w(s_i, l_j) < w(s_i, l_k)$. Since s_i can remain more energy as it moves to l_j, we thus match s_i with l_j.
   - $B_j = B_k$ and s_i is the only candidate in PL_j but not in PL_k. In this case, if s_i is not matched with l_j, the bound B_j has to be increased, but the bound B_k may not have to be increased. So we match s_i with l_j.

If l_j wins, we will replace the pair (s_i, l_k) in P by the new pair (s_i, l_j), remove s_i from PL_k, and enqueue l_k into Q. Otherwise, s_i is removed from PL_j since l_j will not consider s_i any more.

If l_j cannot find a match in (i) and (ii) and there are still unvisited candidates in PL_j, l_j will examine the sensor that can remain the most energy in PL_j and repeat steps (i) and (ii) again, until there are no more candidates.

5. If l_j cannot be matched with any mobile sensor in the above step 5, we will increase bound B_j by an amount of Δ_B and go back to step 5, until a match is found for l_j.

6. The above steps 5 and 6 will be repeated until the queue Q becomes empty.

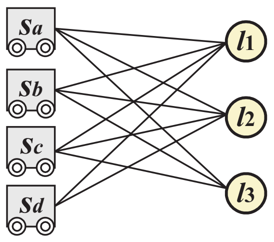
The above procedure must terminate since $|S| \geq |L|$. Note that we should carefully select the value of Δ_B to control the number of candidates to be considered in each iteration. This relates to the maximum and the minimum weights of edges incident to each mobile sensor, and the total number of possible matches (i.e., mn) between S and L. So we recommend Δ_B to be set as follows:

$$\Delta_B = \frac{\alpha}{mn} \times \left(\sum_{j=1}^{m} \max_{\forall i,(s_i,l_j)\in S\times L} \{w(s_i, l_j)\} - \sum_{j=1}^{m} \min_{\forall i,(s_i,l_j)\in S\times L} \{w(s_i, l_j)\} \right), \tag{5.2}$$

where α is an adjustable coefficient.

Fig. 5.2 gives an example with $\alpha = 2$. The energies of mobile sensors are all set to 500 units. Initially, $B_1 = B_2 = B_3 = B_4 = \frac{1}{3} \times (106 + 79 + 94) = 93$ and $\Delta_B = \frac{2}{4\times 3} \times ((217 + 231 + 215) - (106 + 79 + 94)) = 64$. Let $Q = (l_1, l_2, l_3, l_4)$ and we start with l_1. Since there is no candidate mobile sensor in PL_1 with weight smaller than or equal to B_1, B_1 will be increased to $93 + 64 = 157$. In this case, since there are three candidates s_a, s_b, and s_c, l_1 will be matched with s_b (because s_b can remain the maximum energy after it arrives at l_1). In the next iteration, the pair (s_c, l_2) will be matched, as shown in Fig. 5.2(b). However, when examining l_3, it will find that the only candidate s_b has been matched with l_1. Thus, l_3 will compete with l_1 for s_b. Since $B_3 = B_1 = 157$ and $w(s_b, l_3) < w(s_b, l_1)$, (s_b, l_1) will be replaced by (s_b, l_3) as shown in Fig. 5.2(c) and then l_1 will be put into Q. Following the procedure, (s_c, l_2) will be replaced by (s_c, l_1) as shown in Fig. 5.2(d), and finally a match (s_d, l_2) will be found, as shown in Fig. 5.2(e).

Below, we analyze the time complexity of our MSD method when $|S| \geq |L|$. In the MSD method, we first construct a complete bipartite graph G, which requires $O(mn)$ time because we need to assign the weight of each edge. Then calculating the preference lists for all event locations in L takes $O(mn \lg n)$ time because we have to sort the mobile sensors in S. The worst case to match an event location with a mobile sensor is $O(n)$ because the event location has to go through its whole preference list. So it takes $O(mn)$



$w(s_i,l_j)$	s_a	s_b	s_c	s_d
l_1	154	106	127	217
l_2	231	138	79	99
l_3	181	94	215	181

$$PL_1 = \{s_b, s_c, s_a, s_d\}$$
$$PL_2 = \{s_c, s_d, s_b, s_a\}$$
$$PL_3 = \{s_b, s_a, s_d, s_c\}$$

(a)

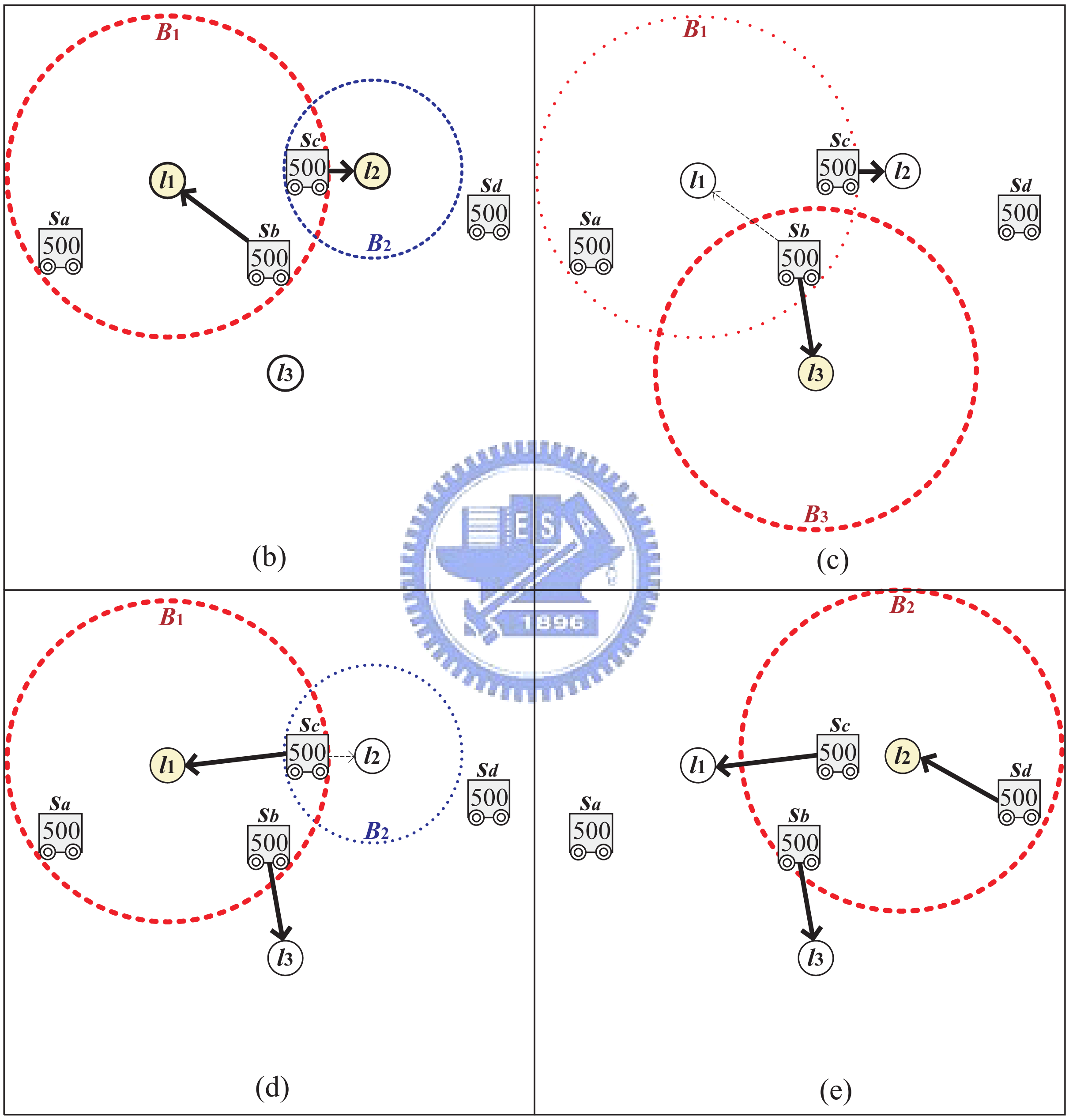


Figure 5.2: An example to execute the MSD method when $|S| \geq |L|$: (a) the edge weights and preference lists of event locations, (b) $P = \{(s_b, l_1), (s_c, l_2)\}$, (c) $P = \{(s_b, l_3), (s_c, l_2)\}$, (d) $P = \{(s_b, l_3), (s_c, l_1)\}$, and (e) $P = \{(s_b, l_3), (s_c, l_1), (s_d, l_2)\}$.

time to compute the maximum matching P. Therefore, the total time complexity of the

MSD method will be $O(mn + mn \lg n + mn) = O(mn \lg n)$.

5.2.2 *Case of* $|S| < |L|$

When event locations are more than mobile sensors, we will cluster L into n groups, each to be visited by one mobile sensor. The following method is built on top of the MSD method discussed in the previous section.

1. We first classify the event locations into $n = |S|$ clusters according to their inter distances. (How to cluster will be discussed later.)

2. Then we define an approximated cost function $\phi(\hat{c}_j)$ for each cluster $\hat{c}_j$, which represents the energy cost required to visit all event locations in $\hat{c}_j$. (The definition of $\phi(\cdot)$ will be given later.) We then repeatedly split and merge some clusters in a way such that the number of clusters remains unchanged and the total cost $\sum_{j=1}^{n} \phi(\hat{c}_j)$ is minimized.

3. Let the final set of n clusters be $\hat{C} = \{\hat{c}_1, \hat{c}_2, \cdots, \hat{c}_n\}$. We define the energy cost $c(s_i, \hat{c}_j)$ for each sensor $s_i \in S$ to visit each member $\hat{c}_j \in \hat{C}$ as:

   $$c(s_i, \hat{c}_j) = e_{move} \times (d(s_i, \hat{l}_{i,j}) + \phi(\hat{c}_j)) + e_{job} \times |\hat{c}_j|,$$

   where $\hat{l}_{i,j}$ is the event location in $\hat{c}_j$ which is closest to s_i. Specifically, the total energy consumption includes moving s_i to $\hat{l}_{i,j}$ and then to all other event locations in $\hat{c}_j$, and the energy cost to conduct jobs at each event locations in $\hat{c}_j$. We then construct a weighted complete bipartite graph $G' = (S \cup \hat{C}, S \times \hat{C})$ such that the vertex set contains all mobile sensors and all clusters and the edge set contains the edge $(s_i, \hat{c}_j)$ from each $s_i \in S$ to each $\hat{c}_j \in \hat{C}$. The edge weight is defined as $w(s_i, \hat{c}_j) = c(s_i, \hat{c}_j)$.

4. With G', we execute the MSD method in Section 5.2.1 to find a matching P' of G'.

5. For each $(s_i, \hat{c}_j) \in P'$, we dispatch mobile sensor s_i to $\hat{l}_{i,j}$, and then to all other event locations in $\hat{c}_j$. The way to traverse from $\hat{l}_{i,j}$ to all other locations in $\hat{c}_j$ can be calculated by any solution to the *traveling salesman problem* (*TSP*) [11] by minimizing the total traversing length.

In step 1, the clustering of L can be done by the classical K-*means* method [48]. In this method, event locations are first randomly separated into n non-empty clusters $\hat{c}_1, \hat{c}_2, \cdots, \hat{c}_n$. Then for each cluster $\hat{c}_i$, we can calculate the fulcrum $\hat{f}_i$ of the event locations in $\hat{c}_i$. We then dissolve the current clustering and allow each event location to pick one of the n fulcrums that is closet to it. All event locations picking the same fulcrum will then form a new cluster. With these new n clusters, we can repeat the above process by calculating their fulcrums and re-cluster all locations. This can be repeated until there is no event relocation.

To evaluate the cost of each cluster in step 2, we define $\phi(\hat{c}_k)$ of each cluster $\hat{c}_k$ as

$$\phi(\hat{c}_k) = \sum_{(s_i,l_j)\in MST(\hat{c}_k)} w(s_i, l_j),$$

where $MST(\hat{c}_k)$ is the minimum spanning tree in $\hat{c}_k$. For example, in Fig. 5.3(b), $\phi(A) = 79$, $\phi(B) = 14$, $\phi(C) = 11$, and $\phi(D) = 55$.

Since K-means may not ensure minimizing the total cost of the derived clusters, especially when there are several event locations far away from others, we have to adjust the cluster result by properly splitting and merging clusters. Intuitively, those clusters containing sparse event locations should be split. However, in order not to change the number of clusters, we have to merge two clusters when splitting a large one. In particular, let us denote w_{intra_max} as the maximum edge weight among edges in all clusters and w_{inter_min} as the minimum edge weight among edges between clusters. When $w_{intra\_max} > w_{inter\_min}$, we can split the cluster with the edge whose weight is w_{intra_max} (by removing that edge) and then merge the two corresponding clusters by adding the edge whose weight is w_{inter_min}. This procedure will be repeated until $w_{intra\_max} \leq w_{inter\_min}$. In this way, we can prevent some clusters from containing too large costs and thus reduce the total cost of clusters. Fig. 5.3 gives an example. In Fig. 5.3(b), $w_{intra\_max} = 60$ (in cluster A) and $w_{inter\_min} = 16$ (between clusters C and D). So we split cluster A into two clusters $A1$ and $A2$, and then merge clusters C and D into single one cluster, as shown in Fig. 5.3(c). Following the same procedure, we can further split cluster D and then merge two clusters $A2$ and B to reduce the total cost of clusters. The final result is shown in Fig. 5.3(d).

5.3 Experimental Results

In this section, we present several experiment results to verify the effectiveness of our proposed MSD method. We design a sensing field as a 450m $\times$ 300m rectangle, on which there are 400 static sensors (used as the event locations) uniformly deployed. In addition, there are some mobile sensors randomly distributed over the sensing field. Each mobile sensor has an initial energy of 3960J (joule). The energy consumption for a mobile sensor to move one-unit distance is set to 8.27J[1].

[1]To observer the effect of dispatch, we ignore the energy cost e_{job} in the experiments.

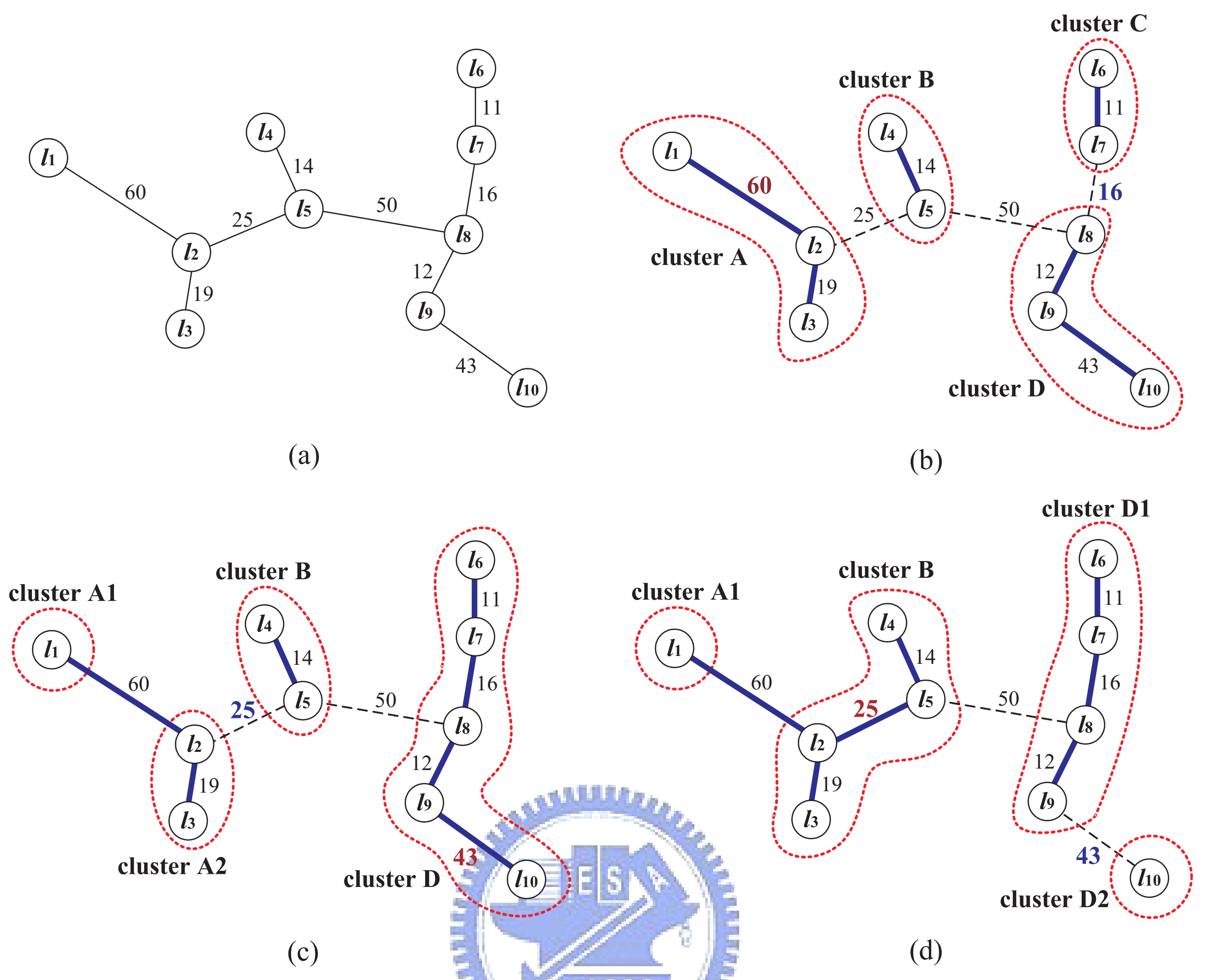


Figure 5.3: An example to group event locations into four clusters: (a) the initial topology, (b) the cluster result after adopting K-means, where the total cost is 159, (c) the cluster result after splitting cluster A and merging clusters C and D, where the total cost is 115, and (d) the cluster result after splitting cluster D and merging clusters $A2$ and B, where the total cost is 97.

5.3.1 *Performance of the MSD Method*

In this experiment, we evaluate the system lifetimes of our proposed MSD method and the dispatch method by simply maximizing the remaining energy of mobile sensors during each one-round dispatch (In the following, we call this method as *iteratively-maximizing method* for short). In the experiment, there are 10 to 15 events randomly generated in each round, and there are 50 mobile sensors to be dispatched to these event locations[2]. We compare the ratio of survived mobile sensors under these two dispatch methods in each round. When the number of survived sensors becomes fewer than that of event locations, the proposed clustering scheme is applied to cluster event locations. The system lifetime

[2]In our experiments, an event location is represented by a static sensor's location.

is referred as the round when the ratio decreases as zero (in the sense that all mobile sensors exhaust their energies).

The system lifetimes of these two dispatch methods are shown in Fig. 5.4. We can observe that our proposed MSD method can have a longer system lifetime compared with the iteratively-maximizing method. This is because the latter does not consider to balance the loads of mobile sensors, which makes some mobile sensors fast to exhaust their energy. The death of these exhausted sensors will burden the remaining survived ones with heavy loads. The situation will become worse as the number of mobile sensors run out of their energies. On the contrary, our MSD method not only attempts to make mobile sensors remain more energy but also balances their loads, so that it will have a longer system lifetime.

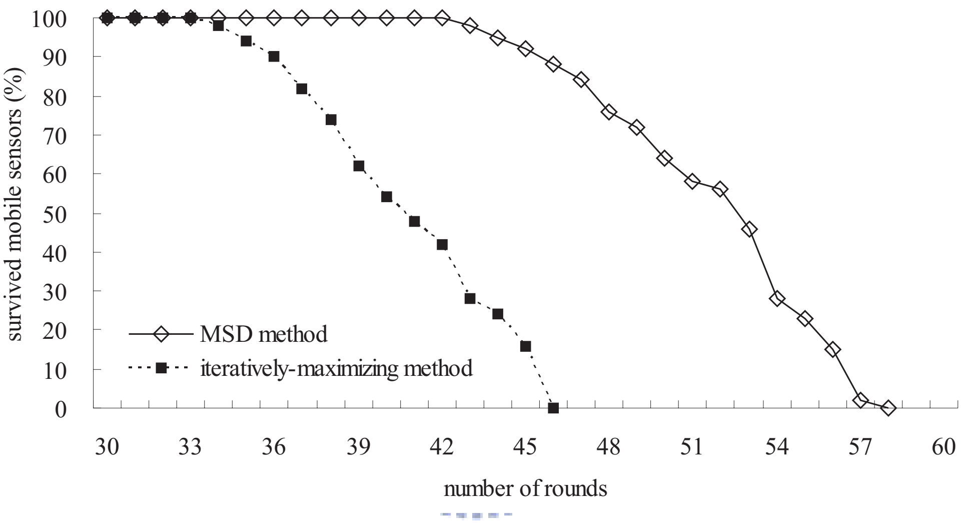
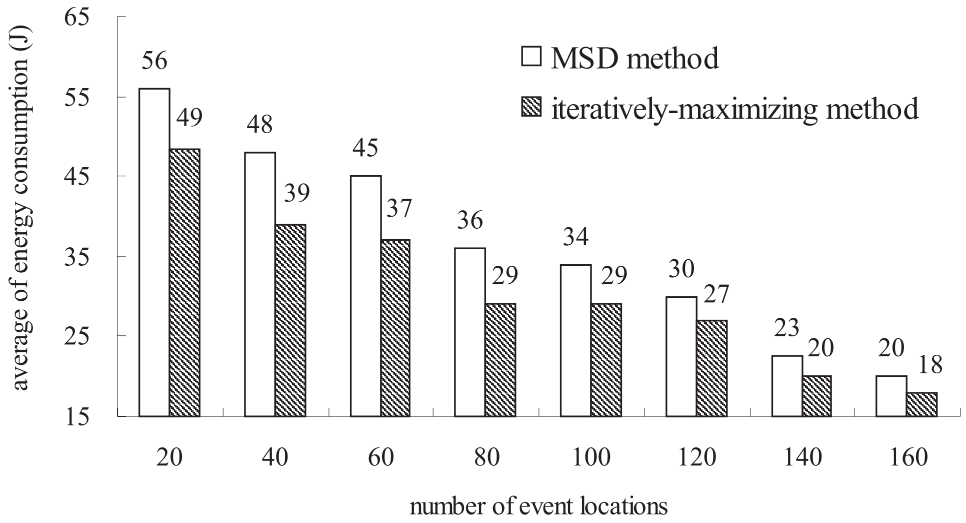


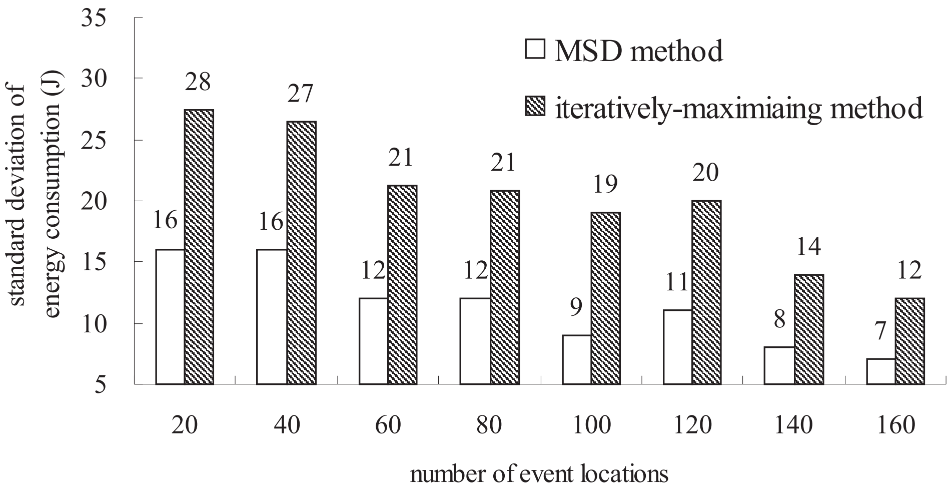
Figure 5.4: Comparison on system lifetimes of the proposed MSD method and the iteratively-maximizing method.

In the next experiment, we measure the degree of load-balance under these two dispatch methods. In particular, we evaluate the averages and standard deviations of energy consumption of mobile sensors. In the experiment, we increase the number of events from 20 to 160. The number of mobile sensors is set as equal to that of event locations, so that each mobile sensor will be assigned with exactly one event location.

Fig. 5.5(a) shows the average of energy consumption of mobile sensors. Because the iteratively-maximizing method always tries to keep the most energy of mobile sensors, it will have a smaller average of energy consumption compared with our MSD method. However, the standard deviation of the iteratively-maximizing method is almost twice than that of our MSD method as shown in Fig. 5.5(b), which indicates that the iteratively-maximizing method will result in seriously unbalance loads among mobile sensors, and thus shortening the system lifetime.



(a)



(b)

Figure 5.5: Comparisons on energy consumption of mobile sensors under the proposed MSD method and the iteratively-maximizing method: (a) the average of energy consumption and (b) the standard deviation of energy consumption.

5.3.2 Effect of the Clustering Scheme

When event locations are more than mobile sensors, we will first cluster event locations and then dispatch mobile sensors to visit these clusters. In this experiment, we study the effect of our clustering scheme on the average of energy consumption of mobile sensors. From Fig. 5.6, we can observe that when the clustering scheme is adopted, mobile sensors can consume less energy. This is because our clustering scheme groups event locations according to their inter distances, and thus the mobile sensors will not travel around locations far away from each other.

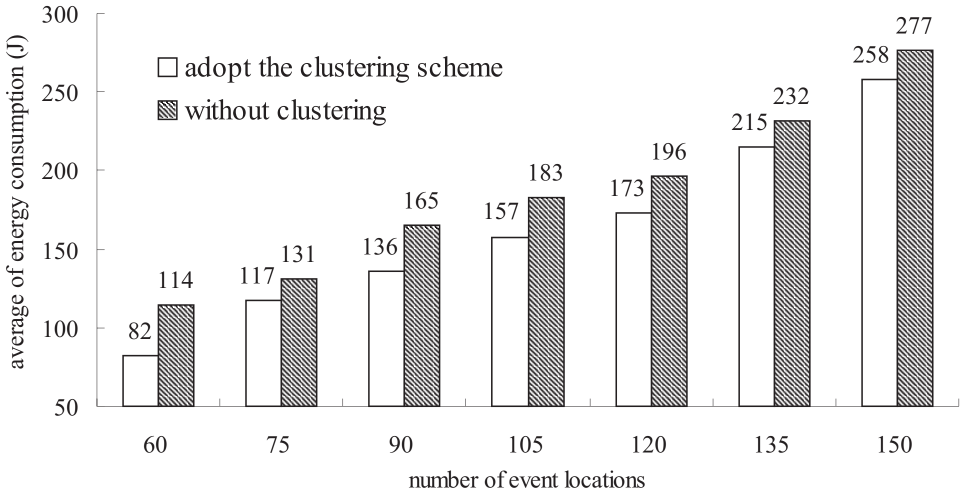


Figure 5.6: The effect of our clustering scheme on the average of energy consumption of mobile sensors, where the number of mobile sensors is set to 50.

5.3.3 *Analysis on the Coefficient α*

In the last experiment, we analyze the effect of the coefficient α on the increasing level Δ_B in Eq. (5.2). The value of α affects both the computation time and result of our MSD method, as shown in Fig. 5.7. In particular, we adopt the number of *redundant iterations* that an event location has to continuously repeat Eq. (5.2) to find candidate mobile sensors as the metric to measure the computation time. In addition, we adopt the product of average and standard deviation of energy consumption of mobile sensors to evaluate the result of the MSD method. Obviously, a smaller product means a better result because mobile sensors can consume less energy while have a more balanced load. Fig. 5.7 presents the effect of coefficient α on the number of redundant iterations and the energy consumption of mobile sensors. We can observe that a smaller δ will cause more redundant iterations while a larger δ will make mobile sensors consume more energy and become unbalanced. From Fig. 5.7, we recommend the optimal value of coefficient δ as 2.0 since both the redundant iterations and the product can be minimized.

5.4 *Summary*

In this chapter, we have developed an efficient dispatch method to schedule mobile sensors to visit event locations in a hybrid sensor network. Our dispatch method can balance the moving distances of mobile sensors while conserve their energies as much as possible, and thus avoiding the early-exhausted mobile sensors burdening other still alive ones. Our dispatch method is general in which the numbers of event locations and mobile sensors can be arbitrary. When the number of event locations is smaller than or equal to that

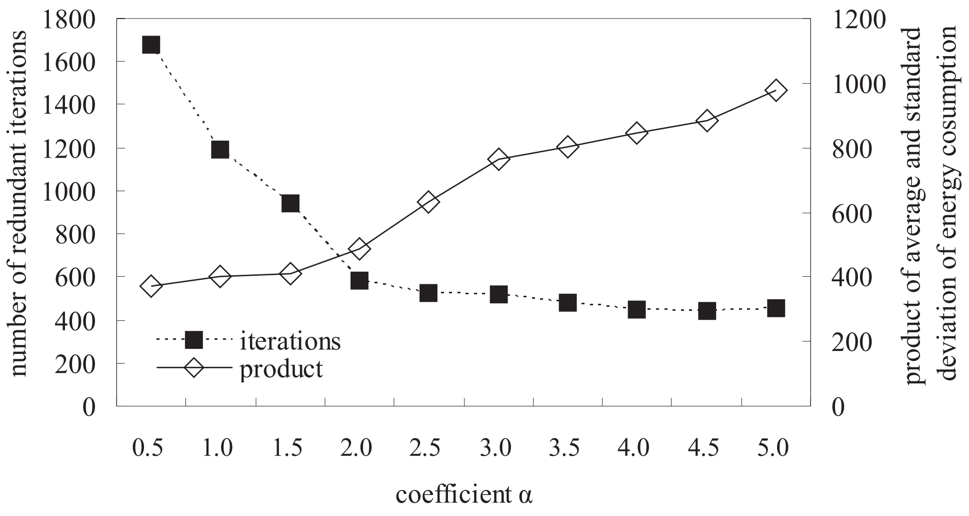


Figure 5.7: The effect of coefficient α on the number of redundant iterations and the energy consumption of mobile sensors, where both the numbers of event locations and mobile sensors are set to 50.

of mobile sensors, we convert the dispatch problem to a maximum matching problem in a weighted bipartite graph. When the number of event locations is larger than that of mobile sensors, we propose efficient clustering schemes to cluster event locations so that the previous matching approach can be applied again. Simulation results have shown that our proposed dispatch method can extend the system lifetime as compared with the iteratively-maximizing method.

Chapter 6

Packet Scheduling for Data Aggregators in a Wireless Sensor Network

After constructing a wireless sensor network, sensors will report their sensing data periodically or transmit emergency notifications to the *data aggregators* (or the sink) when they detect a predefined event. In addition, mobile sensors that have arrived at event locations will also send back their analyzed information to the data aggregators. According to the types and time-constraints of these reported data, we can classify them into *real-time flows* and *non-real-time flows*. For example, events reported from static sensors and analyzed data sent from mobile sensors are classified into real-time flows since events may disappear soon latter, while periodical sensing reports by static sensors are classified into non-real-time flows. Among these flows, if we make sensors compete to transmit their reports to the data aggregators, real-time flows may miss their delay constraints and thus important messages may be dropped. On the contrary, if we allow real-time flows always to preempt non-real-time flows, the latter will be starved. Moreover, since wireless channels are characterized by more serious bursty errors and location-dependent errors, the aforementioned delay-missing and starving situations will become more worse.

To solve these problems, in this chapter we propose two packet fair scheduling algorithms for data aggregators to manage the messages reported from sensors. We consider two wireless environments: a *sing-rate environment* and a *multi-rate environment*. In a single-rate environment, sensors can transmit their packets in a fixed rate and the wireless channels can be either in a *good (error-free)* state or in a *bad (error)* state. Transmissions in a good state will succeed but completely fail in a bad state. Under these assumptions, we propose a *Traffic-Dependent wireless Fair Queuing (TD-FQ)* algorithm that takes traffic types of flows into consideration when scheduling packets. TD-FQ gives a higher priority for real-time flows to alleviate their queuing delays, but it still maintains the fairness among flows so that non-real-time flows will not be starved.

In this chapter, we further consider a multi-rate environment in which sensors can adopt different modulation techniques to transmit their packets under different channel conditions. In particular, A simpler modulation (and thus a higher data rate) can be used when the signal-to-noise ratio (SNR) is sufficiently high, while a more complicated modulation (and thus a lower rate) can still be used under a bad channel [99]. Adopting multi-rate transmissions poses several challenges:

- There is a mismatch between the amount of service that a sensor receives and the amount of time that the sensor actually be served. To transmit the same amount of data, a sensor using a lower rate will take a longer time than one using a higher rate. Thus the concept of virtual time in fair scheduling algorithms may need to be redefined.

- When a flow that suffered from a bad channel exits from errors, it may take a different amount of time for the system to compensate the flow depending on its channel condition, thus making the design of compensation difficult.

- The overall system performance may be degraded if there are too many low-rate flows.

In this chapter, we propose a fair scheduling algorithm called *Multi-Rate wireless Fair Queueing (MR-FQ)* that addresses the aforementioned issues. MR-FQ can adjust a flow's transmission rate according to its channel condition and lagging degree. A flow is allowed to transmit at a lower rate to alleviate its lags only if it is lagging up to a certain degree. Specifically, the more serious a flow is lagging, the lower rate the flow is allowed to use. Such differentiation can take care of both fairness and system performance. Lower rate flows thus will not prolong other flows' delays. In this way, MR-FQ can satisfy the delay-sensitive property of real-time flows, while still maintaining fairness and bounded delays for all flows.

6.1 The TD-FQ Algorithm

In this section, we propose our TD-FQ algorithm used in a single-rate wireless environment.

6.1.1 System Model

We consider a part of the wireless sensor network as in Fig. 6.1. There is a *data aggregator* responsible for collecting data reported from nearby sensors. Packets arriving

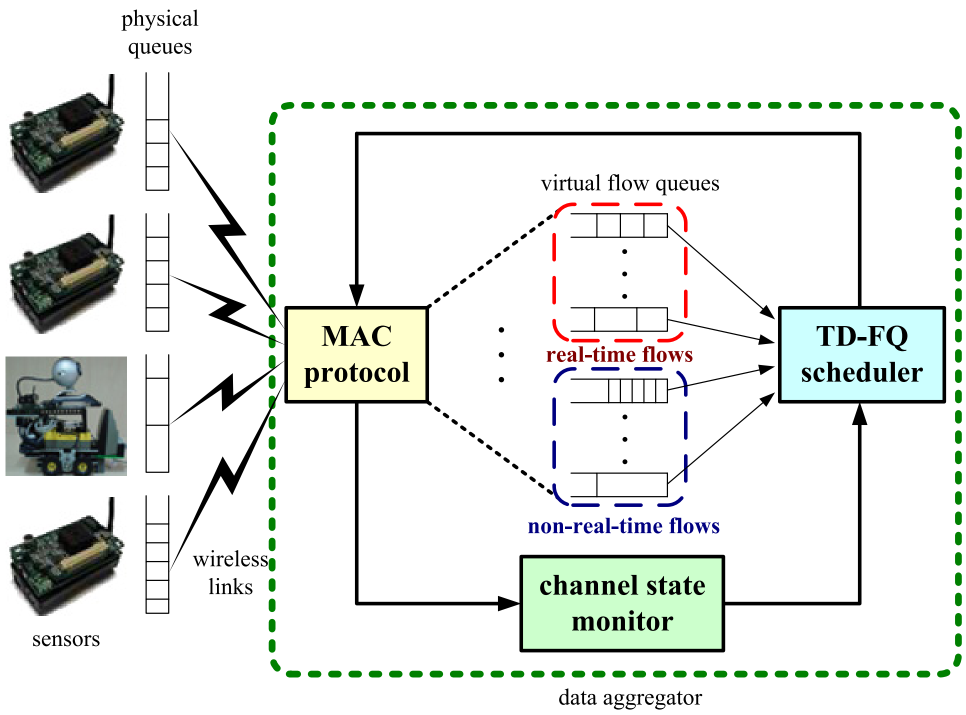


Figure 6.1: The system architecture of TD-FQ.

at the data aggregator are classified into real-time traffic and non-real-time traffic and the data aggregator maintains a set of *virtual flow queues* to record the physical queues' conditions of sensors. The *TD-FQ scheduler* then schedules these flow queues and notifies the corresponding sensor to send its packets to the data aggregator via the *MAC protocol*. The *channel state monitor* provides information about the channel state of each sensor. (There are different alternatives to obtain or predict queues' conditions and channel states of sensors, but these issues are out of the scope of this chapter). For simplicity, we assume that the data aggregator has immediate and accurate knowledge of queues' conditions and channels' states of sensors.

In this section, we focus on the design of TD-FQ scheduler. Sensors may suffer from bursty and location-dependent channel errors. However, error periods are assumed to be sporadic and short relative to the whole lifetime of flows so that long-term unfairness would not happen.

6.1.2 *Scheduling Policy*

In TD-FQ, each flow i is assigned a *weight* w_i to represent the ideal fraction of bandwidth that the system commits to it. However, the real services received by flow i may not match exactly its assigned weight. Thus we maintain a *virtual time* v_i to record the nominal services received by it, and a *lagging level* lag_i to record its credits/debits. The former is to compete with other flows for services, while the latter is to arrange compensatory services. The actual normalized service received by flow i is $v_i - \frac{lag_i}{w_i}$. Flow i is called *leading* if $lag_i < 0$, called *lagging* if $lag_i > 0$, and called *satisfied* if $lag_i = 0$. Further, depending on its queue content, a flow is called *backlogged* if its queue is nonempty, and called *unbacklogged* otherwise. A flow is considered as *active* if it is backlogged or unbacklogged but leading. Note that TD-FQ will only choose active flows to serve. When an unbacklogged but leading flow (i.e., an active flow) is chosen, its service will actually be transferred to another flow for compensation purpose. In addition, when a flow i transits from unbacklogged to backlogged, its virtual time v_i is set to $\max\{v_i, \min_{j\in A}\{v_j\}\}$, where A is the set of all active flows.

Fig. 6.2 outlines the scheduling policy of TD-FQ. First, the active flow i with the minimum virtual time v_i is selected. If flow i is backlogged and its channel condition is good, the head-of-line packet of flow i can be transmitted if flow i is non-leading. In this case, the service is called a *normal service (NS)*. We then update the virtual time v_i as $v_i + \frac{l_p}{w_i}$, where l_p is the length of the packet. In case that flow i has to give up its service because of an empty queue or a bad channel condition, the service will become an *extra service (ES)*. On the other hand, if flow i is over-served (i.e., leading), the *Gradual Degradation Scheme* will be invoked to determine if flow i is still eligible for the service. If flow i has to give up its service, the service will be transferred to a *compensatory service (CS)*. In both cases of CS and ES, the *Compensation Scheme* will be triggered, trying to select another flow j to serve. If the Compensation Scheme cannot select any flow, this service is wasted, called a *forsaken service (FS)*. If the Compensation Scheme still chooses flow i to serve, we update v_i and transmit its head-of-line packet. If a flow j ($\neq i$) is selected, flow j's packet will be transmitted and the values of v_i, lag_i, and lag_j will be updated as follows:

$$v_i = v_i + \frac{l_{p'}}{w_i}, \tag{6.1}$$

$$lag_i = lag_i + l_{p'}, \tag{6.2}$$

$$lag_j = lag_j - l_{p'}, \tag{6.3}$$

where p' is the packet being sent. Note that in this case we "charge" to flow i by increasing its virtual time v_i, but "credit" (respectively, debit) to lag_i (respectively, lag_j) of flow i (respectively, j).

When the scheduler serves the head-of-line packet of any flow i, it has to check the queue size of flow i. If it finds that flow i's queue is empty, it will invoke the *Lag Redistributing Scheme* to adjust flow i's lag, if necessary.

Below, we introduce the Gradual Degradation Scheme, Compensation Scheme, and Lag Redistributing Scheme in TD-FQ. Table 6.1 summarizes the notations used in TD-FQ.

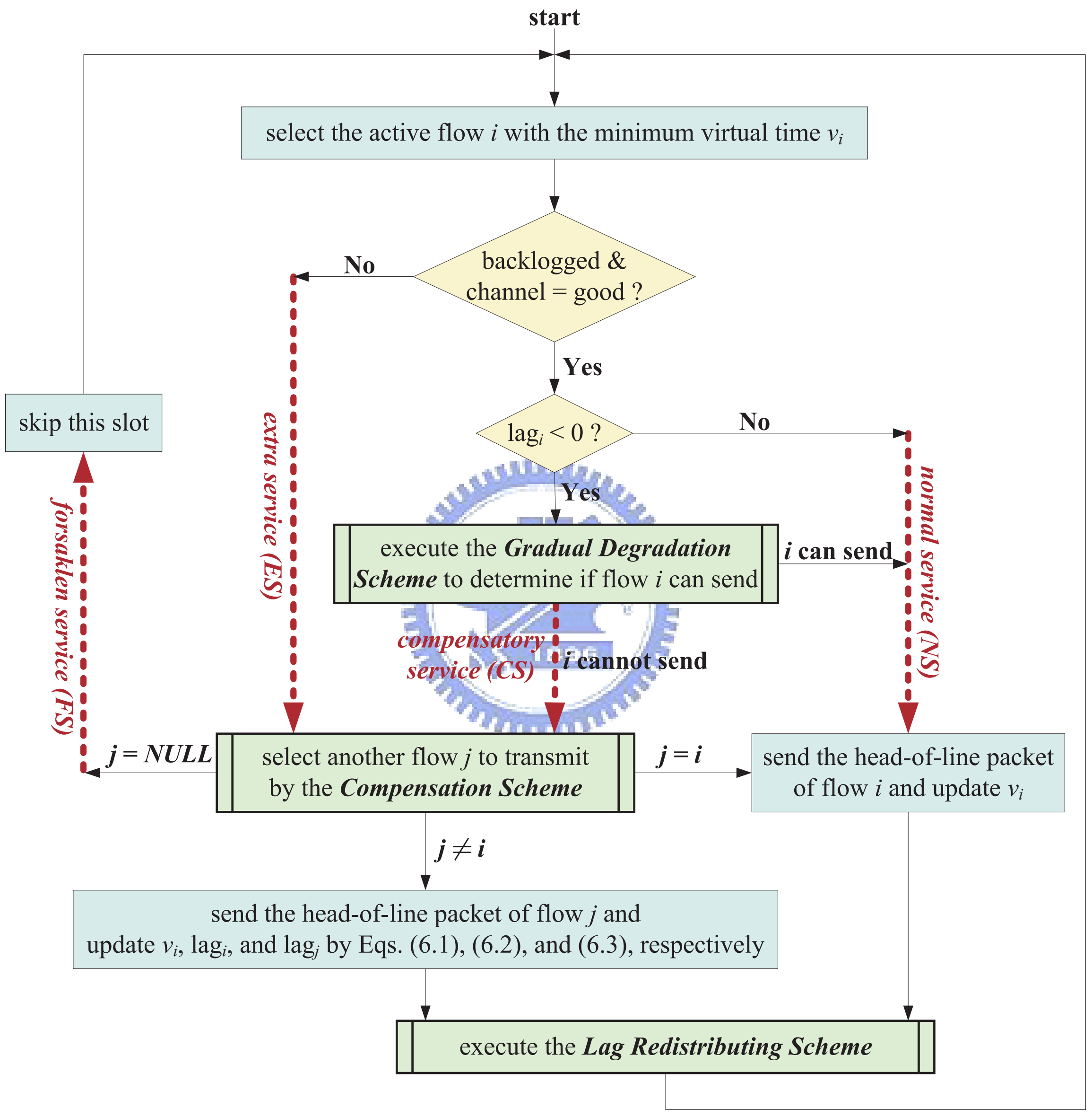

start
select the active flow i with the minimum virtual time v_i
backlogged & channel = good ?
No
Yes
lag$_i$ < 0 ?
No
Yes
skip this slot
extra service (ES)
forsaken service (FS)
normal service (NS)
execute the ***Gradual Degradation Scheme*** to determine if flow i can send
i can send
compensatory service (CS)
i cannot send
j = NULL
select another flow j to transmit by the ***Compensation Scheme***
$j = i$
send the head-of-line packet of flow i and update v_i
$j \neq i$
send the head-of-line packet of flow j and update v_i, lag$_i$, and lag$_j$ by Eqs. (6.1), (6.2), and (6.3), respectively
execute the ***Lag Redistributing Scheme***

Figure 6.2: The scheduling policy of TD-FQ.

Table 6.1: Summary of notations used in TD-FQ.

notations	definition
w_i	weight of flow i
v_i	virtual time of flow i
lag_i	credits/debits of flow i
g_i	gradual degradation service index of flow i when $lag_i < 0$
α_R, α_N	gradual degradation ratios for real-time and non-real-time flows
δ	the threshold to distinguish seriously/moderately lagging flows
L_R, L_N, L_R^S, L_R^M, L_N^S, L_N^M	sets of lagging flows (defined in SWC)
W_R, W_N, W_R^S, W_R^M, W_N^S, W_N^M	weights of lagging flows L_R, L_N, L_R^S, L_R^M, L_N^S, and L_N^M, respectively
V_R, V_N, V_R^S, V_R^M, V_N^S, V_N^M	normalized amounts of ES/CS received by L_R, L_N, L_R^S, L_R^M, L_N^S, and L_N^M, respectively
B	bound of differences of services (used in SWC)
c_i^S, c_i^M	normalized amounts of ES/CS received by flow i when $\frac{lag_i}{w_i} \geq \delta$ and $0 < \frac{lag_i}{w_i} < \delta$, respectively
f_i	normalized amount of ES received by flow i when $lag_i \leq 0$

6.1.3 *Gradual Degradation Scheme*

When a leading flow i is selected by the scheduler, the Gradual Degradation Scheme will be invoked to check flow i's leading amount. A leading flow is allowed to receive an amount of additional service proportional to its normal services. Specifically, when a flow i transits from lagging/satisfied to leading, we set up a parameter $g_i = \alpha \cdot v_i$, where $0 < \alpha < 1$ is a system-defined parameter. Later on, flow i's virtual time will be increased each time when it is selected by the scheduler. Let v_i' be flow i's current virtual time when it is selected. We will allow flow i to be served if $g_i \leq \alpha v_i'$. If so, g_i is updated as $g_i + \frac{l_p}{w_i}$, where l_p is the length of the packet. Intuitively, flow i can enjoy approximately $\alpha(v_i' - v_i)$ of its services.

Further, to distinguish real-time from non-real-time flows, we substitute α by a parameter α_R for real-time flows, and by α_N for non-real-time flows. We set $\alpha_R > \alpha_N$ to distinguish their priorities.

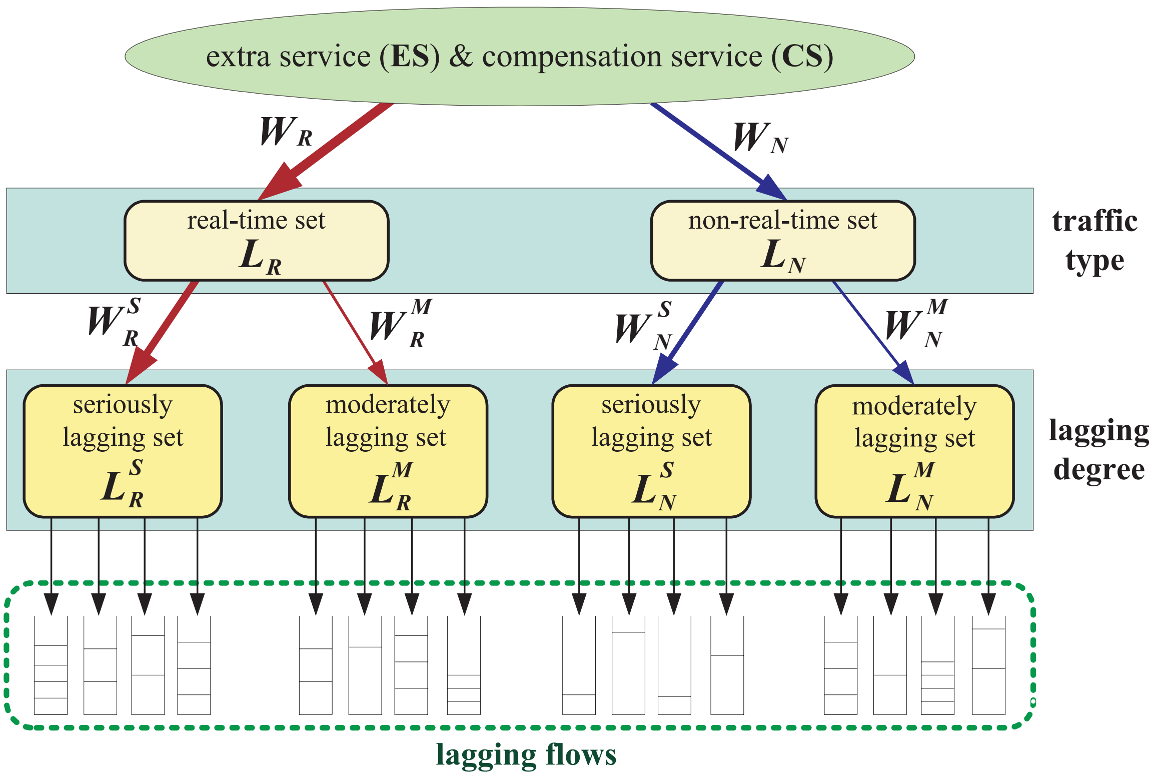

extra service (ES) & compensation service (CS)
W_R
W_N
real-time set L_R
non-real-time set L_N
traffic type
W_R^S
W_R^M
W_N^S
W_N^M
seriously lagging set L_R^S
moderately lagging set L_R^M
seriously lagging set L_N^S
moderately lagging set L_N^M
lagging degree
lagging flows

Figure 6.3: The set-based weight compensation (SWC) scheme.

6.1.4 *Compensation Scheme*

When the selected flow i suffers from channel errors or it cannot satisfy the gradual degradation condition, the Compensation Scheme will be invoked to arrange another flow to receive this additional service (reflected by ES and CS in Fig. 6.2). In this case, lagging flows should always have a higher priority over non-lagging flows to receive such services. Next, we first discusses how to choose a lagging flow to receive ES/CS. Then we deal with the case when all lagging flows are experiencing errors.

Dispatching ES and CS to Lagging Flows

The Compensation Scheme first tries to dispatch ES/CS to lagging flows. Here we propose a *Set-based Weight Compensation (SWC)* scheme, as illustrated in Fig. 6.3. SWC first classifies lagging flows into a *real-time set* L_R and a *non-real-time set* L_N. These two sets are each further classifies into a *seriously lagging set* and a *moderately lagging set*. Individual flows are at the bottom. The concept of weight is used to dispatch services to these sets.

In order to dispatch ES/CS to L_R and L_N, we assoicate them with a weight W_R and W_N, respectively. Normally, we would set $W_R > W_N$. In addition, a variable V_R (respectively, V_N) is used to record the normalized ES/CS received by L_R (respectively,

L_N). When both L_R and L_N contain error-free flows, the service will be given to L_R if $V_R \leq V_N$, and to L_N otherwise. When only one of L_R and L_N contains error-free flows, the service will be given to that one, independent of the values of V_R and V_N. When L_R receives the service, V_R is updated as

$$V_R = \min\left\{V_R + \frac{l_p}{W_R}, \frac{W_N V_N + B}{W_R}\right\}, \tag{6.4}$$

where l_p is the length of the transmitted packet, and B is a system parameter to bound the difference between V_R and V_N. Similarly, when L_N receives the service, V_N is updated as

$$V_N = \min\left\{V_N + \frac{l_p}{W_N}, \frac{W_R V_R + B}{W_N}\right\}. \tag{6.5}$$

Note that to prevent the cases of $V_R \gg V_N$ or $V_N \gg V_R$, which may cause L_R or L_N to starve when the other set recovers from errors, we set a bound $|W_R V_R - W_N V_N| \leq B$. This gives the second term in the right-hand side of Eqs. (6.4) and (6.5).

The flows in L_R are further classified into a seriously lagging set L_R^S and a moderately lagging set L_R^M. We assign a real-time lagging flow i to L_R^S if $\frac{lag_i}{w_i} \geq \delta$, where δ is a system parameter. Otherwise, flow i is assigned to L_R^M. Similarly, the flows in L_N are also classified into a seriously lagging set L_N^S and a moderately lagging set L_N^M. Again, services are dispatched to sets L_R^S, L_R^M, L_N^S, and L_N^M according their weights W_R^S, W_R^M, W_N^S, and W_N^M, respectively. To benefit seriously lagging flows, we set $W_R^S > W_R^M$ and $W_N^S > W_N^M$. Services are then dispatched to these sets similar to the earlier case (i.e., the service distribution to L_R and L_N). We use V_R^S, V_R^M, V_N^S, and V_N^M to record the services received by these sets. Again a bound B is set to limit the differences between V_R^S and V_R^M and between V_N^S and V_N^M.

At the bottom of SWC are four groups of individual flows of the same properties (traffic types and lagging degrees). Here we dispatch ES/CS proportional to flows' weights. Specifically, for each flow i, we associate it with two *compensation virtual times* c_i^S and c_i^M, which keep track of the normalized amount of ES/CS received by flow i when $\frac{lag_i}{w_i} \geq \delta$ and $0 < \frac{lag_i}{w_i} < \delta$, respectively. When the scheduler chooses the seriously lagging set (L_R^S or L_N^S), it selects the *error-free* flow i with the smallest c_i^S in the set to serve. Similarly, when the scheduler chooses the moderately lagging set (L_R^M or L_N^M), it selects the error-free flow i with the smallest c_i^M in the set to serve. When a lagging flow i receives such a service, its compensation virtual times are updated as

$$\begin{cases} c_i^S = c_i^S + \frac{l_p}{w_i}, & \text{if } \frac{lag_i}{w_i} \geq \delta \\ c_i^M = c_i^M + \frac{l_p}{w_i}, & \text{otherwise} \end{cases}.$$

When a flow i newly enters one of the sets L_R^S, L_R^M, L_N^S, and L_N^M or transits from one set to another, we have to assign its c_i^S or c_i^M as follows. If flow i is seriously lagging (i.e.,

$\frac{lag_i}{w_i} \geq \delta$), we set

$$c_i^S = \begin{cases} \max\{c_i^S, c_{min}^{SR}\}, & \text{if flow } i \text{ is real-time} \\ \max\{c_i^S, c_{min}^{SN}\}, & \text{if flow } i \text{ is non-real-time} \end{cases}.$$

Otherwise, we set

$$c_i^M = \begin{cases} \max\{c_i^M, c_{min}^{MR}\}, & \text{if flow } i \text{ is real-time} \\ \max\{c_i^M, c_{min}^{MN}\}, & \text{if flow } i \text{ is non-real-time} \end{cases},$$

where c_{min}^{SR} (respectively, c_{min}^{SN}) is the minimum value of c_j^S such that $j \in L_R^S$ (respectively, $j \in L_N^S$), and c_{min}^{MR} (respectively, c_{min}^{MN}) is the minimum value of c_j^M such that $j \in L_R^M$ (respectively, $j \in L_N^M$). One exception is when the set $L_R^S/L_N^S/L_R^M/L_N^M$ is empty, in which case $c_{min}^{SR}/c_{min}^{SN}/c_{min}^{MR}/c_{min}^{MN}$ is undefined. If so, we set $c_{min}^{SR}/c_{min}^{SN}/c_{min}^{MR}/c_{min}^{MN}$ to the value of c_j^S/c_j^M of the "last flow" j that left the set $L_R^S/L_N^S/L_R^M/L_N^M$.

In summary, SWC compensates more services to real-time flows and seriously lagging flows, thus alleviating the queuing delays of these flows. In addition, SWC does not starve other lagging flows since these flows can still share a fraction of ES/CS.

Dispatching ES to Non-lagging Flows

If there is no lagging flow selected in the previous stage (due to errors), the service will be dispatched according to its original type. If the service comes from CS, it will be returned back to the originally selected flow. Otherwise, the service (i.e., ES) will be given to a non-lagging flow. The scheduler dispatches ES proportional to the non-lagging flows' weights. In particular, each flow i is assigned with an *extra virtual time* f_i to keep track of the normalized amount of ES received by flow i when it is non-lagging (i.e., $lag_i \leq 0$). When a backlogged flow i becomes error-free and non-lagging, f_i is set to

$$f_i = \max\{f_i, \min\{f_j \mid \text{flow } j \text{ is error-free, backlogged, and non-lagging; } j \neq i\}\}.$$

The scheduler selects the flow i with the smallest f_i value among all error-free, backlogged, and non-lagging flows to serve. When flow i receives the service, f_i is updated as $f_i + \frac{l_p}{w_i}$. An exception occurs when there is no selectable non-lagging flow, in which case this time slot will simply be wasted.

6.1.5 *Lag Redistributing Scheme*

After a flow is served, if its queue state changes to unbacklogged and it is still lagging, we will distribute its credit to other flows that are in debet and reset its credit to zero. This is because the flow does not need the credit any more [126]. This is done by the Lag Redistribution Scheme.

The scheme examines the flow i that is actually served in this round. After the service, if flow i's queue becomes empty and $lag_i > 0$, we will give its credit to other flows in debet proportional to their weights. Specifically, for each flow k such that $lag_k < 0$, we set

$$lag_k = lag_k + lag_i \times \frac{w_k}{\sum_{lag_m<0} w_m}.$$

Then we reset $lag_i = 0$.

6.2 The MR-FQ Algorithm

In this section, we propose our MR-FQ algorithm used in a multi-rate wireless environment.

6.2.1 System Model

The architecture of MR-FQ (refer to Fig. 6.4) is similar to that discussed in Section 6.1.1, except that there is a *MAC and transmission module* that can transmit at n rates $\hat{C}_1$, $\hat{C}_2$, $\cdots$, and $\hat{C}_n$, where $\hat{C}_1 > \hat{C}_2 > \cdots > \hat{C}_n$. This module also measures the current channel condition to each sensor and determines the most appropriate rate to communicate with that sensor (several works [8, 54, 99, 115] have addressed the rate selection problem, but this is out of scope of this chapter). The information of the best rate is also reported to the *MR-FQ scheduler* for making a decision. For simplicity, we assume that the data aggregator has immediate knowledge of the best rate for each sensor. Note that this also includes the worst case where the channel is too bad to be used, in which case we can regard the best rate to be zero.

6.2.2 Service Fairness vs. Time Fairness

With the emergence of multi-rate communication, the concept of fairness may be defined in two ways. One is *service fairness*, which means that the difference between services received by any two flows should be bounded, and the other is *time fairness*, which means that the difference between the amounts of transmission time of two any flows should be bounded. Formally, let $\Phi_i^s(t_1, t_2)$ and $\Phi_i^t(t_1, t_2)$ be the amount of services and the amount of time that a flow i receives/utilizes during the time interval $[t_1, t_2)$, respectively. Then for any two flows i and j, during any $[t_1, t_2)$,

$$\left| \frac{\Phi_i^s(t_1, t_2)}{w_i} - \frac{\Phi_j^s(t_1, t_2)}{w_j} \right| \leq \sigma_s, \tag{6.6}$$

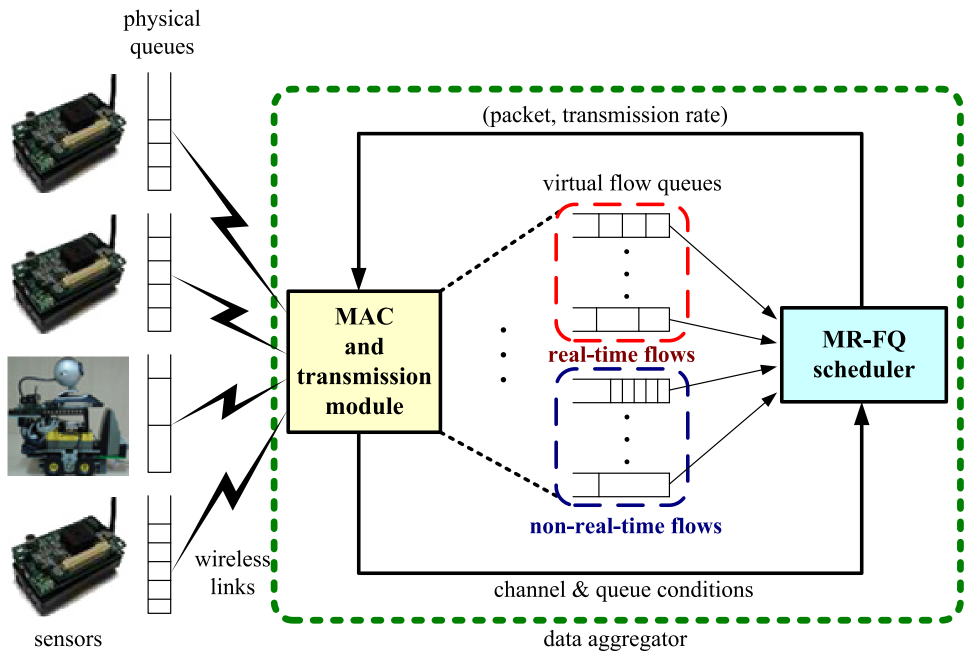


Figure 6.4: The system architecture of MR-FQ.

holds if service fairness is desired, and

$$\left| \frac{\Phi_i^t(t_1, t_2)}{w_i} - \frac{\Phi_j^t(t_1, t_2)}{w_j} \right| \leq \sigma_t, \tag{6.7}$$

holds if time fairness is desired, where σ_s and σ_t are small, non-negative numbers.

We observe that in a single-rate environment, Eq. (6.6) and Eq. (6.7) are equivalent. However, in a multi-rate environment, Eq. (6.6) and Eq. (6.7) may not be satisfied at the same time. If service fairness is desired, then flows using lower rates will occupy more medium time. On the contrary, if time fairness is desired, flows using higher rates will transmit more data. The concept is illustrated in Fig. 6.5. Furthermore, when the rates used by sensors exhibit higher variation, the tradeoff between service and time fairness is more significant (solid line in Fig. 6.5). When the variation is lower, the tradeoff is less significant (dashed line in Fig. 6.5). When the variation is zero, this degenerates to the single-rate case (thick line in Fig. 6.5).

6.2.3 Scheduling Policy

Fig. 6.5 leads to the following guidelines in the design of MR-FQ. First, the concept of virtual time is redefined based on the concept of time fairness. However, we differentiate flows according to their lagging degrees. A flow is allowed to use a lower transmission rate only if it is suffering from a higher lagging degree. In this way, we can take care of service fairness. Thus the system performance would not be hurt when there exist too many low-rate stations.

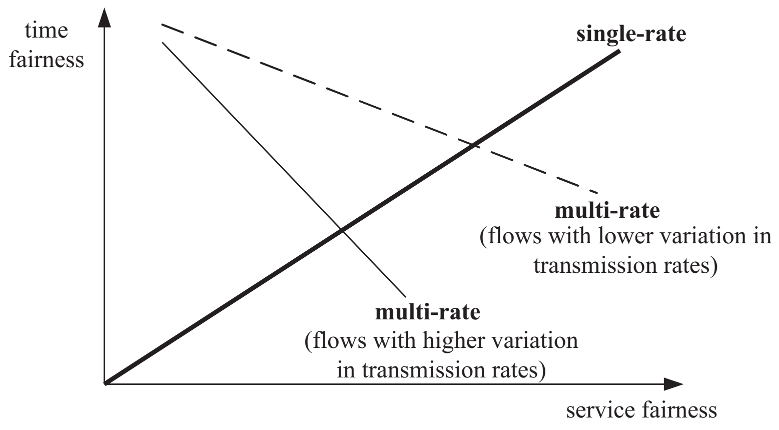


Figure 6.5: The tradeoff between service fairness and time fairness.

Fig. 6.6 outlines the scheduling policy of MR-FQ. First, the active flow i with the smallest virtual time v_i is selected. If flow i is backlogged, the *Rate Selection Scheme* is invoked to compute the best rate r to transmit for flow i. If the result is $r \leq 0$, which means either flow i has a bad channel condition or its current lagging degree does not allow it to transmit (refer to Section 6.2.4 for details). Otherwise, if flow i is non-leading, the head-of-line packet of flow i will be served. Then we update the virtual time of flow i as follows:

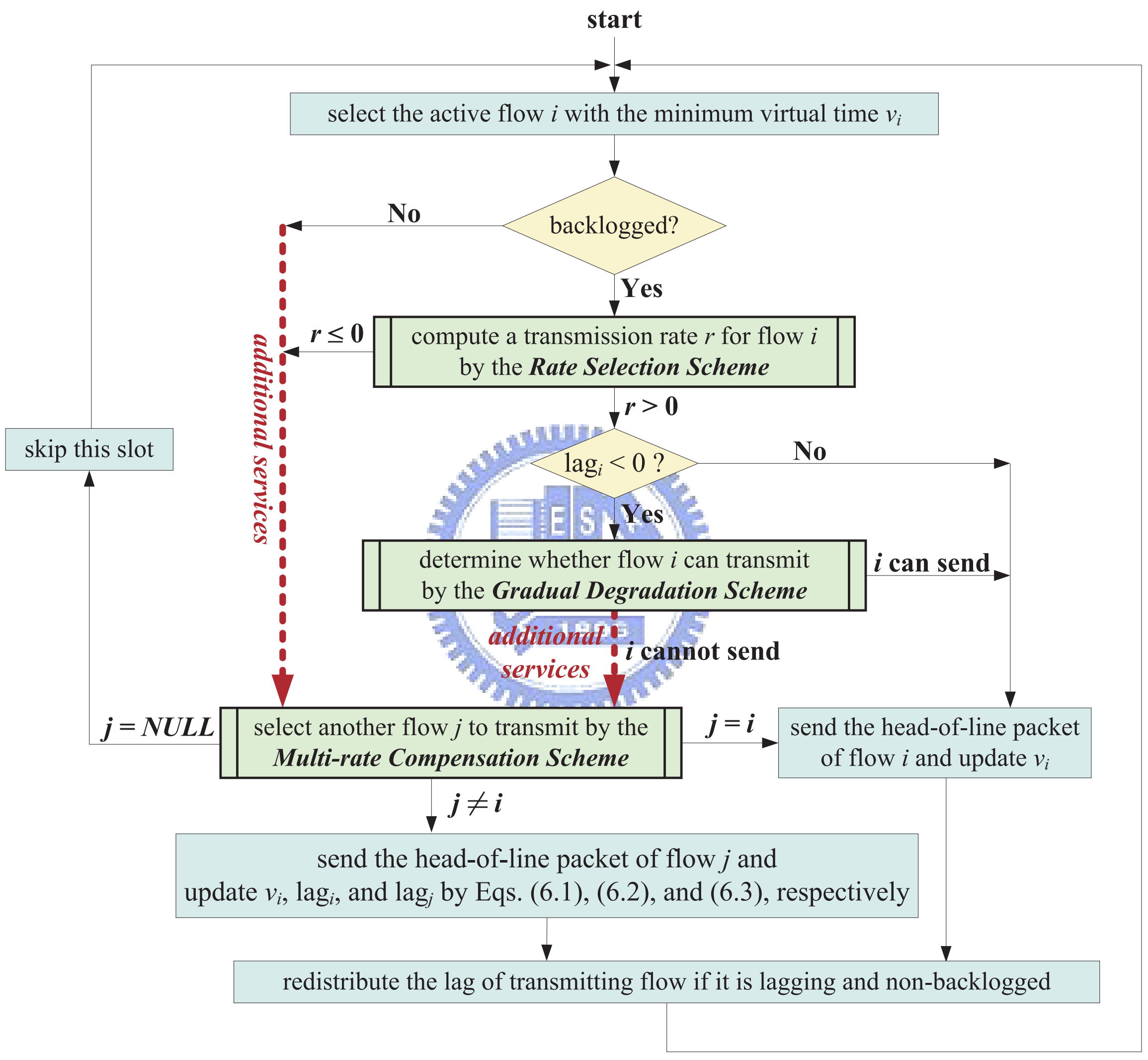
$$v_i = v_i + \left(\frac{l_p}{w_i} \times \frac{\hat{C}_1}{r} \right), \tag{6.8}$$

where l_p is the length of the packet. Note that the ratio $\frac{\hat{C}_1}{r}$ is to reflect the concept of time fairness. The amount of increase in v_i is inverse to the transmission rate r. Thus, if a lower r is used, the less competitive flow i will be in the next round.

If flow i is over-served (i.e., leading), the *Gradual Degradation Scheme* is activated to check if flow i is still eligible for the service (refer to Section 6.1.3). In case that flow i has to give up its service due to an empty queue, a bad channel condition, or a rejection decision by the Gradual Degradation Scheme, the service is transferred to the *Multi-rate Compensation Scheme* to select another flow j to serve (refer to Section 6.2.5). If the scheme fails to select any flow, this service is just wasted. If the scheme still selects flow i to serve, then we send its head-of-line packet and update v_i according to Eq. (6.8). If another flow j ($\neq i$) is selected, flow j's packet is sent and the values of v_i, lag_i, and lag_j are updated according to Eqs. (6.1)–(6.3). Since flow i is not actually served, Eq. (6.1) is equivalent to Eq. (6.8) with $r = \hat{C}_1$.

Whenever the scheduler serves any flow i, it has to check the queue size of flow i. If flow i's queue state changes to non-backlogged and it is still lagging, we distribute its credit to other flows according to the rules in Section 6.1.5.

Below, we introduce the Rate Selection Scheme and Multi-rate Compensation Scheme



start
select the active flow i with the minimum virtual time v_i
backlogged?
No
Yes
$r \le 0$
compute a transmission rate r for flow i by the ***Rate Selection Scheme***
$r > 0$
skip this slot
additional services
$\text{lag}_i < 0$?
No
Yes
determine whether flow i can transmit by the ***Gradual Degradation Scheme***
i can send
additional services
i cannot send
$j = NULL$
select another flow j to transmit by the ***Multi-rate Compensation Scheme***
$j = i$
send the head-of-line packet of flow i and update v_i
$j \ne i$
send the head-of-line packet of flow j and update v_i, lag_i, and lag_j by Eqs. (6.1), (6.2), and (6.3), respectively
redistribute the lag of transmitting flow if it is lagging and non-backlogged

Figure 6.6: The scheduling policy of MR-FQ.

in MR-FQ.

6.2.4 *Rate Selection Scheme*

When a backlogged flow i is selected, the Rate Selection Scheme is invoked to choose a suitable transmission rate for flow i according to its lagging degree and channel condition. The basic idea is to permit different ranges of transmission rates according to flow i's normalized lag (i.e., $\frac{lag_i}{w_i}$). In order to help a seriously lagging flow to alleviate its huge lag, we allow it to use a larger range of rates. Specifically, we set up $n-1$ levels of lagging thresholds δ_1, δ_2, $\cdots$, δ_{n-1}. A flow with a normalized lag exceeding δ_i is allowed to use a rate as low as $\hat{C}_{i+1}$, $i \leq n-1$. Fig. 6.7 shows the mapping of lagging degrees to allowable transmission rates. If flow i's current best rate falls within the allowable range, the rate is returned. Otherwise, a negative value is returned to indicate a failure. For example, if flow i satisfies $\delta_2 < \frac{lag_i}{w_i} \leq \delta_3$ and its current best rate is $\hat{C}_2$, then $\hat{C}_2$ is returned. If the current best rate is $\hat{C}_5$, then a negative value is returned.

lagging degrees	$\hat{C}_1$	$\hat{C}_2$	$\hat{C}_3$	$\cdots$	$\hat{C}_{n-2}$	$\hat{C}_{n-1}$	$\hat{C}_n$
$\frac{lag_i}{w_i} \leq \delta_1$	✓						
$\delta_1 < \frac{lag_i}{w_i} \leq \delta_2$	✓	✓					
$\delta_2 < \frac{lag_i}{w_i} \leq \delta_3$	✓	✓	✓				
$\vdots$				$\vdots$			
$\delta_{n-3} < \frac{lag_i}{w_i} \leq \delta_{n-2}$	✓	✓	✓	$\cdots$	✓		
$\delta_{n-2} < \frac{lag_i}{w_i} \leq \delta_{n-1}$	✓	✓	✓	$\cdots$	✓	✓	
$\delta_{n-1} \leq \frac{lag_i}{w_i}$	✓	✓	✓	$\cdots$	✓	✓	✓

Figure 6.7: The mapping of lagging degrees to allowable transmission rates (indicated by check marks) in the Rate Selection Scheme.

6.2.5 *Multi-rate Compensation Scheme*

When the selected flow i does not have a satisfactory channel condition or fails to pass the Gradual Degradation Scheme, the Multi-rate Compensation Scheme is triggered (reflected by *additional services* in Fig. 6.6). Fig. 6.8 shows how to dispatch additional services. Flows are prioritized according to the following rules:

1. Lagging flows have a higher priority over non-lagging flows to receive such services.

2. Flows that can use higher rates to transmit have a higher priority over flows that can only use lower rates.

3. Among lagging flows of the same best rate, real-time flows and non-real-time ones will share the services according to some ratio.

Note that the last rule is not applied to leading flows because such flows suffer no lagging.

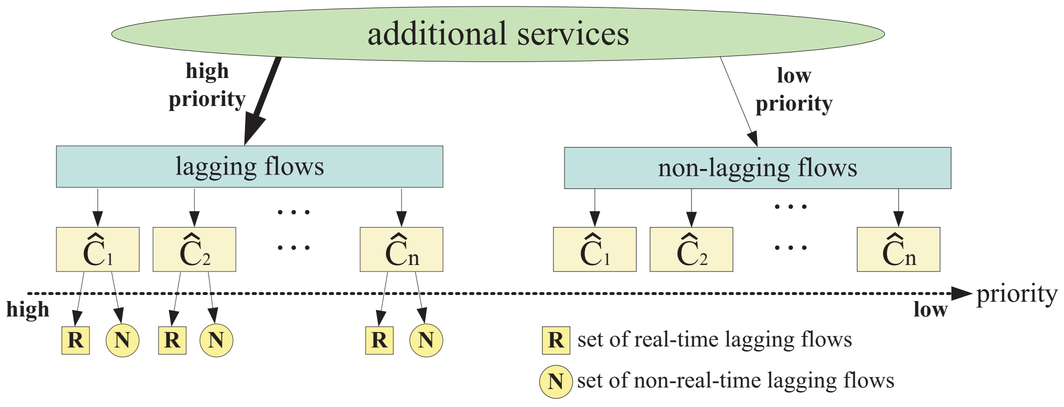


Figure 6.8: Dispatching additional services in the Multi-rate Compensation Scheme.

Next, we elaborate on the last rule. When dispatching additional services to lagging flows (i.e., flows on the left-hand side in Fig. 6.8), we keep track of the services received by real-time ones and non-real-time ones. Let $L_R = L_R^1 \cup L_R^2 \cup \cdots \cup L_R^n$ be the set of real-time, lagging flows, and $L_N = L_N^1 \cup L_N^2 \cup \cdots \cup L_N^n$ the set of non-real-time, lagging flows. To let real-time lagging flows receive more fraction of additional services without starving non-real-time lagging flows, we assign weights W_R and W_N (system parameters) to L_R and L_N, respectively, to control the fractions of additional services they already received, where $W_R > W_N$. A virtual time V_R (respectively, V_N) is used to record the normalized additional services received by L_R (respectively, L_N). Flows in Fig. 6.8 are checked from left to right. When both L_R^k and L_N^k are non-empty, where $1 \leq k \leq n$, the service is given to L_R if $V_R \leq V_N$, and to L_N otherwise. When only one of L_R^k and L_N^k is non-empty, the service is given to that one, independent of the values of V_R and V_N. When a flow in L_R receives the service, V_R is updated by Eq. (6.4). Similarly, when a flow in L_N receives the service, V_N is updated by Eq. (6.5).

When the scheduler selects either L_R^k or L_N^k, it distributes additional services proportional to the weights of flows in that set. Specifically, for each flow i, we maintain a compensation virtual time c_i to keep track of the normalized amount of additional services received by flow i. The scheduler selects the flow i with the smallest c_i to serve, and then updates c_i as

$$c_i = c_i + \left(\frac{l_p}{w_i} \times \frac{\hat{C}_1}{\hat{C}_k} \right),$$

Initially, when a flow i newly enters L_R or L_N, its c_i is set to

$$c_i = \max\{c_i, \min\{c_j \mid \text{flow } j \text{ belongs to the same set of flow } i\ (L_R \text{ or } L_N);\ j \neq i\}\}.$$

If there is no lagging flow in the previous stage, the service is returned back to the originally selected flow if it is a leading flow but rejected by the Gradual Degradation Scheme. Otherwise, the services is given to a non-lagging flow that can use the highest rate. In case of a tie, MR-FQ dispatches the services proportional to some weights. Specifically, each flow i is assigned with an extra virtual time f_i to keep track of the normalized amount of additional services received by flow i when it is non-lagging ($lag_i \leq 0$). When a backlogged flow i that can send packets becomes non-lagging, f_i is set to

$$f_i = \max\{f_i, \min\{f_j \mid \text{flow } j \text{ is backlogged, non-lagging and can send; } j \neq i\}\}.$$

The scheduler selects the flow i with the smallest f_i to serve. When flow i receives the service, f_i is updated as

$$f_i = f_i + \left(\frac{l_p}{w_i} \times \frac{\hat{C}_1}{r}\right),$$

where r is the current best rate for flow i.

6.3 *Theoretical Analyses on Fairness and Delay Bounds*

6.3.1 *Analyses of TD-FQ*

In this section, we analyze the fairness and delay properties of TD-FQ. Our analyses rely on the following assumptions: (i) $\alpha_R \geq \alpha_N$, (ii) $W_R \geq W_N$, (iii) $W_R^S \geq W_R^M$, (iv) $W_N^S \geq W_N^M$, and (v) $B \geq l_{\max}$, where $l_{\max}$ is the maximum length of a packet. The complete proofs can be found in Appendix A.

Fairness Properties

Theorems 6.1–6.3 show the fairness property guaranteed by TD-FQ. Theorem 6.1 is for flows of the same traffic type, while Theorem 6.2 is for flows of different types. Theorem 6.3 provides some bounds on differences of services received by L_R, L_N, L_R^S, L_R^M, L_N^S, and L_N^M.

Theorem 6.1. *For any two active flows i and j of the same traffic type, the difference between the normalized services received by flows i and j in any time interval $[t_1, t_2)$ during which both flows are continuously backlogged, error-free, and remain in the same state (leading, seriously lagging, moderately lagging, or satisfied) satisfies the inequality:*

$$\left|\frac{\Phi_i(t_1,t_2)}{w_i} - \frac{\Phi_j(t_1,t_2)}{w_j}\right| \leq \beta \cdot \left(\frac{l_{\max}}{w_i} + \frac{l_{\max}}{w_j}\right),$$

where $\Phi_i(t_1, t_2)$ represents the services received by flow i during $[t_1, t_2)$, $\beta = 3$ if both flows belong to the same lagging set (L_R^S, L_R^M, L_N^S, or L_N^M) or both flows are satisfied, $\beta = 3+\alpha_R$ if both flows are real-time leading flows, and $\beta = 3 + \alpha_N$ if both flows are non-real-time leading flows.

Theorem 6.2. *For any real-time flow i and non-real-time flow j, the difference between the normalized services received by flows i and j in any time interval $[t_1, t_2)$ during which both flows are continuously backlogged, error-free, and remain leading satisfies the inequality:*

$$\left|\frac{\Phi_i(t_1,t_2)}{w_i} - \frac{\Phi_j(t_1,t_2)}{w_j}\right| \leq 3 \cdot \left(\frac{l_{\max}}{w_i} + \frac{l_{\max}}{w_j}\right) + 2\alpha_N \frac{l_{\max}}{w_j}. \tag{6.9}$$

Theorem 6.3. *The difference between normalized ES/CS received by any two lagging sets in any time interval $[t_1, t_2)$ during which both sets remain active satisfies the inequalities:*

$$\begin{aligned}
&\textit{(1) for } L_R \textit{ and } L_N\textit{:} && \left|\frac{\Phi_R(t_1,t_2)}{W_R} - \frac{\Phi_N(t_1,t_2)}{W_N}\right| \leq \frac{B+l_{\max}}{W_R} + \frac{B+l_{\max}}{W_N},\\
&\textit{(2) for } L_R^S \textit{ and } L_R^M\textit{:} && \left|\frac{\Phi_R^S(t_1,t_2)}{W_R^S} - \frac{\Phi_R^M(t_1,t_2)}{W_R^M}\right| \leq \frac{B+l_{\max}}{W_R^S} + \frac{B+l_{\max}}{W_R^M},\\
&\textit{(3) for } L_N^S \textit{ and } L_N^M\textit{:} && \left|\frac{\Phi_N^S(t_1,t_2)}{W_N^S} - \frac{\Phi_N^M(t_1,t_2)}{W_N^M}\right| \leq \frac{B+l_{\max}}{W_N^S} + \frac{B+l_{\max}}{W_N^M},
\end{aligned}$$

where $\Phi_R(t_1, t_2)$, $\Phi_N(t_1, t_2)$, $\Phi_R^S(t_1, t_2)$, $\Phi_R^M(t_1, t_2)$, $\Phi_N^S(t_1, t_2)$, and $\Phi_N^M(t_1, t_2)$ represents ES/CS received by L_R, L_N, L_R^S, L_R^M, L_N^S, and L_M^N during $[t_1, t_2)$, respectively.

Delay Bounds

When a backlogged flow suffers from errors, it becomes lagging. Theorem 6.4 shows that if a lagging flow becomes error-free and has sufficient service demand, it can get back all its lagging services within bounded time.

Theorem 6.4. *If an active but lagging flow i becomes error-free at time t and remains backlogged continuously after time t, it is guaranteed that flow i will become non-lagging (i.e., $lag_i \leq 0$) within time Δ_t, where*

$$\Delta_t \leq \frac{\varpi(\varepsilon + 2l_{\max})}{w_{\min}(1-\alpha_R)\widehat{R}} + (m + 1 + \frac{\varpi}{w_{\min}})\frac{l_{\max}}{\widehat{R}},$$

m is the number of active flows, $\widehat{R}$ is the transmission rate, ϖ is the total weight of all flows, ϖ_R is the total weight of all real-time flows, ϖ_N is the total weight of all non-real-

time flows, $w_{\min}$ is the minimum weight of all flows, and

$$\varepsilon = \frac{(W_R + W_N)(W_R^S + W_R^M)}{W_R W_R^S}\left(\frac{lag_i(t)}{w_i}\varpi_R + (\frac{\varpi_R}{w_i} + m - 2)l_{\max} + B\right) + \frac{W_R + W_N}{W_R}\left(\delta\varpi_R + (\frac{2\varpi_R}{w_i} + m - 1)l_{\max} + B\right)$$

if flow i is real-time, and

$$\varepsilon = \frac{(W_R + W_N)(W_N^S + W_N^M)}{W_N W_N^S}\left(\frac{lag_i(t)}{w_i}\varpi_N + (\frac{\varpi_N}{w_i} + m - 2)l_{\max} + B\right) + \frac{W_R + W_N}{W_N}\left(\delta\varpi_N + (\frac{2\varpi_N}{w_i} + m - 1)l_{\max} + B\right)$$

if flow i is non-real-time.

6.3.2 Analyses of MR-FQ

In this section, we demonstrate that MR-FQ can guarantee fairness (including service fairness and time fairness) and bounded delays for packet flows by mathematical analyses. Our analyses rely on the following assumptions: (i) $\alpha_R \geq \alpha_N$, (ii) $W_R \geq W_N$, (iii) $B \geq l_{\max}$, and (iv) $r_i \in \{\hat{C}_1, \cdots, \hat{C}_n\}$, where r_i is the transmission rate used by flow i. A flow is called *allowed-to-send* if the Rate Selection Scheme returns a positive transmission rate to it, and is called a *candidate* if it can use a higher rate compared to other flows such that the scheduler may choose it to receive additional services in the Multi-rate Compensation Scheme. Besides, we let $r_i^{\min}$ be the smallest transmission rate that flow i has ever used during the nearest time interval when flow i is active. The complete proofs can refer to Appendix B.

Fairness Properties

Theorems 6.5 and 6.6 show the service fairness guaranteed by MR-FQ. Theorem 6.5 is for flows that have the similar conditions and Theorem 6.6 provides some bounds on differences of services received by L_R and L_N. Theorem 6.7 shows the time fairness guaranteed by MR-FQ.

Theorem 6.5. *For any two active flows i and j, assume that both flows are continuously backlogged and allowed-to-send, and remain in the same state (leading, lagging, or satisfied) during a time interval $[t_1, t_2)$. Let r_{RSC} and r_{MCS} be the transmission rates used by the these flows in the Rate Selection Scheme and the Multi-rate Compensation Scheme during $[t_1, t_2)$, respectively, where r_{RSC} and r_{MCS} are both in $\{\hat{C}_1, \cdots, \hat{C}_n\}$, and their values do not change during $[t_1, t_2)$. Then the difference between the normalized services* *received by flows i and j during $[t_1, t_2)$ satisfies the following inequality:*

$$\left|\frac{\Phi_i^s(t_1,t_2)}{w_i} - \frac{\Phi_j^s(t_1,t_2)}{w_j}\right| \leq \beta \cdot \frac{l_{\max}}{w_i} + \gamma \cdot \frac{l_{\max}}{w_j},$$

where $\Phi_i^s(t_1, t_2)$ represents the services received by flow i during $[t_1, t_2)$, and

$$(\beta,\gamma) = \begin{cases} (\frac{r_{RSC}}{r_i^{\min}}+1, \frac{r_{RSC}}{r_j^{\min}}+1), & \textit{if both flows are lagging but not candidates} \\ (\frac{r_{RSC}+r_{MCS}}{r_i^{\min}}+1, \frac{r_{RSC}+r_{MCS}}{r_j^{\min}}+1), & \textit{if both flows are lagging and candidates} \\ (\frac{r_{RSC}+r_{MCS}}{r_i^{\min}}+1, \frac{r_{RSC}+r_{MCS}}{r_j^{\min}}+1), & \textit{if both flows are satisfied} \\ (\frac{r_{MCS}+\alpha_R\hat{C}_1}{r_i^{\min}}+2, \frac{r_{MCS}+\alpha_R\hat{C}_1}{r_j^{\min}}+2), & \textit{if both flows are real-time leading flows} \\ (\frac{r_{MCS}+\alpha_N\hat{C}_1}{r_i^{\min}}+2, \frac{r_{MCS}+\alpha_N\hat{C}_1}{r_j^{\min}}+2), & \textit{if both flows are non-real-time leading flows} \\ (\frac{r_{MCS}}{r_i^{\min}}+2, \frac{r_{MCS}+2\alpha_N\hat{C}_1}{r_j^{\min}}+2), & \textit{if } i \textit{ and } j \textit{ are real-time and non-real-time} \\ & \textit{leading flows, respectuively.} \end{cases}.$$

Theorem 6.6. *The difference between normalized additional services received by L_R and L_N in any time interval $[t_1, t_2)$ during which both sets remain active (i.e., there exists at least one candidate in each set) satisfies the following inequality:*

$$\left|\frac{\Phi_R(t_1,t_2)}{W_R} - \frac{\Phi_N(t_1,t_2)}{W_N}\right| \leq \frac{B+l_{\max}}{W_R} + \frac{B+l_{\max}}{W_N},$$

where $\Phi_R(t_1, t_2)$ and $\Phi_N(t_1, t_2)$ are additional services received by L_R and L_N during $[t_1, t_2)$, respectively.

Theorem 6.7. *For any two active flows i and j, the difference between the normalized transmission time used by flows i and j in any time interval $[t_1, t_2)$ during which both flows are continuously backlogged and allowed-to-send, and remain in the same state (leading, lagging, or satisfied) satisfies the following inequality:*

$$\left|\frac{\Phi_i^t(t_1,t_2)}{w_i} - \frac{\Phi_j^t(t_1,t_2)}{w_j}\right| \leq \beta \cdot \frac{l_{\max}}{w_i} + \gamma \cdot \frac{l_{\max}}{w_j},$$

where $\Phi_i^t(t_1, t_2)$ represents the transmission time used by flow i during $[t_1, t_2)$, and

$$(\beta,\gamma) = \begin{cases} (\frac{\hat{C}_1}{r_i^{\min}}+1, \frac{\hat{C}_1}{r_j^{\min}}+1), & \textit{if both flows are lagging but not candidates} \\ (\frac{2\hat{C}_1}{r_i^{\min}}+1, \frac{2\hat{C}_1}{r_j^{\min}}+1), & \textit{if both flows are lagging and candidates} \\ (\frac{2\hat{C}_1}{r_i^{\min}}+1, \frac{2\hat{C}_1}{r_j^{\min}}+1), & \textit{if both flows are satisfied} \\ (\frac{(\alpha_R+1)\hat{C}_1}{r_i^{\min}}+2, \frac{(\alpha_R+1)\hat{C}_1}{r_j^{\min}}+2), & \textit{if both flows are real-time leading flows} \\ (\frac{(\alpha_N+1)\hat{C}_1}{r_i^{\min}}+2, \frac{(\alpha_N+1)\hat{C}_1}{r_j^{\min}}+2), & \textit{if both flows are non-real-time leading flows} \\ (\frac{\hat{C}_1}{r_i^{\min}}+2, \frac{(2\alpha_N+1)\hat{C}_1}{r_j^{\min}}+2), & \textit{if } i \textit{ and } j \textit{ are real-time and non-real-time} \\ & \textit{leading flows, respectuively.} \end{cases}.$$

Delay Bounds

Theorem 6.8 shows that if a lagging flow which has sufficient service demand becomes allowed-to-send and is always a candidate in the Multi-rate Compensation Scheme, it can get back all its lagging services within bounded time.

Theorem 6.8. *If an active but lagging flow i which remains backlogged continuously becomes allowed-to-send and is always a candidate in the Multi-rate Compensation Scheme, it is guaranteed that flow i will become non-lagging (i.e., $lag_i \leq 0$) within time Δ_t, where*

$$\Delta_t < \frac{\varpi(\varepsilon + 2l_{\max})}{w_{\min}(1-\alpha_R)\hat{C}_n} + \left(\frac{\hat{C}_1}{\hat{C}_n}(m + \frac{\varpi}{w_{\min}}) + 1\right)\frac{l_{\max}}{\hat{C}_n},$$

m is the number of active flows, ϖ is the total weight of all flows, ϖ_R is the total weight of all real-time flows, ϖ_N is the total weight of all non-real-time flows, $w_{\min}$ is the minimum weight of all flows, and

$$\varepsilon = \frac{W_R + W_N}{W_R}\left(\frac{\hat{C}_1}{\hat{C}_n}(\frac{\varpi_R \cdot lag_i(t)}{w_i} + (\frac{2\varpi_R}{w_i} + m - 2)l_{\max}) + 2l_{\max} + B\right),$$

if flow i is real-time, and

$$\varepsilon = \frac{W_R + W_N}{W_N}\left(\frac{\hat{C}_1}{\hat{C}_n}(\frac{\varpi_N \cdot lag_i(t)}{w_i} + (\frac{2\varpi_N}{w_i} + m - 2)l_{\max}) + 2l_{\max} + B\right),$$

if flow i is non-real-time.

6.4 Experimental Results

In this section, we present some experimental results to verify the effectiveness of the proposed TD-FQ and MR-FQ algorithms. In Section 6.4.1, we compare the packet dropping ratios and queuing delays of real-time flows and the throughput of flows in TD-FQ and CIF-Q. In Section 6.4.2, we verify the effectiveness of MR-FQ in a multi-rate environment. In addition, we also evaluate the effect of time fairness property on MR-FQ.

6.4.1 Performance Evaluation of TD-FQ

Dropping Ratios and Delays of Real-time Flows

In the first experiment, we mix real-time and non-real-time traffics together. We observe the packet dropping ratios and queuing delays of real-time flows in TD-FQ and CIF-Q, respectively. Eight flows are used, as shown in Table 6.2. The first six flows are real-time flows, which have two traffic models: constant-bit-rate (CBR) and ON-OFF model. The latter can be used to model a periodic event reported by sensors. The average durations of ON and OFF states are set to 2.5 and 0.5 seconds, respectively. During ON period, packets are generated with fixed intervals. No packet is generated during OFF period. The last two flows are non-real-time flows without delay constraints, and their traffics are modeled as greedy sources whose queues are never empty. As for error scenarios, we use two parameters P_{good} and P_{bad} to control the average time when the channel stays in error-free and error states, respectively. The total channel capacity is set to 5 Mb/s. The total simulation time in this experiment is 100 seconds.

Table 6.2: Traffic specification of the flows used in the first experiment in Section 6.4.1.

flow	guaranteed bandwidth	packet size	error scenario
periodic1	64 Kb/s	2 Kb	no error occurs
periodic2	32 Kb/s	1 Kb	$P_{good} = 6$ sec., $P_{bad} = 1.5$ sec.
periodic3	32 Kb/s	1 Kb	$P_{good} = 5$ sec., $P_{bad} = 0.5$ sec.
CBR1	512 Kb/s	2 Kb	no error occurs
CBR2	256 Kb/s	1 Kb	$P_{good} = 6$ sec., $P_{bad} = 1.5$ sec.
CBR3	256 Kb/s	1 Kb	$P_{good} = 5$ sec., $P_{bad} = 0.5$ sec.
greedy1	2 Mb/s	4 Kb	no error occurs
greedy2	2 Mb/s	4 Kb	$P_{good} = 6$ sec., $P_{bad} = 1.5$ sec.

For CIF-Q, we set $\alpha = 0.5$, while for TD-FQ we set $\alpha_R = 0.8$ and $\alpha_N = 0.2$, respectively. The weights assigned to lagging sets are $W_R : W_N = 3 : 1$, $W_R^S : W_R^M = 3 : 1$, and $W_N^S : W_N^M = 3 : 1$. The packet dropping ratios and queuing delays of real-time flows are shown in Fig. 6.9 and Fig. 6.10, respectively, where the packet dropping ratio is defined as

$$\frac{\text{Number of packets dropped because of exceeding their deadlines}}{\text{Number of packet generated}},$$

where the deadline of a packet is set to twice of the packet inter-arrival time. From Fig. 6.9 and Fig. 6.10, we can observe that the packet dropping ratios and queuing delays of real-time flows in TD-FQ are smaller than those in CIF-Q, especially when the flows are periodic traffic. This is because TD-FQ not only lets real-time flows give up less services to compensate other lagging flows, but also gives more services to real-time lagging flows for compensation. From this observation, we conclude that TD-FQ can alleviate the packet dropping ratios and queuing delays of real-time flows as compared to CIF-Q.

Throughput of Flows

In the second experiment, we observe the throughput of flows in TD-FQ and CIF-Q. Four flows are used, as shown in Table 6.3. The first two flows are real-time CBR flows, and the

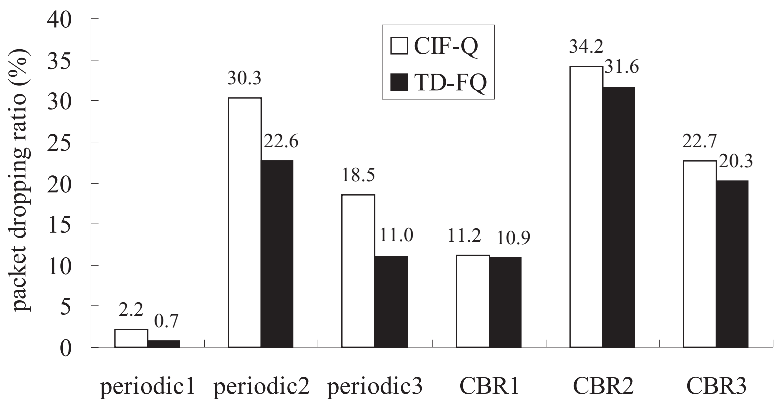


Figure 6.9: Packet dropping ratios of real-time flows.

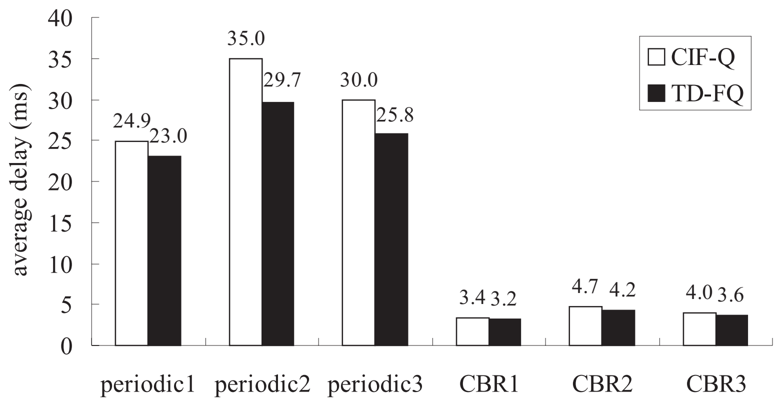


Figure 6.10: Average queuing delays of real-time flows.

last two flows are non-real-time greedy flows. Suffering from channel errors during $[0, 15)$ period, flows CBR2 and greedy2 will become active but lagging after the 15th second. The other flows are all leading in this experiment. For CIF-Q, we set $\alpha = 0.5$, while for TD-FQ we set $\alpha_R = 0.8$, $\alpha_N = 0.2$, $W_R = 3$, and $W_N = 1$. The channel capacity in this experiment is set to 2 Mb/s.

Table 6.3: Traffic specification of the flows used in the second experiment Section 6.4.1.

flow	guaranteed bandwidth	packet size	error scenario
CBR1	1.25 Mb/s	4 Kb	no error occurs
CBR2	1.25 Mb/s	4 Kb	error occurs during [0,15) sec.
greedy1	2 Mb/s	8 Kb	no error occurs
greedy2	2 Mb/s	8 Kb	error occurs during [10,15) sec.

Fig. 6.11 shows the throughput of flows after the 16th second. We see that real-time flows can receive more services in TD-FQ as compared to CIF-Q. This is because TD-FQ favors real-time flows over non-real-time flows. However, the cost, as shown in Fig. 6.11(b), is at lower throughput for non-real-time flows.

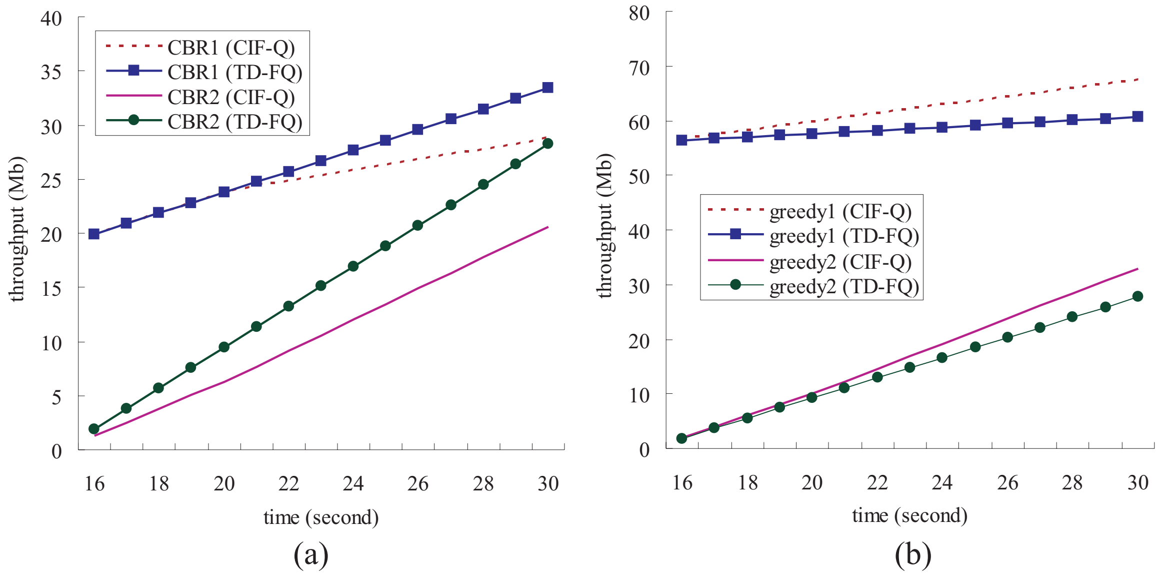


Figure 6.11: Throughput of (a) real-time flows CBR1 and CBR2 and (b) non-real-time flows greedy1 and greedy2.

6.4.2 *Performance Evaluation of MR-FQ*

Impact of a Multi-rate Environment

In the first experiment, we evaluate the impact of a multi-rate environment on our MR-FQ method and other wireless fair scheduling algorithms. We mix real-time and non-real-time flows together. We mainly observe the packet dropping ratios and the average queuing delays of real-time flows and the average throughput of non-real-time flows. We compare CIF-Q and TD-FQ with MR-FQ. CIF-Q and TD-FQ both assume that the wireless channel is either in a good state or a bad state. We assume that the MAC protocol can provide 11 Mb/s, 5.5 Mb/s, 2 Mb/s, and 1 Mb/s transmission rates. Ten flows are used, as shown in Table 6.4. The first six flows are real-time flows, which represent three traffic models: periodic, variable-bit-rate (VBR), and CBR traffics. The average durations of ON and OFF periods of a periodic traffic are set to 2.5 and 0.5 seconds, respectively. For a VBR traffic, packets arrive in a Poisson fashion. The last four flows are non-real-time greedy flows. For error scenarios, we also use P_{good} and P_{bad} to adjust the average time when a channel stays in good and bad states, respectively. When the channel is in the good state, the flow can use 11 Mb/s to transmit. When the channel is in the bad state, the best transmission rate that a flow can use in MR-FQ is randomly selected from 5.5, 2, 1, and 0 Mb/s. However, both CIF-Q and TD-FQ simply treat the channel is bad and no packet can be transmitted. The total simulation time in this experiment is 30 minutes.

For CIF-Q, we set its parameter $\alpha = 0.5$, while for TD-FQ and MR-FQ, we set their

Table 6.4: Traffic specification of the flows used in the first experiment in Section 6.4.2.

flow	guaranteed bandwidth	packet size	error scenario
periodic1	64 Kb/s	2 Kb	P_{good} = 8 sec., P_{bad} = 1.5 sec.
periodic2	32 Kb/s	1 Kb	P_{good} = 5 sec., P_{bad} = 1 sec.
VBR1	2 Mb/s	4 Kb	P_{good} = 8 sec., P_{bad} = 1.5 sec.
VBR2	1 Mb/s	2 Kb	P_{good} = 5 sec., P_{bad} = 1 sec.
CBR1	512 Kb/s	2 Kb	P_{good} = 8 sec., P_{bad} = 1.5 sec.
CBR2	256 Kb/s	1 Kb	P_{good} = 5 sec., P_{bad} = 1 sec.
greedy1	2 Mb/s	4 Kb	P_{good} = 9.5 sec., P_{bad} = 0.5 sec.
greedy2	2 Mb/s	4 Kb	P_{good} = 8 sec., P_{bad} = 1.5 sec.
greedy3	2 Mb/s	4 Kb	P_{good} = 5sec., P_{bad} = 1 sec.
greedy4	2 Mb/s	4 Kb	P_{good} = 3 sec., P_{bad} = 1 sec.

parameters $\alpha_R = 0.8$ and $\alpha_N = 0.2$, respectively. In TD-FQ, the weights assigned to lagging sets are $W_R : W_N = 3 : 1$, $W_R^S : W_R^M = 3 : 1$, and $W_N^S : W_N^M = 3 : 1$. In MR-FQ, we set $W_R : W_N = 3 : 1$. In addition, the values of δ_1, δ_2, δ_3, and B in MR-FQ are set to 32, 64, 128, and 1024, respectively.

The packet dropping ratios and the average queuing delays of real-time flows are shown in Figs. 6.12 and 6.13, respectively. We can observe that real-time flows have the highest packet dropping ratios and average queuing delays when we apply CIF-Q to the scheduler. This is because CIF-Q does not separate real-time flows from non-real-time flows and treat all flows in the same way. Real-time flows then have to compete with non-real-time flows, thus causing higher dropping ratios and queuing delays. The packet dropping ratios and the average queuing delays of real-time flows in TD-FQ are smaller than those in CIF-Q because TD-FQ gives higher priorities to real-time flows to reduce their queuing delays. MR-FQ allows flows in a bad state to transmit packets using lower rates (if possible), so the packet dropping ratios and the average queuing delays of real-time flows in MR-FQ will be the smallest.

The similar effect can be observed in Fig. 6.14, where the average throughput of non-real-time flows in MR-FQ are larger than those in CIF-Q and TD-FQ.

From this experiment, we can conclude that by considering multi-rate capability of a wireless channel, the proposed MR-FQ method can reduce the packet dropping ratios and average queuing delays of real-time flows and increase the overall system performance.

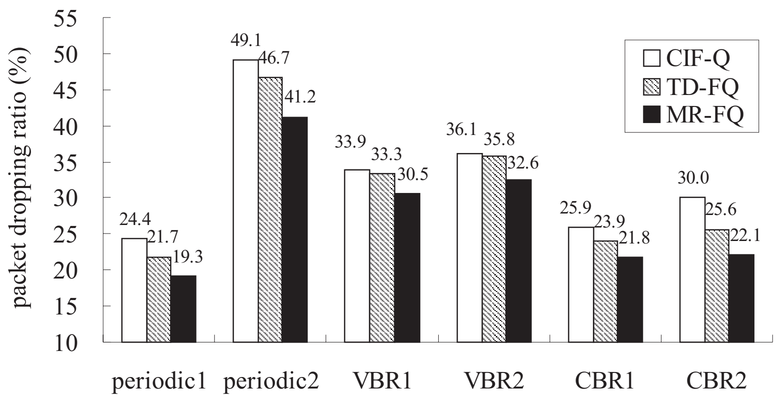


Figure 6.12: Packet dropping ratios of real-time flows.

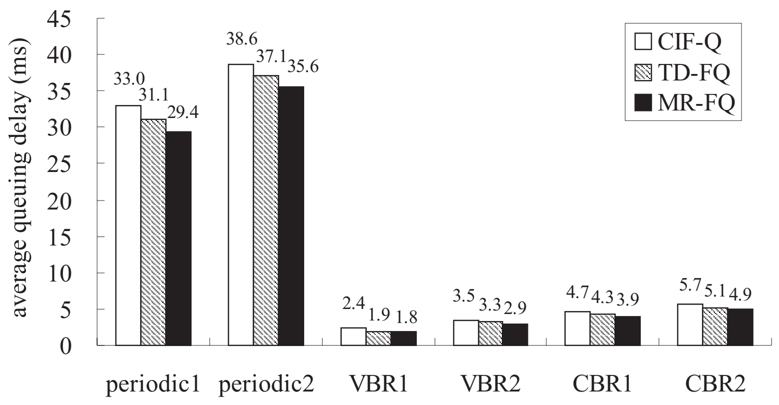


Figure 6.13: Average queuing delays of real-time flows.

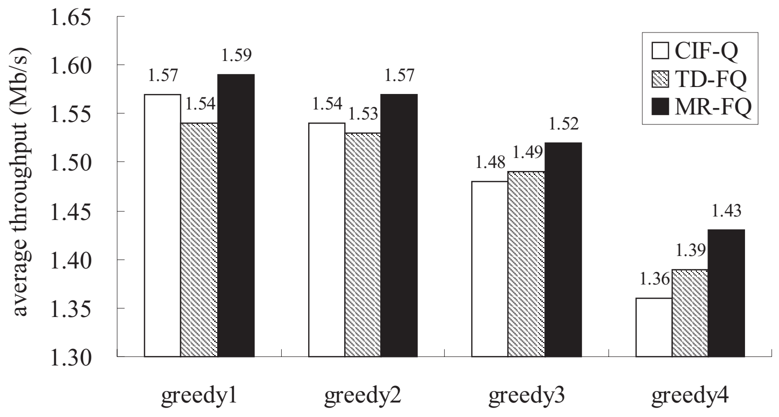


Figure 6.14: Average throughput of non-real-time flows.

Time Fairness Property

In the second experiment, we evaluate the effect of time fairness property on MR-FQ. Recall that there are two parts in MR-FQ that address the time fairness issue. One is the Rate Selection Scheme, which chooses a suitable transmission rate for the selected flow according to its lagging degree and channel condition. A flow is allowed to use a lower rate for transmission only if it suffers from seriously lagging. Another is the ratio

$\frac{\hat{C}_1}{r}$ used to update a flow's virtual time, where r is the transmission rate used by the flow. To observe the effect of time fairness property, we design a modified version of MR-FQ that does not consider the time fairness property. This modified version removes the Rate Selection Scheme and updates a flow i's virtual time as $v_i = v_i + \frac{l_p}{w_i}$, $c_i = c_i + \frac{l_p}{w_i}$, and $f_i = f_i + \frac{l_p}{w_i}$, where l_p is the length of the packet being transmitted. We mainly observe the total services received by flows and the total medium time used by flows. Two greedy flows are used, as shown in Table 6.5. The total simulation time in this experiment is 100 seconds.

Table 6.5: Traffic specification of the flows used in the second experiment in Section 6.4.2.

flow	guaranteed bandwidth	packet size	error scenario
greedy1	6 Mb/s	8 Kb	$P_{good} = 10$ sec., $P_{bad} = 1$ sec.
greedy2	6 Mb/s	8 Kb	$P_{good} = 4$ sec., $P_{bad} = 2.5$ sec.

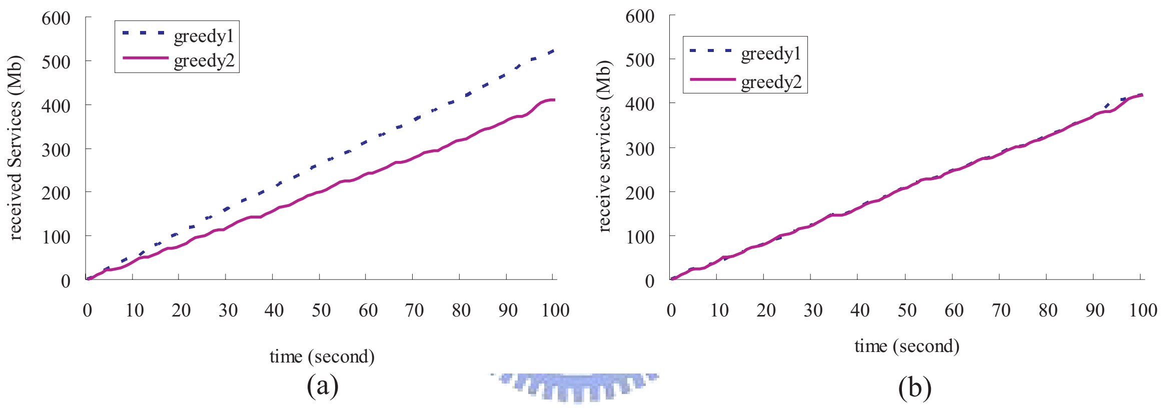


Figure 6.15: Total services received by the two greedy flows: (a) MR-FQ and (b) MR-FQ without considering time fairness.

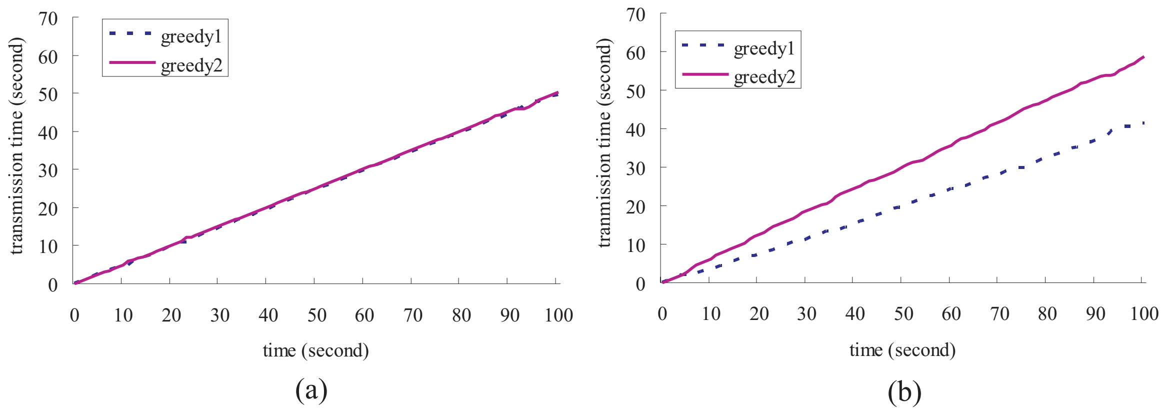


Figure 6.16: Total medium time used by the two greedy flows: (a) MR-FQ and (b) MR-FQ without considering time fairness.

Figs. 6.15 and 6.16 show the total services received and the total medium time used by these two greedy flows, respectively. Since the channel condition of the flow greedy1 is better than that of the flow greedy2, MR-FQ will let the flow greedy1 receive more services, as shown in Fig. 6.15(a). However, the medium time used by both flows are the same in MR-FQ, as shown in Fig. 6.16(a). On the contrary, although the modified version of MR-FQ can achieve better service fairness (as shown in Fig. 6.15(b)), it makes the flow greedy2 occupy too much medium time, as shown in Fig. 6.16(b). Since the flow greedy2 has a worse channel condition, it will often use lower transmission rates to send packets, thus causing a longer transmission time. By comparing Fig. 6.15 (a) and (b), we can observe that the total services received by the flow greedy1 in the modified version of MR-FQ are quite lower than that in MR-FQ. This reflects the fact that if we do not consider the time fairness issue, the flows using lower transmission rates will degrade the amount of services received by other flows, and thus decreasing the overall system performance.

6.5 Summary

In this chapter, we have proposed two fair scheduling algorithms for data aggregators to manage reporting from sensor nodes. The proposed TD-FQ algorithm takes traffic types of flows into consideration and gives a higher priority for real-time flows to reduce their queuing delays. Although TD-FQ benefits real-time flows, it does not starve non-real-time flows. By taking both time fairness and service fairness into account, the proposed MR-FQ algorithm allows a flow to transmit at different rates according to its channel condition and lagging degree. MR-FQ not only increases the overall system throughput, but also guarantees fairness and bounded delays for flows. In this chapter, we have analytically derived the fairness properties and delay bounds of TD-FQ and MR-FQ. Simulation results have also shown that TD-FQ incurs less packet dropping for real-time flows when compared with CIF-Q, while MR-FQ has a larger throughput of flows compared with both CIF-Q and TD-FQ when multi-rate transmissions are allowed.

Chapter 7

Implementation of a Mobile Sensor Platform: the iMouse System

Wireless mobile sensor networks provide us a convenient manner to monitor physical environments. It can be an attractive research direction to integrate the context-aware capability of such network into a surveillance system . In this chapter, we therefore propose an *integrated mobile surveillance and wireless sensor (iMouse) system*. The iMouse system is composed of a large number of static sensors and a small number of more powerful mobile sensors. The former can be used to detect unusual events in the environment, while the latter is able to visit the event locations reported from static sensors to conduct more advanced analyses of events. The iMouse system is thus a mobile, context-aware surveillance system. We demonstrate our current prototyping system for a home/office security application.

7.1 Motivation

One of the major goals of this chapter is to study the possibility to integrate wireless sensor networks into surveillance systems. Most of traditional visual surveillance systems handle the real-time monitoring of stationary and moving objects in the environment. In particular, the chief goal of these systems is to provide an automatic interpretation of scenes and thus to realize or even predict actions of the observed objects from the information obtained from wall-board cameras or closed circuit television (CCTV) [116]. For example, the work in [38] proposes a *video-based surveillance network*. In such a surveillance network, the information gathered by each video camera will be transmitted through an IEEE 802.11 WLAN[1] card. The similar issue has also been addressed in

[1]wireless local area network

the field of robotics [55, 67, 68], which discuss how to navigate a robot by the visual information provided from cameras installed on walls. Specifically, with these information, the server (or the robot itself) can recognize the stationary obstacles or humans in the field, so that the robot can bypass these obstacles. However, the aforementioned surveillance systems have to extract the relatively little meaningful information from massive visual messages, which requires a large amount of manpower or computation to analyze.

As for wireless sensor networks, the issue of object/event tracking has been widely discussed [5, 47, 60, 65, 72, 129]. Most of these works consider that the intrusive object will emit some signals such as light or sound, or the object itself is a phenomenon (e.g., chemical liquid or diffused gas [60]). However, events described by sensor nodes are typically quite brief and lack of more in-depth information. This motivates us to study the possibility to combine surveillance systems with wireless sensor networks to support intelligent surveillance services.

7.2 The System Architecture

The system architecture of our iMouse system is illustrated in Fig. 7.1. This system is composed of a large number of static sensors, a small number of mobile sensors, and a control server. These static sensors form a sensor network to monitor the environment and they will notify the server if there are events occurring. When something unusual is reported from static sensors, the server will dispatch mobile sensors to visit these emergency sites. The mobile sensors will then move to these sites and conduct more in-depth analyses of events. In the following, we detail each system component separately.

Each static sensor is composed of a sensing board and a Mote. The former is used to collect information from the environment, while the latter is used to communicate with other sensors. In our current implementation, three kinds of sensing data can be gathered, including light, sound, and temperature. An *event* is defined when the sensory input is higher or lower than a predefined threshold, depending on the type of sensing data and the application. Different sensory inputs can be combined to define a new event. For example, a combination of light and temperature thresholds can be used to define a potential fire emergency. To detect an explosion, a combination of temperature and sound thresholds can be adopted. For home/office security, an unusual sound or light may be used. More sophisticated sensing devices can be added to increase the capabilities of static sensors. To conserve the energies of static sensors, reporting of events is *reactive*, in the sense that static sensors will report their sensing data only when they detect some specified events.

Each mobile sensor has the following functionalities:

- Receive commands from the server and then move to the specified emergency sites.

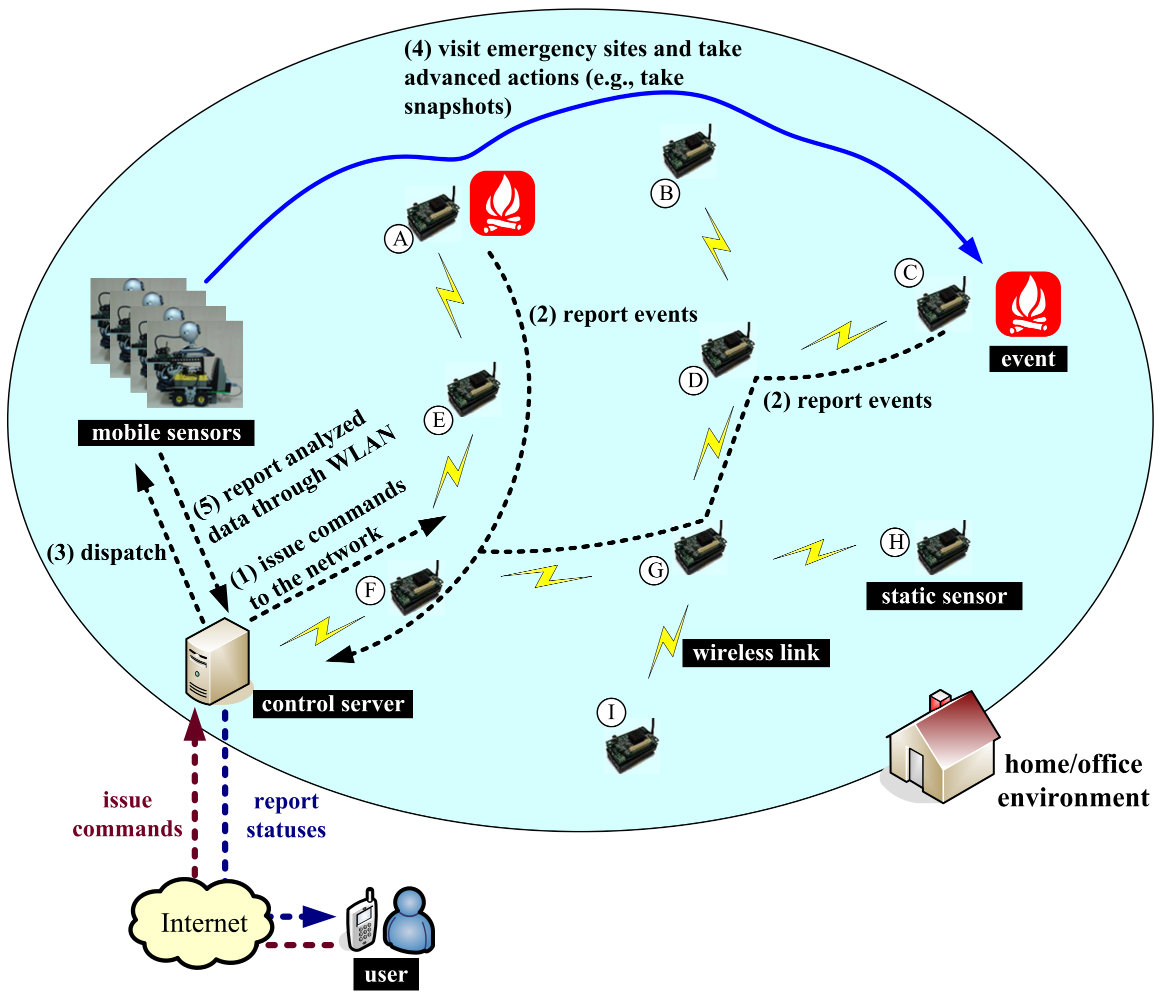


Figure 7.1: The system architecture of the proposed iMouse system.

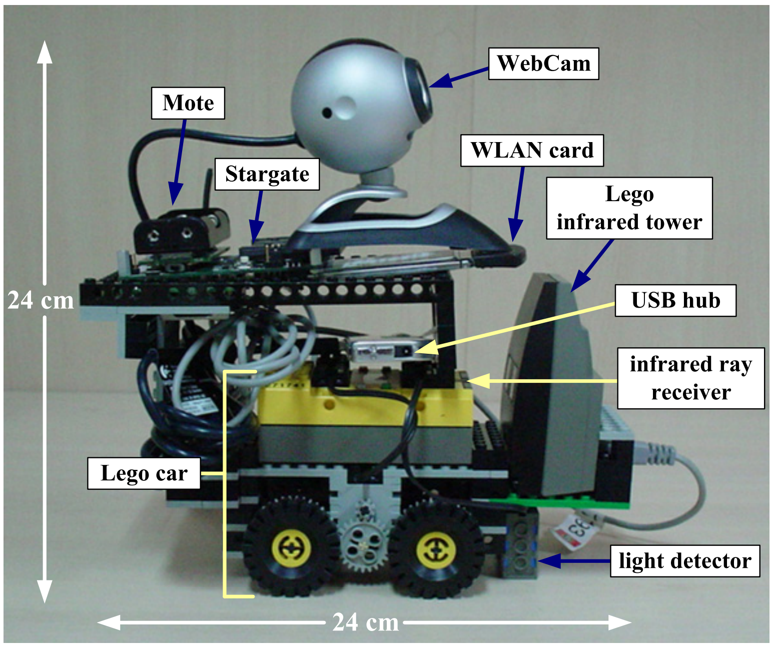


Figure 7.2: The mobile sensor in our iMouse system.

- Exchange messages with neighboring static or mobile sensors.
- Take snapshots from the vicinity of event locations.
- Transmit snapshots and analyzed data to the server.

To implement the aforementioned functions, each mobile sensor is equipped with a processing platform (called *Stargate*), which is connected with a Lego car, a Mote, a WebCam, and an IEEE 802.11 WLAN card, as illustrated in Fig. 7.2. The Lego car provides mobility for the mobile sensor. The Mote is used to communicate with other static sensors. The WebCam is used to take snapshots around the emergency sites. To support a larger communication range and a higher bandwidth, the 802.11 WLAN card is used to communicate with other mobile sensors and to report visual information to the server. Finally, the Stargate is the processing unit of the mobile sensor, which controls the actions of the mobile sensor (such as movement and taking snapshots) according to the messages read from the Mote or WLAN interface.

The control server coordinates all the system components in the iMouse system. It acts as a remote sink to collect information (such as the locations of events) from the static sensors and then dispatches mobile sensors according to the received information. This server also provides a user interface to help people access the statuses of the environment (e.g., locations and snapshots of events) and issue commands to the system. The users can communicate with the server through the Internet.

Next, we present a fire emergency scenario to demonstrate how the iMouse system works, as shown in Fig. 7.1. In the beginning, the control server issues a command to make static sensors form a spanning tree to build up the network and to monitor the environment. Suppose that static sensors A and C have reported unusual high temperatures and are thus suspected of fire emergency in their vicinity. Once receiving such reports, the server will notify the users and then dispatch mobile sensors to visit these emergency sites. After moving to A and C, the mobile sensors will take snapshots in their neighborhoods and send these pictures back to the server for further actions[2].

7.3 Design of the iMouse System

7.3.1 System Operations and Control Flows

Below, we detail the operations and control flows of each system component separately. The control server periodically broadcasts a *tree-maintenance* message to maintain the sensor network. The server also records each static sensor's location (which can be obtained when deployment) and its status, which is set to *normal* initially. When the server receives a *status-change* message from a static sensor, if the static sensor's current status is normal, then it means that there is a new event occurring and thus the server will set this static sensor's status as *abnormal* and then dispatch mobile sensors to visit this emergency site (i.e., the static sensor). Otherwise, the server will reset the static sensor's status as normal, in the sense that the old event has disappeared. To manage mobile sensors, the server maintains a *mobile sensor table*, with each entry having a format of ⟨mobile sensor's ID, current location, visiting sites⟩, where the last two items indicate the current location of the corresponding mobile sensor and the emergency sites that it has to visit, respectively. The server will dispatch mobile sensors according to this table. In particular, the server will first select the mobile sensors whose visiting sites are empty (i.e., no mission assigned), and assign an emergency site to the mobile sensor whose current location is closest to it. After the assignment, the server will send a *visit* message to the mobile sensors with the locations that they have to visit.

Each static sensor executes the procedure in Fig. 7.3. Specifically, a static sensor will wait for messages from the server or other sensors and then conduct actions according to the received message. A *tree-maintenance* message will command the static sensor to check if its parent node is null (or has expired). If the parent node is null, the static sensor will set the sender as its tree parent. It then rebroadcasts the *tree-maintenance* message to its neighbors. To distinguish new from old messages, each *tree-maintenance* message is associated with a unique sequence number. The objective is to maintain a spanning tree

[2]For example, the user can determine whether this event is a false alarm or a true fire emergency and thus decide to call the firemen if needed.

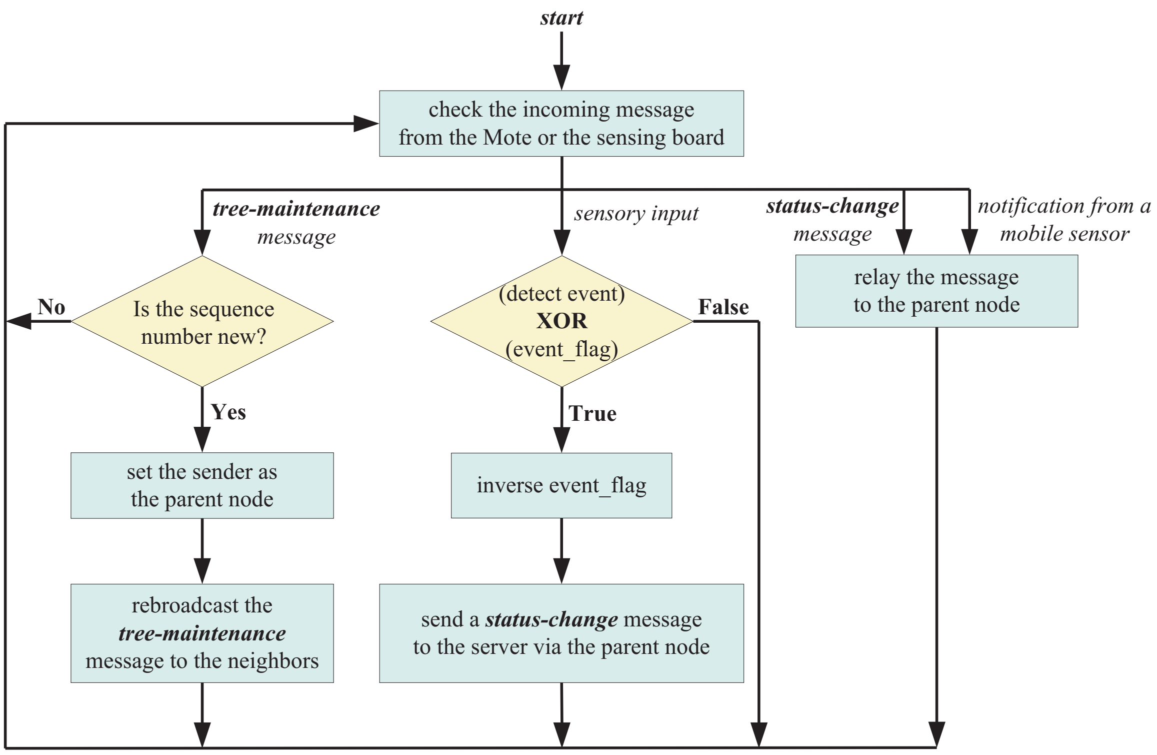

start
check the incoming message from the Mote or the sensing board
tree-maintenance message
sensory input
status-change message
notification from a mobile sensor
No
Is the sequence number new?
(detect event) XOR (event_flag)
False
relay the message to the parent node
Yes
True
set the sender as the parent node
inverse event_flag
rebroadcast the tree-maintenance message to the neighbors
send a status-change message to the server via the parent node

Figure 7.3: The procedure executed by the static sensors.

of the network to collect information from the environment. To determine if the current status changes from the sensory input, each static sensor keeps an *event_flag*, which is initially set to false, to indicate whether it has detected an event or not. The static sensors will decide whether to report a status-change message to the server according to Table 7.1, which reflects an XOR relationship of whether to detect an event and the value of event_flag. Using the event_flag can help reduce communication overheads of static sensors since events usually occur in a non-short period and the static sensors do not need to continuously report to the server during the occurrence of events.

detect event	event_flag	Does the static sensor need to report a *status-change* message?
true	true	No (This is an old event that has been reported before.)
true	false	Yes (This is a new event.)
false	true	Yes (The old event has disappeared.)
false	false	No (Nothing happens.)

Table 7.1: Decision of whether to report a *status-change* message.

Fig. 7.4 illustrates the procedure executed by mobile sensors. On receiving the server's *visit* message, which contains the emergency sites that it has to visit, the mobile sensor can use the modified version of approximate traveling-salesman algorithm [97] APPROX-TSP-TOUR in Fig. 7.5 to compute its patrolling path. During the movement, two neighboring mobile sensors will exchange their current locations with each other to avoid potential collision. In particular, when two mobile sensors want to pass the same location, the one with a larger ID can always have the highest priority to move through that location while the one with a smaller ID has to wait. When the mobile sensor move to a specified emergency site, it will take snapshots in the vicinity and saves the pictures in its Stargate's flash memory. The mobile sensor also notifies the server of the emergency site that it currently visits to make the server update its current location. This notification will be transmitted through the static sensor network to make sure that the server can receive the message. After visiting all emergency sites assigned by the server, the mobile sensor will move into the WLAN communication range of the server to transmit the pictures of all visited emergency sites that it has taken, and the server will also update the corresponding entry in its mobile sensor table.

7.3.2 Implementation Details and User Interface

In this section, we report our prototyping experiences of the proposed iMouse system. We adopt the MICAz Motes [25] as the static sensors. The MICAz is an IEEE 802.15.4-compliant [66] Mote module enabling low-power operations and providing a data rate of 250 kbps with a DSSS[3] radio in 2.4 GHz frequency band. For mobile sensors, we use the Stargate [26] as their processing platforms. The Stargate contains a 32-bits, 400-MHz Intel PXA-255 XScale RISC processor with a 64 MB main memory and a 32 MB extended flash memory. It also has a daughter board with a PCMCIA slot, an RS-232 serial port, a USB port, and a 51-pin extension connector, which can be attached to a Mote. The Stargate can drive a WebCam through its USB port, and an IEEE 802.11 WLAN card through its PCMCIA slot. The Stargate controls the Lego car [84] through a USB port connected to a Lego infrared tower, as shown in Fig. 7.2. The Lego car has an infrared ray receiver in the front to receive commands from the infrared tower, and two motors on the bottom to provide mobility. The Lego car also contains a light detector, and we use it for the navigation purpose. Specifically, this can be realized by different colors of the tapes that we stick on the ground. This mechanism also helps a mobile sensor localize itself in the sensing field.

In our current prototyping system, an experimental 6×6 grid-like sensing field is implemented, as shown in Fig. 7.6. On the ground, we stick black tapes as the roads and golden tapes as the intersections. Two mobile sensors and 17 static sensors are placed on the sensing field to construct the system. For static sensors, a light reading below 800 is to simulate an event, so we can cover a static sensor with a box to model a potential

[3]Direct Sequence Spread Spectrum

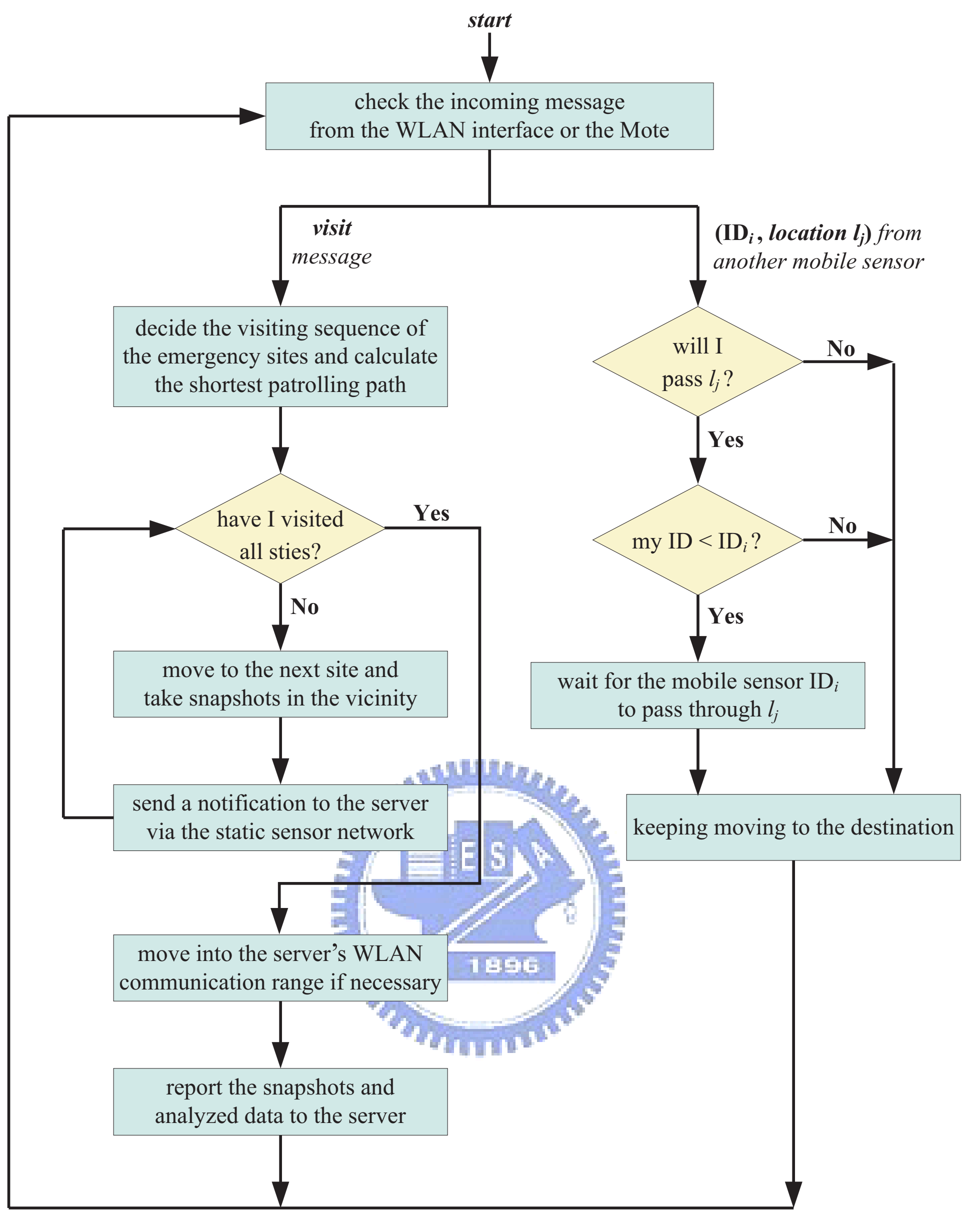

start
check the incoming message from the WLAN interface or the Mote
visit message
(ID$_i$, location l_j) from another mobile sensor
decide the visiting sequence of the emergency sites and calculate the shortest patrolling path
will I pass l_j?
No
Yes
have I visited all sties?
Yes
No
my ID < ID$_i$?
No
Yes
move to the next site and take snapshots in the vicinity
wait for the mobile sensor ID$_i$ to pass through l_j
send a notification to the server via the static sensor network
keeping moving to the destination
move into the server's WLAN communication range if necessary
report the snapshots and analyzed data to the server

Figure 7.4: The procedure executed by the mobile sensors.

```
algorithm APPROX-TSP-TOUR
Input:   $G = (V,E)$: a complete graph, where $V$ is the set of visiting sites
Output:  a Hamiltonian path to visit all sites
1:  select the nearest site $s \in V$ as the root vertex;
2:  find the minimum spanning tree $T$ in $G$ from the root $s$;
3:  let $L$ be the list of verities visited in a pre-order tree of $T$;
4:  return the Hamiltonian path $P$ that visits the vertices in the order $L$;
```

Figure 7.5: The approximate traveling-salesman algorithm APPROX-TSP-TOUR.

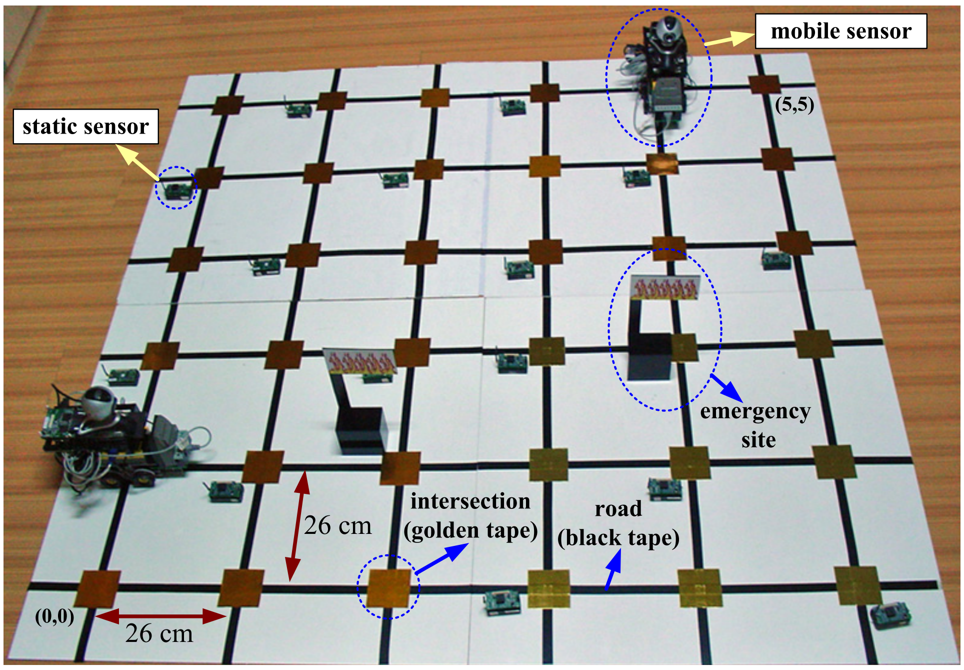
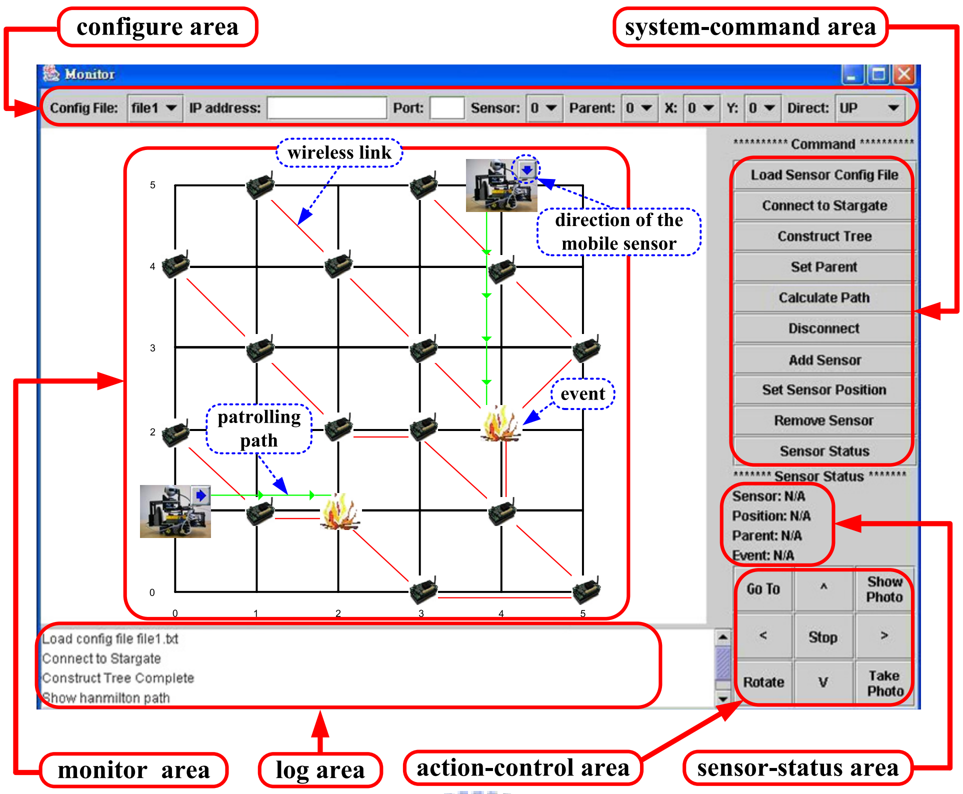


Figure 7.6: A 6 × 6 grid-like sensing field in prototyping system.

emergency.

At the control server, we design a user interface to help users monitor the statuses of the system and control the actions of mobile sensors, as shown in Fig. 7.7. The user interface consists of the following components.

- **Configure area:** The configure area is used to set up the system parameters, such as mobile sensors' IP addresses, ports, and sensors' positions.
- **System-command area:** The system-command area provides an interface to let users issue commands to control the system, such as sending a *tree-maintenance* message to the static sensors, adjusting the network's topology, and connecting or disconnecting a specified mobile sensor.
- **Sensor-status area:** The sensor-status area illustrates the current status of the static sensor being queried.
- **Action-control area:** The action-control area is used to control the actions of a specified mobile sensor, such as changing the moving direction of the mobile sensor and commanding it to take snapshots.
- **Monitoring area:** The monitor area illustrates the network topology of the static sensor network and the patrolling paths of mobile sensors. When a static sensor has detected an event, there will be a fire icon shown in the corresponding location.
- **Log area:** The log area provides some status messages of the system.



Config File: file1 IP address: Port: Sensor: 0 Parent: 0 X: 0 Y: 0 Direct: UP

```
Load config file file1.txt
Connect to Stargate
Construct Tree Complete
Show hanmilton path
```

Sensor: N/A
Position: N/A
Parent: N/A
Event: N/A

Figure 7.7: The user interface at the control server.

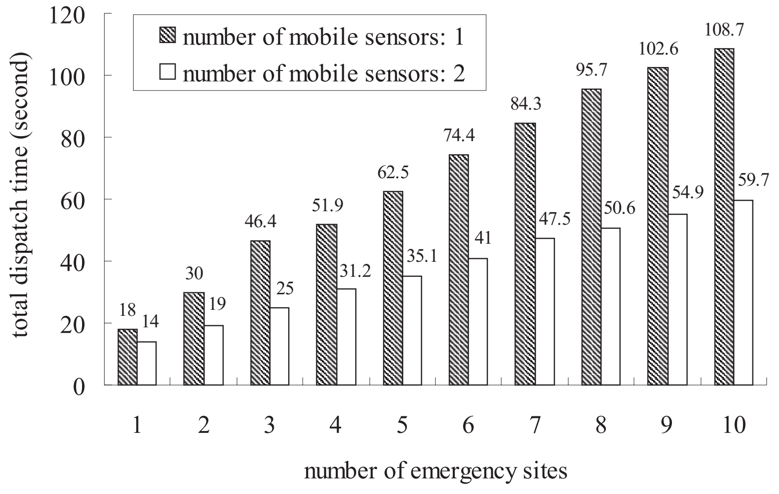
7.4 Experimental Results

In this section, we present some experimental results to evaluate the dispatch time of mobile sensors. In our current implementation, the dispatch time of a mobile sensor to visit an emergency site is determined by the following three times:

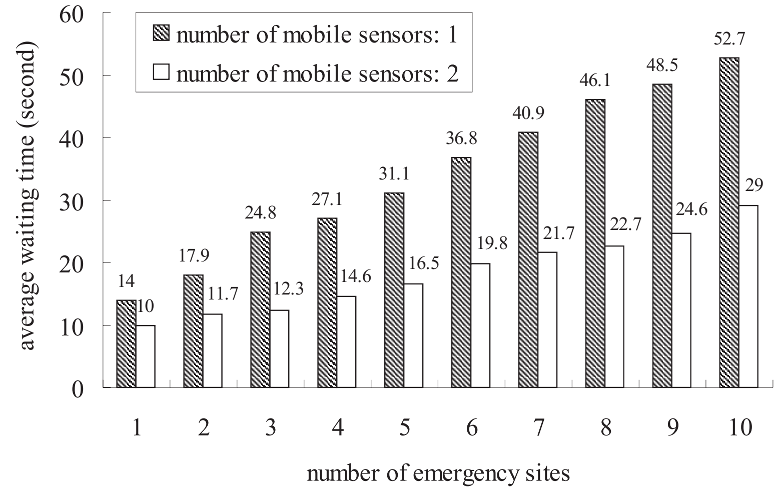
- The time that a mobile sensor needs to move one grid-unit (approximates to 26 cm).
- The time that a mobile sensor needs to turn its direction with an angle of 90 degrees.
- The time that a mobile sensor needs to take snapshots and report to the server.

We measure the averages values of these three times with our system as 2.5, 2.2, and 4.0 seconds, respectively.

Fig. 7.8 shows the experimental results of our prototyping system, where the two mobile sensors are initially placed at locations $(0, 0)$ and $(5, 5)$, respectively. The total dispatch time of a mobile sensor is calculated from the time that the server is notified with events to the time that the mobile sensor visits all event locations and reports its analyzed data, while the average waiting time of an emergency site is calculated from the time that an event is detected by the static sensor to the time that a mobile sensor



(a)



(b)

Figure 7.8: Experimental results of our current prototyping system: (a) total dispatch time of mobile sensors and (b) average waiting time of emergency sites.

moves to the event's location. From Fig. 7.8, we can observe that by adding more mobile sensors, both the dispatch time and waiting time can be greatly reduced.

To evaluate the dispatch time of mobile sensors under more larger scenarios, we setup four grid-size of sensing fields in the simulations: 30×30, 60×60, 90×90, and 120×120. Note that a 30×30 grid-size of sensing field approximately has a 7.5m $\times$ 7.5m area, which approximates to a room's area. In the simulations, event locations and mobile sensors' initial locations are randomly selected inside the sensing field, but any two events will not appear in the same place simultaneously. The number of events and mobile sensors increases proportionally to the size of the sensing field. For each experiment, we repeat 100 rounds and take their averages. Note that in the end of each round, mobile sensors will stay in their last-visiting locations.

Fig. 7.9 shows the total dispatch time of mobile sensors under different grid-size of sensing fields. The dispatch time increases as the number of events increases, but the gaps decrease as we increase the number of mobile sensors. In addition, when the number of events is fixed, the dispatch time decreases as the number of mobile sensors increases. However, such trend becomes smoother when the number of mobile sensors becomes larger. This phenomenon can help us to select the suitable number of mobile sensors under different grid-size of sensing fields.

7.5 Summary

In this chapter, we have proposed the iMouse system that combines both the wireless mobile sensor networks and surveillance technologies to support intelligent mobile surveillance services. On one hand, the mobile sensors can improve the weakness of traditional static sensor networks that they can only provide rough description of the environment by including mobile cameras to conduct more advanced analyses of events. On the other hand, the wireless sensor network introduces the context-aware capability to the surveillance system. Thus, the overheads of traditional visual surveillance systems can be greatly reduced since the real important video sections or images can be immediately retrieved and proactively sent to the users.

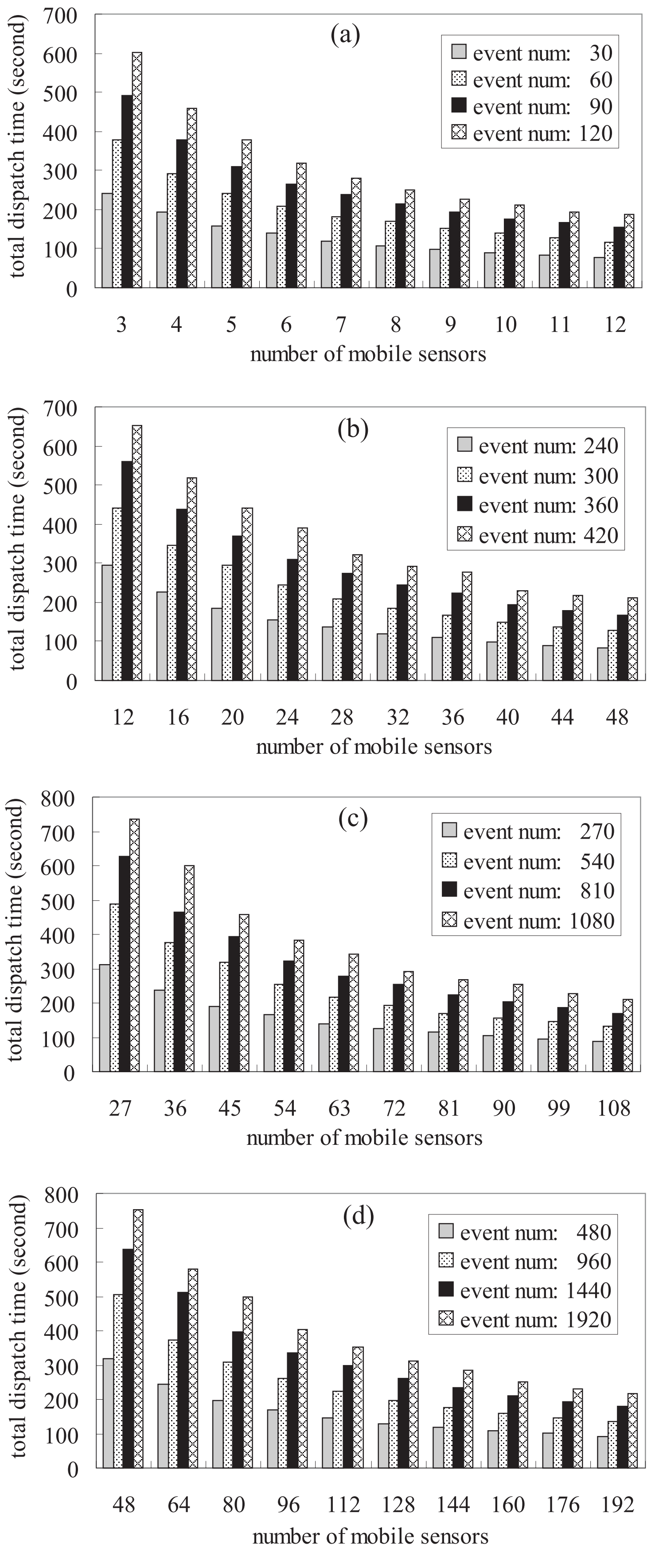


Figure 7.9: Total dispatch time of mobile sensors under different sizes of sensing fields: (a) 30×30 grid-size, (b) 60×60 grid-size, (c) 90×90 grid-size, and (d) 120×120 grid-size.

Chapter 8

Conclusions and Future Directions

8.1 Conclusions

In this dissertation, we have studied the three issues of deployment, dispatch, and packet-scheduling in a mobile wireless sensor network. For the deployment issue, we have first proposed systematical solutions for general sensor placement and dispatch problems. Our solution to the sensor placement problem allows the deployed region as an arbitrary-shaped polygon possibly with obstacles, so the results can be adopted to an indoor environment. Our solution also allows an arbitrary relationship of sensors' communication distances and their sensing distances. We have verified that the proposed placement scheme requires fewer sensors to guarantee full coverage of the deployed region and connectivity of the network as compared with the coverage-first and connectivity-first schemes in different shapes of deployed regions. Moreover, we have proposed two energy-efficient dispatch algorithms to help deploy sensors in a way that sensors can move to the target locations determined by our placement scheme.

We have further developed systematical solutions to the k-coverage sensor placement and distributed dispatch problems. Our solutions to the k-coverage placement problem allow the relationship of sensors' communication distances and their sensing distances as arbitrary. We have shown that the interpolating placement scheme can use fewer sensors to ensure k-coverage of the deployed region and connectivity of the network as compared with the intuitive duplicate scheme. Our solutions to the distributed dispatch problem are based on a competitive nature in a distributed network. Simulation results have shown that the competition-based dispatch scheme performs better than the pattern-based dispatch scheme. However, the performance of pattern-based scheme can approximate to that of the competition-based scheme if there are more than 40% target locations selected as the seed locations.

For the dispatch issue, we have designed an efficient dispatch method to schedule mobile sensors to visit event locations in a hybrid sensor network with the purpose of maximizing the system lifetime. Our dispatch method can balance the moving distances of mobile sensors while conserve their energies as much as possible, and thus avoiding the early-exhausted mobile sensors burdening other still alive ones. Our dispatch method is general in which the numbers of event locations and mobile sensors can be arbitrary. When the number of event locations is not larger than that of mobile sensors, we transform the dispatch problem into a maximum matching problem in a weighted bipartite graph. Otherwise, we propose efficient clustering schemes to group event locations so that the previous matching approach can be adopted. Simulation results have shown that by considering the load-balance of mobile sensors, our proposed dispatch method can prolong the system lifetime compared with the dispatch method that simply maximizes the total remaining energy of mobile sensors during each one-round dispatch.

For the packet-scheduling issue, we have proposed TD-FQ and MR-FQ algorithms for data aggregators to manage the messages reported from sensors. TD-FQ is developed for a single-rate wireless environment, and it takes traffic types of flows into account when scheduling packets. TD-FQ alleviates the queuing delays of real-time flows by giving them a higher priority, but it still guarantees the fairness among flows so that non-real-time flows will not be starved. MR-FQ is developed for a more complicated multi-rate wireless environment. It can adjust flows' transmission rates based on their channel conditions and lagging degrees. In MR-FQ, when a flow is more serious lagging, it is allowed to use a lower rate to transmit its packets, if its channel condition is not good. With this differentiation, MR-FQ can guarantee the fairness property among flows' transmissions while improves the system performance. In this dissertation, we have derived the fairness properties and delay bounds of TD-FQ and MR-FQ by mathematical analyses. Simulation results have also verified their effectiveness.

Finally, in this dissertation we have implemented a mobile sensor platform called the iMouse system. This system combines the mobile wireless sensor networks with the surveillance technologies to support intelligent mobile surveillance services. On one hand, the mobile sensors can help improve the weakness of traditional static sensor networks that they only provide rough information of the environment by including mobile cameras to conduct more in-depth analyses of events. On the other hand, the wireless sensor network can provide the context-aware capability to the surveillance system. Thus, the overheads of traditional visual surveillance systems can be reduced because the real critical information can be immediately sent to the users. In this dissertation, we have reported the prototyping experiences of our iMouse system. We have also analyzed the dispatch time of mobile sensors by simulations.

8.2 *Future Directions*

Based on the research results presented in this dissertation, several issues worth further investigation are summarized as follows:

- **Deploy sensors in a real world:** In this dissertation, we assume that an obstacle may disconnect two sensors and diminish the coverage of a sensor. However, in a real situation, this depends on the materials, thickness, and possibly location of the obstacle. In addition, the signal strength also affects the connectivity between two sensors and the covered area of a sensor. Thus, it deserves to investigate how to further improve our deployment solution to fit for a real world.

- **Multi-type sensor deployment:** Most sensor deployment schemes are based on the assumption that the deployed sensors are homogenous. Reference [19] provides a way to deploy a sensor network with two different types (i.e., cost and sensing distance) of sensors by a grid-based structure. This provides us a research direction to further investigate how to efficiently deploy a multi-type sensor network, where the sensors can have different properties, such as sensing distances, communication distances, costs, and sensing types.

- **Sensor dispatch with other issues:** In our sensor dispatch algorithm, we focus on how to conserve the mobile sensors' energies so that the system lifetime can be extended. However, several issues can be also taken into consideration when dispatching mobile sensors. For example, each event location can be associated with a delay constraint and mobile sensors have to visit these locations before their deadlines expire. Besides, the dispatch problem in a multi-type sensor network is worth further investigation. In such problem, sensors are assumed to have multiple sensing capabilities and different types of events will appear in the sensing field. When an event with certain type occurs, we can only dispatch those mobile sensors that can detect the corresponding type of event to visit that event location.

- **Hierarchical packet scheduling architecture:** In this dissertation, we have proposed TD-FQ and MR-FQ for data aggregators to manage the messages reported from neighboring sensors. However, the management of packet flows between the data aggregators and the sink deserves further investigation. In this case, we have to consider the fairness and delay bounds of flows along the whole pathes from sensors to the sink, rather than just the one-hop distance from sensors to the data aggregator.

- **Improvement of the iMouse system:** In this dissertation, we have reported how we prototyped the iMouse system. We believe that this prototype can be further improved or extended 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.

In this dissertation, we have address the deployment, dispatch, and packet-scheduling issues in mobile wireless sensor networks. We have not only proposed several schemes to solve these problems, but also implemented a prototyping mobile sensor platform. Hopefully these research results presented in this dissertation can be served as useful foundations for future study in the field of mobile wireless sensor networks and thus motivates other researchers to develop more solutions in this field.

Appendix A

Theoretical Analyses of TD-FQ

In this appendix, we analyze the fairness and delay properties of TD-FQ proposed in Section 6.1. Our proof relies on the following assumptions: (i) $\alpha_R \geq \alpha_N$, (ii) $W_R \geq W_N$, (iii) $W_R^S \geq W_R^M$, (iv) $W_N^S \geq W_N^M$, and (v) $B \geq l_{\max}$, where $l_{\max}$ is the maximum length of a packet.

A.1 Fundamental Lemmas

Lemmas A.1–A.3 show that there are bounds on the differences between virtual times (v_i's), extra virtual times (f_i's), and compensation virtual times (c_i^S's and c_i^M's) of two any active flows.

Lemma A.1. *Let $v_i(t)$ be the virtual time of flow i at time t. For any two active flows i and j such that $t \geq 0$, we have*

$$-\frac{l_{\max}}{w_j} \leq v_i(t) - v_j(t) \leq \frac{l_{\max}}{w_i}. \tag{A.1}$$

Proof. This proof is by induction on t.

Basic step: When $t = 0$, all virtual times are zero, so Eq. (A.1) holds trivially.

Induction step: Suppose that at time t, Eq. (A.1) holds. Let $t + \Delta_t$ be the nearest time when any flow changes its virtual time. We want to prove Eq. (A.1) for time $t + \Delta_t$. Observe that a flow's virtual time may be updated in two cases: (1) it is selected by the scheduler and the service does not become a lost service, and (2) it becomes active.

In case (1), let flow i be selected by the scheduler. Then its virtual time becomes

$$v_i(t + \Delta_t) = v_i(t) + \frac{l_p}{w_i},$$

where l_p is the length of the packet being transmitted (not necessarily flow i's). By TD-FQ, it follows that $v_i(t) \le v_j(t)$, for all $j \in A$. Since v_i is increased, by induction hypothesis, we have

$$-\frac{l_{\max}}{w_j} \le v_i(t+\Delta_t) - v_j(t) = v_i(t+\Delta_t) - v_j(t+\Delta_t).$$

Further, since $v_i(t) \le v_j(t)$, we have

$$v_i(t+\Delta_t) - v_j(t+\Delta_t) = \left(v_i(t) + \frac{l_p}{w_i}\right) - v_j(t) \le \frac{l_p}{w_i} \le \frac{l_{\max}}{w_i}.$$

So Eq. (A.1) holds at $t+\Delta_t$.

In Eq. (A.1), if flow j is selected by the scheduler, then $v_i(t+\Delta_t) - v_j(t+\Delta_t) \le \frac{l_{\max}}{w_i}$ holds trivially. Further,

$$v_i(t+\Delta_t) - v_j(t+\Delta_t) = v_i(t) - \left(v_j(t) + \frac{l_p}{w_j}\right) \ge -\frac{l_p}{w_j} \ge -\frac{l_{\max}}{w_j}.$$

So Eq. (A.1) still holds at $t+\Delta_t$.

In case(2), suppose that flow i becomes active at $t+\Delta_t$. By TD-FQ, $v_i(t+\Delta_t)$ is set to $\max\{v_i(t), \min_{k\in A-\{i\}}\{v_k(t+\Delta_t)\}\}$. If $v_i(t+\Delta_t) = \min_{k\in A-\{i\}}\{v_k(t+\Delta_t)\}$, then Eq. (A.1) holds trivially . Otherwise, $v_i(t+\Delta_t) = v_i(t)$, which means that $v_i(t) \ge \min_{k\in A-\{i\}}\{v_k(t+\Delta_t)\}$. So we have

$$v_i(t+\Delta_t) - v_j(t+\Delta_t) \ge \min_{k\in A-\{i\}}\{v_k(t+\Delta_t)\} - v_j(t+\Delta_t) \ge -\frac{l_{\max}}{w_j}.$$

Since the virtual time is non-decreasing, we have

$$v_i(t+\Delta_t) - v_j(t+\Delta_t) \le v_i(t) - v_j(t) \le \frac{l_{\max}}{w_i}.$$

So Eq. (A.1) holds at $t+\Delta_t$. When flow j (instead of i) becomes active, the proof is similar, so we can conclude the proof. □

Because TD-FQ updates f_i, c_i^S, and c_i^M similarly to that of the v_i, proofs of the next two lemmas are similar to that of Lemma A.1. So we omit the proofs.

Lemma A.2. *Let $f_i(t)$ be the extra virtual time of flow i at time t. For any two active flows i and j such that $t \ge 0$, we have*

$$-\frac{l_{\max}}{w_j} \le f_i(t) - f_j(t) \le \frac{l_{\max}}{w_i}.$$

Lemma A.3. *Let $c_i^S(t)$ and $c_i^M(t)$ be the compensation virtual times of flow i at time t. For any two active flows i and j which have the same traffic type (real-time or non-real-time) such that $t \ge 0$, we have*

$$\begin{cases} -\frac{l_{\max}}{w_j} \le c_i^S(t) - c_j^S(t) \le \frac{l_{\max}}{w_i}, & \textit{if both flows are seriously lagging} \\ -\frac{l_{\max}}{w_j} \le c_i^M(t) - c_j^M(t) \le \frac{l_{\max}}{w_i}, & \textit{if both flows are moderately lagging} \end{cases}.$$

Lemma A.4 shows that there is also a bound on the difference between the normalized services received by a leading flow i (i.e., g_i) and the maximum amount that the flow can receive (i.e., $\alpha_i v_i$).

Lemma A.4. *Let $g_i(t)$ be the value of g_i at time t. For any flow i that is error-free, backlogged, and leading during the time interval $t \in [t_1, t_2)$, we have*

$$(\alpha - 1)\frac{l_{\max}}{w_i} \le \alpha v_i(t) - g_i(t) \le \alpha \frac{l_{\max}}{w_i}, \tag{A.2}$$

where $\alpha = \alpha_R$ if flow i is a real-time flow, and $\alpha = \alpha_N$ otherwise.

Proof. The proof is by induction on time $t \in [t_1, t_2)$.
Basic step: When $t = t_1$, flow i just becomes leading, and the Gradual Degradation Scheme will set $g_i(t) = \alpha v_i(t)$, so the lemma is trivially true.
Induction step: Suppose that at time t, the lemma holds. Observe that v_i and/or g_i change only when flow i is selected. So we consider two cases: (1) flow i is actually served, and (2) another flow $j \ne i$ is served. Let $t + \Delta_t \le t_2$ be the nearest time that v_i and/or g_i are updated. We want to prove that the lemma still holds at $t + \Delta_t$.

Based on TD-FQ, case (1) occurs only when $g_i(t) \le \alpha v_i(t)$, so we have

$$\begin{aligned}
&\alpha v_i(t + \Delta_t) - g_i(t + \Delta_t) \\
&= \alpha \left(v_i(t) + \frac{l_p}{w_i} \right) - \left(g_i(t) + \frac{l_p}{w_i} \right) \\
&= (\alpha - 1)\frac{l_p}{w_i} + \alpha v_i(t) - g_i(t) \ge (\alpha - 1)\frac{l_{\max}}{w_i},
\end{aligned}$$

where l_p represents the length of the packet being transmitted.

Case (2) implies $g_i(t) > \alpha v_i(t)$. Also, v_i is updated but g_i is not. So we have

$$\alpha v_i(t + \Delta_t) - g_i(t + \Delta_t) = \alpha(v_i(t) + \frac{l_p}{w_i}) - g_i(t) < \alpha \frac{l_p}{w_i} \le \alpha \frac{l_{\max}}{w_i}. \qquad \square$$

Lemma A.5. *Let $V_R(t)$, $V_N(t)$, $V_R^S(t)$, $V_R^M(t)$, $V_N^S(t)$, and $V_N^M(t)$ be the value of V_R, V_N, V_R^S, V_R^M, V_N^S, and V_N^M at time t, respectively. For $t \ge 0$, we have*

$$\begin{cases}
-\frac{B}{W_N} \le V_R(t) - V_N(t) \le \frac{B}{W_R} \\
-\frac{B}{W_R^M} \le V_R^S(t) - V_R^M(t) \le \frac{B}{W_R^S} \\
-\frac{B}{W_N^M} \le V_N^S(t) - V_N^M(t) \le \frac{B}{W_N^S}
\end{cases}.$$

Proof. This proof is by induction on time $t \ge 0$.
Basic step: When $t = 0$, $V_R(t) = V_N(t) = 0$, so the lemma is trivially true.

Induction step: Assume that the lemma holds at time t. V_R (resp., V_N) is updated only when L_R or L_N is non-empty. We consider two cases: (1) only one set is non-empty, and (2) two sets are non-empty. Let $t+\Delta_t$ be the nearest time that V_R or V_N is updated. We want to prove the lemma to be true at time $t+\Delta_t$.

In case (1), if L_R is active, then ES/CS will be given to L_R. In TD-FQ, we bound the total difference of ES/CS received by L_R and L_N at any time by $|W_R V_R - W_N V_N| \leq B$. So at time $t+\Delta_t$, $W_R V_R(t+\Delta_t) - W_N V_N(t+\Delta_t) \leq B$. Since $W_R \geq W_N$, we have

$$\begin{aligned} & W_R V_R(t+\Delta_t) - W_R V_N(t+\Delta_t) \leq W_R V_R(t+\Delta_t) - W_N V_N(t+\Delta_t) \leq B \\ \Rightarrow\ & V_R(t+\Delta_t) - V_N(t+\Delta_t) \leq \frac{B}{W_R}. \end{aligned}$$

On the other hand, if L_N is active, we can similarly derive that

$$V_R(t+\Delta_t) - V_N(t+\Delta_t) \geq -\frac{B}{W_N}.$$

So the first inequality in the lemma holds at $t+\Delta_t$.

In case (2), since both sets are non-empty, the scheduler gives ES/CS to L_R if $V_R(t) \leq V_N(t)$. Let l_p represent the length of the packet being transmitted. We have

$$V_R(t+\Delta_t) - V_N(t+\Delta_t) = \left(V_R(t) + \frac{l_p}{W_R}\right) - V_N(t) \leq \frac{l_p}{W_R} \leq \frac{l_{\max}}{W_R} \leq \frac{B}{W_R}.$$

Note that it is trivially true that $-\frac{B}{W_N} \leq V_R(t+\Delta_t) - V_N(t+\Delta_t)$. Similarly, if $V_R(t) > V_N(t)$, the service is given to L_N, so we have

$$V_R(t+\Delta_t) - V_N(t+\Delta_t) = V_R(t) - \left(V_N(t) + \frac{l_p}{W_N}\right) > -\frac{l_p}{W_N} \geq -\frac{l_{\max}}{W_N} \geq -\frac{B}{W_N}.$$

Note that it is trivially true that $V_R(t+\Delta_t) - V_N(t+\Delta_t) \leq \frac{B}{W_R}$. Therefore, the first inequality in this lemma still holds at $t+\Delta_t$. The other two inequalities in this lemma can be proved in a similar way. $\square$

A.2 Fairness Properties

Theorems A.1–A.3 show the fairness property guaranteed by TD-FQ. Theorem A.1 is for flows of the same traffic type, while Theorem A.2 is for flows of different types. Theorem A.3 provides some bounds on differences of services received by L_R, L_N, L_R^S, L_R^M, L_N^S, and L_N^M.

Theorem A.1. *For any two active flows i and j of the same traffic type, the difference between the normalized services received by flows i and j in any time interval $[t_1, t_2)$*

during which both flows are continuously backlogged, error-free, and remain in the same state (leading, seriously lagging, moderately lagging, or satisfied) satisfies the inequality:

$$\left|\frac{\Phi_i(t_1,t_2)}{w_i} - \frac{\Phi_j(t_1,t_2)}{w_j}\right| \le \beta \cdot \left(\frac{l_{\max}}{w_i} + \frac{l_{\max}}{w_j}\right),$$

where $\Phi_i(t_1,t_2)$ represents the services received by flow i during $[t_1,t_2)$, $\beta = 3$ if both flows belong to the same lagging set (L_R^S, L_R^M, L_N^S, or L_N^M) or both flows are satisfied, $\beta = 3+\alpha_R$ if both flows are real-time leading flows, and $\beta = 3+\alpha_N$ if both flows are non-real-time leading flows.

Proof. We consider the four cases: flows i and j are both (1) seriously lagging, (2) moderately lagging, (3) satisfied, and (4) leading and backlogged during the entire time interval $[t_1,t_2)$.

Case (1): In this case, any flow i that is seriously lagging can receive services each time when it is selected (by v_i), or when it receives ES/CS from another flow (by c_i^S). Since v_i and c_i^S are updated *before* a packet is transmitted, the services received by flow i may deviate from what really reflects by its virtual times by one packet, so

$$\begin{aligned} v_i(t_2) - v_i(t_1) + c_i^S(t_2) - c_i^S(t_1) - \frac{l_{\max}}{w_i} &\le \frac{\Phi_i(t_1,t_2)}{w_i} \\ &\le v_i(t_2) - v_i(t_1) + c_i^S(t_2) - c_i^S(t_1) + \frac{l_{\max}}{w_i}. \end{aligned} \tag{A.3}$$

Applying Eq. (A.3) to flows i and j, we have

$$\begin{aligned} &v_i(t_2) - v_i(t_1) + c_i^S(t_2) - c_i^S(t_1) - \frac{l_{\max}}{w_i} - \left(v_j(t_2) - v_j(t_1) + c_j^S(t_2) - c_j^S(t_1) + \frac{l_{\max}}{w_j}\right) \\ &\le \frac{\Phi_i(t_1,t_2)}{w_i} - \frac{\Phi_j(t_1,t_2)}{w_j} \\ &\le v_i(t_2) - v_i(t_1) + c_i^S(t_2) - c_i^S(t_1) + \frac{l_{\max}}{w_i} - \left(v_j(t_2) - v_j(t_1) + c_j^S(t_2) - c_j^S(t_1) - \frac{l_{\max}}{w_j}\right). \end{aligned}$$

By Lammas A.1 and A.3, the leftmost term can be reduced to

$$\begin{aligned} &v_i(t_2) - v_j(t_2) - (v_i(t_1) - v_j(t_1)) + c_i^S(t_2) - c_j^S(t_2) - \left(c_i^S(t_1) - c_j^S(t_1)\right) - \left(\frac{l_{\max}}{w_i} + \frac{l_{\max}}{w_j}\right) \\ &\ge -3\left(\frac{l_{\max}}{w_i} + \frac{l_{\max}}{w_j}\right). \end{aligned}$$

Similarly, the rightmost term would be less than or equal to $3\left(\frac{l_{\max}}{w_i} + \frac{l_{\max}}{w_j}\right)$, which leads to

$$\left|\frac{\Phi_i(t_1,t_2)}{w_i} - \frac{\Phi_j(t_1,t_2)}{w_j}\right| \le 3\left(\frac{l_{\max}}{w_i} + \frac{l_{\max}}{w_j}\right).$$

Case (2): This case is similar to case 1. So we can replace c_i^S and c_j^S by c_i^M and c_j^M, respectively, and obtain an inequality similar to Eq. (A.3). This will lead to a $\beta = 3$ too.

Case (3): In this case, both flows can receive services each time when they are selected (by v_i), or when they receive ES from another flow (by f_i). So we have

$$v_i(t_2) - v_i(t_1) + f_i(t_2) - f_i(t_1) - \frac{l_{\max}}{w_i} \le \frac{\Phi_i(t_1,t_2)}{w_i} \le v_i(t_2) - v_i(t_1) + f_i(t_2) - f_i(t_1) + \frac{l_{\max}}{w_i}.$$

Consequently, similar to case 1, by Lemmas A.1 and A.2, we can obtain

$$\left|\frac{\Phi_i(t_1,t_2)}{w_i} - \frac{\Phi_j(t_1,t_2)}{w_j}\right| \le 3\left(\frac{l_{\max}}{w_i} + \frac{l_{\max}}{w_j}\right).$$

Case (4): An error-free, backlogged, and leading flow i can receive NS (by g_i) and ES from other flows (by f_i). So the total services received by flow i during $[t_1, t_2)$ is bounded as

$$\begin{aligned} g_i(t_2) - g_i(t_1) + f_i(t_2) - f_i(t_1) - \frac{l_{\max}}{w_i} &\le \frac{\Phi_i(t_1,t_2)}{w_i} \\ &\le g_i(t_2) - g_i(t_1) + f_i(t_2) - f_i(t_1) + \frac{l_{\max}}{w_i}. \end{aligned} \tag{A.4}$$

Applying Lemma A.4 twice to flows i and j and subtracting one by the other, we have

$$\begin{aligned} &\alpha\left(v_i(t) - v_j(t)\right) + \alpha\left(\frac{l_{\max}}{w_j} - \frac{l_{\max}}{w_i}\right) - \frac{l_{\max}}{w_j} \\ &\le g_i(t) - g_j(t) \le \alpha\left(v_i(t) - v_j(t)\right) + \alpha\left(\frac{l_{\max}}{w_j} - \frac{l_{\max}}{w_i}\right) + \frac{l_{\max}}{w_i}. \end{aligned}$$

By Lemma A.1, we can rewrite the inequality as

$$-\alpha\frac{l_{\max}}{w_i} - \frac{l_{\max}}{w_j} \le g_i(t) - g_j(t) \le \alpha\frac{l_{\max}}{w_j} + \frac{l_{\max}}{w_i}. \tag{A.5}$$

Applying Eq. (A.5) and Lemma A.2 to Eq. (A.4), we have

$$\left|\frac{\Phi_i(t_1,t_2)}{w_i} - \frac{\Phi_j(t_1,t_2)}{w_j}\right| \le (3+\alpha)\left(\frac{l_{\max}}{w_i} + \frac{l_{\max}}{w_j}\right),$$

where $\alpha = \alpha_R$ if these flows are real-time, and $\alpha = \alpha_N$ if they are non-real-time. $\square$

Theorem A.2. *For any real-time flow i and non-real-time flow j, the difference between the normalized services received by flows i and j in any time interval $[t_1, t_2)$ during which both flows are continuously backlogged, error-free, and remain leading satisfies the inequality:*

$$\left|\frac{\Phi_i(t_1,t_2)}{w_i} - \frac{\Phi_j(t_1,t_2)}{w_j}\right| \le 3\cdot\left(\frac{l_{\max}}{w_i} + \frac{l_{\max}}{w_j}\right) + 2\alpha_N\frac{l_{\max}}{w_j}. \tag{A.6}$$

Proof. Applying Lemma A.4 to flows i and j and taking a subtract leads to

$$\alpha_R v_i(t) - \alpha_R \frac{l_{\max}}{w_i} - \left(\alpha_N v_j(t) - (\alpha_N - 1)\frac{l_{\max}}{w_j}\right)$$
$$\le g_i(t) - g_j(t) \le \alpha_R v_i(t) - (\alpha_R - 1)\frac{l_{\max}}{w_i} - \left(\alpha_N v_j(t) - \alpha_N \frac{l_{\max}}{w_j}\right) = T. \quad \text{(A.7)}$$

By Lemma A.1 and the $\alpha_R \ge \alpha_N$ principle, the left-hand side of Eq. (A.7) becomes

$$\alpha_R v_i(t) - \alpha_N v_j(t) + \alpha_N \frac{l_{\max}}{w_j} - \alpha_R \frac{l_{\max}}{w_i} - \frac{l_{\max}}{w_j}$$
$$\ge \alpha_N (v_i(t) - v_j(t)) + \alpha_N \frac{l_{\max}}{w_j} - \alpha_R \frac{l_{\max}}{w_i} - \frac{l_{\max}}{w_j}$$
$$\ge -\alpha_R \frac{l_{\max}}{w_i} - \frac{l_{\max}}{w_j}.$$

Consider the right-hand side of Eq. (A.7). There are two cases for the term $\alpha_R v_i(t) - \alpha_N v_j(t)$. If $\alpha_R v_i(t) - \alpha_N v_j(t) \ge 0$, we have $v_i(t) \ge \frac{\alpha_N}{\alpha_R} v_j(t)$. By Lemma A.1,

$$T \le \alpha_N \left(v_j(t) - v_i(t)\right) + \alpha_N \frac{l_{\max}}{w_j} - \alpha_R \frac{l_{\max}}{w_i} + \frac{l_{\max}}{w_i} \le 2\alpha_N \frac{l_{\max}}{w_j} - \alpha_R \frac{l_{\max}}{w_i} + \frac{l_{\max}}{w_i}.$$

If $\alpha_R v_i(t) - \alpha_N v_j(t) < 0$, we have

$$T \le \alpha_N \frac{l_{\max}}{w_j} - \alpha_R \frac{l_{\max}}{w_i} + \frac{l_{\max}}{w_i}.$$

These two cases together imply $T \le 2\alpha_N \frac{l_{\max}}{w_j} - \alpha_R \frac{l_{\max}}{w_i} + \frac{l_{\max}}{w_i}$. So we have

$$-\alpha_R \frac{l_{\max}}{w_i} - \frac{l_{\max}}{w_j} \le g_i(t) - g_j(t) \le 2\alpha_N \frac{l_{\max}}{w_j} + (1 - \alpha_R)\frac{l_{\max}}{w_i}.$$

Similar to the proof of Theorem A.1, the service received by any leading flow i during $[t_1, t_2)$ satisfies Eq. (A.4). Subtracting Eq. (A.4) of flow i by Eq. (A.4) of flow j leads to

$$g_i(t_2) - g_i(t_1) + f_i(t_2) - f_i(t_1) - \frac{l_{\max}}{w_i} - \left(g_j(t_2) - g_j(t_1) + f_j(t_2) - f_j(t_1) + \frac{l_{\max}}{w_j}\right)$$
$$\le \frac{\Phi_i(t_1, t_2)}{w_i} - \frac{\Phi_j(t_1, t_2)}{w_j}$$
$$\le g_i(t_2) - g_i(t_1) + f_i(t_2) - f_i(t_1) + \frac{l_{\max}}{w_i} - \left(g_j(t_2) - g_j(t_1) + f_j(t_2) - f_j(t_1) - \frac{l_{\max}}{w_j}\right),$$

The leftmost term can be reduced to

$$g_i(t_2) - g_j(t_2) - (g_i(t_1) - g_j(t_1)) + f_i(t_2) - f_j(t_2) - (f_i(t_1) - f_j(t_1)) - \left(\frac{l_{\max}}{w_i} + \frac{l_{\max}}{w_j}\right)$$
$$\ge -\alpha_R \frac{l_{\max}}{w_i} - \frac{l_{\max}}{w_j} - 2\alpha_N \frac{l_{\max}}{w_j} + (\alpha_R - 1)\frac{l_{\max}}{w_i} - 2\left(\frac{l_{\max}}{w_i} + \frac{l_{\max}}{w_j}\right)$$
$$= -3\left(\frac{l_{\max}}{w_i} + \frac{l_{\max}}{w_j}\right) - 2\alpha_N \frac{l_{\max}}{w_j}.$$

Similarly, the rightmost term would be less than or equal to $3\left(\frac{l_{\max}}{w_i} + \frac{l_{\max}}{w_j}\right) + 2\alpha_N \frac{l_{\max}}{w_j}$. Thus, Eq. (A.6) holds. $\square$

Theorem A.3. *The difference between normalized ES/CS received by any two lagging sets in any time interval $[t_1, t_2)$ during which both sets remain active satisfies the inequalities:*

$$\begin{aligned}
&\text{(1) for } L_R \text{ and } L_N: && \left|\frac{\Phi_R(t_1,t_2)}{W_R} - \frac{\Phi_N(t_1,t_2)}{W_N}\right| \le \frac{B+l_{\max}}{W_R} + \frac{B+l_{\max}}{W_N},\\
&\text{(2) for } L_R^S \text{ and } L_R^M: && \left|\frac{\Phi_R^S(t_1,t_2)}{W_R^S} - \frac{\Phi_R^M(t_1,t_2)}{W_R^M}\right| \le \frac{B+l_{\max}}{W_R^S} + \frac{B+l_{\max}}{W_R^M},\\
&\text{(3) for } L_N^S \text{ and } L_N^M: && \left|\frac{\Phi_N^S(t_1,t_2)}{W_N^S} - \frac{\Phi_N^M(t_1,t_2)}{W_N^M}\right| \le \frac{B+l_{\max}}{W_N^S} + \frac{B+l_{\max}}{W_N^M},
\end{aligned}$$

where $\Phi_R(t_1,t_2)$, $\Phi_N(t_1,t_2)$, $\Phi_R^S(t_1,t_2)$, $\Phi_R^M(t_1,t_2)$, $\Phi_N^S(t_1,t_2)$, and $\Phi_N^M(t_1,t_2)$ represents ES/CS received by L_R, L_N, L_R^S, L_R^M, L_N^S, and L_M^N during $[t_1,t_2)$, respectively.

Proof. Since V_R is updated *before* a packet is transmitted, it follows that the total ES/CS received by L_R during $[t_1, t_2)$ is bounded by

$$V_R(t_2) - V_R(t_1) - \frac{l_{\max}}{W_R} \le \frac{\Phi_R(t_1,t_2)}{W_R} \le V_R(t_2) - V_R(t_1) + \frac{l_{\max}}{W_R}.$$

Similarly, for V_N, we have

$$V_N(t_2) - V_N(t_1) - \frac{l_{\max}}{W_N} \le \frac{\Phi_N(t_1,t_2)}{W_N} \le V_N(t_2) - V_N(t_1) + \frac{l_{\max}}{W_N}.$$

Therefore, we have

$$\begin{aligned}
&V_R(t_2) - V_R(t_1) - \frac{l_{\max}}{W_R} - \left(V_N(t_2) - V_N(t_1) + \frac{l_{\max}}{W_N}\right)\\
&\le \frac{\Phi_R(t_1,t_2)}{W_R} - \frac{\Phi_N(t_1,t_2)}{W_N} \le V_R(t_2) - V_R(t_1) + \frac{l_{\max}}{W_R} - \left(V_N(t_2) - V_N(t_1) - \frac{l_{\max}}{W_N}\right).
\end{aligned}$$

By Lemma A.5, we can rewrite the inequality as

$$\begin{aligned}
&-\left(\frac{B+l_{\max}}{W_R} + \frac{B+l_{\max}}{W_N}\right) \le \frac{\Phi_R(t_1,t_2)}{W_R} - \frac{\Phi_N(t_1,t_2)}{W_N} \le \frac{B+l_{\max}}{W_R} + \frac{B+l_{\max}}{W_N}\\
&\Rightarrow \left|\frac{\Phi_R(t_1,t_2)}{W_R} - \frac{\Phi_N(t_1,t_2)}{W_N}\right| \le \frac{B+l_{\max}}{W_R} + \frac{B+l_{\max}}{W_N}.
\end{aligned}$$

This concludes the first inequality. The other two inequalities in this theorem can be proved similarly. □

A.3 Delay Bounds

When a backlogged flow suffers from errors, it becomes lagging. Theorem A.4 shows that if a lagging flow becomes error-free and has sufficient service demand, it can get back all its lagging services within bounded time.

Theorem A.4. *If an active but lagging flow i becomes error-free at time t and remains backlogged continuously after time t, it is guaranteed that flow i will become non-lagging (i.e., $lag_i \le 0$) within time Δ_t, where*

$$\Delta_t \le \frac{\varpi(\varepsilon + 2l_{\max})}{w_{\min}(1-\alpha_R)\widehat{R}} + (m + 1 + \frac{\varpi}{w_{\min}})\frac{l_{\max}}{\widehat{R}},$$

m is the number of active flows, $\widehat{R}$ is the transmission rate, ϖ is the total weight of all flows, ϖ_R is the total weight of all real-time flows, ϖ_N is the total weight of all non-real-time flows, $w_{\min}$ is the minimum weight of all flows, and

$$\begin{aligned}\varepsilon = {} & \frac{(W_R+W_N)(W_R^S+W_R^M)}{W_R W_R^S}\left(\frac{lag_i(t)}{w_i}\varpi_R + (\frac{\varpi_R}{w_i} + m - 2)l_{\max} + B\right) \\ & + \frac{W_R+W_N}{W_R}\left(\delta\varpi_R + (\frac{2\varpi_R}{w_i} + m - 1)l_{\max} + B\right)\end{aligned}$$

if flow i is real-time, and

$$\begin{aligned}\varepsilon = {} & \frac{(W_R+W_N)(W_N^S+W_N^M)}{W_N W_N^S}\left(\frac{lag_i(t)}{w_i}\varpi_N + (\frac{\varpi_N}{w_i} + m - 2)l_{\max} + B\right) \\ & + \frac{W_R+W_N}{W_N}\left(\delta\varpi_N + (\frac{2\varpi_N}{w_i} + m - 1)l_{\max} + B\right)\end{aligned}$$

if flow i is non-real-time.

Proof. Assume that flow i is a real-time flow. Consider the worst case: flow i has the maximum lag among all flows and $lag_i/w_i \ge \delta$ at time t. Since flow i becomes error-free after time t, lag_i is decreased each time when it receives CS. Now let flow i becomes moderately lagging at time t_M, and further become non-lagging at time t_N, $t < t_M < t_N$, i.e., $i \in L_R^S$ during $[t, t_M)$ and $i \in L_R^M$ during $[t_M, t_N)$. Also, let $\Phi_C(t, t_N)$ be the total CS received by all lagging flows during $[t, t_N)$.

To prove this theorem, observe that Δ_t should be an upper bound of $t_N - t$. The largest value of t_N occurs when all flows in the system are error-free (i.e., no ES) and there is only one leading flow, say k, who provides CS such that flow k is a real-time flow and $w_k = w_{\min}$. Since flow k can still receive a fraction α_R of its NS when it is leading and flow k uses g_k to keep track of the amount of such NS when it is leading, this leads to

$$\Phi_C(t, t_N) \ge w_{\min}(v_k(t_N) - v_k(t)) - w_{\min}(g_k(t_N) - g_k(t)) - l_{\max}. \qquad \text{(A.8)}$$

By Lemma A.1, for any active flow j during $[t, t_N)$, we have

$$v_j(t_N) - v_j(t) \le v_k(t_N) - v_k(t) + \frac{l_{\max}}{w_j} + \frac{l_{\max}}{w_{\min}}.$$

This inequality helps to derive the total amount of services provided by the system during $[t, t_N)$:

$$\begin{aligned}
\widehat{R}(t_N - t) &\leq \left(\sum_{j \in A} w_j (v_j(t_N) - v_j(t)) \right) + l_{\max} \\
&\leq \left(\sum_{j \in A} w_j (v_k(t_N) - v_k(t) + \frac{l_{\max}}{w_j} + \frac{l_{\max}}{w_{\min}}) \right) + l_{\max} \\
&\leq (v_k(t_N) - v_k(t)) \sum_{j \in A} w_j + m \cdot l_{\max} + \frac{l_{\max}}{w_{\min}} \sum_{j \in A} w_j + l_{\max} \\
&\leq (v_k(t_N) - v_k(t))\, \varpi + (m + 1 + \frac{\varpi}{w_{\min}}) l_{\max} \\
\Rightarrow\ v_k(t_N) - v_k(t) &\geq \frac{1}{\varpi} \left(\widehat{R}(t_N - t) - (m + 1 + \frac{\varpi}{w_{\min}}) l_{\max} \right). \qquad \text{(A.9)}
\end{aligned}$$

Applying Lemma A.4 to flow k at times t and t_N and taking a subtract, we obtain

$$g_k(t_N) - g_k(t) \leq \alpha_R v_k(t_N) - \alpha_R v_k(t) + \frac{l_{\max}}{w_{\min}}. \qquad \text{(A.10)}$$

By combining Eqs. (A.9) and (A.10) into Eq. (A.8), we can obtain

$$\begin{aligned}
\Phi_C(t, t_N) &\geq w_{\min} \left(v_k(t_N) - v_k(t) - (g_k(t_N) - g_k(t)) \right) - l_{\max} \\
&\geq w_{\min} \left(v_k(t_N) - v_k(t) - \alpha_R v_k(t_N) + \alpha_R v_k(t) - \frac{l_{\max}}{w_{\min}} \right) - l_{\max} \\
&= w_{\min}(1 - \alpha_R) \left(v_k(t_N) - v_k(t) \right) - 2 l_{\max} \\
&\geq \frac{w_{\min}(1 - \alpha_R)}{\varpi} \left(\widehat{R}(t_N - t) - (m + 1 + \frac{\varpi}{w_{\min}}) l_{\max} \right) - 2 l_{\max} \\
\Rightarrow\ t_N - t &\leq \frac{\varpi (\Phi_C(t, t_N) + 2 l_{\max})}{w_{\min}(1 - \alpha_R) \widehat{R}} + (m + 1 + \frac{\varpi}{w_{\min}}) \frac{l_{\max}}{\widehat{R}}. \qquad \text{(A.11)}
\end{aligned}$$

It remains to derive an upper bound for $\Phi_C(t, t_N)$ in Eq. (A.11). Note that there are $m - 1$ lagging flows who are allowed to share the $\Phi_C(t, t_N)$ services. The worst case happens when (1) exactly one of these $m-1$ flows remains in L_N during $[t, t_N)$, (2) exactly $m-3$ flows remain in L_R^S and 1 flow remains in L_R^M during $[t, t_M)$, and (3) no flow remains in L_R^S and exactly $m - 2$ flows remain in L_R^M during $[t_M, t_N)$. Note that in this case L_R can share at most a fraction $\frac{W_R}{W_R + W_N}$ of $\Phi_C(t, t_N)$ during $[t, t_N)$, and L_R^S can share at most a fraction $\frac{W_R^S}{W_R^S + W_R^M}$ of CS received by L_R during $[t, t_M)$.

Let $\Phi_R(t, t_N)$ and $\Phi_N(t, t_N)$ be CS received by L_R and L_N during $[t, t_N)$, respectively, $\Phi_C(t, t_N) = \Phi_R(t, t_N) + \Phi_N(t, t_N)$. According to the first inequality of Theorem A.3, we have

$$\begin{aligned}
\Phi_N(t, t_N) &\leq W_N \left(\frac{\Phi_R(t, t_N)}{W_R} + \frac{B + l_{\max}}{W_R} + \frac{B + l_{\max}}{W_N} \right) \\
\Rightarrow\ \Phi_C(t, t_N) &\leq \frac{W_R + W_N}{W_R} \left(\Phi_R(t, t_N) + B + l_{\max} \right). \qquad \text{(A.12)}
\end{aligned}$$

Next we derive the $\Phi_R(t, t_N)$ in Eq. (A.12). It can be divided into two terms,

$$\Phi_R(t, t_N) = \Phi_R(t, t_M) + \Phi_R(t_M, t_N). \tag{A.13}$$

Let $\Phi_R^S(t, t_M)$ and $\Phi_R^M(t, t_M)$ be CS received by L_R^S and L_R^M during $[t, t_M)$, respectively. Again, by Theorem A.3, we have

$$\begin{aligned}
\Phi_R(t, t_M) &= \Phi_R^S(t, t_M) + \Phi_R^M(t, t_M) \\
&\leq \Phi_R^S(t, t_M) + W_R^M \left(\frac{\Phi_R^S(t, t_M)}{W_R^S} + \frac{B + l_{\max}}{W_R^S} + \frac{B + l_{\max}}{W_R^M} \right) \\
&= \frac{W_R^M + W_R^S}{W_R^S} \left(\Phi_R^S(t, t_M) + B + l_{\max} \right). \qquad \text{(A.14)}
\end{aligned}$$

We further expand the term $\Phi_R^S(t, t_M)$ in Eq. (A.14) as follows:

$$\begin{aligned}
\Phi_R^S(t, t_M) &\leq \sum_{j \in L_R^S(t, t_M)} w_j (c_j^S(t_M) - c_j^S(t)) \\
&\leq \sum_{j \in L_R^S(t, t_M)} w_j \left(c_i^S(t_M) - c_i^S(t) + \frac{l_{\max}}{w_i} + \frac{l_{\max}}{w_j} \right) \\
&= (c_i^S(t_M) - c_i^S(t)) \sum_{j \in L_R^S(t, t_M)} w_j + \frac{l_{\max}}{w_i} \sum_{j \in L_R^S(t, t_M)} w_j + \sum_{j \in L_R^S(t, t_M)} l_{\max} \\
&< \varpi_R (c_i^S(t_M) - c_i^S(t)) + (\frac{\varpi_R}{w_i} + m - 3) l_{\max}. \qquad \text{(A.15)}
\end{aligned}$$

Note that the fourth term in Eq. (A.15) is obtained by applying Lemma A.3 twice on flow i and any flow $j \in L_R^S$

$$c_j^S(t_M) - c_j^S(t) \leq c_i^S(t_M) - c_i^S(t) + \frac{l_{\max}}{w_i} + \frac{l_{\max}}{w_j}.$$

Since L_R^S is empty during $[t_M, t_N)$, $\Phi_R(t_M, t_N) = \Phi_R^M(t_M, t_N)$. Similarly to the derivation of Eq. (A.15), we have

$$\begin{aligned}
\Phi_R(t_M, t_N) = \Phi_R^M(t_M, t_N) &\leq \sum_{j \in L_R^M(t_M, t_N)} w_j (c_j^M(t_N) - c_j^M(t_M)) \\
&\leq \varpi_R (c_i^M(t_N) - c_i^M(t_M)) + (\frac{\varpi_R}{w_i} + m - 2) l_{\max}. \qquad \text{(A.16)}
\end{aligned}$$

By Eqs. (A.14) and (A.15), we have

$$\Phi_R(t, t_M) < \frac{W_R^M + W_R^S}{W_R^S} \left(\varpi_R (c_i^S(t_M) - c_i^S(t)) + (\frac{\varpi_R}{w_i} + m - 2) l_{\max} + B \right). \tag{A.17}$$

Furthermore, by combining Eqs. (A.16) and (A.17) into Eq. (A.13), we have

$$\begin{aligned}\Phi_R(t,t_N) &\le \frac{W_R^M+W_R^S}{W_R^S}\left(\varpi_R(c_i^S(t_M)-c_i^S(t))+(\frac{\varpi_R}{w_i}+m-2)l_{\max}+B\right)\\ &+\varpi_R(c_i^M(t_N)-c_i^M(t_M))+(\frac{\varpi_R}{w_i}+m-2)l_{\max}\\ &=\varpi_R\left(\frac{W_R^S+W_R^M}{W_R^S}(c_i^S(t_M)-c_i^S(t))+c_i^M(t_N)-c_i^M(t_M)\right)\\ &+\frac{2W_R^S+W_R^M}{W_R^S}\left(\frac{\varpi_R}{w_i}+m-2\right)l_{\max}+\frac{(W_R^S+W_R^M)B}{W_R^S}. \qquad (A.18)\end{aligned}$$

By combining Eqs. (A.12) and (A.18), we have

$$\begin{aligned}\Phi_C(t,t_N) &\le \frac{W_R+W_N}{W_RW_R^S}(\varpi_R((W_R^S+W_R^M)(c_i^S(t_M)-c_i^S(t))+W_R^S(c_i^M(t_N)-c_i^M(t_M)))\\ &+\left((2W_R^S+W_R^M)(\frac{\varpi_R}{w_i}+m-2)+W_R^S\right)l_{\max}+(2W_R^S+W_R^M)B). \qquad (A.19)\end{aligned}$$

Since flow i is still lagging after time t_M, it means that $0 < lag_i(t_M) < lag_i(t)$. So

$$c_i^S(t_M)-c_i^S(t)=\frac{|lag_i(t_M)-lag_i(t)|}{w_i}=\frac{lag_i(t)-lag_i(t_M)}{w_i}<\frac{lag_i(t)}{w_i}. \qquad (A.20)$$

After time t_N, flow i becomes non-lagging, so $-l_{\max} < lag_i(t_N) \le 0$. Besides, $0 < lag_i(t_M) < w_i\delta$ since flow i becomes moderately lagging after time t_M, so we have

$$\begin{aligned}c_i^M(t_N)-c_i^M(t_M) &= \frac{|lag_i(t_N)-lag_i(t_M)|}{w_i}\\ &=\frac{lag_i(t_M)-lag_i(t_N)}{w_i}<\delta+\frac{l_{\max}}{w_i}. \qquad (A.21)\end{aligned}$$

By combining Eqs. (A.20) and (A.21) into Eq. (A.19), we have

$$\begin{aligned}\Phi_C(t,t_N) &< \frac{(W_R+W_N)(W_R^S+W_R^M)}{W_RW_R^S}\left(\frac{lag_i(t)}{w_i}\varpi_R+(\frac{\varpi_R}{w_i}+m-2)l_{\max}+B\right)\\ &+\frac{W_R+W_N}{W_R}\left(\delta\varpi_R+(\frac{2\varpi_R}{w_i}+m-1)l_{\max}+B\right). \qquad (A.22)\end{aligned}$$

By combining Eqs. (A.11) and (A.22), the first part of this theorem is proved. When flow i is a non-real-time flow, the proof is similar and we omit the details. □

Appendix B

Theoretical Analyses of MR-FQ

In this appendix, we analyze the fairness and delay properties of MR-FQ proposed in Section 6.2. Our proof relies on the following assumptions: (i) $\alpha_R \geq \alpha_N$, (ii) $W_R \geq W_N$, (iii) $B \geq l_{\max}$, and (iv) $r_i \in \{\hat{C}_1, \cdots, \hat{C}_n\}$, where $l_{\max}$ is the maximum length of a packet and r_i is the transmission rate used by flow i. A flow is called *allowed-to-send* if the Rate Selection Scheme returns a positive transmission rate to it, and is called a *candidate* if it can use a higher rate compared to other flows such that the scheduler may choose it to receive additional services in the Multi-rate Compensation Scheme. Besides, we let $r_i^{\min}$ be the smallest transmission rate that flow i has ever used during the nearest time interval that flow i is active.

B.1 Fundamental Lemmas

The following three lemmas give bounds on the differences between virtual times (v_i's), compensation virtual times (c_i's), and extra virtual times (f_i's) of any two active flows.

Lemma B.1. *Let $v_i(t)$ be the virtual time of flow i at time t. For any two active flows i and j such that $t \geq 0$, we have*

$$-\frac{l_{\max}}{w_j} \times \frac{\hat{C}_1}{r_j^{\min}} \leq v_i(t) - v_j(t) \leq \frac{l_{\max}}{w_i} \times \frac{\hat{C}_1}{r_i^{\min}}. \tag{B.1}$$

Proof. This proof is by induction on t.

Basic step: When $t = 0$, all virtual times are 0, so Eq. (B.1) holds trivially.

Induction step: Suppose that at time t, Eq. (B.1) holds. Let $t + \Delta_t$ be the nearest time when any flow changes its virtual time. We want to prove Eq. (B.1) for time $t + \Delta_t$. Observe that a flow's virtual time may be updated in three cases: (1) it is selected by the scheduler and the service is indeed given to it, (2) it is selected by the scheduler but the service is given to another flow, and (3) it becomes active.

In case (1), let flow i be selected by the scheduler and it use transmission rate r_i ($\geq r_i^{\min}$) to send. Then its virtual time becomes

$$v_i(t+\Delta_t) = v_i(t) + \left(\frac{l_p}{w_i} \times \frac{\hat{C}_1}{r_i}\right),$$

where l_p is the length of the packet being transmitted. By MR-FQ, it follows that $v_i(t) \leq v_j(t)$, for all $j \in A$. Since v_i is increased, by the induction hypothesis, we have

$$-\frac{l_{\max}}{w_j} \times \frac{\hat{C}_1}{r_j^{\min}} \leq v_i(t+\Delta_t) - v_j(t) = v_i(t+\Delta_t) - v_j(t+\Delta_t).$$

Further, since $v_i(t) \leq v_j(t)$, we have

$$v_i(t+\Delta_t) - v_j(t+\Delta_t) = \left(v_i(t) + \frac{l_p}{w_i} \times \frac{\hat{C}_1}{r_i}\right) - v_j(t) \leq \frac{l_{\max}}{w_i} \times \frac{\hat{C}_1}{r_i^{\min}}.$$

So Eq. (B.1) holds at $t + \Delta_t$.

In Eq. (B.1), if flow j is selected by the scheduler and it uses transmission rate r_j ($\geq r_j^{\min}$) to send, then $v_i(t+\Delta_t) - v_j(t+\Delta_t) \leq \frac{l_{\max}}{w_i} \times \frac{\hat{C}_1}{r_i^{\min}}$ holds trivially. Further,

$$v_i(t+\Delta_t) - v_j(t+\Delta_t) = v_i(t) - \left(v_j(t) + \frac{l_p}{w_j} \times \frac{\hat{C}_1}{r_j}\right) \geq -\frac{l_{\max}}{w_j} \times \frac{\hat{C}_1}{r_j^{\min}}.$$

So Eq. (B.1) still holds at $t + \Delta_t$.

Case (2) is similar to case (1), except that we need to replace r_i and r_j by $\hat{C}_1$ in all inequalities.

In case (3), suppose that flow i becomes active at $t + \Delta_t$. By MR-FQ, $v_i(t+\Delta_t)$ is set to $\max\{v_i(t), \min_{k \in A-\{i\}}\{v_k(t+\Delta_t)\}\}$. If $v_i(t+\Delta_t) = \min_{k \in A-\{i\}}\{v_k(t+\Delta_t)\}$, then Eq. (B.1) holds trivially . Otherwise, $v_i(t+\Delta_t) = v_i(t)$, which means that $v_i(t) \geq \min_{k \in A-\{i\}}\{v_k(t+\Delta_t)\}$. So we have

$$v_i(t+\Delta_t) - v_j(t+\Delta_t) \geq \min_{k \in A-\{i\}}\{v_k(t+\Delta_t)\} - v_j(t+\Delta_t) \geq -\frac{l_{\max}}{w_j} \times \frac{\hat{C}_1}{r_j^{\min}}.$$

Since the virtual time is non-decreasing, we have

$$v_i(t+\Delta_t) - v_j(t+\Delta_t) \leq v_i(t) - v_j(t) \leq \frac{l_{\max}}{w_i} \times \frac{\hat{C}_1}{r_i^{\min}}.$$

So Eq. (B.1) holds at $t + \Delta_t$. When flow j (instead of i) becomes active, the proof is similar, so we can conclude the proof. □

Since MR-FQ updates c_i and f_i similarly to that of the v_i, proofs of the next two lemmas are similar to that of Lemma B.1. So we omit the proofs.

Lemma B.2. *Let $c_i(t)$ be the Multi-rate Compensation virtual time of flow i at time t. For any two flows i and j which are both candidates and have the same traffic type (real-time or non-real-time) such that $t \geq 0$, we have*

$$-\frac{l_{\max}}{w_j} \times \frac{\hat{C}_1}{r_j^{\min}} \leq c_i(t) - c_j(t) \leq \frac{l_{\max}}{w_i} \times \frac{\hat{C}_1}{r_i^{\min}}.$$

Lemma B.3. *Let $f_i(t)$ be the extra virtual time of flow i at time t. For any two flows i and j that are both candidates such that $t \geq 0$, we have*

$$-\frac{l_{\max}}{w_j} \times \frac{\hat{C}_1}{r_j^{\min}} \leq f_i(t) - f_j(t) \leq \frac{l_{\max}}{w_i} \times \frac{\hat{C}_1}{r_i^{\min}}.$$

The next lemma gives bounds on the difference between the normalized services received by a leading flow i (i.e., g_i) and the maximum amount that it can receive (i.e., $\alpha_i v_i$).

Lemma B.4. *Let $V_R(t)$ and $V_N(t)$ be the value of V_R and V_N, respectively. For $t \geq 0$, we have*

$$-\frac{B}{W_N} \leq V_R(t) - V_N(t) \leq \frac{B}{W_R}.$$

Proof. This proof is by induction on time $t \geq 0$.
Basic step: When $t = 0$, $V_R(t) = V_N(t) = 0$, so the lemma is trivially true.
Induction step: Assume that the lemma holds at time t. V_R (resp., V_N) can be updated only when L_R^k (resp., L_N^k) is non-empty, where L_R^k (resp., L_N^k) is the subset of L_R (resp., L_N) selected in the Multi-rate Compensation Scheme, respectively. We consider two cases: (1) only one set is non-empty, and (2) two sets are non-empty. Let $t + \Delta_t$ be the nearest time that V_R or V_N is updated. We want to prove the lemma to be true at time $t + \Delta_t$.

In case (1), if L_R^k is non-empty, additional services will be given to L_R. In MR-FQ, we bound the total difference of additional services received by L_R and L_N at any time by $|W_R V_R - W_N V_N| \leq B$. So at time $t + \Delta_t$, $W_R V_R(t + \Delta_t) - W_N V_N(t + \Delta_t) \leq B$. Since $W_R \geq W_N$, we have

$$\begin{aligned} & W_R V_R(t + \Delta_t) - W_R V_N(t + \Delta_t) \leq W_R V_R(t + \Delta_t) - W_N V_N(t + \Delta_t) \leq B \\ \Rightarrow\ & V_R(t + \Delta_t) - V_N(t + \Delta_t) \leq \frac{B}{W_R}. \end{aligned}$$

On the other hand, if L_N^k is non-empty, we can similarly derive that

$$V_R(t+\Delta_t) - V_N(t+\Delta_t) \geq -\frac{B}{W_N}.$$

So the lemma holds at $t+\Delta_t$.

In case (2), since both sets are non-empty, the scheduler gives additional services to L_R if $V_R(t) \leq V_N(t)$. Let l_p represent the length of the packet being transmitted. We have

$$V_R(t+\Delta_t) - V_N(t+\Delta_t) = \left(V_R(t) + \frac{l_p}{W_R}\right) - V_N(t) \leq \frac{l_p}{W_R} \leq \frac{l_{\max}}{W_R} \leq \frac{B}{W_R}.$$

Note that it is trivially true that $-\frac{B}{W_N} \leq V_R(t+\Delta_t) - V_N(t+\Delta_t)$. Similarly, if $V_R(t) > V_N(t)$, the service is given to L_N, so we have

$$V_R(t+\Delta_t) - V_N(t+\Delta_t) = V_R(t) - \left(V_N(t) + \frac{l_p}{W_N}\right) > -\frac{l_p}{W_N} \geq -\frac{l_{\max}}{W_N} \geq -\frac{B}{W_N}.$$

Note that it is trivially true that $V_R(t+\Delta_t) - V_N(t+\Delta_t) \leq \frac{B}{W_R}$. Therefore, the lemma still holds at $t+\Delta_t$. □

B.2 Fairness Properties

Theorems B.1 and B.2 show the service fairness guaranteed by MR-FQ under some constrains. Theorem B.1 is for flows that have the similar conditions and Theorem B.2 provides some bounds on differences of services received by L_R and L_N.

Theorem B.1. *For any two active flows i and j, assume that both flows are continuously backlogged and allowed-to-send, and remain in the same state (leading, lagging, or satisfied) during a time interval $[t_1, t_2)$. Let r_{RSC} and r_{MCS} be the transmission rates used by the these flows in the Rate Selection Scheme and the Multi-rate Compensation Scheme during $[t_1, t_2)$, respectively, where r_{RSC} and r_{MCS} are both in $\{\hat{C}_1, \cdots, \hat{C}_n\}$, and their values do not change during $[t_1, t_2)$. Then the difference between the normalized services received by flows i and j during $[t_1, t_2)$ satisfies the following inequality:*

$$\left|\frac{\Phi_i^s(t_1,t_2)}{w_i} - \frac{\Phi_j^s(t_1,t_2)}{w_j}\right| \leq \beta \cdot \frac{l_{\max}}{w_i} + \gamma \cdot \frac{l_{\max}}{w_j},$$

where $\Phi_i^s(t_1,t_2)$ *represents the services received by flow* i *during* $[t_1,t_2)$*,*

$$\begin{cases} \beta = \frac{r_{RSC}}{r_i^{\min}}+1, \gamma = \frac{r_{RSC}}{r_j^{\min}}+1, & \textit{if both flows are lagging but not candidates} \\ \beta = \frac{r_{RSC}+r_{MCS}}{r_i^{\min}}+1, \gamma = \frac{r_{RSC}+r_{MCS}}{r_j^{\min}}+1, & \textit{if both flows are lagging and candidates} \\ \beta = \frac{r_{RSC}+r_{MCS}}{r_i^{\min}}+1, \gamma = \frac{r_{RSC}+r_{MCS}}{r_j^{\min}}+1, & \textit{if both flows are satisfied} \\ \beta = \frac{r_{MCS}+\alpha_R\hat{C}_1}{r_i^{\min}}+2, \gamma = \frac{r_{MCS}+\alpha_R\hat{C}_1}{r_j^{\min}}+2, & \textit{if both flows are real-time leading flows} \\ \beta = \frac{r_{MCS}+\alpha_N\hat{C}_1}{r_i^{\min}}+2, \gamma = \frac{r_{MCS}+\alpha_N\hat{C}_1}{r_j^{\min}}+2, & \textit{if both flows are non-real-time leading flows} \\ \beta = \frac{r_{MCS}}{r_i^{\min}}+2, \gamma = \frac{r_{MCS}+2\alpha_N\hat{C}_1}{r_j^{\min}}+2, & \textit{if flow } i \textit{ is a real-time leading flow and} \\ & \textit{flow } j \textit{ is a non-real-time leading flow} \end{cases} .$$

Proof. A lagging flow that is allowed-to-send is not necessarily a candidate since there may exist other lagging flows that can use higher rates to transmit. Thus, we have to consider the five cases: (1) flows i and j are both lagging but not candidates, (2) flows i and j are both lagging and candidates, (3) flows i and j are both satisfied, (4) flows i and j are both leading and have the same traffic type, and (5) flows i is a real-time leading flow and j is a non-real-time leading flow during the entire time interval $[t_1,t_2)$.

Case (1): In this case, any flow i that is lagging but not a candidate can only receive services each time when it is selected by v_i. Since v_i is updated *before* a packet is transmitted, the services received by flow i may deviate from what really reflects by its virtual times by one packet. Besides, the services received by flow i is $v_i \times \frac{r_{RSC}}{\hat{C}_1}$. Thus, we have

$$\frac{r_{RSC}}{\hat{C}_1}(v_i(t_2)-v_i(t_1)) - \frac{l_{\max}}{w_i} \le \frac{\Phi_i^s(t_1,t_2)}{w_i} \le \frac{r_{RSC}}{\hat{C}_1}(v_i(t_2)-v_i(t_1)) + \frac{l_{\max}}{w_i}. \qquad \text{(B.2)}$$

Applying Eq. (B.2) to flows i and j, we have

$$\frac{r_{RSC}}{\hat{C}_1}(v_i(t_2)-v_i(t_1)) - \frac{l_{\max}}{w_i} - \left(\frac{r_{RSC}}{\hat{C}_1}(v_j(t_2)-v_j(t_1)) + \frac{l_{\max}}{w_j}\right) \le \frac{\Phi_i^s(t_1,t_2)}{w_i} - \frac{\Phi_j^s(t_1,t_2)}{w_j}$$
$$\le \frac{r_{RSC}}{\hat{C}_1}(v_i(t_2)-v_i(t_1)) + \frac{l_{\max}}{w_i} - \left(\frac{r_{RSC}}{\hat{C}_1}(v_j(t_2)-v_j(t_1)) - \frac{l_{\max}}{w_j}\right).$$

By Lemma B.1, the leftmost term can be reduced to

$$\frac{r_{RSC}}{\hat{C}_1}(v_i(t_2)-v_j(t_2)-(v_i(t_1)-v_j(t_1))) - \left(\frac{l_{\max}}{w_i}+\frac{l_{\max}}{w_j}\right)$$
$$\ge -\left(\frac{r_{RSC}}{r_i^{\min}}+1\right)\frac{l_{\max}}{w_i} - \left(\frac{r_{RSC}}{r_j^{\min}}+1\right)\frac{l_{\max}}{w_j}.$$

Similarly, the rightmost term would be less than or equal to $\left(\frac{r_{RSC}}{r_i^{\min}}+1\right)\frac{l_{\max}}{w_i}+\left(\frac{r_{RSC}}{r_j^{\min}}+1\right)\frac{l_{\max}}{w_j}$, which leads to

$$\left|\frac{\Phi_i^s(t_1,t_2)}{w_i} - \frac{\Phi_j^s(t_1,t_2)}{w_j}\right| \le \left(\frac{r_{RSC}}{r_i^{\min}}+1\right)\frac{l_{\max}}{w_i} + \left(\frac{r_{RSC}}{r_j^{\min}}+1\right)\frac{l_{\max}}{w_j}.$$

Case (2): In this case, both flows can receive services each time when they are selected by v_i/v_j, or receive additional services from another flow by c_i/c_j. Besides, the additional services received by flow i is $c_i \times \frac{r_{MCS}}{\hat{C}_1}$. So we have

$$\frac{r_{RSC}}{\hat{C}_1}(v_i(t_2) - v_i(t_1)) + \frac{r_{MCS}}{\hat{C}_1}(c_i(t_2) - c_i(t_1)) - \frac{l_{\max}}{w_i} \le \frac{\Phi_i^s(t_1,t_2)}{w_i}$$
$$\le \frac{r_{RSC}}{\hat{C}_1}(v_i(t_2) - v_i(t_1)) + \frac{r_{MCS}}{\hat{C}_1}(c_i(t_2) - c_i(t_1)) + \frac{l_{\max}}{w_i}.$$

Similarly to case 1, by Lemmas B.1 and B.2, we can obtain

$$\left|\frac{\Phi_i^s(t_1,t_2)}{w_i} - \frac{\Phi_j^s(t_1,t_2)}{w_j}\right| \le \left(\frac{r_{RSC}+r_{MCS}}{r_i^{\min}} + 1\right)\frac{l_{\max}}{w_i} + \left(\frac{r_{RSC}+r_{MCS}}{r_j^{\min}} + 1\right)\frac{l_{\max}}{w_j}.$$

Case (3): In this case, both flows can receive services each time when they are selected by v_i/v_j, or when they receive additional services from another flow by f_i/f_j. Besides, since the additional services received by flow i is $f_i \times \frac{r_{MCS}}{\hat{C}_1}$, we have

$$\frac{r_{RSC}}{\hat{C}_1}(v_i(t_2) - v_i(t_1)) + \frac{r_{MCS}}{\hat{C}_1}(f_i(t_2) - f_i(t_1)) - \frac{l_{\max}}{w_i} \le \frac{\Phi_i^s(t_1,t_2)}{w_i}$$
$$\le \frac{r_{RSC}}{\hat{C}_1}(v_i(t_2) - v_i(t_1)) + \frac{r_{MCS}}{\hat{C}_1}(f_i(t_2) - f_i(t_1)) + \frac{l_{\max}}{w_i}.$$

Consequently, similar to case 1, by Lemmas B.1 and B.3, we can obtain

$$\left|\frac{\Phi_i^s(t_1,t_2)}{w_i} - \frac{\Phi_j^s(t_1,t_2)}{w_j}\right| \le \left(\frac{r_{RSC}+r_{MCS}}{r_i^{\min}} + 1\right)\frac{l_{\max}}{w_i} + \left(\frac{r_{RSC}+r_{MCS}}{r_j^{\min}} + 1\right)\frac{l_{\max}}{w_j}.$$

Case (4): An allowed-to-send, backlogged, leading flow i can receive services by g_i and additional services from other flows by f_i. So the total services received by flow i during $[t_1, t_2)$ is bounded as

$$g_i(t_2) - g_i(t_1) + \frac{r_{MCS}}{\hat{C}_1}(f_i(t_2) - f_i(t_1)) - \frac{l_{\max}}{w_i} \le \frac{\Phi_i(t_1,t_2)}{w_i}$$
$$\le g_i(t_2) - g_i(t_1) + \frac{r_{MCS}}{\hat{C}_1}(f_i(t_2) - f_i(t_1)) + \frac{l_{\max}}{w_i}.$$

Applying the previous inequality to flows i and j, we have

$$\frac{r_{MCS}}{\hat{C}_1}(f_i(t_2) - f_j(t_2) - f_i(t_1) + f_j(t_1)) + g_i(t_2) - g_j(t_2) - g_i(t_1) + g_j(t_1) - \frac{l_{\max}}{w_i} - \frac{l_{\max}}{w_j}$$
$$\le \frac{\Phi_i^s(t_1,t_2)}{w_i} - \frac{\Phi_j^s(t_1,t_2)}{w_j} \le \frac{r_{MCS}}{\hat{C}_1}(f_i(t_2) - f_j(t_2) - f_i(t_1) + f_j(t_1)) + g_i(t_2) - g_j(t_2) -$$
$$g_i(t_1) + g_j(t_1) + \frac{l_{\max}}{w_i} + \frac{l_{\max}}{w_j}. \tag{B.3}$$

Applying Lemma A.4 twice to flows i and j and subtracting one by the other, we have

$$\alpha\left(v_i(t) - v_j(t)\right) + \alpha\left(\frac{l_{\max}}{w_j} - \frac{l_{\max}}{w_i}\right) - \frac{l_{\max}}{w_j}$$
$$\le g_i(t) - g_j(t) \le \alpha\left(v_i(t) - v_j(t)\right) + \alpha\left(\frac{l_{\max}}{w_j} - \frac{l_{\max}}{w_i}\right) + \frac{l_{\max}}{w_i}.$$

By Lemma B.1, we can rewrite the inequality as

$$-\left(\alpha\frac{\hat{C}_1}{r_j^{\min}}-\alpha+1\right)\frac{l_{\max}}{w_j}-\alpha\frac{l_{\max}}{w_i}\le g_i(t)-g_j(t)$$
$$\le\left(\alpha\frac{\hat{C}_1}{r_i^{\min}}-\alpha+1\right)\frac{l_{\max}}{w_i}+\alpha\frac{l_{\max}}{w_j}. \quad \text{(B.4)}$$

Applying Eq. (B.4) and Lemma B.3 to Eq. (B.3), we have

$$\left|\frac{\Phi_i(t_1,t_2)}{w_i}-\frac{\Phi_j(t_1,t_2)}{w_j}\right|\le\left(\frac{r_{MCS}+\alpha\hat{C}_1}{r_i^{\min}}+2\right)\frac{l_{\max}}{w_i}+\left(\frac{r_{MCS}+\alpha\hat{C}_1}{r_j^{\min}}+2\right)\frac{l_{\max}}{w_j},$$

where $\alpha=\alpha_R$ if these flows are real-time, and $\alpha=\alpha_N$ if they are non-real-time.

Case (5): Applying Lemma A.4 to flows i and j and taking a subtract leads to

$$\alpha_R v_i(t)-\alpha_R\tfrac{l_{\max}}{w_i}-\left(\alpha_N v_j(t)-(\alpha_N-1)\tfrac{l_{\max}}{w_j}\right)\le g_i(t)-g_j(t)\le$$
$$\alpha_R v_i(t)-(\alpha_R-1)\tfrac{l_{\max}}{w_i}-\left(\alpha_N v_j(t)-\alpha_N\tfrac{l_{\max}}{w_j}\right)=S_{right}. \quad \text{(B.5)}$$

By Lemma B.1 and the $\alpha_R>\alpha_N$ principle, the left-hand side of Eq. (B.5) becomes

$$\alpha_R v_i(t)-\alpha_N v_j(t)+\alpha_N\frac{l_{\max}}{w_j}-\alpha_R\frac{l_{\max}}{w_i}-\frac{l_{\max}}{w_j}$$
$$>\alpha_N(v_i(t)-v_j(t))+\alpha_N\frac{l_{\max}}{w_j}-\alpha_R\frac{l_{\max}}{w_i}-\frac{l_{\max}}{w_j}$$
$$\ge-\alpha_R\frac{l_{\max}}{w_i}-\left(\alpha_N\frac{\hat{C}_1}{r_j^{\min}}-\alpha_N+1\right)\frac{l_{\max}}{w_j}.$$

Consider the right-hand side of Eq. (B.5). There are two cases for the term $\alpha_R v_i(t)-\alpha_N v_j(t)$. If $\alpha_R v_i(t)-\alpha_N v_j(t)\ge 0$, we have $v_i(t)\ge\frac{\alpha_N}{\alpha_R}v_j(t)$. By Lemma B.1,

$$S_{right}\le\alpha_N\left(v_j(t)-v_i(t)\right)+\alpha_N\frac{l_{\max}}{w_j}-\alpha_R\frac{l_{\max}}{w_i}+\frac{l_{\max}}{w_i}$$
$$\le\left(\alpha_N\frac{\hat{C}_1}{r_j^{\min}}+\alpha_N\right)\frac{l_{\max}}{w_j}+(1-\alpha_R)\frac{l_{\max}}{w_i}.$$

If $\alpha_R v_i(t)-\alpha_N v_j(t)<0$, we have

$$S_{right}\le\alpha_N\frac{l_{\max}}{w_j}+(1-\alpha_R)\frac{l_{\max}}{w_i}.$$

These two cases together imply $S_{right}\le\left(\alpha_N\frac{\hat{C}_1}{r_j^{\min}}+\alpha_N\right)\frac{l_{\max}}{w_j}+(1-\alpha_R)\frac{l_{\max}}{w_i}$. So we have

$$-\alpha_R\frac{l_{\max}}{w_i}-\left(\alpha_N\frac{\hat{C}_1}{r_j^{\min}}-\alpha_N+1\right)\frac{l_{\max}}{w_j}\le g_i(t)-g_j(t)$$
$$\le\left(\alpha_N\frac{\hat{C}_1}{r_j^{\min}}+\alpha_N\right)\frac{l_{\max}}{w_j}+(1-\alpha_R)\frac{l_{\max}}{w_i}. \quad \text{(B.6)}$$

By applying Eq. (B.6) and Lemma B.3 to Eq. (B.3), we have

$$\left|\frac{\Phi_i^s(t_1,t_2)}{w_i} - \frac{\Phi_j^s(t_1,t_2)}{w_j}\right| \leq \left(\frac{r_{MCS}}{r_i^{\min}} + 2\right)\frac{l_{\max}}{w_i} + \left(\frac{r_{MCS} + 2\alpha_N \hat{C}_1}{r_j^{\min}} + 2\right)\frac{l_{\max}}{w_j}.$$

□

Theorem B.2. *The difference between normalized additional services received by L_R and L_N in any time interval $[t_1, t_2)$ during which both sets remain active (i.e., there exists at least one candidate in each set) satisfies the following inequality:*

$$\left|\frac{\Phi_R(t_1,t_2)}{W_R} - \frac{\Phi_N(t_1,t_2)}{W_N}\right| \leq \frac{B + l_{\max}}{W_R} + \frac{B + l_{\max}}{W_N},$$

where $\Phi_R(t_1,t_2)$ and $\Phi_N(t_1,t_2)$ represents additional services received by L_R and L_N during $[t_1,t_2)$, respectively.

Proof. Since V_R is updated *before* a packet is transmitted, it follows that the total additional services received by L_R during $[t_1,t_2)$ is bounded by

$$V_R(t_2) - V_R(t_1) - \frac{l_{\max}}{W_R} \leq \frac{\Phi_R(t_1,t_2)}{W_R} \leq V_R(t_2) - V_R(t_1) + \frac{l_{\max}}{W_R}.$$

Similarly, for V_N, we have

$$V_N(t_2) - V_N(t_1) - \frac{l_{\max}}{W_N} \leq \frac{\Phi_N(t_1,t_2)}{W_N} \leq V_N(t_2) - V_N(t_1) + \frac{l_{\max}}{W_N}.$$

Therefore, we have

$$\begin{aligned}&V_R(t_2) - V_R(t_1) - \frac{l_{\max}}{W_R} - \left(V_N(t_2) - V_N(t_1) + \frac{l_{\max}}{W_N}\right)\\ &\leq \frac{\Phi_R(t_1,t_2)}{W_R} - \frac{\Phi_N(t_1,t_2)}{W_N} \leq V_R(t_2) - V_R(t_1) + \frac{l_{\max}}{W_R} - \left(V_N(t_2) - V_N(t_1) - \frac{l_{\max}}{W_N}\right).\end{aligned}$$

By Lemma B.4, we can rewrite the inequality as

$$\begin{aligned}&-\left(\frac{B + l_{\max}}{W_R} + \frac{B + l_{\max}}{W_N}\right) \leq \frac{\Phi_R(t_1,t_2)}{W_R} - \frac{\Phi_N(t_1,t_2)}{W_N} \leq \frac{B + l_{\max}}{W_R} + \frac{B + l_{\max}}{W_N}\\ &\Rightarrow \left|\frac{\Phi_R(t_1,t_2)}{W_R} - \frac{\Phi_N(t_1,t_2)}{W_N}\right| \leq \frac{B + l_{\max}}{W_R} + \frac{B + l_{\max}}{W_N}.\end{aligned}$$

□

Theorem B.3 shows the time fairness guaranteed by MR-FQ. Since v_i, c_i, and f_i reflect the transmission time using by flow i, the proof of Theorem B.3 is similar to that of Theorem B.1, except that we do not multiply v_i, c_i, and f_i by $\frac{r_{RSC}}{\hat{C}_1}$ or $\frac{r_{MCS}}{\hat{C}_1}$ factors. Thus, we omit the proof of Theorem B.3.

Theorem B.3. *For any two active flows i and j, the difference between the normalized transmission time used by flows i and j in any time interval $[t_1, t_2)$ during which both flows are continuously backlogged and allowed-to-send, and remain in the same state (leading, lagging, or satisfied) satisfies the following inequality:*

$$\left| \frac{\Phi_i^t(t_1,t_2)}{w_i} - \frac{\Phi_j^t(t_1,t_2)}{w_j} \right| \le \beta \cdot \frac{l_{\max}}{w_i} + \gamma \cdot \frac{l_{\max}}{w_j},$$

where $\Phi_i^t(t_1,t_2)$ represents the transmission time used by flow i during $[t_1,t_2)$,

$$\begin{cases} \beta = \frac{\hat{C}_1}{r_i^{\min}} + 1, \gamma = \frac{\hat{C}_1}{r_j^{\min}} + 1, & \textit{if both flows are lagging but not candidates} \\ \beta = \frac{2\hat{C}_1}{r_i^{\min}} + 1, \gamma = \frac{2\hat{C}_1}{r_j^{\min}} + 1, & \textit{if both flows are lagging and candidates} \\ \beta = \frac{2\hat{C}_1}{r_i^{\min}} + 1, \gamma = \frac{2\hat{C}_1}{r_j^{\min}} + 1, & \textit{if both flows are satisfied} \\ \beta = \frac{(\alpha_R+1)\hat{C}_1}{r_i^{\min}} + 2, \gamma = \frac{(\alpha_R+1)\hat{C}_1}{r_j^{\min}} + 2, & \textit{if both flows are real-time leading flows} \\ \beta = \frac{(\alpha_N+1)\hat{C}_1}{r_i^{\min}} + 2, \gamma = \frac{(\alpha_N+1)\hat{C}_1}{r_j^{\min}} + 2, & \textit{if both flows are non-real-time leading flows} \\ \beta = \frac{\hat{C}_1}{r_i^{\min}} + 2, \gamma = \frac{(2\alpha_N+1)\hat{C}_1}{r_j^{\min}} + 2, & \textit{if flow } i \textit{ is a real-time leading flow and} \\ & \textit{flow } j \textit{ is a non-real-time leading flow} \end{cases} .$$

B.3 Delay Bounds

Theorem B.4 shows that if a lagging flow which has sufficient service demand becomes allowed-to-send and is always a candidate in the Multi-rate Compensation Scheme, it can get back all its lagging services within bounded time.

Theorem B.4. *If an active but lagging flow i which remains backlogged continuously becomes allowed-to-send and is always a candidate in the Multi-rate Compensation Scheme, it is guaranteed that flow i will become non-lagging (i.e., $lag_i \le 0$) within time Δ_t, where*

$$\Delta_t < \frac{\varpi(\varepsilon + 2l_{\max})}{w_{\min}(1-\alpha_R)\hat{C}_n} + \left(\frac{\hat{C}_1}{\hat{C}_n}(m + \frac{\varpi}{w_{\min}}) + 1 \right) \frac{l_{\max}}{\hat{C}_n},$$

m is the number of active flows, ϖ is the total weight of all flows, ϖ_R is the total weight of all real-time flows, ϖ_N is the total weight of all non-real-time flows, $w_{\min}$ is the minimum weight of all flows, and

$$\varepsilon = \frac{W_R + W_N}{W_R} \left(\frac{\hat{C}_1}{\hat{C}_n} \left(\frac{\varpi_R \cdot lag_i(t)}{w_i} + \left(\frac{2\varpi_R}{w_i} + m - 2 \right) l_{\max} \right) + 2l_{\max} + B \right),$$

if flow i is real-time, and

$$\varepsilon = \frac{W_R + W_N}{W_N} \left(\frac{\hat{C}_1}{\hat{C}_n} \left(\frac{\varpi_N \cdot lag_i(t)}{w_i} + \left(\frac{2\varpi_N}{w_i} + m - 2 \right) l_{\max} \right) + 2l_{\max} + B \right),$$

if flow i is non-real-time.

Proof. Assume that flow i is a real-time flow. Consider the worst case: flow i has the maximum lag among all flows. Since flow i becomes allowed-to-send, lag_i is never decreased after time t. Besides, because flow i is always a candidate in the Multi-rate Compensation Scheme, lag_i is decreased each time when it receives additional services. Now let $\Phi_A(t, t_N)$ be the total additional services received by all lagging flows during $[t, t+\Delta_t)$.

To prove this theorem, observe that the largest value of Δ_t occurs when all flows in the system are allowed-to-send and there is only one leading flow, say k, who provides additional services such that flow k is a real-time flow and $w_k = w_{\min}$. Since flow k can still receive a fraction α_R of its services when it is leading and flow k uses g_k to keep track of the amount of such services when it is leading, this leads to

$$\Phi_A(t, t+\Delta_t) \geq w_{\min} \cdot \frac{\hat{C}_1}{\hat{C}_1}(v_k(t+\Delta_t) - v_k(t)) - w_{\min}(g_k(t+\Delta_t) - g_k(t)) - l_{\max}. \quad \text{(B.7)}$$

Not that the best rate of flow k must be $\hat{C}_1$, or it is not allowed-to-send. By Lemma B.1, for any active flow j during $[t, t+\Delta_t)$, we have

$$\begin{aligned} v_j(t+\Delta_t) - v_j(t) &\leq v_k(t+\Delta_t) - v_k(t) + \frac{\hat{C}_1}{r_j^{\min}} \cdot \frac{l_{\max}}{w_j} + \frac{\hat{C}_1}{r_k^{\min}} \cdot \frac{l_{\max}}{w_{\min}} \\ &\leq v_k(t+\Delta_t) - v_k(t) + \frac{\hat{C}_1}{\hat{C}_n}\left(\frac{l_{\max}}{w_j} + \frac{l_{\max}}{w_{\min}}\right). \end{aligned}$$

This inequality helps to derive the total amount of services provided by the system during $[t, t+\Delta_t)$:

$$\begin{aligned} \hat{C}_n \cdot \Delta_t &\leq \left(\sum_{j\in A} w_j \cdot \frac{\hat{C}_1}{\hat{C}_1}(v_j(t+\Delta_t) - v_j(t))\right) + l_{\max} \\ &\leq \left(\sum_{j\in A} w_j(v_k(t+\Delta_t) - v_k(t) + \frac{\hat{C}_1}{\hat{C}_n}(\frac{l_{\max}}{w_j} + \frac{l_{\max}}{w_{\min}}))\right) + l_{\max} \\ &\leq (v_k(t+\Delta_t) - v_k(t)) \sum_{j\in A} w_j + \frac{\hat{C}_1}{\hat{C}_n}\left(m l_{\max} + \frac{l_{\max}}{w_{\min}} \sum_{j\in A} w_j\right) + l_{\max} \\ &\leq (v_k(t+\Delta_t) - v_k(t))\, \varpi + \left(\frac{\hat{C}_1}{\hat{C}_n}(m + \frac{\varpi}{w_{\min}}) + 1\right) l_{\max} \\ \Rightarrow \quad v_k(t+\Delta_t) - v_k(t) &\geq \frac{1}{\varpi}\left(\hat{C}_n \cdot \Delta_t - (\frac{\hat{C}_1}{\hat{C}_n}(m + \frac{\varpi}{w_{\min}}) + 1) l_{\max}\right). \end{aligned} \quad \text{(B.8)}$$

Applying Lemma A.4 to flow k at times t and $t+\Delta_t$ and taking a subtract, we obtain

$$g_k(t+\Delta_t) - g_k(t) \leq \alpha_R v_k(t+\Delta_t) - \alpha_R v_k(t) + \frac{l_{\max}}{w_{\min}}. \quad \text{(B.9)}$$

By combining Eqs. (B.8) and (B.9) into Eq. (B.7), we can obtain

$$\begin{aligned}
\Phi_A(t, t+\Delta_t) &\geq w_{\min}\left(v_k(t+\Delta_t) - v_k(t) - \alpha_R v_k(t+\Delta_t) + \alpha_R v_k(t) - \frac{l_{\max}}{w_{\min}}\right) - l_{\max} \\
&= w_{\min}(1-\alpha_R)\left(v_k(t+\Delta_t) - v_k(t)\right) - 2l_{\max} \\
&\geq \frac{w_{\min}(1-\alpha_R)}{\varpi}\left(\hat{C}_n \cdot \Delta_t - (\frac{\hat{C}_1}{\hat{C}_n}(m + \frac{\varpi}{w_{\min}}) + 1)l_{\max}\right) - 2l_{\max} \\
\Rightarrow \quad \Delta_t &\leq \frac{\varpi(\Phi_A(t,t+\Delta_t) + 2l_{\max})}{w_{\min}(1-\alpha_R)\hat{C}_n} + \left(\frac{\hat{C}_1}{\hat{C}_n}(m + \frac{\varpi}{w_{\min}}) + 1\right)\frac{l_{\max}}{\hat{C}_n}. \qquad \text{(B.10)}
\end{aligned}$$

It remains to derive an upper bound for $\Phi_A(t, t+\Delta_t)$ in Eq. (B.10). The worst case happens when these $n-1$ lagging flows are candidates so that they are all allowed to share the $\Phi_A(t, t+\Delta_t)$ services. Besides, Exactly one of these $n-1$ flows remains in L_N during $[t, t+\Delta_t)$. In this case, L_R can share at most a fraction $\frac{W_R}{W_R+W_N}$ of $\Phi_A(t, t+\Delta_t)$.

Let $\Phi_R(t, t+\Delta_t)$ and $\Phi_N(t, t+\Delta_t)$ be additional services received by L_R and L_N during $[t, t+\Delta_t)$, respectively, $\Phi_A(t, t+\Delta_t) = \Phi_R(t, t+\Delta_t) + \Phi_N(t, t+\Delta_t)$. By Theorem B.2, we have

$$\begin{aligned}
\Phi_N(t, t+\Delta_t) &\leq W_N\left(\frac{\Phi_R(t,t+\Delta_t)}{W_R} + \frac{B + l_{\max}}{W_R} + \frac{B + l_{\max}}{W_N}\right) \\
\Rightarrow \quad \Phi_A(t, t+\Delta_t) &\leq \frac{W_R + W_N}{W_R}\left(\Phi_R(t, t+\Delta_t) + B + l_{\max}\right). \qquad \text{(B.11)}
\end{aligned}$$

By applying Lemma B.2 twice on flow i and any flow $j \in L_R$, we have

$$\begin{aligned}
&\Phi_R(t, t+\Delta_t) \leq \sum_{j \in L_R} w_j \cdot \frac{\hat{C}_1}{\hat{C}_1}(c_j(t+\Delta_t) - c_j(t)) + l_{\max} \\
&\leq \sum_{j \in L_R} w_j \left(c_i(t+\Delta_t) - c_i(t) + \frac{\hat{C}_1}{r_i^{\min}} \cdot \frac{l_{\max}}{w_i} + \frac{\hat{C}_1}{r_j^{\min}} \cdot \frac{l_{\max}}{w_j}\right) + l_{\max} \\
&\leq (c_i(t+\Delta_t) - c_i(t)) \sum_{j \in L_R} w_j + \frac{\hat{C}_1}{\hat{C}_n} \cdot \frac{l_{\max}}{w_i} \sum_{j \in L_R} w_j + \frac{\hat{C}_1}{\hat{C}_n} \sum_{j \in L_R} l_{\max} + l_{\max} \\
&< \varpi_R(c_i(t+\Delta_t) - c_i(t)) + \left(\frac{\hat{C}_1}{\hat{C}_n}(\frac{\varpi_R}{w_i} + m - 2) + 1\right) l_{\max}. \qquad \text{(B.12)}
\end{aligned}$$

After time $t+\Delta_t$, flow i becomes non-lagging, so $-l_{\max} < lag(t+\Delta_t) \leq 0$. Thus, we have

$$\begin{aligned}
&\frac{\hat{C}_n}{\hat{C}_1}\left(c_i(t+\Delta_t) - c_i(t)\right) \leq \frac{|lag_i(t+\Delta_t) - lag_i(t)|}{w_i} < \frac{lag_i(t) + l_{\max}}{w_i} \\
&\Rightarrow c_i(t+\Delta_t) - c_i(t) < \frac{\hat{C}_1}{\hat{C}_n} \cdot \frac{lag_i(t) + l_{\max}}{w_i}. \qquad \text{(B.13)}
\end{aligned}$$

By combining Eqs. (B.12) and (B.13) into Eq. (B.11), we have

$$\Phi_A(t, t+\Delta_t) < \frac{W_R + W_N}{W_R}\left(\frac{\hat{C}_1}{\hat{C}_n}(\frac{\varpi_R \cdot lag_i(t)}{w_i} + \right.$$
$$\left.(\frac{2\varpi_R}{w_i} + m - 2)l_{\max}) + 2l_{\max} + B\right). \qquad \text{(B.14)}$$

By combining Eqs. (B.10) and (B.14), the first part of this theorem is proved. When flow i is a non-real-time flow, the proof is similar and we omit the details. □

Bibliography

[1] D. J. Abraham, R. W. Irving, T. Kavitha, and K. Mehlhorn, "Popular matchings," in *ACM-SIAM Symposium on Discrete Algorithms*, 2005, pp. 424–432.

[2] Acroname, "Garcia robot," http://www.acroname.com/garcia/garcia.html.

[3] 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.

[4] G. W. Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees, "Deploying a wireless sensor network on an active volcano," *IEEE Internet Computing*, vol. 10, no. 2, pp. 18–25, 2006.

[5] 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.

[6] F. Aurenhammer, "Voronoi diagrams – a survey of a fundamental geometric data structure," *ACM Computing Surveys*, vol. 23, no. 3, pp. 345–405, 1991.

[7] D. Avis and G. T. Toussaint, "An optimal algorithm for determining the visibility of a polygon from an edge," *IEEE Transactions on Computers*, vol. 30, no. 12, pp. 910–914, 1981.

[8] K. Balachandran, S. R. Kadaba, and S. Nanda, "Channel quality estimation and rate adaptation for cellular mobile radio," *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 1244–1256, 1999.

[9] P. Basu and J. Redi, "Movement control algorithms for realization of fault-tolerant ad hoc robot networks," *IEEE Network*, vol. 18, no. 4, pp. 36–44, 2004.

[10] V. Bharghavan, S. Lu, and T. Nandagopal, "Fair queuing in wireless networks: issues and approaches," *IEEE Personal Communications*, vol. 6, pp. 44–53, 1999.

[11] M. Bläser, "A new approximation algorithm for the asymmetric TSP with triangle inequality," in *ACM-SIAM Symposium on Discrete Algorithms*, 2003, pp. 638–645.

[12] G. Brasseur, "Robust automotive sensorsy," in *IEEE Instrumentation and Measurement Technology Conference (IMTC)*, vol. 2, 1997, pp. 1278–1283.

[13] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low-cost outdoor localization for very small devices," *IEEE Personal Communications*, vol. 7, no. 5, pp. 28–34, 2000.

[14] Z. Butler and D. Rus, "Controlling mobile sensors for monitoring events with coverage constraints," in *IEEE International Conference on Robotics and Automation*, 2004, pp. 1568–1573.

[15] Z. Butler and D. Rus, "Event-based motion control for mobile-sensor networks," *IEEE Pervasive Computing*, vol. 2, no. 4, pp. 34–42, 2003.

[16] Y. Cao and V. O. K. Li, "Scheduling algorithms in broadband wireless networks," in *Proceedings of the IEEE*, vol. 89, no. 1, 2001, pp. 76–87.

[17] M. Cardei, M. T. Thai, Y. Li, and W. Wu, "Energy-efficient target coverage in wireless sensor networks," in *IEEE INFOCOM*, 2005, pp. 1976–1984.

[18] L. G. Casado, I. García, P. G. Szabó, and T. Csendes, "Packing equal circles in a square II. — new results for up to 100 circles using the TAMSASS-PECS algorithm," *Optimization Theory: Recent Developments from Mátraháza*, pp. 207–224, 2001.

[19] K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho, "Grid coverage for surveillance and target location in distributed sensor networks," *IEEE Transactions on Computers*, vol. 51, no. 12, pp. 1448–1453, 2002.

[20] J. H. Chang and L. Tassiulas, "Maximum lifetime routing in wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 4, pp. 609–619, 2004.

[21] K. Chintalapudi, T. Fu, J. Paek, N. Kothari, S. Rangwala, J. Caffrey, R. Govindan, E. Johnson, and S. Masri, "Monitoring civil structures with a wireless sensor network," *IEEE Internet Computing*, vol. 10, no. 2, pp. 26–34, 2006.

[22] V. Chvátal, "A combinatorial theorem in plane geometry," *Journal of Combinatorial Theory, Series B*, vol. 18, pp. 39–41, 1975.

[23] T. Clouqueur, K. K. Saluja, and P. Ramanathan, "Fault tolerance in collaborative sensor networks for target detection," *IEEE Transactions on Computers*, vol. 53, pp. 320–333, 2004.

[24] Crossbow, "MICA2 Series," http://www.xbow.com/Products/productsdetails.aspx?sid=72.

[25] Crossbow, "MICAz ZigBee Series," http://www.xbow.com/Products/productsdetails.aspx?sid=101.

[26] Crossbow, "Stargate Gateway," http://www.xbow.com/Products/productsdetails.aspx?sid=85.

[27] G. M. Dai, A. H. Du, Q. H. Li, and M. C. Wang, "Planning of moving path based on simplified terrain," in *International Conference on Machine Learning and Cybernetics*, vol. 3, 2003, pp. 1915–1918.

[28] M. Demirbas, A. Arora, and M. Gouda, "A pursuer-evader game for sensor networks," in *Sixth Symposium on Self-Stabilizing Systems*, 2003, pp. 1–16.

[29] S. S. Dhillon and K. Chakrabarty, "Sensor placement for effective coverage and surveillance in distributed sensor networks," in *IEEE Wireless Communications and Networking*, 2003, pp. 1609–1614.

[30] R. Diestel, "Graph theory," *Graduate Texts in Mathematics*, 1997.

[31] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[32] D. Du, F. Hwang, and S. Fortune, "Voronoi diagrams and Delaunay triangulations," *Euclidean Geometry and Computers*, 1992.

[33] X. Du and F. Lin, "Improving sensor network performance by deploying mobile sensors," in *IEEE International Performance, Computing, and Communications Conference (IPCCC)*, 2005, pp. 67–71.

[34] T. Dubejko and K. Stephenson, "Circle packing: experiments in discrete analytic function theory," *Experimental Mathematics*, vol. 4, pp. 307–348, 1995.

[35] S. Dulman, P. Havinga, and J. Hurink, "Wave leader election for wireless sensor networks," in *International Symposium on Mobile Muiltimedia Systems & Applications*, 2002, pp. 43–50.

[36] D. A. Eckhardt and P. Steenkiste, "Effort-limited fair (ELF) scheduling for wireless networks," in *IEEE INFOCOM*, 2000, pp. 1097–1106.

[37] Y. Fang, W. E. Dixon, D. M. Dawson, and P. Chawda, "Homography-based visual servo regulation of mobile robots," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 35, no. 5, pp. 1041–1050, 2005.

[38] 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.

[39] S. Fisk, "A short proof of Chvátal's watchman theorem," *Journal of Combinatorial Theory, Series B*, vol. 24, p. 374, 1978.

[40] F. Fodor, "The densest packing of 19 congruent circles in a circle," *Geometriae Dedicata*, vol. 74, pp. 139–145, 1999.

[41] J. A. George, J. M. George, and B. W. Lamar, "Packing different-sized circles into a rectangular container," *European Journal of Operational Research*, vol. 84, pp. 693–712, 1995.

[42] M. Goldberg, "Packing of 14, 16, 17 and 20 circles in a circle," *Mathematics Magazine*, vol. 44, pp. 134–139, 1971.

[43] S. J. Golestani, "A self-clocked fair queueing scheme for broadband applications," in *INFOCOM*, 1994, pp. 12–16.

[44] P. Goyal, H. M. Vin, and H. Chen, "Start-time fair queueing: a scheduling algorithm for integrated services packet switching networks," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 690–704, 1997.

[45] R. L. Graham and B. D. Lubachevsky, "Dense packings of equal disks in an equilateral triangle: from 22 to 34 and beyond," *The Electronic Journal of Combinatorics*, vol. 2, 1995.

[46] R. L. Graham, B. D. Lubachevsky, K. J. Nurmela, and P. R. J. Östergård, "Dense packings of congruent circles in a circle," *Discrete Mathematics*, vol. 181, no. 1-3, pp. 139–154, 1998.

[47] 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.

[48] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, D. D. Cerra, Ed. Academic Press, 2001.

[49] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh, "Energy-efficient surveillance system using wireless sensor networks," in *ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2004, pp. 270–283.

[50] N. Heo and P. K. Varshney, "Energy-efficient deployment of intelligent mobile sensor networks," *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 35, no. 1, pp. 78–92, 2005.

[51] A. Heppes and J. B. M. Melissen, "Covering a rectangle with equal circles," *Periodica Mathematica Hungarica*, vol. 34, no. 1–2, pp. 65–81, 1996.

[52] F. Hoffmann, M. Kaufmann, and K. Kriegel, "The art gallery theorem for polygons with holes," in *IEEE Symposium on Foundations of Computer Science*, 1991, pp. 39–48.

[53] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, *Global Positioning System: Theory and Practice.* 4th ed., Springer Verlag, 1997.

[54] G. Holland, N. Vaidya, and P. Bahl, "A rate-adaptive MAC protocol for multi-Hop wireless networks," in *ACM Internationl Conference on Mobile Computing and Networking (MobiCom)*, 2001, pp. 236–251.

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

[56] L. Hu and D. Evans, "Localization for mobile sensor networks," in *International Conference on Mobile Computing and Networking*, 2004, pp. 45–57.

[57] S. Hwang and B. P. Kintigh, "Implementation of an intelligent roving robot using multiple sensors," in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 1994, pp. 763–770.

[58] J. O'Rourke, *Art gallery theorems and algorithms.* Oxford University Press, Inc., 1987.

[59] M. R. Jeong, H. Morikawa, and T. Aoyama, "Wireless packet scheduler for fair service allocation," in *Fifth Asia-Pacific Conference and Fourth Optoelectronics and Communications Conference*, 1999, pp. 794–797.

[60] 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.

[61] D. Johnson, T. Stack, R. Fish, D. M. Flickinger, L. Stoller, R. Ricci, and J. Lepreau, “Mobile Emulab: a robotic wireless and sensor network testbed,” in *IEEE INFOCOM*, 2006.

[62] K. Kar and S. Banerjee, “Node placement for connected coverage in sensor networks,” in *Proceedings of the Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, 2003, pp. 50–52.

[63] L. A. Klein, “A boolean algebra approach to multiple sensor voting fusion,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 29, no. 2, pp. 317–327, 1993.

[64] H. W. Kuhn, “Hungarian method for the assignment problem,” *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 1955.

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

[66] LAN/MAN Standards Committee of the IEEE Computer Society, “IEEE Std 802.15.4-2003, Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs),” *IEEE*, 2003.

[67] 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.

[68] 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.

[69] S. Lee, K. Kim, and A. Ahmad, “Channel error and handoff compensation scheme for fair queueing algorithms in wireless networks,” in *IEEE International Conference on Communications (ICC)*, 2002, pp. 3128–3132.

[70] C. W. Lin, D. H. Wang, H. C. Wang, and H. D. Wu, “Prototype development of digital spirometer,” in *International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 4, 1998, pp. 1786–1788.

[71] F. Y. S. Lin and P. L. Chiu, “A near-optimal sensor placement algorithm to achieve complete coverage/discrimination in sensor networks,” *IEEE Communications Letters*, vol. 9, no. 1, pp. 43–45, 2005.

[72] 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.

[73] Y. H. Liu and S. Arimoto, “Finding the shortest path of a disc among polygonal obstacles using a radius-independent graph,” *IEEE Transactions on Robots and Automation*, vol. 11, pp. 682–691, 1995.

[74] L. Lofdahl, G. Stemme, and B. Johansson, “Turbulence measurements using sensors based on silicon technology,” in *IEEE International Congress on Instrumentation in Aerospace Simulation Facilities (ICIASF)*, 1989, pp. 95–103.

[75] S. Lu, V. Bharghavan, and R. Srikant, “Fair scheduling in wireless packet networks,” *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 473–489, 1999.

[76] S. Lu, T. Nandagopal, and V. Bharghavan, “A wireless fair service algorithm for packet cellular networks,” in *ACM International Conference on Mobile Computing and Networking (MobiCom)*, 1998, pp. 10–20.

[77] 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 (WSNA)*, 2002, pp. 88–97.

[78] V. S. Mansouri, B. Afsari, and H. Shahmansouri, “A simple transport protocol for wireless sensor networks,” in *IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ICC)*, 2005, pp. 127–131.

[79] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, “Coverage problems in wireless ad-hoc sensor networks,” in *IEEE INFOCOM*, 2001, pp. 1380–1387.

[80] H. Melissen, “Loosest circle coverings of an equilateral triangle,” *Mathematics Magazine*, vol. 70, no. 2, pp. 118–124, 1997.

[81] J. B. M. Melissen, “Densest packing of eleven congruant circles in a circle,” *Geometriae Dedicata*, vol. 50, pp. 15–25, 1994.

[82] J. B. M. Melissen and P. C. Schuur, “Improved coverings of a square with six and eight equal circles,” *Electronic Journal of Combinatorics*, vol. 3, no. 1, 1996.

[83] J. B. M. Melissen and P. C. Schuur, “Packing 16, 17 or 18 circles in an equilateral triangle,” *Discrete Mathematics*, vol. 145, pp. 333–342, 1995.

[84] MINDSTORM, "Robotics Invention System," http://mindstorms.lego.com.

[85] A. Nasipuri and K. Li, "A directionality based location discovery scheme for wireless sensor networks," in *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002, pp. 105–111.

[86] T. S. E. Ng, I. Stoica, and H. Zhang, "Packet fair queueing algorithms for wireless networks with location-dependent errors," in *IEEE INFOCOM*, 1998, pp. 1103–1111.

[87] D. Nicules and B. Nath, "Ad-hoc positioning system (APS) using AoA," in *IEEE INFOCOM*, 2003, pp. 1734–1743.

[88] D. Niculescu, "Communication paradigms for sensor networks," *IEEE Communications Magazine*, vol. 43, no. 3, pp. 116–122, 2005.

[89] K. J. Nurmela and P. R. J. Östergård, "Covering a square with up to 30 equal circles," Helsinki University of Technology, Laboratory for Theoretical Computer Science, Espoo, Finland, Research Report A62, June 2000.

[90] J. O'Rourke, "Galleries need fewer mobile guards: A variation on Chvátal's theorem," *Geometriae Dedicata*, vol. 14, no. 3, pp. 273–283, 1983.

[91] Özgür B. Akan and I. F. Akyildiz, "Event-to-sink reliable transport in wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 13, no. 5, pp. 1003–1016, 2005.

[92] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 344–357, 1993.

[93] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[94] K. Premaratne, J. Zhang, and M. Dogruel, "Location information-aided task-oriented self-organization of ad-hoc sensor systems," *IEEE Sensors Journal*, vol. 4, no. 1, pp. 85–95, 2004.

[95] P. Ramanathan and P. Agrawal, "Adapting packet fair queuing algorithms to wireless networks," in *ACM International Conference on Mobile Computing and Networking (MobiCom)*, 1998, pp. 1–9.

[96] P. Ramanathan and P. Agrawal, "The havana framework for supporting application and channel dependent QoS in wireless networks," in *IEEE International Conference on Network Protocols (ICNP)*, 1999, pp. 235–244.

[97] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis, "An analysis of several heuristics for the traveling salesman problem," *SIAM Journal on Computing*, vol. 6, no. 3, pp. 563–581, 1977.

[98] I. B. Sachs and D. L. Souvaine, "An efficient algorithm for guard placement in polygons with holes," *Discrete & Computational Geometry*, vol. 13, pp. 77–109, 1995.

[99] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly, "Opportunistic media access for multirate ad hoc networks," in *ACM Internationl Conference on Mobile Computing and Networking (MobiCom)*, 2002, pp. 24–35.

[100] A. Savvides, C. C. Han, and M. B. Strivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2001, pp. 166–179.

[101] C. Sharp, S. Schaffert, A. Woo, N. Sastry, C. Karlof, S. Sastry, and D. Culler, "Design and implementation of a sensor network system for vehicle tracking and autonomous interception," in *European Workshop on Sensor Networks*, 2005, pp. 93–107.

[102] T. C. Shermer, "Recent results in art galleries," *Proceedings of the IEEE*, vol. 80, no. 9, pp. 1384–1399, 1992.

[103] J. P. Sheu, P. W. Cheng, and K. Y. Hsieh, "Design and implementation of a smart mobile robot," in *IEEE International Conference on Wireless And Mobile Computing, Networking And Communications (WiMob)*, 2005, pp. 422–429.

[104] E. Shih, S. H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan, "Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks," in *ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2001, pp. 272–287.

[105] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," *IEEE/ACM Transactions on Networking*, vol. 4, no. 3, pp. 375–385, 1996.

[106] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," in *IEEE International Conference on Communications (ICC)*, 2001, pp. 11–14.

[107] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie, "Protocols for self-organization of a wireless sensor network," *IEEE Personal Communications*, vol. 7, pp. 16–27, 2000.

[108] K. Stephenson, *Introduction to circle packing: the theory of discrete analytic functions.* Cambridge University Press, 2004.

[109] L. Stoica, A. Rabbachin, H. O. Repo, T. S. Tiuraniemi, and I. Oppermann, "An ultrawideband system architecture for tag based wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 54, no. 5, pp. 1632–1645, 2005.

[110] K. Sugihara, M. Sawai, H. Sano, D. S. Kim, and D. Kim, "Disk packing for the estimation of the size of a wire bundle," *Japan Journal of Industrial and Applied Mathmatics*, vol. 21, pp. 259–278, 2004.

[111] T. Sun, L. J. Chen, C. C. Han, and M. Gerla, "Reliable sensor networks for planet exploration," in *IEEE International Conference On Networking, Sensing and Control*, 2005, pp. 816–821.

[112] P. G. Szabó, T. Csendes, L. G. Casado, and I. Garciá, "Packing equal circles in a square I. — problem setting and bounds for optimal solutions," *Optimization Theory: Recent Developments from Mátraháza*, pp. 191–206, 2001.

[113] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin, "Habitat monitoring with sensor networks," *Communications of the ACM*, vol. 47, no. 6, pp. 34–40, 2004.

[114] D. Tian and N. D. Georganas, "A coverage-preserving node scheduling scheme for large wireless sensor networks," in *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002, pp. 32–41.

[115] T. Ue, S. Sampei, N. Morinaga, and K. Hamaguchi, "Symbol rate and modulation level-controlled adaptive modulation/TDMA/TDD system for high-bit-rate wireless data transmission," *IEEE Transactions on Vehicular Technology*, vol. 47, pp. 1134–1147, 1998.

[116] 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.

[117] A. Verma, H. Sawant, and J. Tan, "Selection and navigation of mobile sensor nodes using a sensor network," in *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2005, pp. 41–50.

[118] G. Wang, G. Cao, and T. L. Porta, "Movement-assisted sensor deployment," in *IEEE INFOCOM*, 2004, pp. 2469–2479.

[119] G. Wang, G. Cao, T. L. Porta, and W. Zhang, "Sensor relocation in mobile sensor networks," in *IEEE INFOCOM*, 2005, pp. 2302–2312.

[120] K. C. Wang and Y. L. Chin, "A fair scheduling algorithm with adaptive compensation in wireless networks," in *IEEE Global Telecommunications Conference (GLOBECOM)*, 2001, pp. 3543–3547.

[121] R. Williams, *The geometrical foundation of natural structure: a source book of design.* Dover, New York, 1979.

[122] A. Woo and D. E. Culler, "A transmission control scheme for media access in sensor networks," in *ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2001, pp. 221–235.

[123] J. Wu and S. Yang, "SMART: a scan-based movement-assisted sensor deployment method in wireless sensor networks," in *IEEE INFOCOM*, 2005, pp. 2313–2324.

[124] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 493–506, 2004.

[125] H. H. Yen, F. Y. S. Lin, and S. P. Lin, "Efficient data-centric routing in wireless sensor networks," in *IEEE International Conference on Communications (ICC)*, vol. 5, 2005, pp. 3025–3029.

[126] Y. Yi, Y. Seok, T. Kwon, Y. Choi, and J. Park, "W^2F^2Q: packet fair queuing in wireless packet networks," in *ACM International Workshop on Wireless Mobile Multimedia (WOWMOM)*, 2000, pp. 2–10.

[127] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proceedings of the IEEE*, vol. 83, pp. 1374–1396, 1995.

[128] H. Zhang and J. C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," in *NSF International Workshop on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks*, 2004.

[129] 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.

[130] S. Q. Zheng, J. S. Lim, and S. S. Iyengar, "Finding obstacle-avoiding shortest paths using implicit connection graphs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, pp. 103–110, 1996.

[131] S. Zhou, M. Y. Wu, and W. Shu, "Finding optimal placements for mobile sensors: wireless sensor network topology adjustment," in *IEEE Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication*, 2004, pp. 529–532.

[132] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization based on virtual forces," in *IEEE INFOCOM*, 2003, pp. 1293–1303.

[133] Y. Zou and K. Chakrabarty, "A distributed coverage- and connectivity-centric technique for selecting active nodes in wireless sensor networks," *IEEE Transactions on Computers*, vol. 54, pp. 978–991, 2005.

Curriculum Vitae

You-Chiun Wang was born in Tainan, Taiwan, in 1979. He received his B.S. and M.S. degrees in Computer Science and Information Engineering from the National Chung-Cheng University and the National Chiao-Tung University, Taiwan, in 2001 and 2003, respectively. He is currently a Ph.D. candidate in the Department of Computer Science, National Chiao-Tung University, Taiwan. His research interests include wireless communication and mobile computing, QoS management and wireless fair scheduling, mobile ad hoc network, and wireless sensor networks. He is a member of the Phi Tau Phi Scholastic Honor Society of the Republic of China, in 2001. More information can be found in http://www.csie.nctu.edu.tw/∼wangyc/index.htm
Email address: wangyc@csie.nctu.edu.tw

Publication List

Journal Papers

1. Shiang-Rung Ye, You-Chiun Wang, and Yu-Chee Tseng, "A Jamming-Based MAC Protocol to Improve the Performance of Wireless Multihop Ad Hoc Networks," *Wireless Communications and Mobile Computing*, Vol. 4, No.1, Feb. 2004, pp. 75 – 84. (SCIE, EI)

2. You-Chiun Wang, Shiang-Rung Ye, and Yu-Chee Tseng, "A Fair Scheduling Algorithm with Traffic Classification in Wireless Networks," *Computer Communications*, Vol. 28, No. 10, Jun. 2005, pp. 1225 – 1239. (SCIE, EI)

3. You-Chiun Wang, Yu-Chee Tseng, and Wen-Tsuen Chen, "MR-FQ: A Fair Scheduling Algorithm for Wireless Networks with Variable Transmission Rates," *Simulation: Transactions of The Society for Modeling and Simulation International*, Vol. 81, No. 8, Aug. 2005, pp. 587 – 608. (SCIE, EI)

4. You-Chiun Wang, Kai-Yang Cheng, and Yu-Chee Tseng, "Using Event Detection Latency to Evaluate the Coverage of a Wireless Sensor Network," accepted by *Computer Communications*. (SCIE, EI)

Conference Papers

1. Shiang-Rung Ye, You-Chiun Wang, and Yu-Chee Tseng, "A Jamming-Based MAC Protocol for Wireless Multihop Ad Hoc Networks," *IEEE Vehicular Technology Conference* (*VTC*), 2003-Fall, Orlando, USA, pp. 1396 – 1400.

2. You-Chiun Wang, Shiang-Rung Ye, and Yu-Chee Tseng, "A Fair Scheduling Algorithm with Traffic Classification in Wireless Networks," *SCS International Symposium on Performance Evaluation of Computer and Telecommunication Systems* (*SPECTS*), 2004, San Jose, USA, pp. 502 – 509.

3. You-Chiun Wang, Yu-Chee Tseng, Wen-Tsuen Chen, and Kun-Cheng Tsai, "MR-FQ: A Fair Scheduling Algorithm for Wireless Networks with Variable Transmission

Rates," *IEEE International Conference on Information Technology: Research and Education* (*ITRE*), 2005, Hsinchu, Taiwan, pp. 250 – 254.

4. You-Chiun Wang, Chun-Chi Hu, and Yu-Chee Tseng, "Efficient Deployment Algorithms for Ensuring Coverage and Connectivity of Wireless Sensor Networks," *IEEE Wireless Internet Conference* (*WICON*), 2005, Visegrád, Hungary, pp. 114 – 121.

5. Yu-Chee Tseng, You-Chiun Wang, and Kai-Yang Cheng, "An Integrated Mobile Surveillance and Wireless Sensor (iMouse) System and Its Detection Delay Analysis," *ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems* (*MSWiM*), 2005, Montréal, Canada, pp. 178 – 181.

Book Chapters

1. You-Chiun Wang and Yu-Chee Tseng, "Packet Fair Queuing Algorithms for Wireless Networks," *Design and Analysis of Wireless Networks*, Edited by Yi Pan and Yang Xiao, Nova Science Publishers, 2005, ISBN: 1-59454-186-8.

2. You-Chiun Wang and Yu-Chee Tseng, "Attacks and Defenses of Routing Mechanisms in Ad Hoc and Sensor Networks," *Security in Sensor Networks*, Edited by Yang Xiao, Auerbach Publications, CRC Press, 2006, ISBN: 0-84937-058-2.

Submitted Journal Papers

1. Yu-Chee Tseng, You-Chiun Wang, Kai-Yang Cheng, and Yao-Yu Hsieh, "iMouse: An Integrated Mobile Surveillance and Wireless Sensor System," submitted to *IEEE Computers*, Feb. 2006 (in revision).

Submitted Conference Papers

1. You-Chiun Wang and Yu-Chee Tseng, "Distributed Sensor Deployment of a Wireless Sensor Network for Multi-level Coverage," submitted to *IEEE INFOCOM* 2007.

2. Min-Hsien Chang, You-Chiun Wang, Wen-Chih Peng, and Yu-Chee Tseng, "Energy-Efficient Algorithms for Dispatching Mobile Sensors in a Wireless Sensor Network," submitted to *IEEE INFOCOM* 2007.

3. Yu-Chee Tseng, You-Chiun Wang, and Lun-Wu Yeh, "iPower: An Energy Conservation System for Intelligent Buildings by Wireless Sensor Networks," submitted to *International Computer Symposiums (ICS)* 2006.