Stony Brook University

Self-Adaptive, Scalable and Energy-Efficient Algorithms for Unattended Sensor Networks

A Dissertation Presented

by

Ming Ma

to

The Graduate School
in Partial Fulfillment of the
Requirements
for the Degree of

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

August 2007

Stony Brook University

The Graduate School

Ming Ma

We, the dissertation committee for the above candidate for the
Doctor of Philosophy degree in Electrical Engineering,
hereby recommend acceptance of this dissertation.

Dr. Yuanyuan Yang, Advisor
Professor
Department of Electrical and Computer Engineering

Dr. Armen H. Zemanian, Chair
Distinguished Professor, IEEE Life Fellow
Department of Electrical and Computer Engineering

Dr. Alex Doboli
Associate Professor
Department of Electrical and Computer Engineering

Dr. Jie Gao
Assistant Professor
Department of Computer Science

This dissertation is accepted by the Graduate School
Lawrence Martin
Dean of the Graduate School

Abstract of the Dissertation

Self-Adaptive, Scalable and Energy-Efficient Algorithms for Unattended Sensor Networks

by

Ming Ma

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

2007

This thesis proposes an integrated suite of self-adaptive, scalable and energy-efficient algorithms and protocols for large, unattended sensor networks. The proposed approach includes several closely related topics in mobile sensor networks: (1) self-deployment algorithms which are suitable for different environments and requirements of different tasks; (2) channel reservation MAC protocol for WLANs and sensor networks; (3) reliable position-based routing in mobile ad-hoc/sensor networks; (4) energy-efficient data gathering mechanism.

Wireless sensor networks are playing an increasingly important role in a wide-range of applications, such as medical treatment, outer-space exploration, battlefield surveillance, emergency response, etc. A wireless sensor network is generally composed of hundreds and thousands of distributed sensor nodes, with each node having limited communication, computing and sensing capability. For such large scale and resource-limited networks, energy-efficiency and scalability become two critical issues. Unlike traditional networks,

sensor networks usually work in an unknown or hazardous environment, such as outer-space, seabed and battlefield. Little information about the environment can be obtained before the sensor nodes are deployed. Therefore, each sensor node must be able to "learn and think" itself, and also cooperate with each other to make decisions more efficiently and reliably. Thus energy-efficiency, scalability and self-adaptability are very important capabilities for unattended mobile sensor networks. Due to the special characteristics of sensor nodes and their working environments, classical algorithms and protocols designed for traditional networks may not be suitable for sensor networks.

The thesis presents a suite of energy-efficient, scalable and self-adaptive algorithms and protocols: (1) The adaptive Triangular deployment algorithm (ATRI) provides a self-adaptive deployment solution to mobile sensor networks, which can greatly increases the coverage area and reduces the coverage gap. (2) The channel reservation protocol (CR-MAC) provides a new adaptive collision avoidance mechanism at MAC layer that can decrease collisions and provide higher throughput than traditional approaches with only minimum overhead for exchanging control messages. (3) Single path flooding chain algorithm can improve the end-to-end reliability mobile sensor networks. (4) By introducing hierarchy, clustering algorithm and cluster head positioning algorithm improve the scalability and energy efficiency of large scale sensor networks. (5) Data gathering scheme introduces mobility to the data collector and can greatly prolong the lifetime of static sensor networks.

Contents

List of Tables

List of Figures

Acknowledgements

Firstly, I would like to acknowledge my advisor, Dr. Yuanyuan Yang, for her continual support and advice throughout this work. I could not have imagined having a better advisor and mentor for my Ph.D. study, and without her knowledge, perceptiveness, inspiration, and enthusiasm I would never have finished.

I would like to thank my colleagues Zhenghao Zhang, Hui Zhang, Chi Ma, Deng Pan, Min Yang, Lin Liu, Miao Zhao, Xi Deng, Jin Wang, GuoWen Han and Xiangdong Qin in High Performance Computing and Networking Research Laboratory for their friendship and all of their help over years.

I would also like to express my gratitude to the many individuals in the department who have made my stay at Stony Brook a pleasant and memorable one. Graduate Program Coordinator Deborah Kloppenburg and Judy Eimer have been especially helpful.

Also thanks to my friend, Guodong Zhang for giving me a lot of advice and guidance on my research.

I am also grateful to Dr. Peilin Hong and Dr. Jinsheng Li at University of Science and Technology of China for their invaluable help and especially for initiating the topic on networking.

I am grateful to all my friends from University of Science and Technology of China and Stony Brook University, for being the surrogate family during the many years I stayed there and for their continued moral support there after. Specially thanks to my close friends Fan Li, Xia Hua, Ning Li, Song Yue, Yu Liu, Lei Jiang, Haipeng Li, Heng Yan, Haizhi Liang, Jian Li, Yang Zhao, Lei Wang, Zhe Jin and Liang Sun, who always stand by my side and share my happiness and sadness for so many years.

I am forever indebted to my parents Baoxiang Ma and Kaisheng Li for their unyielding love, endless patience and encouragement. Specially thanks to my wife Ying Wei. She will always be my greatest love, my truest friend, and my closest companion. Also thanks to my parents-in-law Zhenxiang Wei and Huanqun Yin for their love and support. I could not have completed this thesis without them.

Finally, thanks to my committee members, Dr. Armen H. Zemanian, Dr. Alex Doboli, Dr. Sangjin Hong, and Dr. Jie Gao, who offered guidance and support.

Chapter 1

Introduction

This chapter explains the motivation, challenges, design goals, and contributions of the thesis.

1.1 Motivation for This Thesis

In recent years, wireless sensor networks have emerged as a new information-gathering paradigm in a wide-range of applications, such as medical treatment, outer-space exploration, battlefield surveillance, emergency response, etc. [1, 2, 3, 4, 5]. Although most current existing research on sensor networks is still at the prototype level, see, for example, UCB-smart dusts [6], MIT-μAMPS [7], ISI-pc104 [9], UCLA-WINS [8], TmoteSky nodes [10], and Crossbow MICA [11], it is expected that sensor network technologies will be applied widely in various areas in the near future [1, 2, 3, 4, 5]. The sensor nodes are usually thrown into the large scale sensing field without a pre-configured infrastructure. Before monitoring the environment, sensor nodes must be able to discover nearby nodes, organize themselves into a network and subscribe to the network. After the network has

been set up, sensor nodes begin to sense the environment and send data to the outside observer. Although sensor nodes are designed with low power consumption in mind, a sensor node can survive limited lifetime with current technologies [8, 13, 14]. Furthermore, low computing capacity, limited memory and communication bandwidth of sensor nodes prohibit the use of high complexity algorithms and protocols. Moreover, the topology of the sensor network may change abruptly due to unexpected failure of sensor nodes. All these special characteristics of sensor networks bring unique challenges to designing a reliable and efficient sensor network. The network must be designed such that it is **energy-efficient, scalable** and **self-adaptive**.

A sensor network roughly works in three phases: deployment, self-organization, sensing and data gathering. Comparing to first two phases, sensing and data gathering consume dominant percentage energy because the first two phases are needed only when the network is initialized. In the phase of sensing and data gathering, the energy consumption on sensing is relatively stable because it only depends on the sampling rate, and does not depend on network topology or the location of sensors. Therefore the data gathering scheme is the most important factor that determines the network lifetime. Although applications of sensor networks may be diverse, most of them share an essential feature, which is all data packets must be aggregated at the data observer. In a homogeneous network where sensors are organized into a flat topology, sensors close to the observer consume much more energy than sensors at the margin of the network, since they need to relay a lot of packets from sensors far away from the data observer. As a result, after these sensors fail, other nodes cannot reach the observer and the network becomes disconnected, even most of the nodes can still survive for a long period.

1.2 Challenges and Design Goals

In the following we discuss several challenges and design goals related to these properties of sensor networks.

1.2.1 Unattended vs. Human-Controlled

In a human-controlled sensor network, sensor nodes work as passive operators. The motion, location and coverage of all the nodes in the network are planned and scheduled by the human-being or a remote controller. In order to make accurate decisions, a large amount of sensing data and global environment information have to be fed back to the controller. However, the conflict between a lot of energy consumption for uploading data and downloading commands and limited energy of each small sensor node becomes a primary obstacle to apply this approach in large scale sensor networks. In addition, the long communication delay prevents human-control-based protocols from being used in delay-sensitive and remote exploring applications. For instance, in the recent Mars Rover Project [15] developed by NASA, it takes about 12 minutes to download a small picture from the Mars Rover to the earth. Due to the long distance between a sensor and the controller and the low speed of the wireless channel, the commands can hardly been transmitted to the sensor instantly. Therefore, sensor nodes must have self-configurability and self-adaptability in an unknown environment. For example, the Mars Rover can turn itself on/off to prolong the lifetime. Developing an integrated suite of adaptive algorithms and protocols for unattended sensor nodes is a primary research task for the future sensor networks.

1.2.2 Dynamic vs. Static Network Topologies

In some sensor networks, such as a roof monitoring system, all nodes have fixed locations and sufficient energy (usually plugged into electrical outlets). Since no nodes move or run out of power, the network can work well with existing mature protocols for static topologies. However, can this type of network be used in monitoring unknown and dynamically changed environments? The answer is clearly no. First, in an unknown environment, sensors can hardly be placed precisely at the intended locations with little information about the environment available. Second, mobile sensor nodes are more suitable for dynamically changed environments and fulfilling the searching and exploring tasks than static sensor nodes. In addition, limited lifetime and unpredictable damage may also cause the failure of nodes, thereby change the topology of the network. The dynamic topologies have a critical impact on the network layer algorithms and protocols, since the traditional routing protocols for static networks can no longer perform efficiently in dynamic sensor networks.

Most existing routing algorithms for dynamic sensor/ad hoc networks can be divided into two classes: topology-based and position-based. Topology-based algorithms are based on network link information, while position-based algorithms use location information of nodes to achieve packet forwarding. Although some topology-based routing algorithms, such as DSDV [17], OLSR [18], TBRPS [19], DSR [20, 21], TORA [22], AODV [23, 24], try to improve the performance by only maintaining the link information that are currently in use, they still have some inherent limitations. When thousands of moving nodes need to communicate with each other, the overhead storm leads to tremendous power and time consumption. On the other hand, position-based routing algorithms [25, 26, 27, 28, 29, 30, 31] do not need to establish and maintain network links, and the routing decisions are mainly based on the location information of the destination node and the one-hop neighbors of

each forwarding node. In a position-based routing algorithm, each mobile node can obtain its own location information from Global Positioning System (GPS) or some other positioning services [33, 34]. A routing decision at a node is triggered by an incoming packet to the node and is made based on the location information of both the destination node and the one-hop neighbors of each forwarding node. In general, position-based routing algorithms are more preferred for sensor networks due to their efficiency and reliability, and may become dominant routing algorithms in dynamic sensor networks in the near future [25, 26].

1.2.3 Unknown vs. Well-Known Environments

One of the most important functionalities of sensor networks is to sense the human-unreachable area, such as volcano, seabed and outer-space. Unlike in a well-known environment, it is impossible to throw sensor nodes to their expected targets in an unknown working area or provide a map of the working area to sensor nodes before the placement. However, most of previous research work in this area assume all nodes are well-deployed or a global map is pre-stored in the memory of all sensor nodes. In fact, in some situations the environment may be completely unknown to the newly coming sensor nodes. Without the control of the human being, sensor nodes must be "smart" enough to learn the working area by themselves, and then deploy themselves to their expected working targets. There has been some interesting work on self-deployment, such as Potential-Field-Based Deployment algorithm [59], Virtual Force Algorithm(VFA) [62] and Movement-Assisted Sensor Deployment algorithms [61] (VEC, VOR, MiniMax). These algorithms focused on maximizing the coverage area by assuming that the motion of each node can be affected by virtual force from other nodes and obstacles. In these approaches, sensor nodes can

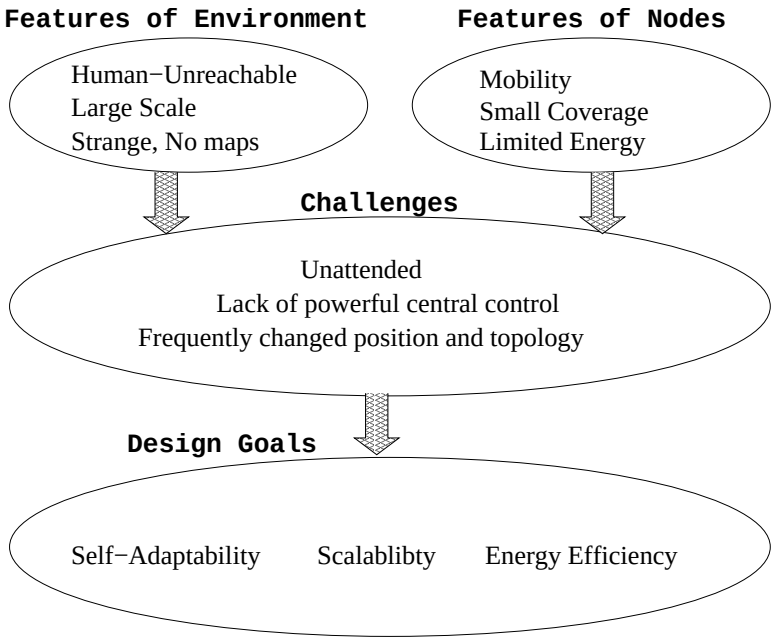


Figure 1.1: Features, challenges and design goals of the future sensor networks.

dynamically adjust their positions according to the change of the network structure. However, sensor nodes were always assumed to be uniformly distributed rather than based on different task requirements or the distribution of interested events. Self-deployment is a fundamental preparation step for the subsequent operations in a sensor network, and in order to prolong the lifetime of the network, this step must be fast and energy-efficient.

Finally, we summary the challenges and design goals of the sensor networks discussed above in Figure 1.1.

1.3 Contributions

The novel contribution of this thesis includes:

- **An adaptive Triangular Deployment Algorithm for Unattended Mobile Sensor**

Networks.[94] The adaptive Triangular deployment algorithm can increase the non-gap coverage of mobile sensors. It also supports adaptive deployment. Without the map and information of the environment, nodes can avoid obstacles and adjust the density dynamically based upon different requirements.

- **A contention-Based MAC protocol with channel reservation Protocol.**[96] The proposed MAC protocol can achieve much better throughput, fairness, packet delay than IEEE 802.11 RTS/CTS protocol. In particular, under saturated traffic, both the throughput and the fairness index of the CR-MAC protocol are very close to the theoretical bound.

- **A reliable single path flooding chain routing algorithm for mobile ad-hoc/sensor Networks, called single-path flooding chain algorithm.**[95] The proposed algorithm can significantly save the bandwidth and power for resource limited mobile nodes, especially in large networks. The single-path flooding chain algorithm consistently performs well for various mobilities and keeps a high successful packet delivery ratio $(> 75\%)$, which is insensitive to the change of node's motion speed.

- **A clustering and Load Balancing mechanism in Hybrid Sensor Networks with Mobile Cluster Heads.**[99, 98, 97, 100] The proposed cluster head positioning algorithm can increase the network lifetime by a significant amount. In addition, the algorithm can recover the network from unexpected failure of sensors and cluster heads.

- **A novel data gathering scheme by introducing mobility and hierarchy into sensor networks.**[101] The proposed data gathering mechanism can prolong the network lifetime about 30 times compared to a network which has only a static observer, and about 4 times compared to a network whose mobile observer can only move along straight lines.

The proposed research combines protocol design, algorithm design, analytical, probabilistic and simulation techniques to conduct comprehensive studies on the above issues. The proposed research will have a significant impact on fundamental design principles and infrastructures for the development of future sensor networks. The outcome of this project will be applicable to a wide spectrum of applications, including space, military, environmental, health care, home and other commercial areas.

1.4 Thesis Outline

The proposed design techniques are presented in a bottom-up fashion. Chapter 3 proposes a new position-based routing algorithm, *Single Path Flooding Chain algorithm* for ad hoc networks. Chapter 4 proposes a novel and more efficient contention-based MAC protocol, called the *Channel Reservation MAC protocol*, by introducing a reservation mechanism. Chapter 5 proposes a heuristic algorithm for clustering sensors, positioning cluster heads and balancing traffic load in the network. Chapter 6 presents data gathering schemes for large scale sensor networks by introducing hierarchy and mobility. Chapter 7 concludes the thesis.

Chapter 2

Adaptive Triangular Self-Deployment Algorithm

This chapter presents a novel sensor deployment algorithm, called *adaptive triangular deployment (ATRI)* algorithm, for large scale unattended mobile sensor networks. ATRI algorithm aims at maximizing coverage area and minimizing coverage gaps and overlaps, by adjusting the deployment layout of nodes close to equilateral triangulations, which is proved to be the optimal layout to provide the maximum no-gap coverage. The algorithm only needs location information of nearby nodes, thereby avoiding communication cost for exchanging global information. By dividing the transmission range into six sectors, each node adjusts the relative distance to its one-hop neighbors in each sector separately. *Distance threshold strategy* and *movement state diagram strategy* are adopted to avoid the oscillation of nodes. The simulation results show that ATRI algorithm achieves a much larger coverage area and less average moving distance of nodes than existing algorithms. We also show that ATRI algorithm is applicable to practical environments and tasks, such as working in both bounded and unbounded areas, and avoiding irregularly-shaped obstacles.

In addition, the density of nodes can be adjusted adaptively to different requirements of tasks.

The rest of the chapter is organized as follows. Section 2.1 summarizes the existing work in this area. Section 2.2 studies the optimum node layout for sensor deployment. Section 2.3 presents the details of the new self-deployment algorithm for mobile sensor networks. Section 2.4 gives the simulation results for the proposed algorithm. Finally, Section 2.5 concludes the chapter.

2.1 Related Work

There has been some previous work on the maximum coverage problem for sensor networks in the literature. *Potential-field-based deployment algorithm* [59] assumes that the movement of each node can be affected by virtual force from other nodes and obstacles. In the algorithm, all nodes explore from a compact region and fill the maximum working area in a way similar to the particles in the micro-world [57]. Although this approach can maximize the coverage area, since the main idea of this algorithm is obtained from the micro-world, the nodes in the network may oscillate for a long time before they reach the static equilibrium state, like the particles in micro world. The oscillation of nodes consumes much more energy than moving to the desired location directly. Moreover, this algorithm can only be used in a bounded area, since nodes must be restricted within the boundary by the virtual force from boundary. Without the boundary, each node will not stop expelling others until there is no other nodes within its transmission range. *The Virtual Force Algorithm (VFA)* [62] divides a sensor network into clusters. Each cluster head is responsible for collecting the location information of the nodes and determining their targets. The cluster architecture may lead to an unbalanced lifetime of the nodes and is

not suitable for the networks that do not have powerful central nodes. *Constrained Coverage algorithm* [60] can guarantee that each node has at least K neighbors by introducing two virtual forces. However, it still does not have any mechanism to limit the oscillation of nodes. *Movement-Assisted Sensor Deployment algorithms* [61], which consist of three independent algorithms *VEC*, *VOR* and *MiniMax*, use Voronoi diagrams to discover the coverage holes and maximize the coverage area by pushing or pulling nodes to cover the coverage gaps based on virtual forces. In the VEC algorithm, the nodes which have covered their corresponding Voronoi cells do not need to move, while other nodes are pushed to fill the coverage gaps. In VOR, nodes will move toward the farthest Voronoi vertices. The MiniMax algorithm moves nodes more gently than VOR, thereby avoiding the generation of new holes in some cases. Compared to potential-field-based deployment algorithm, movement-assisted sensor deployment algorithms reduce the oscillation and save the energy consumed by node movement. All three algorithms assume that each node knows its Voronoi neighbors and vertices. However, Voronoi diagram is a global structure, and all Voronoi vertices and cells can only be obtained when the global location information of the nodes in the network is known [58], which means that each node must exchange its current location information with all other nodes in the network to acquire its corresponding Voronoi vertices and cell. For each location update message, it may take $O(n)$ one-hop transmissions to reach all other nodes, where n is the total number of nodes in the network. In the case when the GPS system is unavailable, the error in one-hop relative locations of the nodes may be accumulated. Thus, the error for two far away nodes may be significant. In addition, so far most of the existing algorithms are only concerned with deploying nodes within a bounded area. VOR and Minimax algorithms are based on Voronoi diagrams and require every Voronoi cell to be bounded. However, by the definition of Voronoi

graph [58], each periphery Voronoi cell is unbounded, since it contains a Voronoi vertex at infinity. Thus, VOR and Minimax algorithms cannot be used in this situation.

Finally, though all the algorithms discussed above intended to maximize the node coverage, minimize the coverage overlap and gap and deploy nodes uniformly, they did not answer a fundamental question in the deployment: What type of node layout can provide the maximum coverage with the smallest overlap and gap? We will address this issue in the next section.

2.2 Ideal Node Layout for Maximum Coverage

Similar to the deployment algorithms discussed in the previous section, one of the important goals of our algorithm is to maximize the coverage area, where the spans of the coverage area on both X and Y dimension are much larger than the sensing range of sensors. However, before we design a maximum coverage algorithm, we need to know what type of node layout can provide the maximum coverage for a given number of nodes. In order to find the ideal node layout for maximum coverage, we introduce the Delaunay triangulation [58] to describe the layout of the network. Let N be a set of n nodes, which are randomly thrown in the plane, and T be a Delaunay triangulation of N, such that no other nodes in N are inside the circumcircle of any triangle in T. Suppose that a large number of sensor nodes are randomly thrown in a two-dimensional field. The entire sensing area can be partitioned into some Delaunay triangles, whose vertices represent sensor nodes. We assume that the number of nodes is so large that the entire working area consists of a large number of Delaunay triangles. We have the following theorem regarding the optimum node layout.

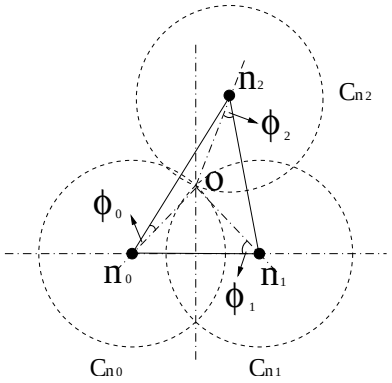


Figure 2.1: The maximum no-gap coverage area in a triangle can be obtained, if and only if the lengths of all three edges of the Delaunay Triangle equal $\sqrt{3}r$.

Theorem 1 *If all Delaunay triangles are equilateral triangles with edge length $\sqrt{3}r$, the coverage area of n nodes is maximum without coverage gap, where the spans of the coverage area on both X and Y dimension are much larger than the sensing range of sensors.*

Proof. Since the entire working area can be decomposed into a large number of Delaunay triangles, if we can prove that the no-gap coverage area in any Delaunay triangle is maximized when the lengths of all its edges equal $\sqrt{3}r$, then the maximum coverage area of n nodes can be obtained. Let C_{n_0}, C_{n_1} and C_{n_2} be the circles centered at the points n_0, n_1 and n_2, respectively, which denote the coverage area of corresponding nodes. Without loss of generality, we assume that circle C_{n_0} and circle C_{n_1} cross at point O, where O and n_2 locate on the same side of edge (n_0, n_1) as shown in Fig. 2.1. Let $\phi_0 = \angle n_2 n_0 O$, $\phi_1 = \angle n_0 n_1 O$ and $\phi_2 = \angle n_1 n_2 O$, where $0 < \phi_0, \phi_1,$ and $\phi_2 < \frac{\pi}{2}$. Let $|\overline{n_i O}|$ denote the distance between node n_i and O, where $n_i \in \{n_0, n_1, n_2\}$. From Fig.2.1, since C_{n_0} and C_{n_1} cross at O, we can obtain $|\overline{n_1 O}| = |\overline{n_0 O}| = r$. In order to maximize the area of the triangle without coverage gap, $|\overline{n_2 O}|$ should equal r. The area of triangle $\Delta n_0 n_1 n_2$, denoted as $A(\Delta n_0 n_1 n_2)$, can be calculated as the summation of the area of the following three triangles $\Delta n_0 n_1 O$, $\Delta n_0 n_2 O$

and $\Delta n_2 n_1 O$.

$$\begin{aligned} A(\Delta n_0 n_1 n_2) &= A(\Delta n_0 n_1 O) + A(\Delta n_0 n_2 O) + A(\Delta n_2 n_1 O) \\ &= r^2(\sin\phi_0\cos\phi_0 + \sin\phi_1\cos\phi_1 + \sin\phi_2\cos\phi_2) \\ &= \frac{r^2}{2} \times (\sin(2\phi_0) + \sin(2\phi_1) + \sin(2\phi_2)) \end{aligned} \tag{2.1}$$

Since

$$|\overline{n_1 O}| = |\overline{n_0 O}| = |\overline{n_2 O}| = r$$

we have

$$\phi_0 + \phi_1 + \phi_2 = \frac{\pi}{2} \tag{2.2}$$

By replacing ϕ_2 with $(\frac{\pi}{2} - \phi_0 - \phi_1)$ in (2.1), we obtain,

$$A(\Delta n_0 n_1 n_2) = \frac{r^2}{2} \times (\sin(2\phi_0) + \sin(2\phi_1) + \sin(2\phi_0 + 2\phi_1)) \tag{2.3}$$

Let

$$f(\phi_0, \phi_1) = \sin(2\phi_0) + \sin(2\phi_1) + \sin(2\phi_0 + 2\phi_1)$$

When

$$\frac{\partial f(\phi_0, \phi_1)}{\partial \phi_0} = 0 \text{ and } \frac{\partial f(\phi_0, \phi_1)}{\partial \phi_1} = 0$$

the maximal value of $A(\Delta n_0 n_1 n_2)$ can be obtained. Thus,

$$\begin{aligned} \frac{\partial f(\phi_0, \phi_1)}{\partial \phi_0} &= 2\cos(2\phi_0) + 2\cos(2\phi_0 + 2\phi_1) = 0 \\ \frac{\partial f(\phi_0, \phi_1)}{\partial \phi_1} &= 2\cos(2\phi_1) + 2\cos(2\phi_0 + 2\phi_1) = 0 \end{aligned} \tag{2.4}$$

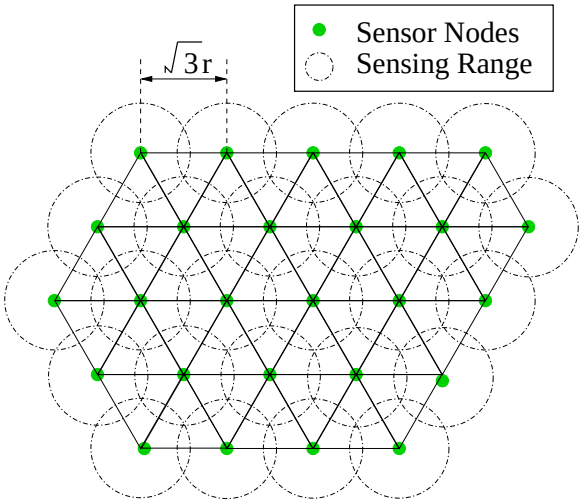


Figure 2.2: The perfect node layout for maximum no-gap coverage.

By solving (2.4), we obtain,

$$\begin{aligned} 2\phi_0+\phi_1 &= \frac{(2k+1)\pi}{2}, \quad k=0,\pm1,\pm2,\ldots \\ \phi_0+2\phi_1 &= \frac{(2m+1)\pi}{2}, \quad m=0,\pm1,\pm2,\ldots \end{aligned} \tag{2.5}$$

Since $0<\phi_0,\phi_1,\phi_2<\frac{\pi}{2}$, we have,

$$2\phi_0+\phi_1=\phi_0+2\phi_1=\frac{\pi}{2} \tag{2.6}$$

From (2.2) and (2.6), we can obtain that the maximum value of $A(\Delta n_0n_1n_2)$ can only be achieved when $\phi_0=\phi_1=\phi_2=\frac{\pi}{6}$. We have shown that when $\phi_0=\phi_1=\phi_2=\frac{\pi}{6}$ and the lengths of all three edges $\overline{n_0n_1}$, $\overline{n_1n_2}$ and $\overline{n_2n_0}$ equal $\sqrt{3}r$, the area of triangle $\Delta n_0n_1n_2$ is maximized. Therefore, as depicted in Fig. 2.2, if all Delaunay triangles are equilateral triangles with edge length $\sqrt{3}r$, the no-gap coverage area in a plane is maximized.

Having considered the maximum coverage problem, we now derive the minimum average moving distance of the nodes under a uniform deployment density. We assume that initially all sensor nodes are in a compact area near the origin of the polar coordinate system and eventually will be deployed to a disk-shaped area S with radius D, that is, the sensors are uniformly distributed over area $S = \pi D^2$. We have the following theorem regarding the minimum average moving distance.

Theorem 2 *When sensor nodes are deployed from a compact area to a disk-shaped area S with radius D, the minimum average moving distance of the nodes is* $\frac{2D}{3}$.

Proof. When nodes are uniformly distributed over area S, the minimum average moving distance D_{avg} can be computed as the average distance from the origin of the polar coordinate system. Let (ρ, θ) denote the polar coordinate of the node. Thus,

$$D_{avg} = E(\rho) = \int_0^{2\pi} \int_0^D \frac{\rho}{\pi D^2} \rho d\rho d\theta = \frac{2D}{3} \tag{2.7}$$

By plugging $S = \pi D^2$ into (2.7), we have

$$D_{avg} = \frac{2}{3}\sqrt{\frac{S}{\pi}} \tag{2.8}$$

It should be pointed out that D_{avg} can be achieved only when every node directly moves to its final position during the deployment. Thus it represents the minimum average moving distance in the ideal situation. However, this optimum value is difficult to achieve when the final position of each node is unknown before the deployment and there are obstacles that may block the movement of the nodes. Nevertheless, D_{avg} can serve as a lower bound on the average moving distance of the nodes for any deployment algorithm. We will compare the average moving distance of our algorithm with D_{avg} in the simulation section.

2.3 Adaptive Triangular Deployment Algorithm

In this section, we present a new adaptive deployment algorithm based on the optimum node layout we obtained in the previous section. For presentational convenience, we start with a simpler version of the algorithm, called *triangular deployment (TRI) algorithm*, then discuss some strategies to improve the basic algorithm, and finally present the complete algorithm.

2.3.1 The Basic Triangular Deployment Algorithm

We have known what type of node layout can maximize the coverage in a plane. Now the issue that needs to be addressed is how to deploy nodes from a compact area or an irregular layout to a perfect layout. A large number of sensor nodes are randomly thrown into a working area or placed in a bunch. As discussed earlier, exchanging global location/topology information during such a dynamical deployment period would put a heavy traffic burden to the network. Furthermore, when a bunch of nodes are located within a compact area and most of them need to communicate with others at the same time, the communication will be very inefficient due to the collision at the MAC layer [43]. Thus, the node movement decision should be based on local information in the deployment process. Since the location information is updated periodically, as a result, each node can only decide its movement periodically.

We now present an algorithm to deploy the sensor nodes close to a perfect equilateral triangular layout with the maximum coverage. The basic idea of the algorithm is to adjust the distance between two Delaunay neighbors to $\sqrt{3}r$ in three different coordinate systems, namely, XY, $X'Y'$ and $X''Y''$, where the angles between X-axis and X'-axis, and between X-axis and X''-axis are $\frac{\pi}{3}$ and $\frac{2\pi}{3}$, respectively. As shown in Fig. 2.3(a), the coverage area of

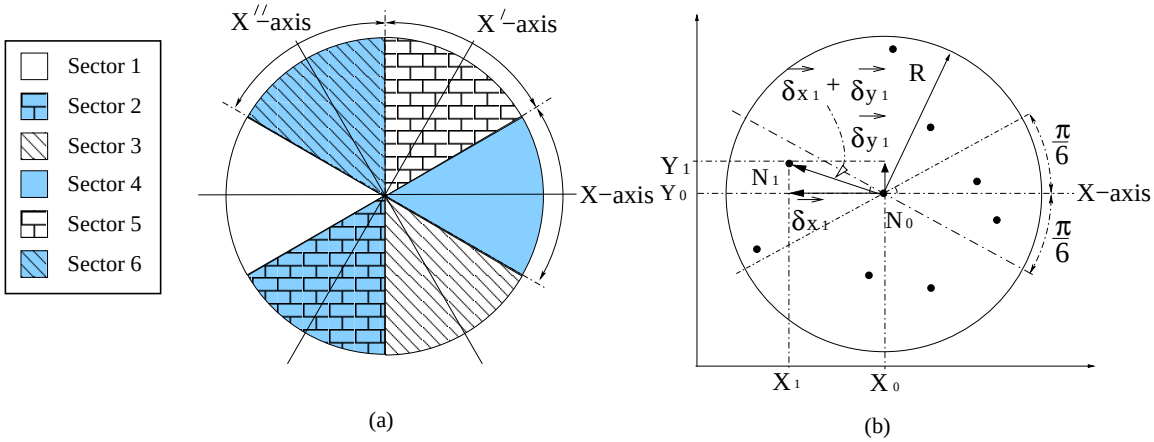


Figure 2.3: Local movement strategies based on the location of one-hop neighbors.

a sensor node is divided into six sector areas, called sectors 1 to 6 counterclockwise, where X-axis, $X^{'}$-axis and $X^{''}$-axis symmetrically partition sectors 1 and 4, sectors 2 and 5, and sectors 3 and 6. The radius of each sector equals the transmission range of the node. The location of the nodes in sectors 1 and 4, sectors 2 and 5, and sectors 3 and 6, are expressed by XY, $X^{'}Y^{'}$ and $X^{''}Y^{''}$ coordinates, respectively.

In each sector, the node adjusts its location along the corresponding axis based on the location of its Delaunay neighbors. However, the adjustment algorithm based on Delaunay diagram suffers the similar limitation as the solutions based on Voronoi graph, since global location of all sensor nodes is needed to determine Delaunay triangulations and Delaunay neighbors. In practice, we use the nearest neighbors to the node in each sector instead of Delaunay neighbors. For example, as shown in Fig. 2.3(b), node N_1 is the nearest neighbor of node N_0 in sector 1, where their coordinates in XY coordinate system are (X_1, Y_1) and (X_0, Y_0). Let the location vector $\overrightarrow{\delta x_1}$ and $\overrightarrow{\delta y_1}$ denote vectors $[(X_1 - X_0), 0]$ and $[0, (Y_1 - Y_0)]$, respectively. If $|\overrightarrow{\delta x_1} + \overrightarrow{\delta y_1}| < \sqrt{3}r$, it means that there is too much coverage overlap between node N_0 and node N_1. Thus, the movement of N_0 should be opposite to N_1

for $\frac{|\overrightarrow{\delta x_1} - \sqrt{3}r\frac{\overrightarrow{\delta x_1}}{|\overrightarrow{\delta x_1}|}|}{2}$ to reduce the coverage overlap. On the contrary, if $|\overrightarrow{\delta x_1} + \overrightarrow{\delta y_1}| > \sqrt{3}r$, a coverage gap may exist between node N_0 and node N_1. We will let N_0 move towards N_1 along X-axis for $\frac{|\overrightarrow{\delta x_2} - \sqrt{3}r\frac{\overrightarrow{\delta x_2}}{|\overrightarrow{\delta x_2}|}|}{2}$ to fill the coverage gap. Besides the movement on X-coordinate, the movement vector of N_0 projected on Y-coordinate equals $\overrightarrow{\delta y_1}/2$. Thus, the movement vector of N_0, $\overrightarrow{\delta v_1} = \frac{\overrightarrow{\delta x_1} - \sqrt{3}r\frac{\overrightarrow{\delta x_1}}{|\overrightarrow{\delta x_1}|} + \overrightarrow{\delta y_1}}{2}$.

In general, for sector s, each node searches the nearest neighbor within the sector and calculates the relative horizontal and vertical location vectors $\overrightarrow{\delta x_s}$ and $\overrightarrow{\delta y_s}$ along its corresponding axis. Here, $\overrightarrow{\delta x_s}$ and $\overrightarrow{\delta y_s}$ are expressed by relative coordinates corresponding to sector s. The movement vector of a node in sector s, $\overrightarrow{\delta v_s}$, can be expressed as

$$\overrightarrow{\delta v_s} = \frac{\overrightarrow{\delta x_s} - \sqrt{3}r\frac{\overrightarrow{\delta x_s}}{|\overrightarrow{\delta x_s}|} + \overrightarrow{\delta y_s}}{2} \tag{2.9}$$

Note that each movement vector is obtained in one of three different coordinate systems, XY, $X'Y'$ and $X''Y''$. After the movement vectors in all six sectors are obtained, they need to be transferred into uniform coordinates and added in order to obtain the total movement vector for the current round. Table 2.1 gives this triangular deployment algorithm. As will be seen in the simulation results, after several rounds of such adjustments, the layout of the network will be close to the ideal equilateral triangle layout. As a result, the coverage area of the network will be maximized.

Table 2.1: Triangular deployment algorithm

Triangular Deployment Algorithm
for each round
  for each node $i = 1$ to n
    Broadcast "Hello" message containing its location information to its one-hop neighbors;
    Receive "Hello" message from nearby nodes and obtain their location information;
    Divide its coverage area into 6 sectors;
    for each sector $s = 1$ to 6
      The node calculates location vector $\overrightarrow{\delta x}$ and $\overrightarrow{\delta y}$ to its nearest neighbor
       in the coordinate system;
      Calculate and store the movement vector for sector s, $\overrightarrow{\delta v_s} = \frac{\overrightarrow{\delta x} - \sqrt{3}r\frac{\overrightarrow{\delta x}}{|\overrightarrow{\delta x}|} + \overrightarrow{\delta y}}{2}$
    Transfer movement vectors of all sectors into uniform coordinates;
    Add them up to obtain the total movement vector for node i;
Move;

2.3.2 Minimizing Oscillation

We have proved that equilateral triangular layout can maximize the coverage, and also proposed a simple algorithm to adjust the network from an irregular layout to the ideal equilateral triangle layout. However, since the global location information of the network is difficult to obtain in the deployment process, it is impossible for each node to move to its desired target directly. Thus, sensor nodes may move back and forth frequently before they reach its desired target. To make the algorithm suitable to real-world applications, another important issue is to reduce the total moving distance of the nodes in the deployment.

Recall that the moving strategy in our triangular deployment algorithm is that if the horizontal distance between two neighbors is longer than $\sqrt{3}r$, the sensors will move towards each other to shorten the gap between them. On the contrary, they will move away from each other to reduce the coverage overlap. According to this strategy, nodes will move all

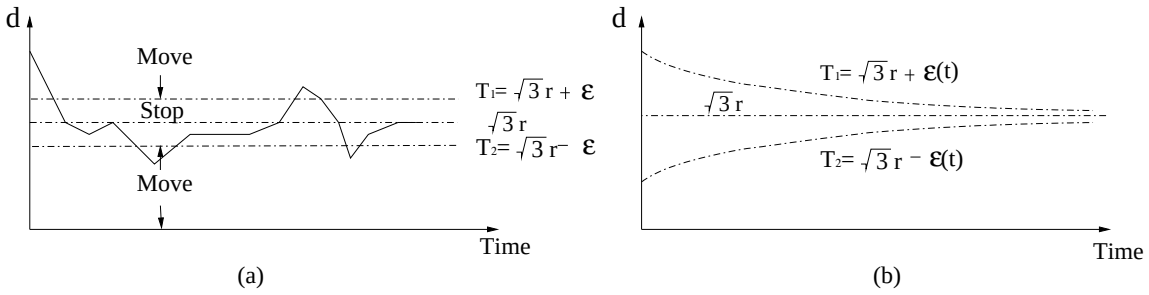


Figure 2.4: Threshold strategy for reducing node oscillation. (a) Constant threshold. (b) Variable threshold.

the time, unless the network reaches the perfect layout or its maximum rounds. In order to reduce the oscillation, we adopt a threshold strategy by using two distance thresholds, T_1 and T_2, instead of $\sqrt{3}r$ for making moving decisions, where $T_1 = \sqrt{3}r + \varepsilon$, $T_2 = \sqrt{3}r - \varepsilon$ and ε is a small constant. As described in Fig. 2.4(a), the Y-coordinate d denotes the distance between the node and its nearest neighbor. When two far away nodes move towards each other and the distance between them decreases to T_1, two nodes stop moving. On the other hand, when two close nodes move apart and the distance between them increases to T_2, they will stop and keep the current distance between them. This moving strategy guarantees that the node will not move if it is located between T_1 and T_2 away from its neighbors, so that the node is affected less when its neighbors move slightly. Note that if the adjustment granularity is too small, which is given by $\Delta d = T_1 - T_2 = 2\varepsilon$, T_1 and T_2 are close to $\sqrt{3}r$ at the beginning of the deployment process and there will be no obvious difference between the algorithm in Table 2.1 and the algorithm with threshold strategy. However, if Δd is too large, it is impossible to adjust the network to the perfect equilateral triangular layout. In order to solve this problem, we let the thresholds T_1 and T_2 be the function of time t. As shown in Fig. 2.4(b), the adjustment granularity decreases as time t increases. That is, $T_1 = \sqrt{3}r + \varepsilon(t)$ and $T_2 = \sqrt{3}r - \varepsilon(t)$, respectively, where $\varepsilon(t)$ is

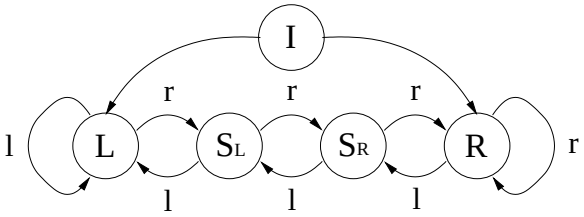


Figure 2.5: Movement state diagram for reducing node oscillation. L: Move left; R: Move Right; S_L and S_R: Stay; I: Initial state; l: Expected moving direction is left; r: Expected moving direction is right.

called the threshold function. In practice, since the algorithm is executed round by round, the threshold can be changed to a function of the number of rounds. In our simulations in Section2.4, we use $\varepsilon(Rd_{cur}) = \frac{\sqrt{3}}{4}r \times e^{-Rd_{cur}/Rd_{total}}$ as the threshold function, where Rd_{cur} and Rd_{total} are the numbers of the current rounds and the total rounds, respectively.

The second strategy, called movement state diagram, is to use a state diagram to reduce the node oscillation. Each movement can be considered as a vector and be decomposed into the projection vectors on X and Y coordinates. For X coordinate, nodes can only move left or right. Oscillation exists when a node moves towards the opposite direction of the previous movement. In order to avoid oscillation, nodes are not allowed to move backwards immediately. Two state diagrams are used in the movement vectors projected on X and Y coordinates separately. Fig. 2.5 shows an example of movement state diagram for X coordinate, which contains 5 states and is used in our simulation. The diagram has 5 states: L, R, S_L, S_R and I, and two transitions l and r. L and R denote the movement to the left and the right respectively. If the state of a node is S_L or S_R, it has to stay where it is till the next round. I is the initial state. l and r represent the moving decision to the left and the right made by the triangular deployment algorithm. For example, a node plans to move left after running the triangular algorithm, which means that the current transition is l. Then, it needs to check its current state on its state diagram. If its current state is L, I or S_L, the

next state will go to L after the transition l. A node can move left only when its next state is L. If the current state of the node is S_R (or R), the next state will transit to S_L (or S_R) upon the transition l. Thus, the node cannot move until next round. The movement control for Y coordinate follows a similar procedure. Simulation results in Section 2.4 shows that the distance threshold strategy and movement state diagram strategy can reduce a significant amount of movements during the deployment.

2.3.3 Adaptive Triangular Deployment Algorithm

We have discussed how to deploy nodes with equal or almost equal density in an open area where the entire area needs to be sensed uniformly. However, in many real-world applications, the working area is partially or entirely bounded. Also, some irregularly-shaped obstacles may be in the working area. In other situations, sensors may need to be deployed with different density based on the requirements of tasks. Without the control from human being or central controller, and without map and global information, sensors have to be smart enough to make decisions themselves.

In order to make the deployment algorithm more practical, sensors must be able to avoid obstacles and boundaries. Because an accurate map of the sensing area may not be always available before the deployment, we assume that each sensor is equipped with an ultrasonic obstacle detecting module [64] which makes it possible to detect obstacles when it moves close enough to the obstacles. As discussed above, the triangular deployment algorithm can only adjust the relative positions of two sensor nodes. However, unlike sensor nodes, obstacles and boundaries usually have irregular shapes and continuous outlines. In order to enable the triangular deployment algorithm to adjust the relative positions between sensor nodes and obstacles and boundaries with only a minor modification, the outlines of

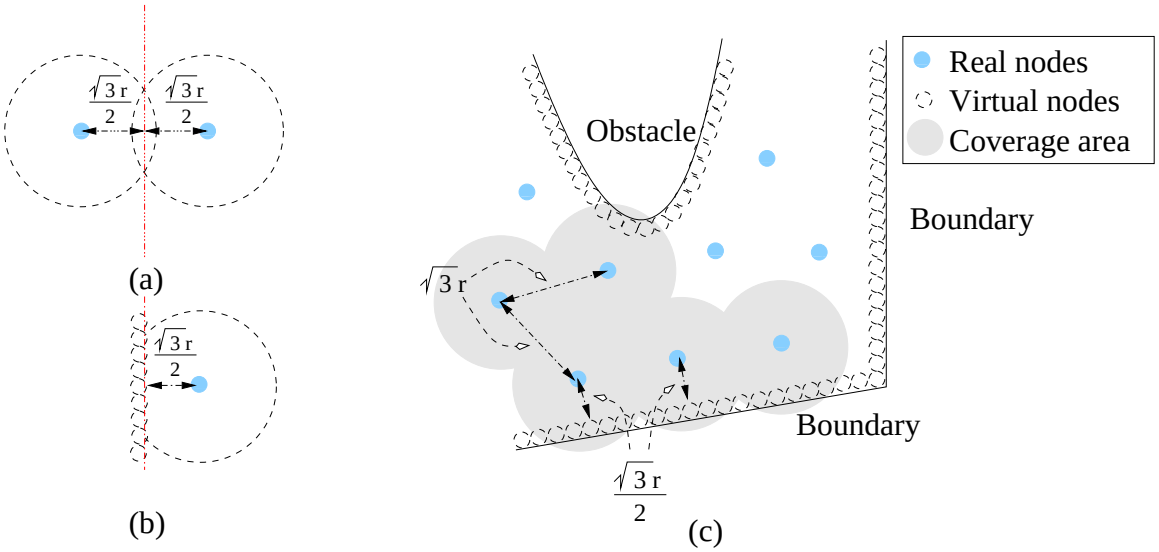
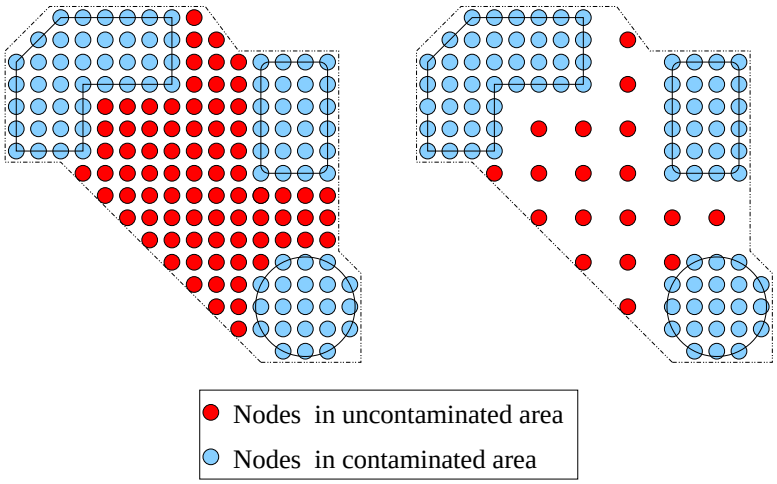


Figure 2.6: Examples of the adaptive deployment in a bounded area with obstacles.

obstacles and boundaries are abstracted as many virtual nodes, which surround obstacles and boundaries closely. As shown in Fig. 2.6, each small dotted circle around obstacles and boundaries denotes a virtual node. In practice, after each sensor node detects the outlines of obstacles or boundaries within its coverage area, from a sensor node's point of view, the outlines of obstacles or boundaries can be considered as many virtual nodes. Like real sensor nodes, these virtual nodes also "push" real nodes away when real nodes are located too close to them, or "pull" real nodes close to them when real nodes are located too far away from them. Similar to the basic triangular deployment algorithm, after a real sensor node divides its coverage area into six sectors, it takes account of both real nodes and virtual nodes in each sector of its coverage area. When the real node finds other real or virtual nodes are located too close to itself in each sector, it moves away from them. On the contrary, when it is located too far away from other real or virtual nodes, it moves towards them to fill the coverage gap. However, unlike real nodes, virtual nodes can neither move nor cover the sensing area. Since virtual nodes cannot actually cover any area, in order

to avoid the coverage gap or overlap between real nodes and virtual nodes, intuitively, the optimum distance between adjacent real nodes and virtual nodes should be shorter than $\sqrt{3}r$. In Fig. 2.6(a), the coverage areas of two real nodes are equally divided by a straight line. As discussed earlier, both real nodes need to be at least $\sqrt{3}r/2$ away from the line bisector in the optimum layout. The straight line can be considered as the virtual "boundary" of the coverage areas of two nodes. In Fig. 2.6(b), when the line becomes a real boundary of an obstacle or a wall, the node on the right still needs to keep the distance to the line as $\sqrt{3}r/2$ to avoid the coverage gap. Thus, the distance between a real node and a virtual node needs to be adjusted to $\sqrt{3}r/2$ instead of $\sqrt{3}r$. We revise our algorithm as follows for deploying nodes in an area with obstacles. Let r_d denote the deployment radius. In each sector, if the nearest neighbor of the node is a virtual node, the node sets its deployment radius r_d to $\frac{r}{2}$. If its nearest neighbor is a real node, its r_d still equals its sensing radius r. And then, the node runs the triangular deployment algorithm by replacing r with r_d. Fig. 2.6 shows an example of the layout after nodes are deployed in a partially bounded area with irregularly-shaped obstacles. We can see that the distance between real nodes is still $\sqrt{3}r$, while the distance between real nodes and virtual nodes (obstacles or boundaries) is $\sqrt{3}r/2$.

Besides avoiding the obstacles in the working area, the location of nodes should also follow the occurring distribution of the interested events. The higher frequency or density of the event occurring within a given area, the more sensor nodes are needed to monitor the status change in that area. On the contrary, it is unnecessary to monitor the event-blank area where there will never exist interested events. In other words, nodes should be deployed with different densities based on the requirements of tasks. For example, suppose sensor nodes have two different types of tasks: sensing and communication. For



(a) Uniform Distribution

(b) Non−uniform Distribution Based On Interested Events

Figure 2.7: Examples of the adaptive deployment based on different task requirements.

different tasks or regions, the coverage range of a node may be different. As shown in Fig. 2.7, a bunch of sensors are thrown into a contaminated chemistry factory to monitor the density of leaked chemicals, in which contaminated sources locate in separated buildings, which may not be uniformly distributed. The task of some nodes with high density is to cover and monitor the contaminated area. And the task of other nodes located in the event-blank area with low density is to provide communications between any two separated contaminated buildings. Thus, the algorithm should not only maximize the coverage area where the nodes need to be deployed uniformly, but also adjust the node density based on the different distribution of event occurring. It is easy to revise the triangular deployment algorithm to make it suitable for the adaptive deployment based on different requirements. In the triangular deployment algorithm, in order to obtain maximum no-gap coverage, each node tries to adjust the distance to its nearest neighbors to $\sqrt{3}r$. When nodes move into a highly concerned region and find that they need to be deployed more densely, they set

a new shorter deployment radius, say, r_d, instead of the sensing radius r, and then run the adaptive triangular deployment algorithm by replacing r with r_d. Note that the deployment radius r_d of each sensor can be decided by itself based on the measurement obtained from the environment. When a region does not need to be sensed with high density, nodes extend to a sparser density by choosing a longer deployment radius.

By applying distance threshold, movement state diagram and adaptive adjustment strategies, we obtain the adaptive triangular deployment algorithm (ATRI) that is summarized in Table 2.2. As will be seen in our simulation results, by incorporating these strategies, our adaptive triangular deployment algorithm can drive nodes to avoid obstacles in the area and deploy them with different densities based on the requirements of tasks.

2.3.4 Discussions on Some Practical Issues: Synchronizing Sensors and Reducing Packet Collision

So far we have assumed that sensors are well-synchronized and the location information can be exchanged between sensors without packet collision. However, in practice, the clocks of sensors can be imprecise due to several reasons. First, the clocks may not be initially synchronized well. Sensors may be turned on at different times. The clock may also be affected by the changes of the environment, such as temperature and pressure or the battery voltage. Without global synchronization, some sensors with "faster clock" may reach the maximum rounds and stop moving, while others still seek and try to move to better positions to improve the coverage. In order to make all sensors move at the same pace, sensors need to be synchronized globally. Synchronization in sensor networks has been studied in the literature by many researchers. For instance, in [65] Qun et al proposed

Table 2.2: Adaptive triangular deployment algorithm

Adaptive Triangular Deployment Algorithm
for each round
for each node $i = 1$ to n
Broadcast "Hello" message containing its location information to its one-hop neighbors;
Receive "Hello" message from nearby nodes and obtain their location information;
Detect obstacles or boundaries within its coverage area and obtain location of virtual nodes;
Divide its coverage area into 6 sectors;
for each sector $s = 1$ to 6
Adjust its sensing radius r_d adaptively based on the requirement of tasks or location of virtual nodes;
Calculate the threshold value THR;
Calculate location vector $\overrightarrow{\delta x}$ and $\overrightarrow{\delta y}$ to its nearest neighbor/virtual node ;
if $(0 < |\delta x_s| \leq |\sqrt{3}r_d - THR|)$ **or** $(|\delta x_s| \geq |\sqrt{3}r_d + THR|)$
Calculate and store the movement vector for sector s, $\overrightarrow{\delta v_s} = \frac{\overrightarrow{\delta x} - \sqrt{3}r_d \frac{\overrightarrow{\delta x}}{|\overrightarrow{\delta x}|} + \overrightarrow{\delta y}}{2}$;
else
$\overrightarrow{\delta v_s} = \overrightarrow{0}$;
end if
Transfer movement vectors of all sectors into uniform coordinates;
Add them up to obtain the total movement vector for node i;
Check the state diagram to decide if move or not and make transitions on the state diagram;
Move;

a fully localized diffusion-based synchronization method, which scales well in a large network. This algorithm can be adopted in our deployment algorithm for the synchronization purpose.

Moreover, in some applications, a large number of sensors are initially placed in a compact area. When every sensor wants to obtain the shared channel to broadcast its location information, the channel may become so busy that many packets are collided. Though our main focus of this chapter is on the movement planning of sensors, nevertheless next we briefly discuss how sensors can exchange location information reliably in such situation. A simple way to reduce collision is to deploy sensors as sparsely as possible if the environment and the application permit. As studied in some existing work [50, 51], another feasible solution is to adaptively adjust the contention-window size based on how busy the channel is. However, both solutions cannot completely avoid collision. Unlike some existing work on MAC protocols, here we are more concerned with the reliability than the throughput or the channel utilization during the deployment phase. In order to avoid collision, instead of using a flat topology, we can introduce a hierarchy into the network. During each round of the ATRI algorithm, we let some sensors act as cluster heads and poll other sensors to avoid packet collision. Since the network topology keeps changing before the maximum round of the ATRI algorithm is reached, cluster heads are not fixed and should be selected at the beginning of each round. To become a cluster head, sensors first operate in a contention-based mode and compete with each other to obtain the channel. Once a node obtains the channel by successfully sending out a broadcast message, it becomes a cluster head. All sensors that receive the broadcast message stop trying to send out the message and become cluster members. After becoming a cluster head, the sensor polls all sensors one by one. If the polled sensor is in the transmission range of the cluster head, the

sensor sends its location information to the cluster head. Otherwise the channel keeps idle for a short period, and then the cluster head will poll the next sensor. Though the procedure of electing the cluster head is contention-based, after a sensor is selected as the cluster head, sensors in the cluster stop competing with each other and can upload their location information to the cluster head in a contention-free manner. After acquiring positions of all sensors in its transmission range, the cluster head broadcasts all newly updated location information to all nodes in its transmission range. Thus, by running the polling protocol, the inner-cluster collision can be avoided.

Unlike traditional one-hop wireless networks, such as WLANs and Bluetooth networks, a sensor network may consist of multiple clusters. Each cluster head may not know the activities of other cluster heads. Thus, if two nearby cluster heads broadcast packets at the same time, the packets may collide. The inter-cluster collision problem basically is a hidden terminal problem, which has been extensively investigated in the literature. Busy-tone-based approaches can solve the hidden terminal problem by using a busy tone to warn nodes not to send packets. A busy tone can be a simple unmodulated SINE wave transmitted in a separate narrow-band channel. Togagi et al proposed a busy tone multiple access (BTMA) scheme [66] to solve the hidden terminal problem by requiring a receiver to power up a busy tone to warn hidden nodes. Haas et al. presented a dual busy tone multiple access (DBTMA) scheme [67], which uses two physically separate tones, one indicating transmitting busy, and another indicating receiving busy. Other nodes that hear the busy tone will postpone their transmission to prevent the collision. The idea of busy tones can be used to solve the inter-cluster collision problem. Once a cluster head obtains the channel, it broadcasts a busy tone in a separate channel with a much longer transmission range than that for transmitting regular data to disable the transmission of all nearby cluster heads and sensors.

Since the busy tone is transmitted in a much simpler waveform and at lower frequency band than regular data, the busy tone can be transmitted more energy efficiently, even it needs to be sent further than regular data. Thus, by using the combination of the polling protocol and the busy-tone scheme, the inner-cluster and inter-cluster packet collision can be avoided, and location information can be exchanged reliably.

2.4 Performance Evaluations

This section presents a set of experiments designed to evaluate the performance and cost of the proposed algorithm. Besides the ideal flat open area, the simulation is also run in the more practical environments, where irregularly-shaped obstacles may block the movement of nodes. In addition, we implement the adaptive deployment based on different deployment requirements of the regions. All sensor nodes are equipped with Chipcon CC2420 Zigbee transceivers [68], which can reach as far as 50 meters away. Each sensor node can sense the occurring of events within a radius of 3 meters away from itself. At the beginning of the experiments, sensors are randomly placed within a $1m \times 1m$ compact square which is centered at point $(25m, 25m)$. Then the nodes explode to a large, evenly deployed layout. In order to limit the oscillation of the nodes, the same movement state diagram depicted in Fig. 2.5 is used in all scenarios. The distance threshold function $\varepsilon(Rd_{cur}) = \frac{\sqrt{3}}{4} r_d \times e^{-Rd_{cur}/Rd_{total}}$, where Rd_{cur} and Rd_{total} are the numbers of current rounds and total rounds. We measure the total coverage area and the average moving distance per node and compare them with existing algorithms.

2.4.1 Performance and Cost Evaluation

In this subsection, we compare the performance and cost of VEC and ATRI algorithms. VEC algorithm has similar performance as VOR and Minimax algorithm in a bounded area. In addition, like ATRI algorithm, VEC algorithm can be used in both unbounded and bounded areas, while VOR and Minimax algorithms have to know the boundary information. For the sake of simplicity, we did not take account of the communication cost for exchanging location information and assume that location information is error-free, though VEC algorithm needs global location information and is more vulnerable to inaccurate location information than ATRI algorithm. In order to see the effects of various node densities, 100 nodes are randomly placed into a $1m \times 1m$ square around point $(25m, 25m)$ at the beginning, then they explode from a compact area to a large area. Both algorithms run for 100 rounds. In order to evaluate the performance and cost of the two algorithms, two metrics are measured for each simulation round: total coverage area and average moving distance, which are defined as the coverage area of 100 nodes and accumulated moving distance per node from the beginning of the simulation, respectively. Fig. 2.8(a) shows the total coverage area of both algorithms as the simulation rounds increase. We can see that the total coverage of both algorithms increases rapidly to as high as $2200m^2$ before round 40, and then goes smoothly after round 40. At round 100, ATRI stops at about $2600m^2$, while VEC is close to $2500m^2$. From rounds 10 to 100, ATRI always leads VEC for about $50m^2$ to $100m^2$. Fig. 2.8(b) describes the average moving distance when simulation rounds range from 10 to 100. Average moving distance has a similar increasing trend to the total coverage area, as the simulation rounds increase. In both algorithms, after round 60, nodes are deployed evenly and their average distance is close to $\sqrt{3}r$. Most nodes do not need to move except minor adjustments. As shown in Fig. 2.9, after round 60, most nodes

form the equilateral triangle layout. There are no obvious changes between the layout of round 60 and that of round 80, because the layout of nodes is already very close to the ideal equilateral triangular layout after round 60. In addition, from total coverage area of both algorithms, we also calculate optimum average moving distances and plot them in Fig. 2.8(b). As discussed earlier, given a fixed total coverage area, the optimum average moving distance can be calculated by (2.8). Recall that optimum average moving distances can only be obtained when the working environment is well known and each node knows its expected target before the deployment. Without the map of the environment, nodes have to move in a zigzag manner, which makes the average moving distances of both algorithms longer than the optimum values. However, compared to VEC algorithm, ATRI still saves up to 50% of the optimum average moving distance from rounds 10 to 100.

2.4.2 Non-Uniform Deployment

In this subsection, we simulate the scenario that some dangerous chemical is leaking at point $(25m, 25m)$. Without loss of generality, we assume that the contaminated area is disk-shaped, which is centered at point $(25m, 25m)$. The radius of the contaminated area is $10m$, which is unknown to the sensors before the deployment. In order to detect the region of the contaminated area, 200 sensor nodes are thrown within the compact square area close to the origin of the chemical. The circle around $(25m, 25m)$ in Fig. 2.10(a) represents the contaminated area. The sensor nodes within the contaminated area need to be more densely deployed than non-contaminated area to detect the small change of chemical density. Within the contaminated area, the sensing radius is $1m$, while the sensing radius within the non-contaminated area is $3m$. The whole sensing area is bounded within $50m \times 50m$, which is also centered at point $(25m, 25m)$. Fig. 2.10(b) and (c) plot the deployment

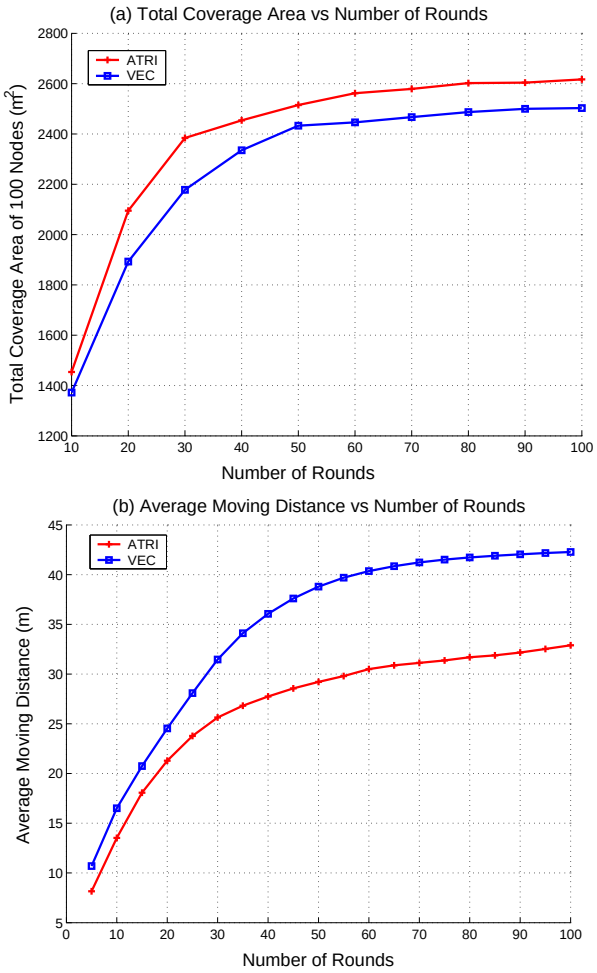


Figure 2.8: The total coverage area and average moving distance of ATRI and VEC algorithms for 100 runs of simulations.

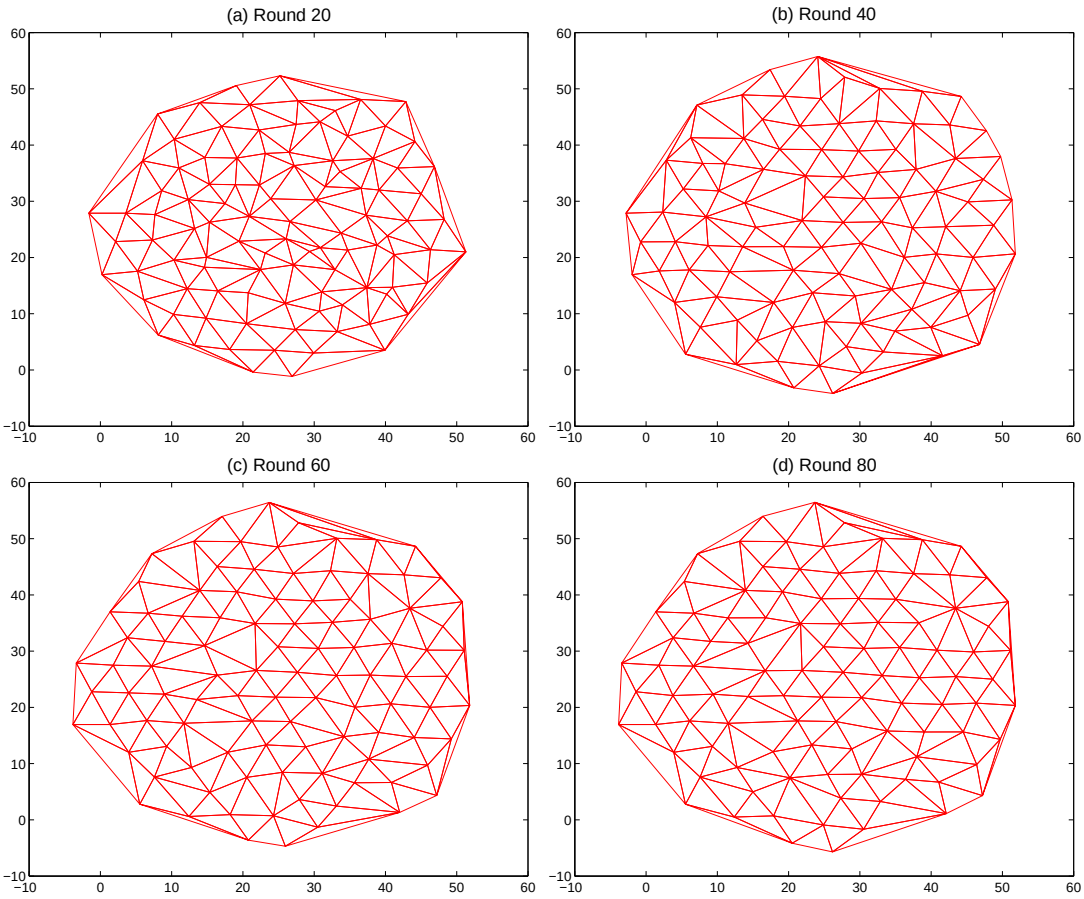


Figure 2.9: Triangular layout of 100 nodes at rounds 20, 40, 60 and 80.

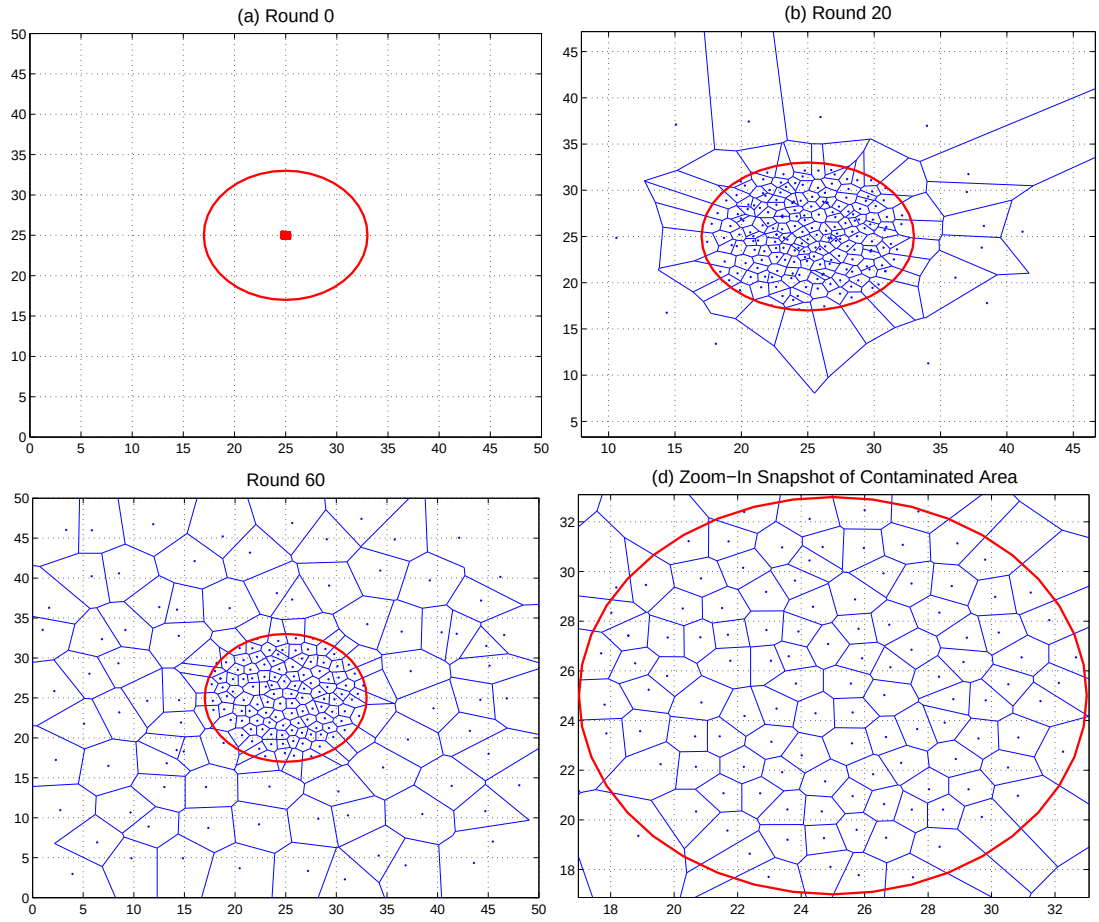


Figure 2.10: Simulation results for non-uniform deployment: (a) Round 0; (b) Round 20; (c) Round 60; (d) Zoom-in Snapshot of the contaminated area.

layout at round 20 and round 60, respectively. Fig. 2.10(d) is the zoom-in snapshot of the contaminated area at round 60. We can see that at round 60, sensor nodes in both contaminated area and non-contaminated area are deployed evenly with the corresponding deployment radius. Our adaptive algorithm can also be used in various irregularly-shaped contaminated areas, or highly concerned areas. In addition, in a more complicated case that the contaminated area enlarges or shrinks from time to time, the sensor nodes can also change their deployment density dynamically to satisfy the requirement.

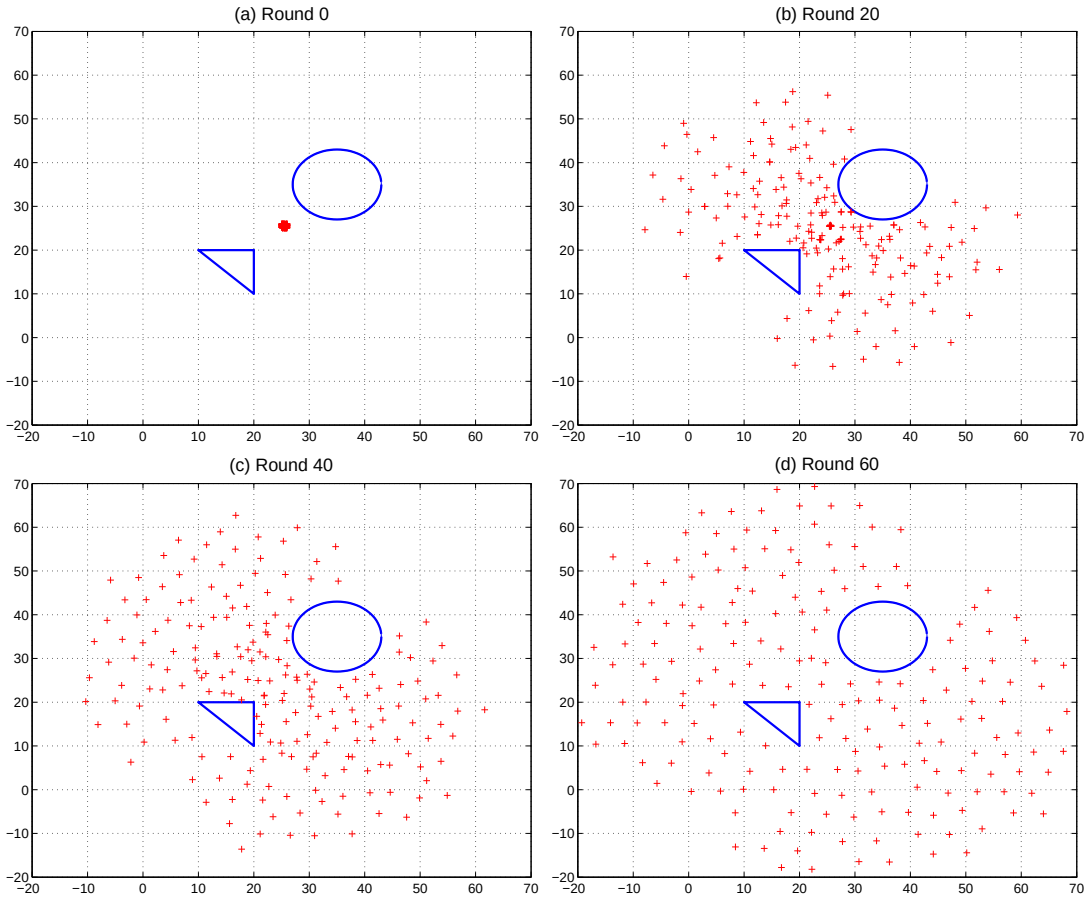


Figure 2.11: Simulation results for the environment with obstacles: (a) Initial snapshot; (b) Round 20; (c) Round 40; (d) Round 60.

2.4.3 Exploring the Area with Obstacles

In order to show that ATRI algorithm works well in the sensing environment with obstacles, a circular and a triangular obstacles are placed in the sensing area. As shown in Fig. 2.11, the circular obstacle is centered at point $(35m, 35m)$ with radius $8m$. Vertices of the triangular obstacle are $(10m, 20m)$, $(20m, 10m)$ and $(20m, 20m)$, respectively. Fig. 2.11(a), (b), (c) and (d) illustrate the deployment layouts at rounds 0, 20, 40 and 60, respectively, where each "+" symbol denotes the position of its corresponding node. At the beginning,

Table 2.3: Exploring sensing area with obstacles : moving distance vs coverage area

Number of rounds	20	40	60
Total coverage area (m^2)	2032	3504	4415
Average moving distance (m)	17.45	33.33	46.13
Optimum average moving distance (m)	16.95	22.26	25.00

similar to the previous scenario, 200 sensor nodes are randomly thrown into a $1m \times 1m$ square around point $(25m, 25m)$. As the simulation runs, we can see from Fig. 2.11(b),(c) and (d) that, the total coverage enlarges round by round. At round 60, nodes are evenly deployed and no nodes enter the triangular or the circular region during the deployment. We can see that though obstacles block the movement of some nodes, ATRI algorithm still performs very efficiently. During the simulation, we measure the total coverage area and average moving distance at rounds 20, 40 and 60. In addition, we also calculate optimum average moving distance by plugging the total coverage area into (2.8). All three metrics are shown in Table 2.3.

In practice, optimum average moving distance is difficult to reach unless the desired optimum position of each node is known before the deployment and no obstacles block the movement of the nodes. In the situation that the map of the environment is unavailable and the movement of some nodes is blocked by obstacles, we can see that the movement cost of ATRI algorithm is very reasonable compared to the optimum value.

2.5 Conclusions

In this chapter, we have proposed a new adaptive deployment algorithm for unattended mobile sensor networks, namely, adaptive triangular deployment (ATRI) algorithm. We

have introduced an equilateral triangle deployment layout of sensor nodes and proved that it can produce the maximum no-gap coverage area in a plane. By using the only location information of one-hop neighbors for the adjustment of each node, the algorithm can make the overall deployment layout close to equilateral triangulations. In order to reduce the back-and-forth movement of nodes, the distance threshold strategy and movement state diagram strategy are adopted, which limit the oscillation and reduce the total movement distance of nodes. ATRI algorithm can be used in both bounded and unbounded areas. It also supports adaptive deployment. Without the map and information of the environment, nodes can avoid obstacles and adjust the density dynamically based upon different requirements. Without the control from the human being or central controller, each node can make decisions itself. In addition, ATRI algorithm is run in a completely distributed fashion by each node and based only on the location information of nearby nodes.

Chapter 3

Single Path Flooding Chain Routing Algorithm

In the first chapter, we have introduced an adaptive deployment algorithm for mobile sensor networks. After the deployment phase, sensor nodes begin to sense the field and communicate with each other to exchange sensing data. In the remaining parts of the thesis, we will introduce some novel algorithms to provide reliable and efficient communication for sensors.

This chapter gives a theoretical analysis about the effect of the out-of-date location information on the performance of the single path routing algorithm, and then present a new position-based routing algorithm for ad hoc networks. As will be seen, the new algorithm achieves much lower communication complexity than the existing flooding-based algorithms, which is measured by the average number of one-hop transmissions required to send a packet from a source node to a destination node, and can consistently perform well for various mobilities.

The rest of the chapter is organized as follows. In Section 3.1 we give a brief overview

of some position-based routing algorithms. In Section 3.2, we analyze the error probability of the existing single path routing algorithm caused by the location information periodical update. Based on the analysis in Section 3.2, we present in Section 3.3 our new single path flooding chain routing algorithm. Section 3.4 contains some simulation results of the algorithm, and finally, Section 3.5 concludes the chapter.

3.1 Related Work

The growing interest in ad-hoc/sensor networks has resulted in many routing algorithms and protocols proposed for such dynamic, self-organizing and resource-limited networks. Most of work in this area focuses on two types of routing algorithms: *topology-based* routing and *position-based* routing algorithms. Traditional topology-based routing algorithms, which are widely used in wired networks, depend on the link information to make routing decisions. On the other hand, position-based routing algorithms require the physical positions of nodes to perform packet forwarding. In general, the topology of an ad hoc network changes too frequently to be updated timely. Maintaining a routing table at each node introduces a significant amount of network traffic to an ad hoc network. Thus, position-based routing algorithms were proposed to eliminate some limitations of topology-based routing algorithms. In a position-based routing algorithm, there is no need to establish and maintain links, and routing decisions are mainly based on the location information of the destination node and the one-hop neighbors of the current node. Thus the algorithm can avoid the overhead of maintaining global information of the network.

In a position-based routing algorithm, each mobile node in an ad hoc network can obtain its own location information from the *Global Positioning System* (GPS) or some other positioning services [33, 34]. A routing decision at a node is triggered by an incoming

packet to the node and is made based on the location information of both the destination node and the one-hop neighbors of the sender. However, it is generally not sufficient to establish routing paths between the source and the destination if all nodes only know their own location information. In most position-based algorithms, each node broadcasts its location information to its one-hop neighbors periodically and requests location information of the destination node by contacting the *location service* [35, 36, 37, 38]. Location services are the mechanisms that provide the location information of a specific node in the network to any node which sends a request to them. Mobile nodes register their location information with the location service. When a node needs the location of a desired node, it contacts the location service to obtain the location of that node. An example of location services is the base station in cellular networks. Each base station is a location server and provides the location information to all mobile nodes in the cell. A survey on location services and position-based algorithms can be found in [25]. Position-based routing in ad hoc networks has been studied extensively in recent years. Most of proposed work can be divided into *flooding-based* algorithms [27, 28] or *greedy packet forwarding* (single-path based) algorithms [31]. In a flooding-based algorithm, a packet is flooded to the entire or most part of the network. On the other hand, in a greedy packet forwarding algorithm, a packet is transmitted through a single routing path and has only one copy in the network at any time. In this section we discuss some existing flooding-based routing algorithms and greedy packet forwarding algorithms.

3.1.1 Flooding-Based Routing Algorithms

The simplest flooding-based routing algorithm in an ad hoc network, called *blind flooding*, is to flood a packet from the source node to all other nodes in the network hop by

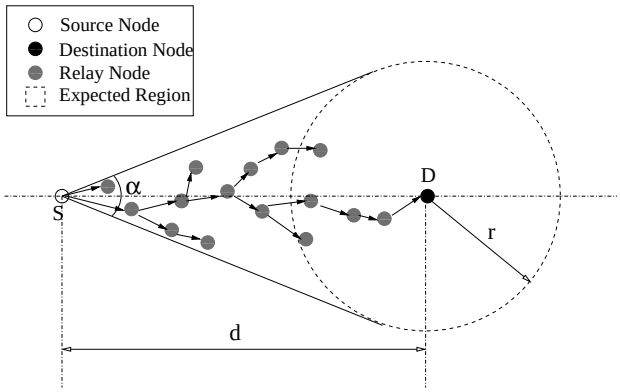


Figure 3.1: Illustration of the expected region in the DREAM Algorithm.

hop. This approach does not need the location information of the nodes in the network and eliminates the power and bandwidth overhead for exchanging location information. However, this algorithm has a serious scalability problem. Since each packet passes through all nodes in the network, the communication complexity of the blind flooding algorithm is $O(n)$, where n is the number of nodes in the network. In this chapter, we say two mobile nodes are *connected* if and only if their distance is less than their transmission radius. Thus there exists a *connected path* between a source node and a destination node if any two adjacent nodes in the path are connected. Although the blind flooding does not require the location information, the high communication complexity prohibits its using in a large ad hoc network.

An improved flooding-based algorithm, *Distance Routing Effect Algorithm For Mobility* (DREAM), was proposed in [27]. In the DREAM algorithm, instead of flooding a packet to the entire network, a source node S floods a packet only to a restricted area which is determined by the *expected region* of the corresponding destination node D. As shown in Figure 3.1, the circle around node D is its expected region whose radius can be represented by $r = V_{max}(t_1 - t_0)$, where V_{max} is the maximum speed of node D, t_1 is the current

time, and t_0 is the time-stamp of node D's location information maintained by S. Since node D locates in the center of its expected region at t_0, it is unlikely to move out of its expected region at t_1, even if it always moves at its maximum speed. Thus in order to cover the expected region of the destination node, the flooding can be restricted only within the angle α, as shown in Figure 3.1, which is defined by the angle between two tangent lines of the expected region of node D. Compared to the blind flooding algorithm, the DREAM algorithm does not need to flood each packet to all connected nodes. However, though only part of nodes in the network participate in flooding in the DREAM algorithm, as analyzed in [25], the communication complexity of the DREAM algorithm is still $O(n)$. This is because that in the worst case when the distance between the source and the destination node equals to the diameter of the network, each packet covers a sector of the entire network. Furthermore, when the destination node moves too slowly and the distance d between the source and the destination node is too far, the flooding must be constrained in a very small angle α, where $\alpha = 2\sin^{-1}(\frac{(t_1-t_0)V_{max}}{d})$. If all flooding nodes are restricted in such a very narrow area, it is difficult to find a connected path between the source and the destination node. In an extreme case that the destination node is static, the expected region of the destination node becomes a point. Therefore, the packet must be forwarded along the straight line between the source and the destination node. If not all nodes on this line are connected, the DREAM algorithm will fail. Compared to the blind flooding, the DREAM algorithm avoids covering the area outside of the expected region of the destination node, therefore saves a significant amount of power and bandwidth. However, the $O(n)$ communication complexity of the DREAM algorithm affects its scalability in large networks. As can be seen in Section 3.3, our newly proposed algorithm eliminates a lot of redundancy of the DREAM algorithm while achieving comparable performance by flooding packets in each

limited sub-area.

3.1.2 Greedy Packet Forwarding Algorithms

A group of more efficient algorithms are *greedy packet forwarding* algorithms [39, 40, 41, 31]. In a greedy packet forwarding algorithm, there is only one routing path between the source node and the destination node, if it exists. Any node that is not contained in the routing path will not participate in forwarding packets. When a packet was forwarded to a node in a routing path at t_1, it can find the next-hop node among its one-hop neighbors by using some strategies, for example, *Most Forward Within R*(MFR) [39], *Nearest with Forward Progress* (NFP) [40] and *Compass Routing* [41]. However, based on the above forwarding strategies, sometimes the greedy packet forwarding algorithm may fail if the current node cannot find the next-hop node in the forwarding direction to the destination node among its neighbors. For instance, in the MFR protocol, if the sender cannot find a node that is closer to the destination node than itself within its transmission range, it will fail. To solve this problem, researchers have proposed some recovery mechanisms, such as the perimeter routing strategy of the *Greedy Perimeter Stateless Routing protocol* (GPSR) [31]. In these algorithms, a packet will be transferred into the recovery mode, when it fails to find a "better" node than itself. Because the communication complexity of forwarding a packet from one node to the other is a linear function of their distance, the communication complexity of the greedy packet forwarding is $O(\sqrt{n})$. It should be pointed out that all algorithms discussed above assume the location information is always accurate. However, since the location information is usually updated periodically, all routing decisions have to be based on the most recent updated location information rather than the current location information. When one node wants to forward a packet to the other

node, it thought the other node is still within its transmission range based on the old location information. But the other node may move too far away to be reached after the last location information update. Due to the out-of-date location information and the motion of nodes between two consecutive location information updates, even robust single-path algorithms, such as GPSR [31], may fail to forward the packet successfully. Thus, we say that greedy packet forwarding algorithms are more sensitive to the motion of nodes than flooding-based algorithm. As can be seen from the analysis in Section 3.2 and the simulation results in Section 3.4, a single routing path has higher failure probability when nodes move intensively. Thus, greedy packet forwarding algorithms are only suitable to static or low mobility ad hoc networks.

Some researchers try to add some redundancy to enhance the greedy forwarding algorithm. The approach is called *multi-path routing algorithm* [32], in which the source sends several copies of each packet through several separate paths at the same time. The multi-path forwarding algorithm can perform much better than the single path algorithm. The successful packet delivery probability of a multi-path forwarding algorithm $P_{mul} = \sum_k P_k$, where k is the number of multiple paths and P_k is the successful packet delivery probability of the k_{th} path. The equation above can be satisfied only when any two of paths have no dependence. For example, in the case that two paths overlap in some nodes, if any overlapping node dies or moves away, both paths will fail. In order to minimize dependence between any two paths, the source may need to acquire accurate information of the global node distribution to decide several independent paths, whereas single path algorithm only needs the location information of nearby nodes. The overhead of exchanging the global information leads to the tremendous energy and time consumption in large ad hoc networks. As can be seen in section 3.3, our new algorithm is a kind of "local" algorithm like single

path algorithm and improve the performance of the single path algorithm by flooding packets to the nearby area of each forwarding node in the single routing path instead forwarding packets in an arbitrary direction.

3.2 Path Connectivity Analysis for Greedy Packet Forwarding Algorithms

Before we present our new algorithm, in this section we first analyze the error probability of a routing path caused by the outdated location information. Consider an *n*-node ad hoc network, where all nodes are located in a two dimensional square and have the same transmission radius *R*. Each node is able to communicate with its neighbors at most *R* units away from it. In most position-based routing algorithms, each node broadcasts control packets to update its location information maintained by other nodes in every short period T. The motion trace of a node is recorded by a series of discrete positions which are updated periodically. Since all forwarding decisions are made at these discrete points, it is not necessary to acquire the actual motion curve between any two adjacent points. Moreover, during each short period between two consecutive location information updates, it is very likely for nodes to keep the same direction and the same speed or only change them slightly. For an easy analysis, we assume that each node keeps the same speed and motion direction between two consecutive location information updates. During each location update period, the speed and the motion direction of each node are given randomly according to the uniform distribution in $(0, V_{max})$ and $(0, 2\pi)$, respectively, where V_{max} is the maximum speed of the node. Thus the motion curve of a node can be described by a chain-like curve instead of its actual curve. Also, since the signal transmission time and

the computing time of each node has little impact on the performance and the cost of the algorithm compared to the location information update period, we simplify the theoretical analysis and simulations by ignoring these short times.

In an ad hoc network, a routing decision is made based on the most recently updated location information. However, all mobile nodes move from time to time, and as a result, two connected nodes may become disconnected after a short while. There is no guarantee that the packet coming at time t_1 still can pass through the routing path based on the location information at time t_0 (where $t_0 + T > t_1 > t_0$). We define the probability that two connected nodes become disconnected after Δt as q, where $\Delta t = t_1 - t_0$. Assume that there are a total of m hops from the source node to the destination node and the connection status of any two nodes is independent. Then the probability that a routing path determined at time t_0 is still connected at time t_1, P_{path}, can be calculated by

$$P_{path} = (1-q)^m \tag{3.1}$$

In the following, we calculate the probability q that two connected nodes become disconnected after Δt. Figure 3.2(a) gives two connected nodes N_1 and N_2 which have absolute velocities $\overrightarrow{V_1}$ and $\overrightarrow{V_2}$, respectively. Let ϕ_1 and ϕ_2 be the motion directions of nodes N_1 and N_2, respectively. To see the upper bound of the error probability, we assume a worst case that both node N_1 and node N_2 move at the maximum speed, that is, $|\overrightarrow{V_1}| = |\overrightarrow{V_2}| = V_{max}$. Without loss of generality, we choose node N_1 as the motion reference object and $\phi_1 = 0$. In this relative motion system, as shown is Figure 3.2(c), node N_1 becomes the origin of the polar coordinate, and node N_2 is located at (r, θ), where (r, θ) is the position of node N_2 in the polar coordinate. As shown in Figure 3.2(b), now node N_1 has velocity $\overrightarrow{V_1'} = 0$ and node N_2 has velocity $\overrightarrow{V_2'} = (\overrightarrow{V_2} - \overrightarrow{V_1})$, where the motion direction of $\overrightarrow{V_2'}$, $\phi_2' = \angle\overrightarrow{V_2'} = \frac{\phi_2 - \phi_1 + \pi}{2} =$

$\frac{\phi_2+\pi}{2}$. We assume that ϕ_1 and ϕ_2 can take any direction within 0 to 2π with the same probability. Thus, ϕ_2' is uniformly distributed in $(\frac{\pi}{2}, \frac{3\pi}{2})$, and the probability mass function of ϕ_2', $f(\phi_2') = \frac{1}{\pi}$. In Figure 3.2(c), S is defined as the distance from point (r,θ) to the edge of the circle in the direction of ϕ_2', where $S = \sqrt{R^2 - r^2\sin^2(\phi_2' - \theta)} - r\cos(\phi_2' - \theta)$. When $\theta = \phi_2'$, S reaches its minimum value, $S_{min} = (R - r)$. In the relative motion system, if the distance of node N_2 moving in the direction of ϕ_2' during t_0 and t_1 is longer than S, that is, when

$$|\overrightarrow{V_2'}|(t_1 - t_0) > S \tag{3.2}$$

node N_2 will move out of the transmission range of node N_1. Thus, the probability that the link between nodes N_1 and N_2 becomes disconnected at t_1,

$$q = P_r\{|\overrightarrow{V_2'}|(t_1 - t_0) > S\} \tag{3.3}$$

Denoting $|\overrightarrow{V_2'}|(t_1 - t_0)$ in (3.3) as C, we have

$$\begin{aligned} q &= P_r\{C > S\} \le P_r\{C > S_{min}\} \\ &\le P_r\{C > R - r\} = P_r\{r > R - C\} \\ &= \int_{\frac{\pi}{2}}^{\frac{3\pi}{2}} \int_0^{2\pi} \int_{R-C}^{R} f(r,\theta,\phi_2')\, dr\, d\theta\, d\phi_2' \end{aligned} \tag{3.4}$$

We assume mobile nodes are distributed uniformly in the transmission range of node N_1 and the relative motion direction of a node does not depend on its location. Thus,

$$f(r,\theta,\phi_2') = f(r,\theta)\, f(\phi_2') = \frac{r}{\pi^2 R^2} \tag{3.5}$$

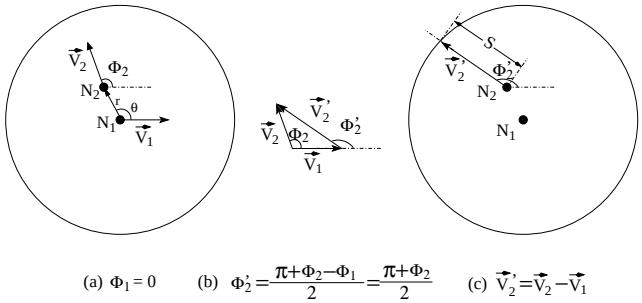


Figure 3.2: The motion models of two nodes. (a) Absolute motion model. (b) Velocity composition. (c) Relative motion model.

Plugging (3.5) into (3.4), we obtain

$$\begin{aligned} q &\leq \int_{\frac{\pi}{2}}^{\frac{3\pi}{2}} \int_0^{2\pi} \int_{R-C}^{R} \frac{r}{\pi^2 R^2} dr d\theta d\phi_2' \\ &\leq \frac{1}{2\pi^2 R^2} \int_{\frac{\pi}{2}}^{\frac{3\pi}{2}} \int_0^{2\pi} 2RC - C^2 d\theta d\phi_2' \\ &\leq \frac{1}{2\pi^2 R^2} \int_{\frac{\pi}{2}}^{\frac{3\pi}{2}} \int_0^{2\pi} 2RC d\theta d\phi_2' \\ &\leq \frac{2C}{R} \end{aligned} \tag{3.6}$$

Replacing $C = |\overrightarrow{V_2'}|(t_1 - t_0)$ in (3.6), we have

$$\begin{aligned} q &\leq \frac{2|\overrightarrow{V_2'}|(t_1 - t_0)}{R} \\ &\leq \frac{2(|\overrightarrow{V_1}| + |\overrightarrow{V_2}|) \times (t_1 - t_0)}{R} \\ &\leq \frac{4V_{max} \times T}{R} \end{aligned} \tag{3.7}$$

Where T is the location information update period and V_{max} is the maximum speed of a

node. When $V_{max} \cdot T \ll R$, $q \to 0$ and $P_{path} = (1-q)^m \to 1$. This means that the routing path based on the location information at t_0 is still connected at t_1 with probability $P_{path} \to 1$. On the other hand, when $\frac{4V_{max} \times T}{R}$ increases, the single routing path between the source node and the destination node becomes increasingly instable. A possible solution to this problem is that after sending a packet to its next-hop node, each node keeps a copy of the packet in the memory until it receives an acknowledgment from the next-hop node. If the next-hop node moves out of the transmission range of the current node, it cannot receive the packet and reply to the current node. If the current node does not receive the acknowledgment within a certain amount of time, it will wait for the next location information update and then look for a new next-hop node based on the new location information. After the current node makes a new forwarding decision, it will re-forward the packet to the new next-hop node. Although this approach can reduce the error probability of a greedy forwarding algorithm, these handshaking messages introduce extra traffic burden to a resource-limited network. Moreover, waiting for the new location information update delays the transmission of packets.

3.3 Single Path Flooding Chain Routing Algorithm

As discussed earlier, flooding-based algorithms are more robust than single-path-based routing algorithms. However, they usually have $O(n)$ communication complexity for an n-node ad hoc network, which makes the consumption of energy and bandwidth heavily depend on the number of nodes and affects the scalability of the network. In this section, we propose a new routing algorithm that compromises between flooding-based algorithms and greedy forwarding algorithms. While keeping the low communication complexity, the new algorithm enhances the robustness of single-path-based routing algorithms by flooding

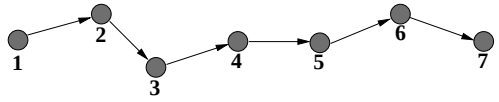
the packet only within a limited area near two adjacent nodes in a single routing path.

3.3.1 Sub-area Flooding

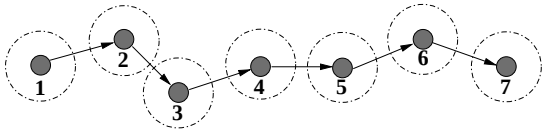
A packet forwarding decision is usually based on the location information updated earlier than the current time. Thus the next-hop node may move out of the transmission range of the current node and cannot receive the packet. If each node in the single routing path floods the packet within a limited area rather than forwards the packet in an arbitrary direction, it is highly possible for the packet to re-capture the "escaped" next-hop node in constant steps with the help of *relay nodes*. Note that the communication complexity of this approach is $O(\sqrt{n})$, since the flooding only occurs in the nearby nodes of each forwarding node.

Figure 3.3(a) shows a single routing path along nodes $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$, which is determined by the location information updated at t_0. However, when a packet needs to pass through this path from node 1 to node 7 at t_1, the routing path may become disconnected because of the movement of nodes during t_0 and t_1. For example, in Figure 3.3(b), node 5 moves out of the transmission range of node 4 at t_1. The link between node 4 and node 5 becomes disconnected. As a result, the packet can no longer reach the destination node 7 along this path.

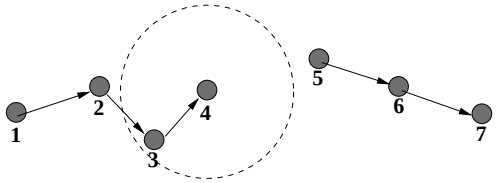
In Table 3.1, we give a new position-based algorithm, called *single-path flooding chain algorithm*. Instead of forwarding the packet along a single path, we can choose the "limited flooding area" for each forwarding node in the single path to flood the packet. At the current time t_1, just like the single path routing algorithms, the current forwarding node decides the next-hop node in the single routing path based on the location information updated at t_0. And then it calculates the expected region of its next-hop node, which is the



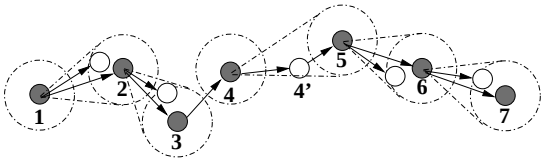
(a) Greedy Packet Forwarding Algorithm:
A single path from 1 to 7 determined by the most recent location information updated at t_0 .



(b) Circles around nodes are their expected regions.



(c) Node 5 runs out of the transmission range of node 4 at t_1
Single routing path becomes disconnected.



(d) Single Path Flooding Chain Algorithm:
With the relay of node 4', the packet re−capture node 5.

Figure 3.3: Examples of greedy packet forwarding and single-path flooding chain algorithms.

circle around the location of the next-hop node at t_0 with the radius of $r = V_{max}(t_1 - t_0)$. Figure 3.3(b) shows the expected region of forwarding nodes $1, 2, 3, 4, 5, 6$ and 7. Each sender in the single routing path tries to cover the expected region of the next-hop node by simply flooding the packet to the nodes within two tangent lines of the expected region of the next-hop node. In Figure 3.3(d), node 4 can re-capture node 5 through the relay of node $4'$.

Table 3.1: Single path flooding chain algorithm

Algorithm : Single Path Flooding Chain
for each packet starts from the source node at t_1
  current node := source node;
    while current node $\neq$ destination node
      current node finds the next-hop node based on the location information updated at t_0;
      current node calculates the expected region of the next-hop node;
      current node floods the packet to cover the expected region of the next-hop node;
      current node := next-hop node;
    end while;
end for
End

3.3.2 Communication Complexity

In this subsection, we analyze the communication complexity of the proposed single path flooding chain algorithm. Intuitively, the higher density of an ad hoc network, the more connectivity of the network. Simulation results in [42] showed that six to eight neighbors can make a small size network connected with high probability. Thus in a high density network where each node has more than eight neighbors, routing algorithms may not have a great impact on the performance. Therefore, in order to make a fair comparison on the performance and the complexity of the routing algorithms, we assume that the network has a general density, in which each node has a small constant number of neighbors within its transmission range in average.

In the single path flooding chain algorithm, the packets flooded by a node will cover an area to reach the next-hop node. We call this area *Covered Area*. The covered area has two possible shapes. As shown in Figure 3.4(a), when the distance d between the current and the next-hop node is less than the radius r of the expected region (where $r = V_{max}(t_1 - t_0)$),

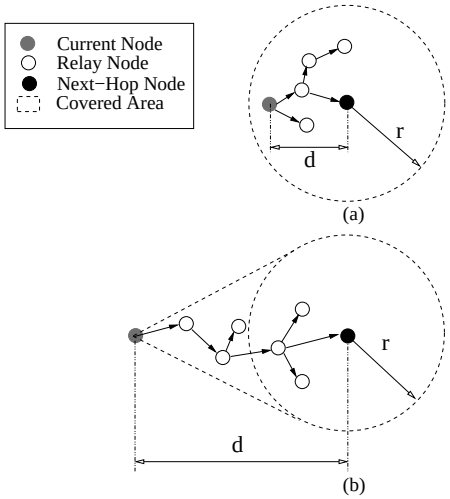


Figure 3.4: Covered area of the next hop node. (a) $d < r$. (b) $d > r$.

the covered area is bounded by two tangent lines and the arc between two tangent points, which includes the center of the circle. Figure 3.4(b) is the other case, which shows that when d is longer than r, the covered area is a circle centered at the position of the next-hop node. In both cases, d is less than R, because two adjacent nodes in the single routing path must be in the transmission ranges of each other. r is in fact a measure for the node mobility during the short period between two location information updates. Few existing position-based routing algorithms can perform well in a network with a very high mobility ($r >> R$) except some algorithms which use the flooding strategy, such as the DREAM and the single path flooding chain algorithms. As will be seen in Section 3.4, the greedy packet forwarding algorithm starts to have very poor performance after $\frac{r}{R} > 0.5$. In fact, when r is much larger than R and is similar to the network diameter, the expected region of a node in the DREAM or the single path flooding chain algorithm may cover most nodes in the network. In this case, there is little difference among the DREAM, the single path flooding chain and the blinding flooding algorithms in communication complexity. In order

to compare three position-based routing algorithms in a network with a general mobility, we assume that the radius r of the expected region is at most a constant times the transmission radius R. Thus, both r and d can be expressed as a constant times R. Since the size of the covered area is a quadratic function of r and d, it can be represented as $(k \cdot R^2)$, where k is a constant. For given density of nodes, say ρ, each covered area contains a constant number of nodes, which equals $k \cdot \rho \cdot R^2$.

In the algorithm, before the current node floods a packet to the next-hop node, it puts the maximum speed and the location information of the next-hop node into the packet header. When a node receives the packet, it calculates the covered area based on the information in the packet header and its own location, and then checks if it is in the covered area with respect to the packet. If yes, it will act as a "relay node" and floods this packet to its one-hop neighbors in the covered area. Otherwise, the packet will not be forwarded to other nodes. This strategy guarantees that only a constant number of nodes contained in each covered area participate in relaying packets between two adjacent forwarding nodes in a single path. Thus, the complexity of the newly proposed single-path flooding chain algorithm is only a constant factor higher than that of a single-path-based routing algorithm and is still $O(\sqrt{n})$.

3.4 Simulation Results

We have simulated the greedy packet forwarding, DREAM and the single-path flooding chain algorithms and compare their performance and complexities. In the simulations, we use the same node motion model as discussed in Section 3.2. n nodes are placed randomly in a $1 \times 1km^2$ square. Each node has the same transmission range $R = 100m$. Each time the location information is updated, the speed and the direction of each node will change according to the uniform distribution in $(0, V_{max})$ and $(0, 2\pi)$, respectively, where V_{max} is

the maximum speed of the node. In both greedy forwarding algorithm and single-path flooding chain algorithm, without loss of generality, MFR [39] strategy is used to decide the next-hop node in the single routing path. In order to analyze the effect of the node mobility, we introduce a metric $\beta = \frac{V_{max}\Delta t}{R}$ to measure the mobility of a mobile node, where Δt is the time between the packet arrival and the last location information update, and R is the transmission radius of the node. We first consider the effect of the node mobility to the *successful packet delivery ratio*, which represents the probability that a packet is sent successfully from a source node to a destination node. We also compare the communication complexities of the algorithms by varying the network size.

3.4.1 Successful Packet Delivery Ratio

In this scenario, we compare the successful packet delivery ratios of three algorithms in a network with 400 mobile nodes by ranging β from 0 to 1. For each β, we generate 100 unicast packets between randomly chosen source-destination pairs. Figure 3.5 shows the relationship between the successful packet delivery ratio P_d and the node mobility metric β. Three curves correspond to the greedy packet forwarding, DREAM and single-path flooding chain algorithms, respectively. As shown in Figure 3.5, when nodes have very low mobility ($\beta < 0.1$), the P_d of the greedy packet forwarding algorithm is greater than 50%. Since the radius of the expected region is proportional to $V_{max}\Delta t$ in the DREAM algorithm, when $\beta < 0.1$, the P_d of the DREAM algorithm is less than 50%. When β increases, the P_d of the greedy packet forwarding algorithm becomes too low to be acceptable. On the other hand, the successful packet delivery ratio of the DREAM algorithm ascends as β increases and exceeds the single-path flooding chain algorithm slightly when the network has high mobility ($\beta > 0.6$). We can see that unlike other two algorithms, the single-path flooding

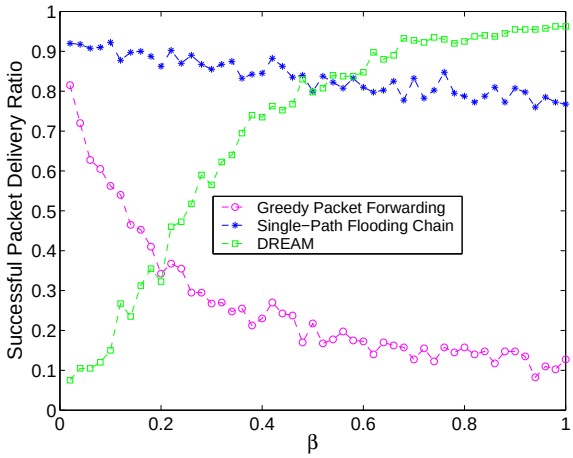


Figure 3.5: The successful packet delivery ratio between a randomly chosen source-destination pair.

chain algorithm is not sensitive to the change of the node's speed and can keep a stable high successful packet delivery ratio ($P_d > 75\%$) in both low mobility and high mobility ad hoc networks.

3.4.2 Communication Complexity

From Figure 3.5, we observe that the DREAM algorithm has slightly better successful packet delivery ratio than the single-path flooding chain algorithm when nodes have high mobilities ($\beta > 0.6$). We now compare the communication complexity to see the tradeoff between the DREAM algorithm and the single-path flooding chain algorithm. We analyze the communication complexity by varying the network size from 100 nodes to 1000 nodes, for $\beta = 0.6, 0, 7, 0.8$ and 0.9, respectively. We can see from Figure 3.6 that the successful packet delivery ratios of three algorithms all have an increasing trend when the network size increases from 100 to 1000. In general, the higher network density is, the more connectivity of the network has. However, we notice that the difference in the communication

complexity between the DREAM algorithm and the single-path flooding chain algorithm becomes larger as the network size increases. Besides the network size, the mobility of the nodes also affects the communication complexity. The faster the node's speed is, the larger the expected region and the higher communication complexity. From Figure 3.6, we also observe that the DREAM algorithm has higher communication complexity than the single-path flooding chain algorithm when $\beta = 0.9$ than that of $\beta = 0.6, 0,7$ and $0,8$. Thus, the single-path flooding chain algorithm is more scalable than the DREAM algorithm for large size and high mobility ad hoc networks and always has a better successful packet delivery ratio than the greedy packet forwarding algorithm.

3.5 Conclusions

In this chapter, we have presented a new position-based routing algorithm called single-path flooding chain algorithm for mobile ad hoc networks. Compared to flooding-based routing algorithms with $O(n)$ communication complexity, the newly proposed algorithm reduces the communication complexity to $O(\sqrt{n})$, which is as low as the greedy packet forwarding algorithm. The new algorithm can significantly save the bandwidth and power for resource limited mobile nodes, especially in large networks. In addition, simulation results have showed that single-path flooding chain algorithm consistently performs well for various mobilities and keeps a high successful packet delivery ratio $(> 75\%)$, which is insensitive to the change of node's motion speed.

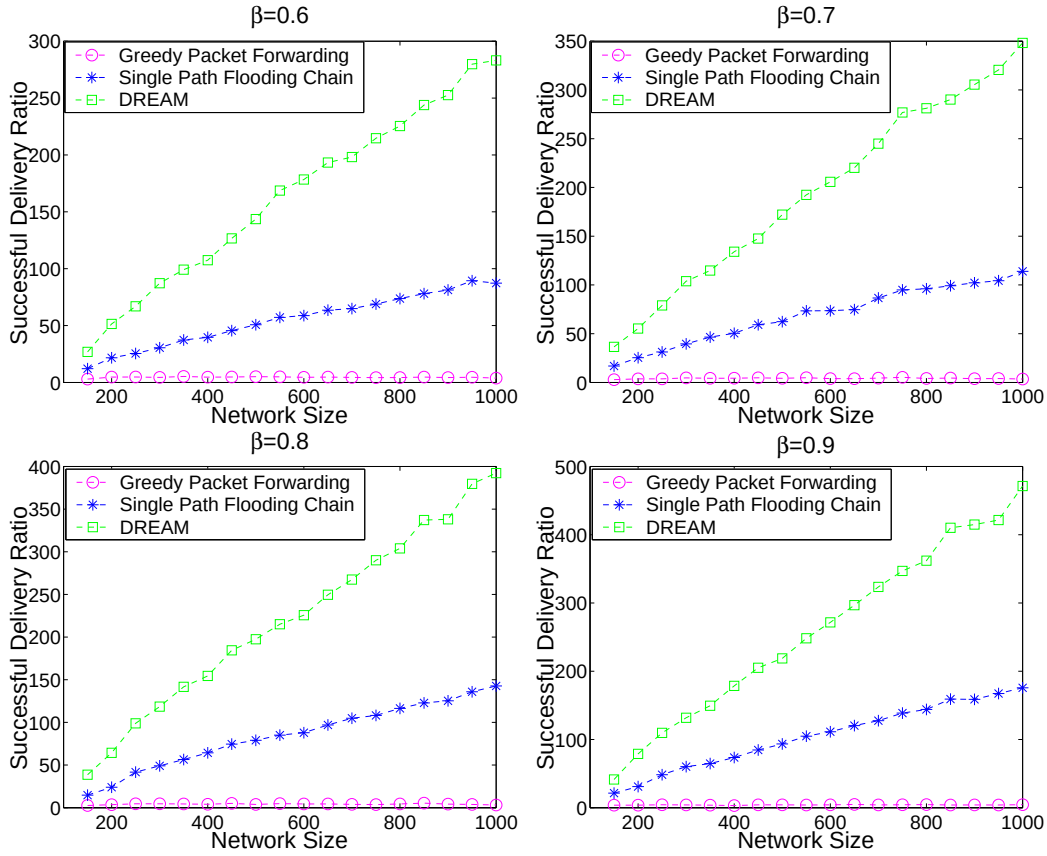


Figure 3.6: Communication complexity when $\beta = 0.6, 0.7, 0.8$ and 0.9, respectively.

Chapter 4

Channel Reservation MAC Protocol

In chapter 3, single path flooding chain routing algorithm is designed to provide reliable end-to-end communication at network layer for mobile ad-hoc/sensor networks. This chapter focuses on MAC layer and presents a novel contention-based MAC protocol, called the *Channel Reservation MAC protocol*, by introducing a reservation mechanism. As will be seen, the proposed protocol achieves much better throughput and fairness and shorter packet delay than the IEEE 802.11 protocol.

The rest of the chapter is organized as follows. In Section 4.1, we review some technique details of the IEEE 802.11 MAC protocol. Section 4.2 gives a brief overview of the existing work. In Section 4.3, we present the new Channel Reservation MAC protocol. Section 4.4 gives theoretical analyses on the saturated throughput of the protocol under the ideal channel condition. In Section 4.5, we discuss some related practical issues. Section 6.4 contains simulation results, and finally, Section 6.5 concludes the chapter.

4.1 IEEE 802.11 MAC Protocol

Wireless local area networks (WLANs) have achieved a tremendous amount of growth in recent years. Advanced signal processing and RF technologies accelerate the development and applications of WLANs. IEEE 802.11-based WLANs [43], [44], [45], [46] have emerged as a prevailing technology for the broadband wireless access in both enterprises and homes. IEEE 802.11 is a family of standards for medium access control (MAC) and physical layer (PHY) specifications to provide multiple accesses for a shared channel. The most popular IEEE 802.11b standard operates on the unlicensed 2.4GHZ radio spectrum and supports data rate up to 11$Mbps$. On PHY layer, it uses either *direct sequence spread spectrum (DSSS)*, *frequency-hopping spread spectrum (FHSS)*, or *infrared* (IR) for modulation. Moreover, the newer IEEE 802.11a and IEEE 802.11g provide higher data rate up to 54$Mbps$. IEEE 802.11 MAC protocol is designed based on the *Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA)* protocol [43], [45].An IEEE 802.11-based WLAN can be configured in either an infrastructure mode or an ad-hoc mode. Most WLANs in small-scale homes or offices are usually deployed in the infrastructure mode and use a central controller to allocate the channel resource. On the other hand, the ad-hoc mode can provide more flexible network access than the infrastructure mode without the central control due to its peer-to-peer nature. In recent years, the rapid growth of large-scale wireless sensor networking applications has boosted the development of ad-hoc MAC protocols. The MAC protocol primarily determines the performance of a WLAN. Although IEEE 802.11 MAC protocol is adequate for the basic connectivity and applications, its performance still needs to be improved to meet the rapidly increasing demand for more advanced broadband technology.

Before we present our new protocol, in this section we first review some technique

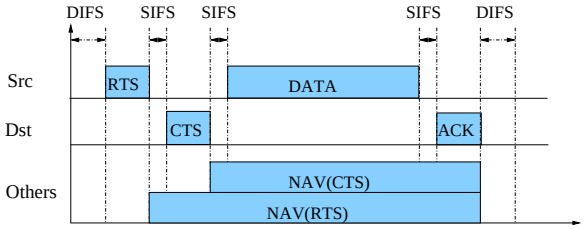


Figure 4.1: IEEE 802.11 RTS/CTS mode.

details of IEEE 802.11 MAC protocol. IEEE 802.11 MAC contains two coordination functions, namely, *Point Coordination Function (PCF)* and *Distributed Coordination Function (DCF)* [43], [44], [45], [46] which support the infrastructure configuration and the ad-hoc configuration, respectively. PCF depends on a central controller to allocate the channel resource and provide contention-free services, while DCF is a typical contention-based protocol. PCF is built on the top of DCF and performs more efficiently than DCF. However, PCF requires the presence of a point coordinator and can only be used in the infrastructure configuration. On the other hand, DCF belongs to the class of CSMA/CA protocols. Similar to other CSMA/CA protocols, DCF protocol provides a mechanism for collision avoidance by using the carrier sense and the exponential back-off algorithm. In some networks, such as large scale sensor networks, due to lack of powerful central controllers, contention-based distributed channel access protocols are more preferred. Next we describe some specific operations of IEEE 802.11 DCF protocol.

DCF defines two channel access modes: basic mode and RTS/CTS access mode. Both modes adopt the discrete-time back-off algorithm. Whenever a node has a packet to transmit, it generates a random *back-off counter* uniformly from $[0, CW-1]$, where CW represents the size of the *contention window*. As long as the carrier is sensed idle for a period of *Distributed Inter Frame Space (DIFS)*, the node begins to decrement its back-off counter

by one. After that, the back-off counter is reduced by one for each idle slot. If the carrier is busy, the back-off counter is frozen until the next idle DIFS is sensed. When the back-off counter is reduced to zero, the node begins the transmission. The minimum and maximum values of CW, CW_{min} and CW_{max}, are determined by the physics layer characteristics. For example, CW_{min} and CW_{max} are set to 16 and 1024, respectively for FHSS physical layer in IEEE 802.11b standard. After each successful transmission, CW is reset to its minimum value CW_{min}. After each unsuccessful transmission, the value of the back-off stage m will be increased by 1 and CW will be doubled until it reaches its maximum value CW_{max}, that is, $CW = \max\{2^m CW_{min}, CW_{max}\}$. The difference between the basic mode and the RTS/CTS mode is that the basic mode follows a 2-way handshaking mechanism. Each successful transmission of the data frame is confirmed by a positive *acknowledgement (ACK)* message sent by the destination, whereas the RTS/CTS mode uses a 4-way handshaking (RTS-CTS-DATA-ACK) instead of the 2-way handshaking in the basic mode. As shown in Fig. 4.1, two short conversation frames, *Request to Send (RTS)* and *Clear to Send (CTS)*, are sent prior to the transmission of the data frame. When the source node is ready to send a packet, it sends a RTS message instead of the data frame. The RTS message contains the expected duration information for the remaining transmission, as shown in Fig. 4.1, which includes three *Short Inter Frame Space (SIFS)* intervals, one CTS message, one data frame and one ACK message. If no collision occurs, the destination node replies with a CTS message after a SIFS, which also contains the new expected duration information of the remaining transmission. Other nodes receiving the RTS and CTS messages update their *Network Allocation Vector (NAV)* based on the duration information in RTS and CTS messages. NAV is maintained by each node to indicate when the channel is available and is updated by RTS and CTS messages. All other nodes know when the current transmission will complete

according to the updated NAV and avoid transmitting packets during this period. Since the length of the RTS/CTS message is usually much shorter than that of the data frame, even if the collision of several RTS messages occurs, it still can reduce the collision overhead compared to the collision of long data frames.

It should be pointed out that although the RTS/CTS mode has much merit compared to the basic mode, its performance is still very sensitive to the number of active nodes in the network. In recent years, many researchers have attempted to improve the performance of contention-based MAC protocols in WLANs. We will give a brief overview of the existing work in the next section.

4.2 Related Work

Bianchi [48] analyzed the saturated throughput by using the Markov chain model and the study revealed that the saturated throughput of IEEE 802.11 DCF decreases as the number of nodes increases. In DCF, the contention window (CW) size is an important parameter to govern the network throughput and energy consumption. When there are only a few active nodes in the network, a small CW can reduce the average idle time of the channel and increase the channel utilization. When there are a large number of nodes that have packets to transmit, the collision probability can be decreased by using a large CW. In IEEE 802.11 DCF, the CW size is simply doubled for each packet collision. If the number of active nodes can be obtained, the CW size can be adjusted adaptively. Based on the analytical results in [48], Bianchi [49] proposed a method to estimate the number of competing nodes by using the Kalman filter. The dynamic tuning algorithm in [50] provides a method to tune the contention window to the optimal size dynamically based on

the estimated number of active nodes. Unlike the IEEE 802.11 DCF protocol, the back-off counter in this protocol is randomly generated following the geometric distribution, instead of the exponential distribution. Fast Collision Resolution (FCR) algorithm [51] can speed up the collision resolution by actively redistributing the back-off counters. It uses a smaller contention window for the nodes that just complete the transmission successfully and reduces the back-off counter exponentially when a fixed number of consecutive idle slots are sensed. A gentle CW adjustment strategy GDCF, proposed in [52], improved the throughput and fairness of the DCF by decreasing the CW size gently rather than resets it to CW_{min} after each successful transmission. DCF^{+} algorithm [53] improved the throughput for TCP traffic by combining the TCP ACK and the MAC ACK messages. However, the improvement of the DCF^{+} algorithm is only limited to TCP traffic.

Although some of the above algorithms allow nodes to adjust their back-off counters flexibly rather than run the exponential back-off algorithm passively, the adjustment strategy is still based on the past channel status. Thus, in the standard DCF algorithm or any of the above adaptive algorithms, when a node reduces its back-off counter to zero and prepares to transmit a packet, it does not know if other nodes also have packets to send in the same time slot. If a node can reserve the channel by informing others when it plans to transmit the next packet before the next transmission, other nodes may avoid colliding with the packet. Accordingly, the collision can be avoided or reduced. In the infrastructure mode, the central controller can easily allocates the channel resource without collision. However, if a network is configured in the ad-hoc mode, the nodes can hardly reserve the channel without the central control. Furthermore, it is difficult for nodes to exchange the reservation information without sending extra packets.

4.3 A New MAC Protocol with Channel Reservation

In this section, we present a novel MAC protocol with channel reservation mechanism, called Channel Reservation MAC (or CR-MAC for short) protocol. We first propose a mechanism for exchanging the reservation information by utilizing the overhearing feature of the shared wireless channel. And then we describe how nodes reserve the channel by using these control messages.

4.3.1 Exchanging Reservation Information by Utilizing the Overhearing Feature

In a contention-based algorithm, each node needs to overhear all traffic sent by nearby nodes within its transmission range. Control messages, such as RTS, CTS and ACK messages specified in IEEE 802.11, are received by all neighbors within the transmission range of the sender, and then discarded by non-destination nodes. The overhearing feature of the contention-based algorithms can be used to exchange information between nodes without introducing too much overhead. For instance, both RTS and CTS messages in IEEE 802.11 contain the remaining duration of the current transmission. Other active nodes update their NAVs based on the duration in RTS and CTS messages, and they will not try to transmit packets until the current transmission is completed. Besides the duration of the current transmission, if each node can inform other nodes more information about the next transmission during the current transmission, other nodes would try to avoid colliding with the next transmission of the node. A novel idea of our protocol is to include the reservation information for the next transmission into the control messages of the current transmission instead of sending an extra control packet. Thus, the reservation information can be heard

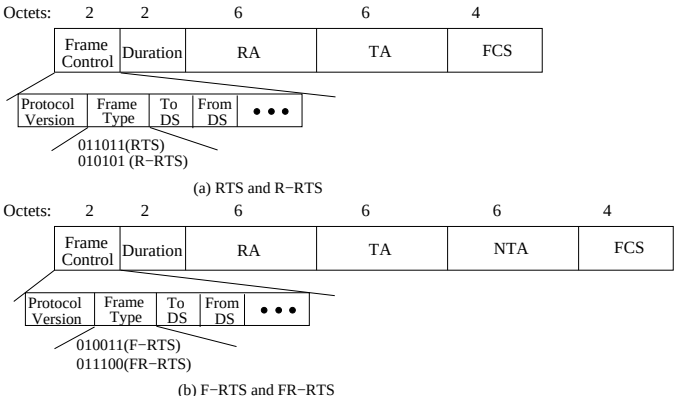


Figure 4.2: RTS, R-RTS, F-RTS and FR-RTS frame formats.

by all nodes within the transmission range of the sender with little extra overhead.

Before we explain the details of our new protocol, we introduce three additional control messages, termed *Reservation RTS (R-RTS)*, *Forwarding RTS (F-RTS)* and *Forwarding-Reservation RTS (FR-RTS)* messages, respectively. The frame formats of RTS, R-RTS, F-RTS and FR-RTS are given in Fig.4.2. As shown in Fig.4.2, R-RTS has the same length and fields as the basic RTS message, including *Frame Control*, *Duration*, *Receiver Address (RA)*, *Transmitter Address (TA)* and *Frame Check Sequence (FCS)*. Besides all the above fields in the basic RTS, F-RTS and FR-RTS also contain a 6-byte field called *Next Transmitter Address (NTA)*. Each Frame Control field consists of a 6-bit binary code indicating the type of the frame, for example, RTS(011011), CTS(011100) and Data(100000). In order to differentiate new messages from the existing ones, we put a different code in the control field for each new frame type. Since the codes from 010000 to 011001 are reserved in IEEE 802.11, we can use some of them for the new messages without conflicting with the existing ones. We choose 010101, 010011 and 011100 for R-RTS, F-RTS and FR-RTS messages, respectively. Besides these three new messages, all other messages are kept same as in IEEE 802.11.

Given the above formats of new control messages, we now see how much extra transmission cost would be needed for exchanging the reservation information. In our new protocol CR-MAC, the sender may send a basic RTS, R-RTS, F-RTS or FR-RTS message in different situations, where F-RTS and FR-RTS are 6 bytes longer than RTS and R-RTS. To see the worst case, we can assume that the sender always begins the transmission with the F-RTS/FR-RTS message. We assume that the transmission cost per bit for different messages is the same. Then the ratio of transmission cost of the new protocol to the standard RTS/CTS mode, $\frac{TransCost_{CR-MAC}}{TransCost_{RTS/CTS}}$, can be calculated as follows:

$$\frac{TransCost_{CR-MAC}}{TransCost_{RTS/CTS}} = \frac{L_{Data} + L_{RTS^*} + L_{CTS}}{L_{Data} + L_{Basic\_RTS} + L_{CTS}} \tag{4.1}$$

where L_{Data}, L_{Basic_RTS}, L_{RTS^*} and L_{CTS} represent the lengths of the data frame, basic RTS message, F-RTS/FR-RTS message, and CTS message, respectively. Let $L_e = L_{RTS^*} - L_{Basic\_RTS}$. Compared to L_{data}, L_e has only 6 bytes in length and is usually much shorter than L_{data}. We obtain

$$\frac{TransCost_{CR-MAC}}{TransCost_{Basic}} = \frac{L_e + L_{Data} + L_{Basic\_RTS} + L_{CTS}}{L_{Data} + L_{Basic\_RTS} + L_{CTS}} \approx 1 \tag{4.2}$$

The above equation indicates that the extra transmission cost introduced by the new protocol is negligible.

4.3.2 Channel Reservation Algorithm

In this subsection, we describe how each node can reserve the channel by using the above control messages. Besides new control messages, in the CR-MAC protocol, each node keeps a reservation waiting list to record the transmission sequence of the nodes

within its transmission range, which contains a series of IDs or addresses of the nodes. In addition, we assume that all the packets wait in the transmission queue before the transmission and are sent following the FIFO queueing discipline. Also, a boolean variable is used to indicate if there are more than one packets waiting in the transmission queue.

In the CR-MAC protocol, each node in the network works alternatively in two periods: *competing period* and *reservation period*. Accordingly, all nodes can be classified into two subsets: *competing nodes* and *sequential transmitting nodes* (or simply *transmitting nodes*). Each competing node runs the exponential back-off algorithm as in the RTS/CTS mode.

As will be explained later, the channel cannot keep idle for longer than a SIFS during the reservation period. Since the back-off counter of each competing node is frozen during the reservation period, it can only attempt to transmit a packet during the competing period. When a competing node obtains the channel and has no more packets waiting in its transmission queue, it begins the transmission with a basic RTS message and completes the transmission in the same way as the standard RTS/CTS mode. If there are already some other packets waiting in its transmission queue before the transmission of the current packet, the competing node informs other nodes by sending a R-RTS message instead of a basic RTS, which has a different field from the basic RTS message. Thus, all nearby nodes that receive the R-RTS message are informed the fact that the current sender still has packets ready to send. Then, they put the address of the sender to the end of their reservation waiting lists, which are used to store the transmission sequence based on the FIFO discipline. After a successful transmission with a R-RTS message, the sender has reserved a transmission slot in the next reservation period and becomes a transmitting node. And then, the node regenerates its back-off counter uniformly from $[0, CW_{Res} - 1]$, instead of the

range $[0, CW_{min} - 1]$ in the standard RTS/CTS mode, where CW_{Res} denotes the contention window of the transmitting node and is usually larger than CW_{min}. Each transmitting node also reduces its back-off counter for each idle slot and attempts to transmit packets when the back-off counter is reduced to zero. One difference between a transmitting node and a competing node is that a transmitting node resets its contention window to CW_{Res} when the collision occurs, instead of doubling the contention window.

Once a transmitting node obtains the channel successfully, the network enters the reservation period. Before a transmitting node transmits a packet, it checks its reservation waiting list. If the list is not empty, the node puts the address of the first node in its reservation waiting list into the NTA field of the F-RTS or FR-RTS message and starts the transmission with the F-RTS or FR-RTS message. All nearby nodes acquire the address of the next sender from the NTA field of the F-RTS or FR-RTS and hence know which node will be the next sender. After the transmission, nearby nodes remove the address of the sender from the top of their waiting lists. During the reservation period, transmitting nodes pass the "transmission token" from one to another. Since only the node designated by the previous sender can send its packet, the collision can be avoided. In addition, because there is only a SIFS between two consecutive transmissions, the channel cannot be sensed idle for a DIFS $(DIFS > SIFS)$ during the reservation period. Therefore, every node has to freeze its back-off counter and stops competing with others until the reservation period ends.

Like competing nodes, transmitting nodes can also reserve the channel by sending a FR-RTS message. If a transmitting node has more than one packets to send when it just obtains the channel, it sends a FR-RTS message to reserve the channel for the next packet in the queue. Nearby nodes then update their reservation waiting lists by putting the address of the current sender to the end of their waiting lists, which is obtained from the TA field

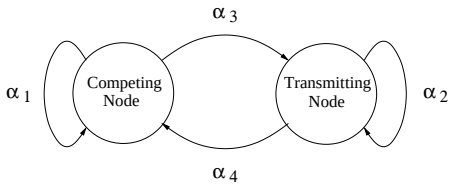
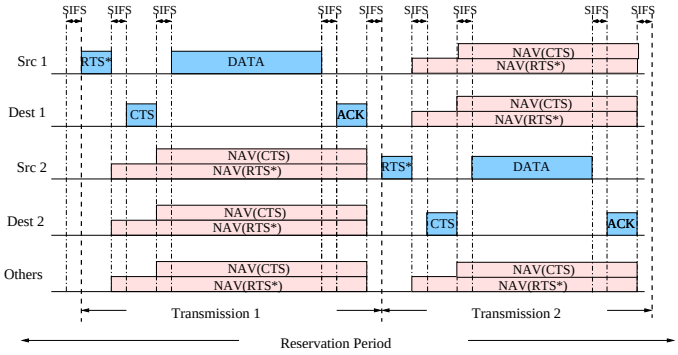


Figure 4.3: Transitions between competing nodes and transmitting nodes: α_1 and α_4 : Transmission queue of the current sender is empty; α_2 and α_3 : Transmission queue of the current sender is non-empty.

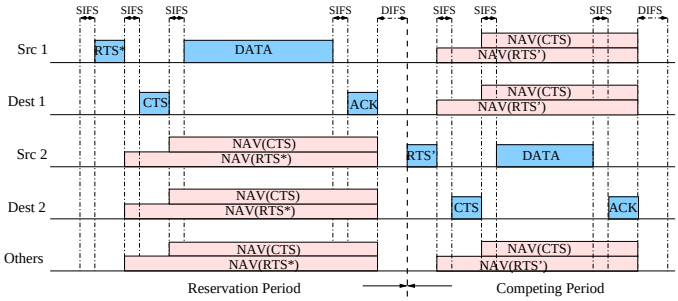
of FR-RTS. Also, the NTA field of the FR-RTS message indicates the address of the next sender. If the sender has no other packets in the queue except the current packet, the sender sends a F-RTS message to forward the transmission token to the next sender. Both F-RTS and FR-RTS messages contain the NTA field indicating the next sender. FR-RTS informs other nodes to update their waiting lists, while the F-RTS does not have this function. In the second case, since the sender does not reserve the channel for the next transmission, it will become the competing node after the transmission. Fig.4.3 describes the transitions between competing nodes and transmitting nodes.

There are two cases where the reservation period would end and the network would return to the competing period. One case is that the waiting list of the current sender is empty. That means there are no other transmitting nodes within its transmission range. Then the sender cannot pass the transmission token to another nearby transmitting node. The other case is that when a transmitting node has been designated twice as the next sender during a reservation period, it will give up the transmission and keep idle for a DIFS. After the channel is sensed idle for a DIFS, the network returns to the competing period, and all active nodes begin to reduce their back-off counters to compete for the channel.

The timing diagram of the protocol is depicted in Fig.4.4. Fig.4.4(a) describes two



(a) Two consecutive transmissions during a reservation period.



(b) The network changes from the reservation period to the competing period after the channel is sensed idle for a DIFS.

Figure 4.4: Timing diagram of the CR-MAC protocol. RTS* denotes F-RTS or FR-RTS message and RTS' denotes RTS or R-RTS message.

consecutive transmissions during a reservation period. At the beginning of the first transmission, a F-RTS/FR-RTS message is sent by the first sender, which includes the duration of the remaining transmission and the address of the next sender. The destination node replies with a CTS message after receiving the F-RTS/FR-RTS message correctly. All other nearby nodes update their NAVs based on the duration information in the F-RTS/FR-RTS and CTS messages. Then a data frame is sent by the first sender, and confirmed by a ACK message. During the reservation period, since the next sender has been designated by the previous sender, the next sender only needs to wait for a SIFS to begin the second transmission after the ACK of the first transmission ends. Thus, no node can activate its back-off counter during the reservation period. Fig.4.4(b) shows the process that the network changes from a reservation period to a competing period. We can see that, similar to Fig.4.4(a), Fig.4.4(b) also contains two complete transmissions. The first one is the last transmission in a reservation period, while the second one is the first transmission of the following competing period. After the ACK message of the last transmission in the reservation period is over, no transmitting node would use the channel and the channel keeps idle for at least a DIFS. Then, all active nodes unfreeze their back-off counters and compete with each other by running the back-off algorithm. In conclusion, during the reservation period, the time interval between any two consecutive frames is SIFS, thus, competing nodes cannot reduce their back-off counters. On the other hand, during the competing period, the time interval between two consecutive transmissions must be longer than a DIFS, and the back-off counters of active nodes can be unfrozen after each transmission. Fig.4.5 summaries the transitions between competing periods and reservation periods.

Intuitively, in the CR-MAC protocol, higher throughput can be obtained when the network is in the reservation period than in the competing period, since one transmitting node

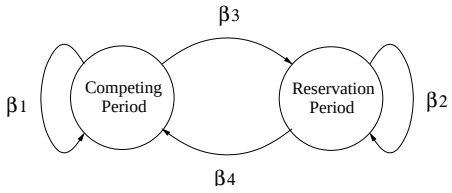


Figure 4.5: Transitions between competing periods and reservation periods: β_1 : Waiting list of the current sender is empty; β_2 : Waiting list of the current sender is non-empty and no node has been designated twice as the next transmitter during the current reservation period; β_3 : Waiting list of the current sender is non-empty; β_4 : Waiting list of the current sender is empty or a node has been designated twice as the next transmitter during the current reservation period.

can send the packet after another in a collision-free manner. However, a potential problem with the CR-MAC protocol is that transmitting nodes may be favored more than competing nodes. As fairness is another important metric to evaluate MAC protocols in addition to throughput, to address this issue, we forcibly let the reservation period end when a transmitting node obtains the second chance to transmit during the same reservation period. Next, we explain in more detail on how the CR-MAC protocol balances the transmission chances between competing nodes and transmitting nodes.

4.3.3 Improving the Fairness of the CR-MAC Protocol

In a WLAN, all nodes should have almost the same chance to access the channel. In the ideal situation, as described above, one transmitting node can send packets after another in a collision-free manner. On the other hand, each competing node has to reduce its back-off counter before the transmission attempt and may encounter collision during the transmission. Furthermore, every competing node competes for itself, while all transmitting nodes can transmit without any collision once any of them obtains the channel. Apparently, competing nodes have less chance to obtain the channel than transmitting nodes in the protocol.

In order to enhance the fairness of the CR-MAC protocol, we adopt three strategies to balance the transmission chances between the transmitting nodes and the competing nodes.

The first strategy is that, as discussed above, the transmitting node must give up the transmission chance, if it obtains the transmission token twice during the same reservation period. Because only transmitting nodes can obtain the channel during the reservation period, competing nodes cannot send packets until the reservation period is over. If a node that always has a non-empty queue obtains the channel, the reservation period will not end unless it is forced to do so. In order to maintain the fairness among all the nodes, CR-MAC protocol does not allow any node to send more than one packets during each reservation period.

The second strategy is that all transmitting nodes will choose a larger initial contention window than that of competing nodes. In the protocol, the back-off counter of a transmitting node is chosen uniformly from the range $[0, CW_{Res} - 1]$, where $CW_{Res} = CW_{min} \times N_{Res}$, CW_{min} is the initial contention window size of a competing node, and N_{Res} denotes the length of the waiting list of a transmitting node. When $N_{Res} = 1$, the transmitting node has the same initial contention window as competing nodes. When there are many transmitting nodes within the transmission range of the node, it will choose a large contention window. Since the larger a contention window, the less chance to obtain the channel, we increase the size of the contention window of transmitting nodes based on the number of transmitting nodes to balance the chances to obtain the channel between transmitting nodes and competing nodes.

The last strategy is that, at the beginning of each competing period, transmitting nodes should freeze their back-off counters until a certain number, say, W, of consecutive idle slots are sensed. The percentage of the idle slots reflects how busy the channel is. When

W ($W > 1$) is a fixed constant, in general, the more nodes compete for the channel, the longer time is needed to obtain W consecutive idle slots. At the beginning of each competing period, no transmitting nodes are allowed to reduce their back-off counters before W consecutive idle slots are sensed. This strategy can adaptively adjust the duration where transmitting nodes are not allowed to compete with competing nodes, according to how busy the channel is. As will be seen in Section 6.4, our simulation results show that these three strategies can help maintain transmission balance between transmitting nodes and competing nodes, and the CR-MAC protocol can achieve better throughput and fairness simultaneously than the standard RTS/CTS protocol.

4.4 Analysis of Throughput under Saturated Traffic

The saturated throughput, which is obtained under the overloaded traffic, is an important metric to evaluate the performance of MAC protocols. In this section, we analyze the saturated throughput of the CR-MAC protocol by using two random processes to model the states of the channel and back-off counter of a given node. Given specific parameters of the network, we calculate the saturated throughput of the CR-MAC protocol and compare it with the standard RTS/CTS protocol. In order to simplify the analysis, in this section, we study the saturated throughput under the ideal channel condition, which assumes that there is no unexpected packet loss or collision in the network, and only one node can use the shared channel at the same time.

4.4.1 Characteristics of CR-MAC Protocol under Saturated Traffic

In order to obtain the saturated throughput, we first study some characteristics of the CR-MAC protocol under saturated traffic. Under saturated traffic, every node in the network always has packets ready to send, and hence each node will work as a transmitting node after its first successful transmission. In this section, we assume that all nodes will become transmitting nodes after the warming-up period and analyze the saturated throughput in the steady state. The saturated throughput is defined as the payload transmitted per unit time under saturated traffic. In order to compute the saturated throughput in the steady state, we need to obtain the average duration of a competing period and a reservation period. From the beginning of each competing period, every transmitting node has to freeze its back-off counter for W consecutive idle slots to maintain the fairness, so the competing period begins with W idle slots in the steady state. Then every transmitting node starts to reduce its back-off counter and competes with others. Once a transmitting node obtains the channel, the network enters the reservation period. During each reservation period, every node sends exactly one packet. Therefore, each reservation period consists of N consecutive successful transmissions, where N is the number of nodes in the network. After that, the reservation period is forced to cease and the network goes back to the competing period.

The average duration of a successful transmission consists of not only the duration for transmitting a data frame, but also the duration for transmitting all the control messages (e.g. RTS, CTS, ACK, etc.) and inter frame spaces (e.g. SIFS and DIFS). From the timing diagram in Fig.4.4, we can see that every successful transmission except the last one during each reservation period consists of a FR-RTS/F-RTS, a CTS, a ACK, a data frame and four SIFSs, where the data frame includes a MAC header, a PHY header and a payload. The duration of the last successful transmission of each reservation period is slightly different

from previous ones, and it ends with a DIFS instead of a SIFS to reactivate the back-off counters of other nodes. Let T_S be the average transmission duration of the first $(N-1)$ successful transmissions, and $T_S^{'}$ be the average duration of the last successful transmission in each reservation period. $T_S^{'}$ and T_S can be expressed as follows.

$$\begin{cases} T_S &= T_{FR-RTS/F-RTS} + T_{CTS} + T_{ACK} + 4T_{SIFS} + \\ & T_{MACHeader} + T_{PHYHeader} + T_{Payload} \\ T_S^{'} &= T_{FR-RTS/F-RTS} + T_{CTS} + T_{ACK} + 3T_{SIFS} + \\ & T_{MACHeader} + T_{PHYHeader} + T_{Payload} + T_{DIFS} \\ & = T_S + T_{DIFS} - T_{SIFS} \end{cases} \tag{4.3}$$

where $T_{FR-RTS/F-RTS}$, T_{CTS}, T_{ACK}, T_{SIFS}, $T_{MACHeader}$, $T_{PHYHeader}$, $T_{Payload}$ and T_{DIFS} denote the durations of $FR-RTS/F-RTS$, CTS, ACK, $SIFS$, $MAC\ Header$, $PHY\ Header$, $Payload$ and $DIFS$, respectively. Thus, the average duration of the reservation period can be easily calculated as follows.

$$\overline{D}_{Res} = (N-1)T_S + T_S^{'} = N * T_S + T_{DIFS} - T_{SIFS} \tag{4.4}$$

The average duration of a competing period is more difficult to obtain than that of a reservation period. In the steady state, before any transmitting node obtains the channel, only collisions and idle slots occur in the competing period. Let T_I and T_C be the average duration of an idle slot and a collision, respectively. Clearly, T_I equals to the length of a slot. When a collision occurs, the remaining transmission after the F-RTS/FR-RTS frame is canceled. After the channel keeps idle for a DIFS, every node becomes unfrozen. Hence, a collision contains a DIFS and a F-RTS/FR-RTS, that is,

$$T_C = T_{F-RTS/FR-RTS} + T_{DIFS} \tag{4.5}$$

Under saturated traffic, each competing period contains only idle slots and collisions. In order to obtain the average duration of the competing period, in the following, we use two random processes to model the states of the channel and the back-off counter of a given node during the competing period.

4.4.2 Random Process of Channel States

The first random process is used to model the channel state during the remaining competing period after W consecutive idle slots. The channel must be in one of the three possible states: idle, collision and successful transmission, which are represented by I, C and S in abbreviation in the following. The channel changes from one state to another step by step. In each step, the states I, C and S last T_I, T_C and T_S, respectively. Let $\{X_1, X_2, \ldots\}$ be a sequence of random variables to model the channel states, where X_k represents the channel state at the k_{th} step and $X_k \in \{I, C, S\},\ k = 1, 2, \ldots$.

We now derive the state and transition probabilities of this random process. Unlike the Markov process, the channel state at the k_{th} step depends not only on the state of the $(k-1)_{th}$ step, but also on all previous $(k-1)$ steps. For example, the channel can be in I or C state at the k_{th} step if and only if there is no successful transmission at previous $(k-1)$ steps, because any successful transmission will cease the competing period. Let ϕ_k denote the event that there is no successful transmission occurs during first k steps. In other words, $\phi_k = (X_k \neq S, X_{k-1} \neq S, \ldots,\ X_1 \neq S)$. Let S_k, C_k and I_k represent the events that the channel has a successful transmission, a collision, and an idle slot at the k_{th} step, respectively. Let $P(S_k)$ be the probability that a node sends the first packet successfully at the k_{th} step, and $P(S_k, \phi_{k-1})$ be the joint probability that a node sends the first packet successfully at the k_{th} step after $(k-1)$ idle slots or collisions. We can obtain $P(S_k) =$

$P(S_k, \phi_{k-1})$ and $\sum_{k=1}^{\infty} P(S_k, \phi_{k-1}) = 1$, since only idle slots and collisions can occur before the successful transmission.

Let $\overline{D}_C(k)$ be the average duration of the competing period except first W consecutive idle slots, which is under the condition that the first successful transmission occurs at the k_{th} step. Let $L_{Payload}$ be the average length of the payload. The average duration of a reservation period $\overline{D}_{Res}$ can be obtained from (4.4). We can calculate the saturated throughput ρ as the transmitted payload during one competing period and one reservation period as follows.

$$\begin{aligned} \rho &= \sum_{k=1}^{\infty} P(S_k, \phi_{k-1}) \cdot \frac{N \times L_{Payload}}{W \times T_I + \overline{D}_C(k) + \overline{D}_{Res}} \\ &= \sum_{k=1}^{\infty} P(S_k | \phi_{k-1}) P(\phi_{k-1}) \cdot \\ &\quad \frac{N \times L_{Payload}}{W \times T_I + \overline{D}_C(k) + N \times T_S + T_{DIFS} - T_{SIFS}} \end{aligned} \tag{4.6}$$

Let D_i be the duration at the i_{th} step $(i < k)$, we have

$$\overline{D}_C(k) = \sum_{i=1}^{k-1} E(D_i | S_k) = \sum_{i=1}^{k-1} E(D_i | S_k, \phi_{k-1}) \tag{4.7}$$

Since D_i depends only on the channel states of the first i steps. we obtain

$$E(D_i | S_k, \phi_{k-1}) = E(D_i | \phi_i) \tag{4.8}$$

Before a successful transmission occurs at the k_{th} step, the channel state at the i_{th} step can be either I or C. We have

$$E(D_i | \phi_i) = P(C_i | \phi_i) T_c + P(I_i | \phi_i) T_I$$

$$
\begin{aligned}
&= \frac{P(C_i,\phi_i)T_c + P(I_i,\phi_i)T_I}{P(\phi_i)} \\
&= \frac{P(C_i,\phi_{i-1})T_c + P(I_i,\phi_{i-1})T_I}{P(C_i,\phi_{i-1}) + P(I_i,\phi_{i-1})} \\
&= \frac{P(C_i|\phi_{i-1})T_c + P(I_i|\phi_{i-1})T_I}{P(C_i|\phi_{i-1}) + P(I_i|\phi_{i-1})}
\end{aligned} \tag{4.9}
$$

By combining (4.7), (4.8) and (4.9), we obtain

$$
\overline{D}_C(k) = \sum_{i=1}^{k-1} \frac{P(C_i|\phi_{i-1})T_c + P(I_i|\phi_{i-1})T_I}{P(C_i|\phi_{i-1}) + P(I_i|\phi_{i-1})} \tag{4.10}
$$

By plugging (4.10) into (4.6), we find that the saturated throughput ρ only depends on the probabilities $P(\phi_k)$, $P(C_k|\phi_{k-1})$, $P(I_k|\phi_{k-1})$ and $P(S_k|\phi_{k-1})$, $k = 1, 2, \ldots$. In addition, $P(\phi_k)$ also can be expressed by $P(C_k|\phi_{k-1})$, $P(I_k|\phi_{k-1})$ and $P(S_k|\phi_{k-1})$ as follows.

$$
\begin{aligned}
P(\phi_k) &= P(\phi_k|\phi_{k-1})P(\phi_{k-1}|\phi_{k-2})\ldots P(\phi_2|\phi_1)P(\phi_1) \\
&= (P(C_k|\phi_{k-1}) + P(I_k|\phi_{k-1}))(P(C_{k-1}|\phi_{k-2}) \\
&\quad + P(I_{k-1}|\phi_{k-2}))\ldots(P(C_1) + P(I_1))
\end{aligned} \tag{4.11}
$$

Therefore, if $P(C_k|\phi_{k-1})$, $P(I_k|\phi_{k-1})$ and $P(S_k|\phi_{k-1})$ can be solved, the saturated throughput ρ can be easily obtained. The conditional probabilities $P(C_k|\phi_{k-1})$, $P(I_k|\phi_{k-1})$ and $P(S_k|\phi_{k-1})$ are determined by the states of back-off counters of nodes. For example, if more than one nodes reduce their back-off counters at the end of the $(k-1)_{th}$ step, the collision must occur at the k_{th} step. If only one node has the zero back-off counter at the end of the $(k-1)_{th}$ step, it can make a successful transmission at the k_{th} step. Thus, we can find $P(C_k|\phi_{k-1})$, $P(I_k|\phi_{k-1})$ and $P(S_k|\phi_{k-1})$ from the states of the back-off counters.

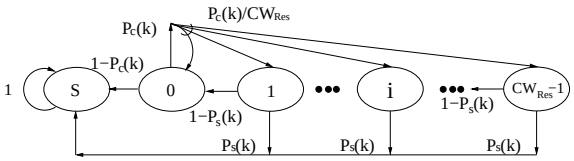


Figure 4.6: Markov chain model for transmitting nodes.

4.4.3 Random Process of Back-off Counter States

Next, we analyze the random process for the back-off counter of a given node. While the channel state changes step by step, the state of the back-off counter of a given node follows the Markov process. At the end of each step in the random process of the channel state, every node changes its back-off counter based on the channel state of the current step. For example, an idle slot makes every node reduce its back-off counter by one. Thus, we can say that the change of the back-off counter state is triggered by the channel state. On the other hand, the state of the back-off counter also causes the change of the channel state. For instance, when more than one nodes reduce their back-off counters to zero at the end of the same slot, a collision must occur in the next slot.

We use the Markov chain in Fig.4.6 to model the back-off counter state of a given node during the remaining competing period after W consecutive idle slots. The Markov chain contains $(CW_{Res}+1)$ states, namely, $S, 0, 1, \ldots, (CW_{Res}-1)$, respectively. Each of them represents a state of the back-off counter of a node. States 0 to $(CW_{Res}-1)$ correspond to the values of the back-off counter. State S denotes the state that the node has sent a packet successfully. Unlike a regular Markov chain, not all transition probabilities of this Markov chain are constants. Some transition probabilities depend on the channel state of the current step. Let $P_s(k)$ and $P_c(k)$ be the conditional probabilities that the channel state is in states S and C at the k_{th} step, respectively. We assume that the k_{th} transition of the back-off counter state occurs at the end of the k_{th} step of the channel state, and is caused

by the channel state of the k_{th} step. Let $P_{ij}(k)$ be the transition probability of the back-off counter state at the k_{th} transition from state i to state j, where $i, j = S, 0, 1, \ldots, (CW_{Res} - 1)$. We have

$$P_{ij}(k) = \begin{cases} 1, & i = S, j = S; \\ 1 - P_c(k), & i = 0, j = S; \\ \frac{P_c(k)}{CW_{Res}}, & i = 0, j = 0, 1, \ldots, CW_{Res} - 1; \\ P_s(k), & i = 1, 2, \ldots, CW_{Res} - 1, j = S; \\ 1 - P_s(k), & i = 1, 2, \ldots, CW_{Res} - 1, j = i - 1; \\ 0 & others. \end{cases} \tag{4.12}$$

The first equation accounts for the case that a node sends a packet successfully at the k_{th} step. Thus, the competing period ends, and the back-off counter will stay at state S after that. The second and third equations model the case that a node tries to send a packet at the k_{th} step. After a successful transmission, the back-off counter goes to state S, with the conditional probability $1 - P_c(k)$. Otherwise, the node resets it back-off counter by choosing a number uniformly from 0 to $(CW_{Res} - 1)$ again. In the fourth equation, whenever any other node in the network completes a successful transmission, the node will no longer reduce its back-off counter, and transfer to state S. Finally, the fifth equation accounts for the case that the back-off counter is reduced by one unless a successful transmission occurs at the k_{th} step. From above transition probabilities, we can construct the transition matrix $P_{trans}(k)$ by plugging every $P_{ij}(k)$ into its corresponding entry of $P_{trans}(k)$.

In addition to the transition probabilities, we also need to know the state probabilities of the back-off counter to solve $P(C_k|\phi_{k-1})$, $P(I_k|\phi_{k-1})$ and $P(S_k|\phi_{k-1})$. Let $\vec{\mu}(k)$ be the state probability vector of the back-off counter at the k_{th} step. Since a node chooses its back-off counter uniformly from 0 to $(CR_{Res} - 1)$ at the beginning of each competing period, the

initial state probability vector

$$\vec{\mu}(1) = (0, 1/CW_{Res}, 1/CW_{Res}, \ldots, 1/CW_{Res})^T \tag{4.13}$$

From the transition probabilities in (4.12) and the initial state probabilities in (4.13), we obtain

$$\begin{aligned} \vec{\mu}(k) &= (\mu^S(k), \mu^0(k), \mu^1(k), \ldots, \mu^{CW_{Res}-1}(k))^T \\ &= \vec{\mu}(1) \times \prod_{i=1}^{k-1} P_{trans}(i) \end{aligned} \tag{4.14}$$

where $\mu^i(k)$, $i = S, 0, 1, \ldots, CW_{Res} - 1$, represents the probability that the back-off counter stays at state i at the k_{th} step. Let $\tau(k)$ denote the conditional probability that a node tries to transmit a packet at the k_{th} step, under the condition that no successful transmission occurs before the k_{th} step. We have

$$\tau(k) = \frac{\mu^0(k)}{1 - \mu^S(k)} \tag{4.15}$$

The conditional probabilities $P_C(k)$ and $P_S(k)$ can be expressed by $\tau(k)$. When a given node tries to send a packet at the k_{th} step, if all other $N-1$ nodes have non-zero back-off counters, the packet can be transmitted successfully. If some of remaining nodes also try to send the packets at the k_{th} step, the collision will occur. Thus,

$$\begin{cases} P_S(k) = (1 - \tau(k))^{N-1} \\ P_C(k) = 1 - (1 - \tau(k))^{N-1} \end{cases} \tag{4.16}$$

From (4.12) to (4.16), we can solve the state probability vector $\vec{\mu}(k)$ and conditional packet transmission probability $\tau(k)$ recursively for any $k > 0$. Moreover, we can derive $P(C_k|\phi_{k-1})$, $P(I_k|\phi_{k-1})$ and $P(S_k|\phi_{k-1})$ from $\tau(k)$. We have

$$\begin{cases} P(I_k|\phi_{k-1}) = (1-\tau(k))^N \\ P(S_k|\phi_{k-1}) = N\tau(k)(1-\tau(k))^{N-1} \\ P(C_k|\phi_{k-1}) = 1 - P(I_k|\phi_{k-1}) - P(S_k|\phi_{k-1}) \end{cases} \tag{4.17}$$

Though the values of $P(C_k|\phi_{k-1})$, $P(I_k|\phi_{k-1})$ and $P(S_k|\phi_{k-1})$ for any $k > 0$ can be calculated from (4.12) to (4.17), it is still difficult to give a neat expression of ρ by plugging them into (4.6) and (4.10). Furthermore, because ρ is obtained by calculating the summation of the function of k in (4.6), when k ranges from 1 to ∞, it is impractical to get precise value for ρ by repeating the addition operation for infinite iterations. Therefore, we will find the approximate value of ρ in the next subsection.

4.4.4 Numerical Analysis of Saturated Throughput

In order to calculate the approximate value of the saturated throughput ρ, we repeat the addition operations in (4.6) for a sufficiently large number, M, times instead of infinite times. That is,

$$\rho = \sum_{k=1}^{M} P(S_k|\phi_{k-1})P(\phi_{k-1}) \cdot \frac{N \times L_{Payload}}{W \times T_I + \overline{D}_C(k) + N \times T_S + T_{DIFS} - T_{SIFS}} \tag{4.18}$$

Table 4.1: System parameters used for numerical analyses and simulations

Average Packet Payload	8184 bits
MAC Header	272 bits
PHY Header	128 bits
F-RTS and FR-RTS	208 bits + PHY Header
Basic RTS and R-RTS	160 bits + PHY Header
CTS	112 bits + PHY Header
ACK	112 bits + PHY Header
SIFS	28 μs
DIFS	128 μs
Slot Length	50 μs
Transmission Rate(R)	1Mbps
CW_{min}	16
CW_{max}	1024
W	4

Table 4.1 illustrates the frame-related parameters and overhead used in the following numerical analyses and simulations. Recall that W is a small integer and transmitting nodes cannot reduce their back-off counters until W consecutive idle slots are sensed from the beginning of each competing period. Here we choose $W = 4$. As discussed above, the length of F-RTS/FR-RTS in Table 4.1 is 6 bytes longer than that of RTS/R-RTS. In order to compare the performance with the results in [48], we use the same average length of the payload as in [48], which is represented by $L_{Payload}$ and computed by the product of the average duration of the payload $T_{Payload}$ and the transmission rate R. All other parameters in Table 4.1 are the same as the specification of IEEE 802.11b FHSS [43], [44]. By plugging these parameters into (4.3) and (4.5), we obtain the values of T_S and T_C. For a given number of nodes, we can calculate $\tau(k)$, $P_S(k)$, $P_C(k)$, $P_{trans}(k)$, $P(I_k|\phi_{k-1}), P(S_k|\phi_{k-1}), P(C_k|\phi_{k-1})$ and $P(\phi_k|\phi_{k-1})$ for $1 \leq k \leq M$, recursively, from (4.12) to (4.17). Finally, from above numerical results for each k, $1 \leq k \leq M$, the approximate value of ρ can be obtained by repeating the addition operation in (4.18) for M times.

In addition, we derive the upper bounds of saturated throughput for both RTS/CTS and CR-MAC. The theoretical upper bound of the saturated throughput can be obtained, if and only if all transmissions are completed one by one without collisions and extra idle slots. Thus, the upper bound of the saturated throughput for RTS/CTS and CR-MAC can be expressed by $\widehat{\rho}_{RTS/CTS}$ and $\widehat{\rho}_{CR-MAC}$ as follows.

$$\widehat{\rho}_{RTS/CTS} = \frac{T_{Payload} \times R}{T_S^{RTS/CTS}} = 855.7(kbps) \tag{4.19}$$

$$\widehat{\rho}_{CR-MAC} = \frac{N \times T_{Payload} \times R}{N \times T_S + T_{DIFS} - T_{SIFS} + W \times T_I} \tag{4.20}$$

where $T_S^{RTS/CTS}$ equals to the average duration of each successful transmission in the RTS/CTS protocol and can be represented as

$$\begin{aligned} T_S^{RTS/CTS} &= T_{RTS} + T_{CTS} + T_{ACK} + 3T_{SIFS} + T_{DIFS} + \\ &\quad T_{MACHeader} + T_{PHYHeader} + T_{Payload} \end{aligned} \tag{4.21}$$

We choose $M = 80, 100$ and 200 and calculate the approximate saturated throughput for each case, when the number of nodes N ranges from 1 to 60. We plot three approximate curves of the saturated throughput for $M = 80, 100$ and 200 in Fig. 4.7. In addition, based on the theoretical analysis in [48], we calculate the numerical results of the saturated throughput of the RTS/CTS protocol by using the same parameters in Table 4.1, and also plot the curve in Fig. 4.7. Finally, we plot the upper bounds for RTS/CTS and CR-MAC in Fig. 4.7 by evaluating equations (4.19) and (4.20).

From Fig.4.7, we have following observations. First, the larger M is, the better approximate saturated throughput can be obtained. We can see that there is very little difference

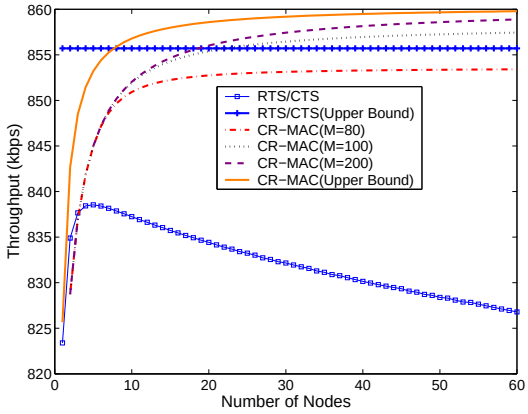


Figure 4.7: Numerical results of saturated throughput.

between the curves for $M = 100$ and $M = 200$. Second, the CR-MAC protocol ($M = 200$) achieves better saturated throughput than RTS/CTS when $N \geq 3$ and even exceeds the upper bound of RTS/CTS when $N \geq 18$. Finally, we can see that the saturated throughput of the CR-MAC protocol increases slightly, while the saturated throughput of RTS/CTS drops sharply, as the number of nodes increases from 1 to 60, due to more collisions after it reaches its maximum value.

4.5 Discussions on Some Practical Issues

4.5.1 Throughput under Unsaturated Traffic

We have shown that the CR-MAC protocol achieves better saturated throughput under the ideal channel condition than the RTS/CTS protocol. However, can the CR-MAC protocol still perform better than the RTS/CTS protocol under more realistic unsaturated traffic? In the following, we analyze the throughput of the CR-MAC protocol in this scenario.

In the CR-MAC, since transmitting nodes transmit packets without collision during the

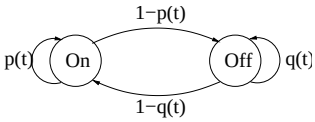


Figure 4.8: ON/OFF traffic model.

reservation period, intuitively, the higher percentage of transmitting nodes in the network, the better throughput can be obtained. The percentage of the transmitting nodes depends on the offered traffic load of every node. For example, if a node always has a non-empty queue, it will work as a transmitting node after the first successful transmission. On the contrary, if a node has no other packets waiting in the queue, when it begins to transmit the current packet, it has to compete with other nodes to transmit the next packet. The heavier the offered load is, the better chance a node can reserve the channel for the next packet when it transmits the current packet.

In order to see the effect of the offered load on the throughput of the CR-MAC protocol, we consider an ON/OFF traffic model shown in Fig. 4.8. The traffic model has two states ON and OFF. ON state means the node is busy and has packets ready to send, while OFF state means the node is idle. Assume that an active node wants to transmit a packet at time slot t. At that time, it has more than one packets waiting in the queue with probability $p(t)$ so that it can keep active after slot t. Also, assume that at the end of each idle slot, an idle node in OFF state produces a packet to send and transfers into ON state with probability $1-q(t)$. It stays in OFF state with probability $q(t)$. Any general traffic, such as TCP and UDP traffic, can be characterized by the ON/OFF model by choosing a specific transition probabilities $p(t)$ and $q(t)$.

To simplify the analysis and examine the extreme cases, we assume that $p(t)$ and $q(t)$ are constants p and q, which range from 0 to 1. When $p = 1$ and $q < 1$, all nodes always have a non-empty queue, and the network works under saturated traffic. When $p = 0$ and

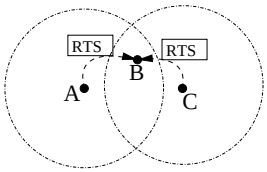


Figure 4.9: Illustration of the hidden node problem.

$1 > q \geq 0$, there are no other packets waiting in the queue when a node begins the current transmission. In this case, no node can reserve the channel for the next transmission. Every node has to compete with each other to obtain the channel all the time. In this case, the CR-MAC protocol degrades to the standard RTS/CTS protocol, and has the same performance. Between these two extreme cases, when $1 \geq p > 0$ and $1 > q \geq 0$, more or fewer nodes will work as transmitting nodes for some time. As will be seen in Section 6.4, the simulation results show that CR-MAC protocol provides better throughput and fairness when $p > 0$, and the same performance when $p = 0$ compared to the standard RTS/CTS protocol.

4.5.2 Error Handling in Networks with Unexpected Packet Loss and Collision

As discussed above, the proposed protocol works well under the ideal channel condition. However, in practice, a wireless channel may not be as stable as the ideal case. Nodes may face unexpected packet loss and collision. We will show that the CR-MAC protocol can also work well in these cases.

Based on the assumption of the ideal network, when a transmitting node broadcasts a RTS message, all nearby nodes within the transmission range of the sender can receive the RTS message correctly. Unfortunately, the RTS may be lost due to the interference in the environment. Furthermore, the RTS collision may occur in a multi-hop network due to the

hidden node problem [54], [55], which is illustrated briefly as follows. As shown in Fig. 4.9, node A and node C stay too far away to hear each other, and they may transmit RTS messages to node B in the same time slot. Thus, node B cannot receive the RTS messages correctly from either of them. This is called the hidden node problem. In this case, when node A and node C cannot receive the CTS message after a SIFS, they know the transmissions has failed. In the CR-MAC protocol, the failure of RTS transmission may occur in both the competing period and the reservation period. When node A or node C works in the reservation period, it cannot forward the "transmission token" to the next sender due to the RTS failure. Then, the node keeps idle for a DIFS and the reservation period transfers into the competing period. If the RTS failure occurs during the competing period, the node will reset their contention window and continue to compete for the channel. In the worst case that no transmitting node can forward the transmission token to the next sender due to the transmission failure, the network will not enter the reservation period at all, and all the nodes compete with each other all the time. Thus, in the worst case, CR-MAC protocol still has similar performance and cost to the standard RTS/CTS protocol.

4.6 Simulation Results

In this section, we present the simulation results of the CR-MAC protocol. In our simulation, we assume that all nodes uniformly distributed within a $50 \times 50m^2$ 2-dimensional square. A node can communicate with any other nodes within the transmission range of $80m$. 95% of packets can be received correctly. The same parameters of the protocol are described in Table 4.1. The simulation was run under the ON/OFF traffic model in Fig.4.8 for various values of (p,q) pairs. We evaluated four important performance metrics, throughput, fairness, maximum packet delay and mean packet delay of each node. Moreover, all

these performance metrics are compared with the standard RTS/CTS protocol under the same condition. The simulation ran for 100 seconds which contained 20 seconds warming-up period. All simulation data were collected from 20 seconds to 100 seconds.

4.6.1 Throughput

Fig.4.10(a,b and c) illustrate the throughput of CR-MAC and RTS/CTS protocols for $(p,q) = (1,0), (0.5,0.5)$ and $(0,0)$, respectively. When $p = 1$, every node always has packets ready to send, so every node works under saturated traffic. Fig.4.10(a) shows the saturated case when $(p,q) = (1,0)$. We can see that as the number of nodes increases, the saturated throughput of CR-MAC protocol increases slightly, while the throughput of RTS/CTS drops sharply due to the increase of collisions. In case (b) $((p,q) = (0.5,0.5))$, a node may work as a competing node or a transmitting node alternatively. The throughput of CR-MAC protocol declines a little as the number of nodes increases, since some nodes work as competing nodes and more collisions occur as the number of nodes increases. However, all nodes in RTS/CTS are always competing nodes. It is not surprising that RTS/CTS protocol has more collisions than the CR-MAC protocol, which results in a fast decline of the throughput. When $p = 0$, no nodes can reserve the channel all the time, because every node always has an empty queue when it wants to send the current packet. In this case, as discussed in Section 4.5, the CR-MAC degrades to the RTS/CTS protocol, we can see from Fig.4.10(c) that they have similar throughput when the number of nodes changes from 1 to 80.

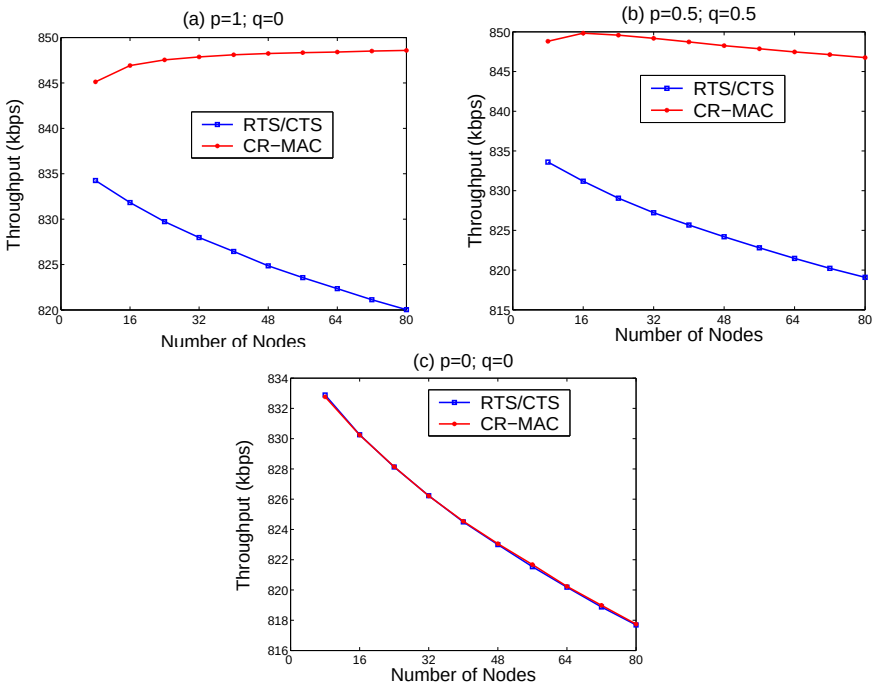


Figure 4.10: Comparison of throughput under ON/OFF traffic. (a) $p = 1, q = 0$. (b) $p = 0.5, q = 0.5$. (c) $p = 0, q = 0$.

4.6.2 Fairness

To evaluate the fairness of the CR-MAC protocol, we adopt the fairness index defined in [56], which is widely accepted and used in [52] [51]. The fairness index is defined as

$$f(x) = \frac{(\sum \rho_i/\rho_i^{'})^2}{N\sum(\rho_i/\rho_i^{'})^2} \tag{4.22}$$

where ρ_i and $\rho_i^{'}$ represent the throughput and the ideal throughput of node i, respectively, and N is the number of nodes in the network. We assume that all nodes has the same ideal throughput. It is easy to show that $0 < f(x) \leq 1$ and $f(x) = 1$ if and only if all nodes have the same throughput ρ_i. Also, the closer to 1 the fairness index is, the better fairness can be obtained.

Fig.4.11(a,b and c) shows the fairness index when $(p,q) = (1,0), (0.5,0.5)$ and $(0,0)$, respectively. Fig.4.11(a) shows the fairness under saturated traffic $((p,q) = (1,0))$. In CR-MAC protocol, since all nodes always have packets to send, they all work as transmitting nodes in the steady state. Thus, every node has equal chance to use the channel and send exactly one packet during each reservation period. Thus, the fairness index of the CR-MAC protocol always equals to 1, while the fairness index of the RTS/CTS protocol decreases as the number of nodes increases. In case (b) $((p,q) = (0.5,0.5))$,we can see that the fairness index of the CR-MAC protocol still keeps above 0.95. However, the fairness index curve of the RTS/CTS protocol decreases fast and drops down to 0.91 when the number of nodes increases to 80. The third case $((p,q) = (0,0))$ accounts for the situation that no node can reserve the channel. In this case the CR-MAC protocol degrades to the RTS/CTS protocol. In Fig.4.11(c), we observes that CR-MAC and RTS/CTS have almost the same declining curve as the number of nodes increases. Besides the above three special cases, we also ran the simulation for various values of (p,q) pairs. We found that the CR-MAC protocol

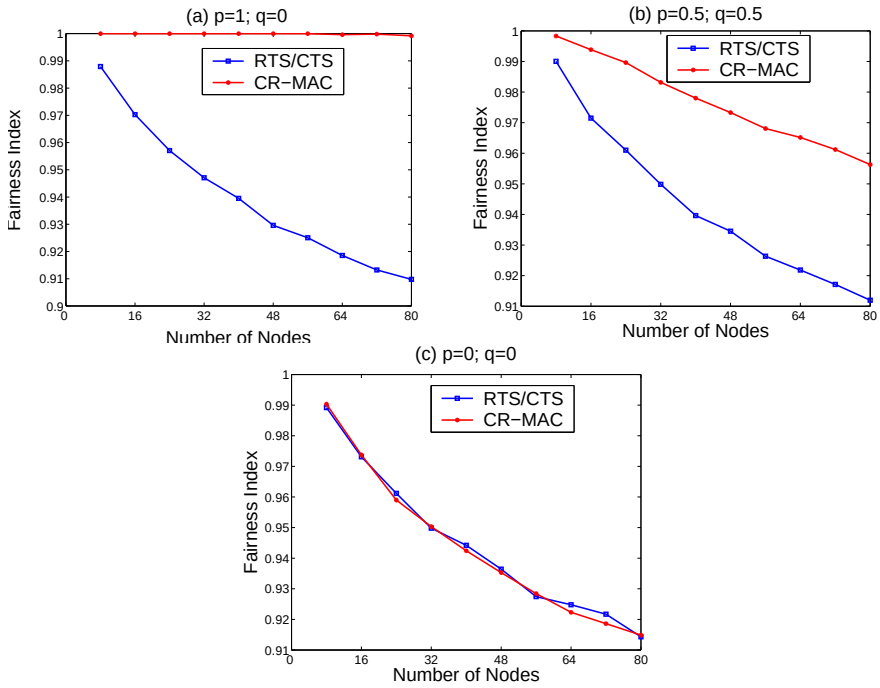


Figure 4.11: Comparison of fairness index under ON/OFF traffic. (a) $p = 1, q = 0$. (b) $p = 0.5, q = 0.5$. (c) $p = 0, q = 0$.

always provides better fairness than the RTS/CTS protocol when $p > 0$, and has a similar declining curve as the RTS/CTS protocol when $p = 0$.

4.6.3 Packet Delay

Packet delay is also an important metric to evaluate a MAC protocol. In the simulation, we compared the maximum and mean packet delay of the CR-MAC and the RTS/CTS protocols for each node. We looked at three traffic cases ($(p,q) = (1,1)$, $(p,q) = (0.5,0.5)$ and $(p,q) = (0,0)$). When the network operates under saturated traffic $(p = 1)$, CR-MAC can obtain shorter packet delay than RTS/CTS, because RTS/CTS causes much more collisions than CR-MAC. When $p = 0$, as discussed above, CR-MAC protocol degrades to RTS/CTS and has the same packet delay as the RTS/CTS protocol. Therefore, we only

Table 4.2: The statistic simulation data for the maximum packet delay, $E(D_{Max})$ and $V(D_{Max})$ denote the average and variance of the maximum packet delay.

Node #	$E(D_{Max}) \pm V(D_{Max})$ $(CR-MAC)$	$E(D_{Max}) \pm V(D_{Max})$ (RTS/CTS)
8	2.7171 ± 1.8148	7.2405 ± 3.1556
16	4.5122 ± 2.4781	11.5075 ± 2.3972
32	8.7888 ± 3.4135	17.0359 ± 4.7697
64	14.6533 ± 4.5074	21.2358 ± 5.0779

Table 4.3: The statistic values for the mean packet delay, $E(\overline{D})$ and $V(\overline{D})$ denote the average and variance of the mean packet delay.

Node #	$E(\overline{D}) \pm V(\overline{D})$ $(CR-MAC)$	$E(\overline{D}) \pm V(\overline{D})$ (RTS/CTS)
8	0.0604 ± 0.0012	0.0651 ± 0.0029
16	0.1338 ± 0.0050	0.1408 ± 0.0108
32	0.2830 ± 0.0146	0.2936 ± 0.0268
64	0.5851 ± 0.0408	0.6048 ± 0.0658

give the simulation results for the case $(p,q) = (0.5, 0.5)$ here. The simulation program was run for 10 independent iterations when the number of nodes N equals to 8, 16, 32 and 64, respectively. The packet delay is defined as the duration from the time the packet enters the queue to the time it leaves. For each node, we collected the maximum packet delay and the mean packet delay during the time from 20 seconds to 100 seconds, and calculated the average and variance of maximum packet delay and mean packet delay of all N nodes. The statistic data of the maximum packet delay and mean packet delay are shown in Tables 4.2 and 4.3, where $E(D_{Max})$, $V(D_{Max})$, $E(\overline{D})$ and $V(\overline{D})$ denote the average values and the variances of the maximum packet delay and mean packet delay, respectively. All data are measured in seconds.

From Table 4.2, we can see that the average maximum delay of the RTS/CTS protocol is nearly twice as that of the CR-MAC protocol. The data in Table 4.3 also reflects that the

RTS/CTS protocol has sightly longer mean packet delay than CR-MAC protocol. Therefore, we can conclude that the CR-MAC protocol is more suitable to the delay-sensitive applications than RTS/CTS, such as real-time voice and video traffic.

4.7 Conclusions

In this chapter, we proposed a new contention-based MAC protocol with channel reservation, called CR-MAC protocol. We showed theoretically and experimentally that the CR-MAC protocol can achieve much better throughput, fairness, packet delay than IEEE 802.11 RTS/CTS protocol. In particular, under saturated traffic, both the throughput and the fairness index of the CR-MAC protocol are very close to the theoretical bound, regardless of the number of nodes in the network. By utilizing the overhearing feature of the shared medium, the proposed protocol includes the reservation information into the control messages, rather than transmit extra control packets. Thus, the proposed protocol does not incur much extra overhead. Moreover, the simulation results show that CR-MAC protocol achieves shorter maximum and mean packet delay than the RTS/CTS protocol, thus the new protocol is more suitable to the delay-sensitive applications.

Chapter 5

Clustering and Load Balancing Algorithms

In chapter 3 and 4, we assume that wireless nodes have very long battery life. Thus, single path flooding algorithm and channel reservation protocol focus more on performance and reliability than energy efficiency. In the next two chapters, the proposed approaches are designed for low speed, low cost applications, in which sensors can only work properly for a limited time. For such applications, energy efficiency becomes the first concern.

This chapter investigates the problem of positioning mobile cluster heads and balancing traffic load in a hybrid sensor network, which consists of two types of nodes: basic static sensor nodes and mobile cluster heads. In such a network, sensor nodes are organized into clusters and form the lower layer of the network. At the higher layer, cluster heads collect sensing data from sensors and forward data to outside observers. Such two-layer hybrid networks are more scalable and energy-efficient than homogeneous sensor networks. We show that the locations of cluster heads can affect network lifetime significantly. The problem of maximizing network lifetime through dynamically positioning cluster heads in

the network (referred to as the CHL problem in this paper) turns out to be NP-hard. We present a heuristic algorithm for positioning cluster heads and balancing traffic load in the network.

The rest of the chapter is organized as follows. Section 5.1 provides some background of the problem studied in this paper and briefly discusses the related work. Section 5.2 gives the description and the formalization of the problem. Section 5.3 presents the new clustering algorithm. Section 5.4 gives the performance evaluation results. Finally, Section 5.5 concludes the chapter.

5.1 Background and Related Work

In a large scale hierarchical network, sensors are organized into clusters with a cluster head in each cluster. Cluster heads act not only as a data aggregation point for collecting sensing data from sensors, but also as a controller/scheduler to make various routing and scheduling decisions. Such hierarchical clustered sensor networks can be classified into two categories: *homogeneous* and *heterogeneous*. In a homogeneous network, all nodes have identical capability and energy at the beginning. Some of the nodes are selected to serve as cluster heads [69, 70, 72, 71]. However, cluster heads will inevitably consume more energy than other sensor nodes. In order to avoid the problem of cluster heads failing faster than other nodes, sensor nodes can act as the cluster head rotationally [69]. In this type of network, since every sensor node may possibly become a cluster head, each of them has to be "powerful" enough to handle incoming and outgoing traffic and cache sensing data, which will increase the overall cost of the entire sensor network. Furthermore, selecting cluster heads dynamically also results in high overhead due to the frequent information exchange among the sensor nodes. On the contrary, a heterogeneous sensor

network, or a hybrid sensor network, contains a small number of resource-rich specialized cluster heads along with a large number of resource-limited basic sensor nodes. Basic sensor nodes have limited communication capability and mainly focus on sensing the environment, while cluster head nodes are equipped with more powerful transceivers and batteries but simpler sensing modules. The cluster head organizes basic sensor nodes around it into a cluster, collects sensing data from sensors and forwards data to the outside observer. A two-layer hybrid sensor network is depicted in Figure 5.1, where large nodes at the higher layer are cluster heads and small nodes at the lower layer denote the basic sensor nodes. The advantages of a heterogeneous sensor network are: first, the hierarchical topology is more scalable than a flat network without hierarchies; second, as the dominant percentage of the network nodes, the basic sensors can be made very simple and inexpensive, thus the overall cost of the network can be reduced.

In a heterogeneous sensor network, after deployed in a two-dimensional area, each cluster head organizes sensor nodes around it into a cluster. The inner-cluster traffic can be roughly divided into two types based on the direction of the traffic flow: uploading sensing data from sensors to the cluster head and downloading commands from the cluster head to sensors. Since the number of cluster heads is much smaller than the number of sensors and each sensor has a very limited transmission range, the packets sent by a sensor node may need multi-hop relaying by other sensors to reach the cluster head. On the other hand, since a cluster head has a more powerful transceiver, every sensor in the cluster can hear from the cluster head directly. Cluster head can decide the routing path and poll sensors one by one in a time-slotted manner. Our recent work [99, 98] focused on the inner-cluster multi-hop polling scheme for a cluster with a static cluster head. In this paper, we consider mobile resource-rich cluster heads, and will focus on optimally positioning such cluster heads in

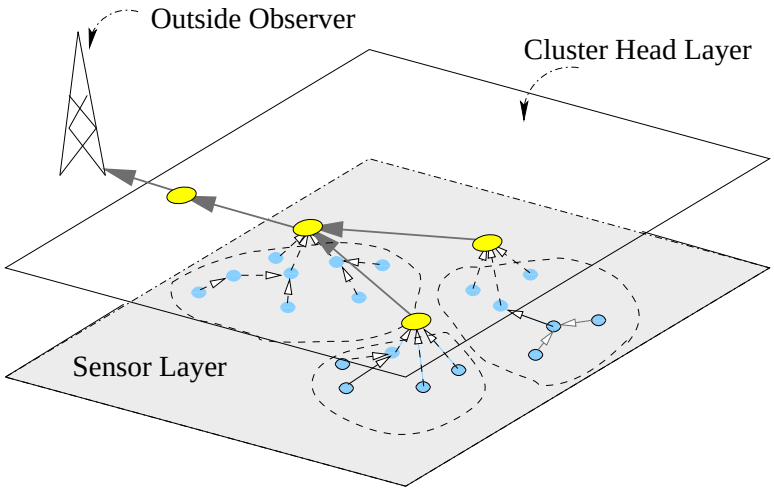


Figure 5.1: A two-layer hybrid sensor network. Large nodes at the higher layer are cluster heads. Small nodes at the lower layer are basic sensor nodes.

the network to balance traffic load and prolong network lifetime.

The problem studied in this paper is related to two well-known problems: the facility location problem [76] and the network flow problem [75]. The facility location problem (FLP) has been extensively studied in the last decade, which can be briefly described as follows. Given a set of demand nodes, a set of facility nodes and cost functions, the problem is to optimize some objectives, for example, the *k*-center problem which minimizes the maximum distance (cost) between any demand node and the nearest facility, the *k*-median problem which minimizes the sum of distances to the nearest facility, and so on. As many other optimization problems, the *k*-center and *k*-median facility location problems are NP-hard. In the problem we consider in this paper, demand nodes and facilities correspond to sensor nodes and cluster heads, respectively. However, different from the traditional facility location problem, we introduce the capacity, or energy limit, to each sensor node, to find the maximum lifetime of the entire network. The network flow problem is to optimize the flow between any pair of nodes in a graph according to various objective and constraint

functions. The network flow problem was first introduced to multi-hop wireless networks by Chang and Tassiulas [73], and then discussed in other literatures as well, see, for example, [74, 99, 98]. In this paper, we will use the network flow as a mathematical tool to balance the traffic load to maximize network lifetime by applying different constraints.

We consider the applications, such as environment monitoring, where the sensing data is generally collected at a low rate and sensing data is not very delay-sensitive so that it can be accumulated into fixed-length data packets and uploaded once a while. For such applications, we have the following assumptions. n_s static sensors and n_c mobile cluster heads are uniformly deployed onto a two-dimensional working area. Due to limited transmission power, sensor nodes can only reach nearby nodes within a limited range. The total power consumption for transmitting a packet, including processing power consumption, such as coding/decoding, modulation/demodulation, A/D-D/A, and so on, and radio power both are assumed to be proportional to the size of the packet. For the sake of simplicity, we only consider the major energy consumption for communications and ignore those for sensing and other tasks. Cluster heads can move to anywhere within the working area, and are equipped with more powerful transceivers and batteries having much longer lifetime than those of sensor nodes. Sensing modules can be optional for cluster heads. In addition, cluster heads are connected directly without the relaying of sensor nodes. We assume that all sensors and cluster heads obtain either their absolute positions from the GPS [33] or relative positions from other location services [34, 77]. In the area without the GPS system, we assume that each sensor can estimate the relative location information to nearby nodes by detecting the relative distance and angle [34, 77].

5.2 Problem Description and Formalization

Before describing the problem we are considering, we first give an example to see how the position of the cluster head affects the network lifetime. Here we define the *network lifetime* as the lifetime of the first failure node in the network. As shown in Figure 5.2(a), two cluster heads and twenty sensor nodes are randomly deployed in the sensing field. The network is organized into two spanning trees rooted at two cluster heads respectively. Due to limited transmission power of sensors, a packet may need multi-hop relays to reach the cluster head. We can see that in the left cluster, node 1 is a bottleneck node, because it has to relay the packets from its six child nodes to the cluster head. An even worse situation occurs in the other cluster: node 2 suffers heavier load than node 1, which has seven children. Since node 1 and node 2 are bottlenecks in the corresponding clusters, they will consume energy much faster than other nodes. After they fail, the network becomes disconnected. Figure 5.2(b) shows the network layout in which two cluster heads are placed at better positions. Some traffic load forwarded by node 1 and node 2 before is now re-directed via other nodes with lighter traffic load. As a result, the most unlucky node in Figure 5.2(b) has only two child nodes. We can see that due to the movement of cluster heads, the traffic load is balanced and the network lifetime is prolonged significantly. From this simple example, we observe that, in order to maximize the network lifetime, the maximum load of any node should be minimized. In addition to load balancing, positions of cluster heads can also affect the directions of traffic flow, thereby have a significant impact on the network lifetime. In the following, we will start with the networks with static cluster heads and then consider the networks with mobile cluster heads.

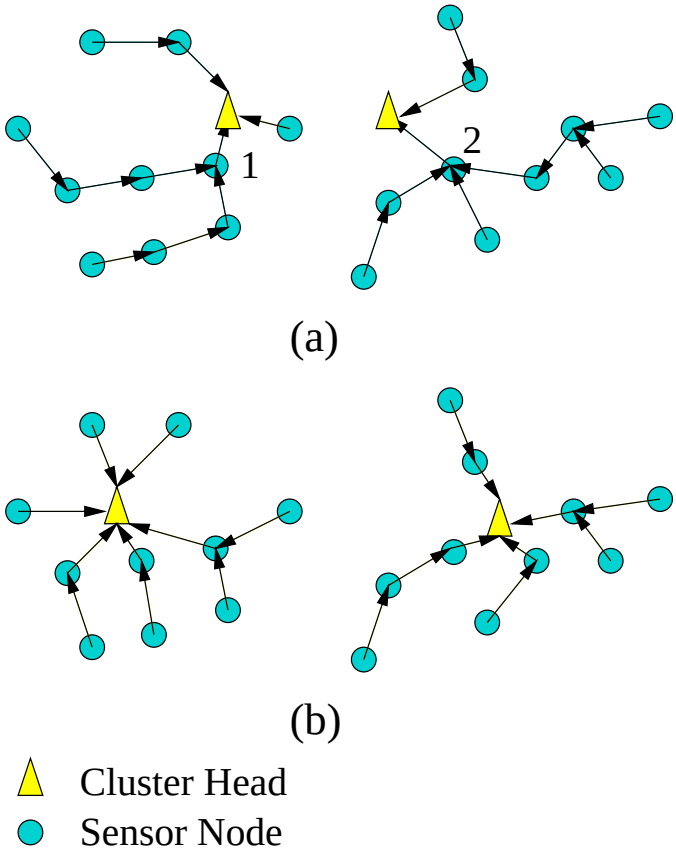


Figure 5.2: (a) Initial topology of two clusters. (b) New topology after two cluster heads move to better locations.

5.2.1 Load Balancing in Networks with Static Cluster Heads

As discussed above, due to the different amount of traffic each sensor node relays, some nodes may fail sooner than others. In order to maximize the network lifetime, relaying paths must be carefully planned to balance traffic load. Load balancing problem in sensor networks has been investigated in some existing work, such as [73, 74, 99], and can be formalized into a network flow problem or a linear program under different objectives and constraints. For a network with static cluster heads, the problem of maximizing network lifetime can be formalized as the following network flow problem.

A static sensor network can be modeled as a directed graph $G(S,C,A)$, where $S = \{s_1, s_2, \ldots, s_{n_s}\}$ is the set of all sensor nodes, $C = \{c_1, c_2, \ldots, c_{n_c}\}$ is the set of all cluster heads and A is the set of all directed links $a(i, j)$ where $i \in S, j \in S \bigcup C$. Given a directed graph G, its corresponding flow graph $G'(S', C', Src, Dst, A')$ can be constructed as follows:

- For each $s_i \in S$, add two vertices s'_i and s''_i to S'; an arc $a(s'_i, s''_i)$ is added into A' with capacity $\frac{Es_i - (r_{s_i} T P_g)}{P_r} + r_{s_i} T$; for each $c_j \in C$, add a vertex c'_j to C';

- For each arc $a(s_i, s_j) \in A$, where $s_i, s_j \in S$, add an arc $a(s''_i, s'_j)$ into A' with infinity capacity;

- For each arc $a(s_i, c_j) \in A$, where $s_i \in S, c_j \in C$, add an arc $a(s''_i, c'_j)$ into A' with infinity capacity;

- A pair of source and destination nodes *Src* and *Dst* are added into G'; for each $s'_i \in S'$, connect *Src* and s'_i by an arc $a(Src, s'_i)$ with capacity $r_{s_i} T$;

- For each $c'_j \in C'$, connect c'_j and *Dst* by an arc $a(c'_j, Dst)$ with infinity capacity.

where r_{s_i} and E_{s_i} denote the data generating rate and energy limit of node s_i, P_g and P_r represent the power consumption for generating and relaying a unit of traffic, respectively, and T is the network lifetime. Since the sensing data is sent out periodically, say, every ΔT time. We can set $T = \Delta T$ at the beginning and increase T by ΔT every time. For any given T, the problem is a regular maximum flow problem and can be solved by Ford-Fulkerson algorithm in polynomial time. In this construction, $r_{s_i}T$ limits the flow from *Src* to s_i and represents the flow generated by s_i within time T, which consumes $(r_{s_i}TP_g)$ energy. Due to the energy constraint of node s_i, the maximum flow that node s_i can relay within time T is $\frac{Es_i-(r_{s_i}TP_g)}{P_r}$. Thus, the total flow a node s_i can generate and relay in time T is limited by $\frac{Es_i-(r_{s_i}TP_g)}{P_r}+r_{s_i}T$. When the maximum flow equals $\sum_{s_i\in S} r_{s_i}T$, it means that until time T, all generated traffic by n_s sensor nodes is received by cluster heads. Thus, all n_s sensors must be alive until T. We can keep increasing T and running Ford-Fulkerson algorithm to obtain the maximum flow for every T value, until the maximum flow is less than $\sum_{s_i\in S} r_{s_i}T$, which means that some nodes have failed before time T. Finally, the value of T obtained before the last run of Ford-Fulkerson algorithm is the maximum network lifetime. An example of the construction is depicted in Figure 5.3.

We now analyze the time complexity of this method. Let U denote the maximum units of traffic any sensor node generates and relays within time T^*, where T^* is the maximum network lifetime obtained by the algorithm. Then

$$U = \max_{s_i\in S}\left\{\frac{Es_i-(r_{s_i}T^*P_g)}{P_r}+r_{s_i}T^*\right\}$$

The running time of this method is $O(UM_s^2)$, where $M_s = n_c + n_s$ is the total number of cluster heads and sensors in the network.

Note that the above optimization needs global location information of sensor nodes

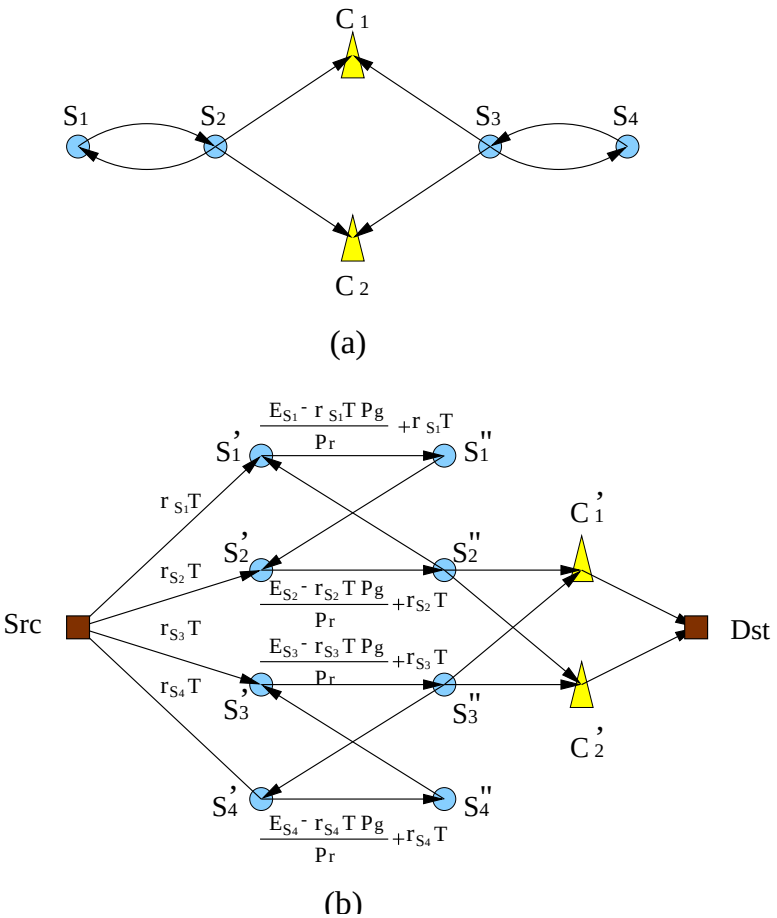


Figure 5.3: (a) Connection patterns of a network with two clusters. (b) Network flow graph for maximizing the network lifetime, where the capacities of unmarked arcs are infinity.

and cluster heads and connection patterns of the network. As we know, exchanging global location information is very energy-consuming and impractical for a large sensor network. Here we give this global optimal solution is only for the purpose of comparing the lifetime of the network with mobile cluster heads later.

5.2.2 Possible Locations of Mobile Cluster Heads

Next we consider the problem of maximizing the lifetime of the network where cluster heads are allowed to move in the sensing field. From the example we discussed earlier, we can see that the position of a cluster head can greatly affect the network lifetime. Before cluster heads decide where to move, they need to know where they can move to. The *possible location set L* of cluster heads is defined as a set of target positions where cluster heads can move to, stay and be reached by sensor nodes. For most applications, cluster heads are able to move continuously on the two-dimensional plane, which includes an infinite number of points. However, not every point in the field can be a possible location for a cluster head, because once a cluster head moves to a point, nearby sensor nodes must be close enough to communicate with the cluster head. We define the *neighbor set* of a point as a set of sensor nodes which can communicate with the cluster head placed at this point. A point is a possible location of the cluster head, if and only if its neighbor set is non-empty and the set has been obtained or estimated by the cluster head.

Possible locations of cluster heads can be classified into two types: known positions whose neighbor sets have been obtained, and unknown positions whose neighbor sets can be estimated. In some cases, such as indoor applications, due to the reflection from walls and obstacles, wireless propagation is difficult to be modeled into a neat form [78]. The transmission range of a node may be in an irregular shape. It is impractical to obtain the

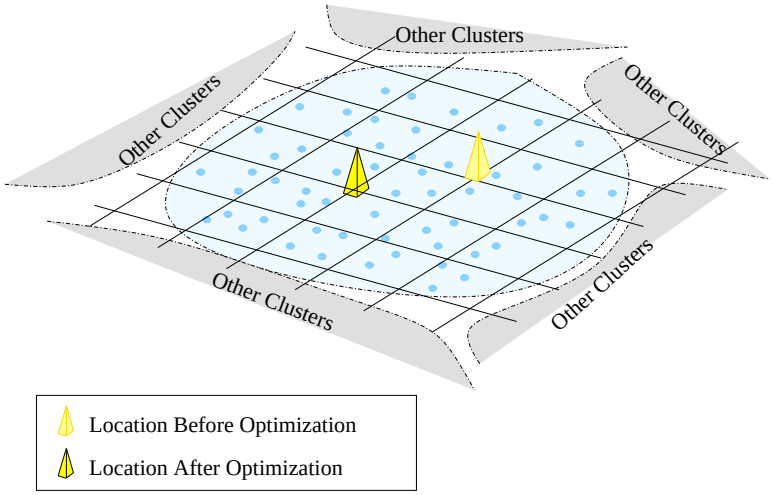


Figure 5.4: Grid scheme: cluster head can only move to crossing points of grid cells.

neighbor set of an unknown point, unless a cluster head moves there or a sensor node has stayed there. In this situation, only the points where sensor nodes or cluster heads have been located can be possible locations, because only the neighbor sets of these points can be obtained precisely. A cluster head will choose its next position from these known possible locations. Though these possible locations can guarantee the cluster head to communicate with its neighbor set, some positions in an unknown area may be better than these known positions. When sensor nodes are deployed in an open two-dimensional field, where received wireless signal strength is inversely proportional to the square of the distance between the transmitter and the receiver. In this case, the two-ray model can be used to model the propagation of wireless nodes. The transmission range can be approximately modeled by a unit disk centered at the transmitter, whose radius is proportional to its transmission power. Given relative positions and maximum signal strengths of all sensor nodes and cluster heads, the neighbor set of any point in the field can be estimated according to the

two-ray propagation model. Thus, the possible location set becomes an infinite set. In order to simplify the problem, we assume that each cluster divides the sensing field into grid cells. Only the crossing points of grids can be the possible locations. We refer to these two methods to obtain possible locations of cluster heads described above as *known position scheme* and *grid scheme*, respectively. An example of using the grid scheme is illustrated in Figure 5.4, where the sensing field is divided into grids, and the cluster head will choose and then move to the best location among all crossing points of the grids.

5.2.3 Problem Definition and Formalization

For a network with n_s sensor nodes and n_c mobile cluster heads, we assume that each cluster head can move to any one of the n_l possible locations. We now consider the problem of finding the positions of n_c cluster heads from n_l $(n_l \geq n_c)$ possible locations, and determining which cluster head serves each of n_s sensors to maximize the network lifetime. We call this problem *Cluster Head Location* problem and refer to it as CHL problem. CHL problem can be formally stated as follows.

The sensor network can be modeled as a directed graph $G(S,L,A)$, where $S = \{s_1, s_2, \ldots, s_{n_s}\}$ is a set of all sensor nodes, $L = \{l_1, l_2, \ldots, l_{n_l}\}$ is a set of all possible locations of cluster heads, and A is the set of all arcs in G. Let N_{s_j} be a set of sensors that can reach s_j in one hop, and N_{l_i} be a set of sensors that can reach the cluster head at l_i in one hop, and $N_{s'_j}$ be a set of sensor nodes and possible locations of cluster heads which are in the transmission range of sensor s_j. An arc $a(s_j, s_k)$ belongs to A, if and only if sensor s_k can be reached by s_j in one hop, and an arc $a(s_j, l_k)$ exists in G, if and only if the possible location l_k is located within the transmission range of sensor s_j. Let $n_s = |S|$, $n_l = |L|$ and n_c be the number of cluster heads, where $n_l \geq n_c$. Each vertex $s_j (s_j \in S)$ has n_l indicator variables $c_{s_j}^{l_i}$, $\forall l_i \in L$.

Each arc $a(s_k,t)$ in A contains n_l flow values $f_{s_k t}^{l_i}$, where $s_k \in S$, $t \in S \bigcup L$ and $l_i \in L$. Each element l_i in L corresponds to a variable I^{l_i} indicating whether a cluster head stays at l_i.

Given a graph $G(S,L,A)$, the problem can be formalized as follows:

$$
\begin{aligned}
&\textit{Maximize } T \\
&\textit{Subject to} \\
&I^{l_i} = \{0,1\}, \forall l_i \in L \\
&\sum_{l_i \in L} I^{l_i} = n_c \\
&c_{s_j}^{l_i} = \{0,1\}, \forall l_i \in L, \forall s_j \in S \\
&\sum_{l_i \in L} c_{s_j}^{l_i} = 1, \forall l_i \in L, \forall s_j \in S \\
&\sum_{s_j \in N_{l_i}} f_{s_j l_i}^{l_i} \leq I^{l_i} \times M \\
&\sum_{l_i \in L} \sum_{t \in N_{s_j'}} f_{s_j t}^{l_i} = \sum_{l_i \in L} \sum_{s_k \in N_{s_j}} f_{s_k s_j}^{l_i} + r_{s_j} T, \forall s_j \in S \\
&P_r \sum_{t \in N_{s_j'}} f_{s_j t}^{l_i} + P_g \sum_{s_k \in N_{s_j}} f_{s_k s_j}^{l_i} \leq c_{s_j}^{l_i} E_{s_j}, \forall s_j \in S \\
&\textit{All variables} \geq 0
\end{aligned}
$$

In this program, T is the network lifetime, and I^{l_i} is an indicator variable denoting whether a cluster head is placed at possible location l_i. Since n_c cluster heads will choose their positions from n_l possible locations, the summation of I^{l_i} for all $l_i \in L$ equals n_c. $c_{s_j}^{l_i}$ is another indicator variable denoting whether sensor s_j belongs to the cluster head at possible location l_i. Since each sensor can only belong to one cluster, only one $c_{s_j}^{l_i}$ for all $l_i \in L$ can be 1, while others equal 0. In the fifth constraint, M is a large positive constant and

represents the maximum traffic a cluster head can handle. Since cluster heads are powerful, M can be assumed to be any constant larger than the total flow in the network. If there is a cluster head staying at possible location l_i, the total flow received by the cluster head must be less than M. Otherwise, no flow goes to l_i. The sixth constraint accounts for the fact that, for each sensor node, the total outgoing traffic equals the sum of total incoming traffic and the total locally generated traffic. r_{s_j} is the data generating rate of sensor s_j. The seventh constraint says that the total energy consumption of sensor s_j must be limited by its maximum energy limit E_{s_j}. From this constraint, we can also see that when $c_{s_j}^{l_i} = 0$, that is, when sensor s_j does not belong to the cluster head at l_i, both $f_{s_j t}^{l_i}(t \in N'_{s_j})$ and $f_{s_k s_j}^{l_i}(s_k \in N_{s_j})$ equal zero, which means no flow goes to l_i via sensor s_j.

5.2.4 NP-Hardness of the CHL Problem

We next show that it is NP-hard to obtain optimal clustering and locations of cluster heads which maximizes the network lifetime.

Lemma 1 *The CHL problem is NP-hard.*

Proof. In order to show that CHL is NP-hard, we give a reduction from the *k*-center problem [76]. The *k*-center problem is a basic facility location problem, which aims at locating *k* facilities in a graph and to assign demand vertices to facilities, so as to minimize the maximum distance from a vertex to the facility to which it is assigned. Given any instance of the *k*-center facility location problem, an instance of CHL problem can be constructed as follows.

Let G denote the graph of the *k*-center problem, $Dem = \{D_1, D_2, \ldots, D_{n_D}\}$ denote a set of demand vertices, and $Fac = \{F_1, F_2, \ldots, F_{n_F}\}$ be the set of possible locations of facilities.

A graph of CHL $G'(S,L,A)$ can be constructed as follows, where S, L and A are as defined in Section 5.2.3.

- For each demand vertex D_j in G, add a vertex s_j to S, with infinity energy capacity E_{s_j} and data generating rate $r_{s_j} = 1$.
- For each vertex denoting a possible facility location F_i in G, add a vertex l_i into L.
- For each pair of D_j and F_i in G, add an intermediate vertex m_j^i into S, with the energy capacity $E_{m_j^i} = \frac{1}{d(D_j,F_i)}$ and data generating rate $r_{m_j^i} = 0$, where $d(D_j,F_i)$ is the distance between demand vertex D_j and possible location F_i in G. Connect m_j^i and l_i, s_j and m_j^i with two arcs $a(m_j^i,l_i)$ and $a(s_j,m_j^i)$, respectively, where $a(m_j^i,l_i), a(s_j,m_j^i) \in A$.

In the construction, the number of cluster heads n_c in CHL equals the number of facilities, k, in the k-center problem, and the number of possible locations of cluster heads n_l equals the number of possible facility locations, n_F, in the k-center problem. Power consumption for generating and relaying a unit of traffic, P_g and P_r, both equal 1. An example of the construction of G' from G is shown in Figure 5.5. Figure 5.5(a) contains four demand vertices and two possible facility locations, while two possible locations of cluster heads vertices and twelve sensor vertices are added in Figure 5.5(b). Among all twelve sensor nodes, $m_j^i, (j = 1,2,3,4; i = 1,2)$, are intermediate nodes which have zero data generating rate and act as relaying-only nodes.

In the optimal solution of CHL, if there is a non-zero flow goes from s_j to l_i via the path $s_j \rightarrow m_j^i \rightarrow l_i$, a demand node D_j is assigned to the facility F_i. In addition, the maximum flow generated by s_j within time T, which equals Tr_{s_j}, must be less than or equal to the maximum flow the intermediate node m_j^i can relay, which equals $\frac{E_{m_j^i}}{P_r}$. Since both P_r and r_{s_j}

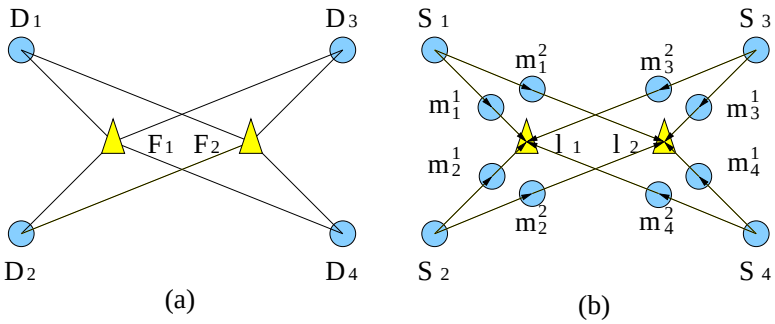


Figure 5.5: Reduction from the k-center problem to CHL. (a) Graph of the k-center. (b) Graph of CHP.

are 1 in this construction, we know that the lifetime T is bounded by $\min_j \max_i E_{m_j^i}$, where $E_{m_j^i}$ is the energy constraint of the intermediate vertex m_j^i, for $i = 1, 2, \ldots, n_F$, $j = 1, \ldots, n_D$. Since $E_{m_j^i} = \frac{1}{d(D_j, F_i)}$, we have, $\max T = \max \min_j \max_i (\frac{1}{d(D_j, F_i)})$.

Therefore, the optimal clustering achieves the maximum lifetime T in G', if and only if demand vertices in G can be organized into k clusters with the maximum-minimum distance from any demand vertex to the facility to which it is assigned can be minimized to $\frac{1}{T}$.

We have shown that it is NP-hard to find a clustering which maximizes the lifetime T in the CHL problem. In addition to the NP-hardness of the problem, for a centralized algorithm, collecting global location information and connection patterns will be a very energy-consuming task for a large scale network. Even if a suboptimal solution can be obtained by relaxation or other methods, the centralized algorithm may not be suitable for a large sensor network. In a network which consists of hundreds and thousands of nodes, it will be a huge burden to forward location and topology information to one central controller instead of sending to multiple distributed cluster heads. For these reasons, in the following we present a practical distributed heuristic algorithm for the above problem.

5.3 Clustering a Hybrid Sensor Network: A Heuristic Clustering Algorithm

We propose a heuristic clustering algorithm as outlined in Figure 5.6. The algorithm can be divided into two phases: organizing sensor nodes into different clusters and finding the best location for every cluster head. These two steps can be executed rotationally as shown in Figure 5.6. We will explain the algorithm in details next.

5.3.1 Organizing Sensor Nodes into Clusters

A straightforward way to organize sensor nodes into clusters is to assign each sensor to the "nearest" cluster head from it. Here, the distance from a sensor node to a cluster head is measured by the hop count. After sensor nodes and cluster heads are deployed in the field, first, they need to discover their neighbor nodes around and transmission links among them. During the neighbor discovery phase, *Hello* messages will be generated and flooded by each sensor node. In order to avoid extra overhead, these *Hello* messages are also used to construct clusters in our algorithm. A cluster will be organized into a spanning tree as follows. At the beginning, each cluster head generates a *Hello* message, which includes three extra fields in addition to other regular contents: *Root*, *Parent* and *Level*, where *Root* and *Parent* are marked by the ID of the cluster head, and *Level* denotes the current level of the cluster head in the spanning tree and is set to zero. Then the cluster head broadcasts the *Hello* message with the same transmission power as sensor nodes. Each sensor node will also maintain its node level, parent ID and root ID in the tree, where level is set to infinity at the beginning. Once a sensor node receives *Hello* messages from the cluster head or other sensor nodes, they will check the *Level* value in the message and compare it with its node

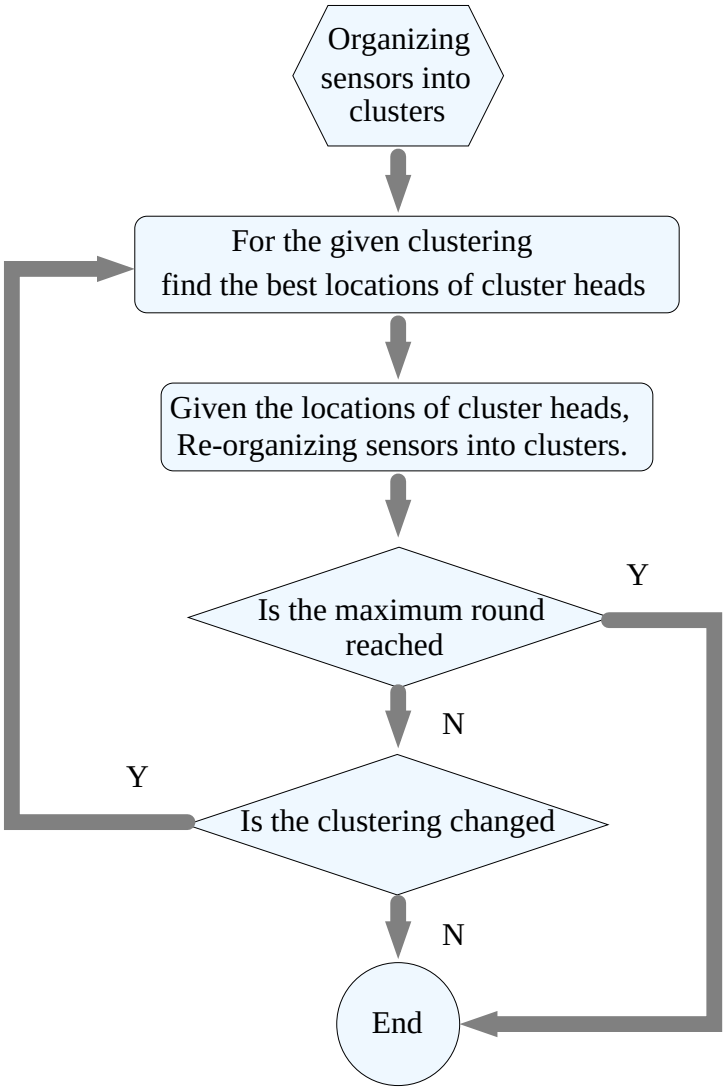


Figure 5.6: The heuristic clustering algorithm.

level. If the *Level* in the message is less than its own node level, it updates its root ID, parent ID and level according to the values of *Root*, *Parent* and *Level* in the message, respectively. Then, it increments the *Level* of the message by one, sets the *Parent* of the message with its node ID, and broadcasts it again. If the *Level* in the message is not less than its own node level, it simply drops the message. Finally, when no more *Hello* messages are broadcast in the network, every sensor node will know which cluster it belongs to and which sensor node is its parent node in the spanning tree. The cluster forming process will end after a fixed-length of the time, which should be long enough to guarantee every sensor can find its nearest cluster head. At the end of the cluster forming phase, each sensor node uploads its location and neighbor information to its nearest cluster head through the branch of the spanning tree. This phase is described in Table 5.1.

We now analyze the complexity of the clustering algorithm. Each sensor has at most n one-hop neighbors, where n is the total number of nodes in the network. Each sensor needs to receive *Hello* messages from all its one-hop neighbors, and also send out its *Hello* message to all of them, which needs $O(n)$ times one-hop transmission. Thus, the total complexity of the clustering algorithm is $O(n^2)$.

5.3.2 Finding Best Locations of Cluster Heads

After the cluster forming phase, the entire network is organized into n_c independent clusters. Each of them consists of a cluster head as the root and a bunch of sensor nodes. The next step is to find the best location for each cluster head from its possible locations. The new location of a cluster head should balance the traffic and minimize the maximum traffic load of each sensor node.

Table 5.1: Clustering forming algorithm

Cluster Forming Algorithm

Procedure of a cluster head:
for each cluster head C_i
Generate a $Hello$ message M_i, including
  $Parent(M_i) = C_i$;
  $Root(M_i) = C_i$;
  $Level(M_i) = 0$;
Broadcast the $Hello$ message M_i with the same
transmission power as sensor nodes;
end for

Procedure of a sensor node:
for each sensor n_j
while (Cluster forming phase is not ended)
  Overhear the channel and receive Hello messages;
  if $Level(M) \geq$Level of n_j
    Discard the message;
  else
    Level of $n_j = Level(M) + 1$;
    Parent ID of $n_j = Parent(M)$;
    Root ID of n_j= $Parent(M)$;
    $Parent(M) = n_j$;
    $Level(M) = Level(M) + 1$;
    Broadcast Hello message M;
  end if
end while
Upload its location and neighbor information
to the nearest cluster head;
end for

Search for Best Locations of Cluster Heads

During the clustering forming phase, each cluster head has acquired a "map" of the entire cluster which includes the positions and connection patterns of all sensor nodes in the cluster. From the map, a cluster head can obtain some possible locations by using the known position scheme or the grid scheme discussed earlier. Then the cluster head tests possible locations one by one. For each possible location, the cluster head assumes that it has moved there and nearby sensors have been organized around it into a cluster. The minimum lifetime of sensor nodes will be maximized by running the network flow algorithm we will introduce next. After all possible locations are tested, the cluster head will compare all max-min lifetime values and choose the maximum one. Its corresponding possible location will become the next position of the cluster head. It should be pointed out that in the grid scheme, the number of possible locations is determined by the size of grids, while the number of known positions is fixed for a network. The more possible locations are available, the better locations of cluster heads may be obtained, though small grid size increases the processing time and complexity. We will compare how different grid sizes affect the network lifetime in Section 5.4.

Load Balancing by Using Network Flow Algorithm

For each possible location of a cluster head, we assume that the cluster head has moved there and find the maximum-minimum lifetime of the cluster. Since each cluster only contains one cluster head and the location of the cluster head has been fixed, the problem of maximizing the minimum lifetime of the cluster becomes a simplified version of the CHL problem with $n_c = n_l = 1$. The problem can be formalized as follows.

$$\textit{Maximize } T$$

$$\textit{Subject to}$$

$$\sum_{t \in N_{s'_j}} f_{s_j t}^{l_1} = \sum_{s_k \in N_{s_j}} f_{s_k s_j}^{l_1} + r_{s_j} T, \forall s_j \in S$$

$$P_r \sum_{t \in N_{s'_j}} f_{s_j t}^{l_1} + P_g \sum_{s_k \in N_{s_j}} f_{s_k s_j}^{l_1} \leq E_{s_j}, \forall s_j \in S$$

$$\textit{All variables} \geq 0$$

All the above notations have the same definitions as those in Section 5.2.3. The two constraints account for the flow and energy constraints at each sensor node, respectively. Compared to the CHL problem, this problem is a regular network flow program, which can be formalized into a linear program and solved in polynomial time.

Time Complexity of Finding the Best Location of the Cluster Head

Let M_L be the number of possible locations and M_s be the total number of sensor nodes and cluster head in the cluster. For each possible location, the above linear program can be solved by the method we introduced in Section 5.2.1. Thus, the maximum lifetime can be obtained in $O(UM_s^2)$ time, where U is as defined in Section 5.2.1. Therefore, the total running time for finding the best location of the cluster head is $O(UM_LM_s^2)$.

5.3.3 Recovering from Unexpected Failure of Cluster Heads and Sensors

In this subsection, we discuss how a two-layer hybrid sensor network can recover from unexpected failures of some sensor nodes and cluster heads. One of the most important functionalities of a sensor network is to sense the human-unreachable area, such as volcano, seabed and outer-space, where unexpected hazards occur from time to time. These hazards may lead to unexpected failure of cluster heads and sensor nodes. Without the remote control from human being, alive cluster heads and sensors must be able to re-organize the network self-adaptively. Cluster head can find the failure of sensor nodes in the cluster, if it cannot receive sensing data from these nodes for a while. When the hazard occurs in a small region, only a small number of sensor nodes are destroyed by the hazard, the cluster head may ignore the failures. If a large percentage of sensor nodes in the cluster fail, or some cluster heads are down, alive cluster heads need to re-organize the network to balance the traffic for the new network layout. What they need to do is to execute the clustering algorithm again. Note that all operations are dynamic and self-adaptive and no control from human being is needed.

5.4 Performance Evaluations

We implemented the proposed clustering scheme on the NS-2 simulator. In the simulation, we assume that 800 sensor nodes and 6 cluster heads are uniformly deployed within a $670 \times 670 m^2$ two-dimensional square. The two-ray propagation model is used to describe the feature of the physical layer. With the maximum transmission power $0.858mw$, each node can communicate with other nodes as far as $40m$ away. The radio bandwidth

is 200kbps. CBR traffic on the top of UDP is generated to measure the throughput. Each packet has a fixed size of 80 bytes, including header and payload. Within each cluster, multi-hop polling protocol [99] is used as the inner-cluster protocol to avoid packets collision at the MAC layer. The performance of the algorithm was evaluated from three aspects: network layout, network lifetime and recovery from failures.

5.4.1 Network Layout

In this scenario, 800 sensor nodes and 6 cluster heads are randomly deployed to cover the sensing field, which will be organized into 6 clusters. The sensing field is divided into $10m \times 10m$ grids by cluster heads. In Figure 5.7, six triangle symbols in different colors denote cluster heads, and other smaller symbols represent sensor nodes. The cluster head and sensor nodes that belong to the same cluster are marked in the same color. Figure 5.7(a), (b), (c) and (d) depict the initial network layout and the layout after the first, second and third round of adjustments, respectively. In the center of Figure 5.7(a), We can see that three cluster heads are initially placed so close that they are near the margins of their corresponding clusters. It is inefficient to upload sensing data to cluster heads for those sensor nodes located too far away from cluster heads. However, after several rounds of adjustments, as shown in Figure 5.7(b), (c) and (d), we can observe that, first, the crowded cluster heads move apart from each other and now are closer to the center of each cluster; second, some sensor nodes are re-allocated from one cluster to the other, which can balance the sizes of clusters.

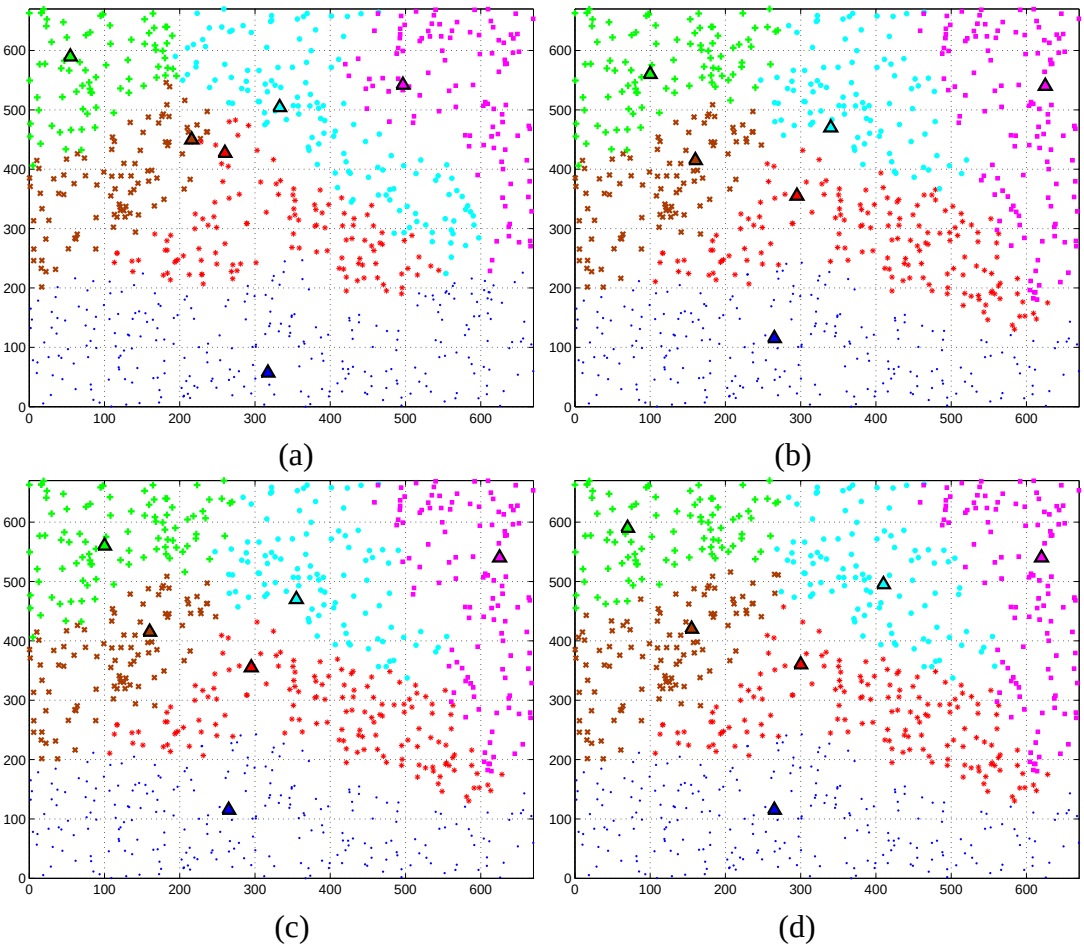


Figure 5.7: Network layout where triangle symbols denote cluster heads, while other symbols represent sensor nodes. Sensor nodes and cluster heads are marked in the same color if they belong to the same cluster. (a) Initial layout. (b) Layout after round 1. (c) Layout after round 2. (d) Layout after round 3.

5.4.2 Network Lifetime

Next we will see how mobile cluster heads can improve the lifetime of a sensor network. After the network is deployed, for the initial layout of a network, the optimal network lifetime can be obtained by running the flow algorithm discussed in Section 5.2.1. Here we assume that the global location information and connection patterns needed by this global algorithm are available, though it may be impractical to obtain such global information in a large network. The optimal lifetime obtained will only be used as a performance reference for the comparison purpose. We also run the known position scheme and grid scheme to optimize the lifetime of the network with mobile cluster heads. Figure 5.8 shows the relative lifetime ratio of the known position scheme and the grid scheme with grid distances $5m$, $10m$ and $20m$, respectively, to the optimal lifetime of the network with static cluster heads. We can see from Figure 5.8 that the relative lifetime ratio is less than 1 at the beginning. This is because that the initialization of the clustering phase is only based on the local information, and the traffic flow is optimized within the range of each cluster, while the maximum lifetime in the static network is obtained from the global optimization. However, after the first round of running the grid scheme, the relative lifetime ratio increases to about 1.2 for the grid size $5m$, and about 1.0 for the grid size $10m$ and $20m$. We can see that by running the grid scheme for 5 rounds, the relative lifetime ratio goes above 1.32 for all three different grid sizes. The known position scheme is not as good as the grid scheme after round 1, but the relative lifetime ratio of the known position scheme keeps increasing and reaches 1.10, 1.26 and 1.30 after round 2, 3 and 4, respectively. We can observe that, both the grid scheme and the known position scheme can achieve at least 30% improvement of the network lifetime after 5 rounds of adjustments.

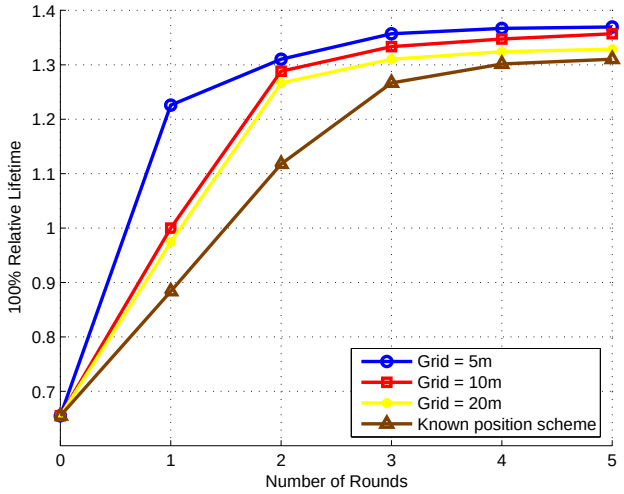


Figure 5.8: The relative lifetime of the grid scheme (Grid size = 5m, 10m and 20m) and known position scheme for the network with mobile cluster heads, compared to the optimal lifetime for the network with only static cluster heads.

We also investigate how the transmission range of sensors can affect the network lifetime. Figure 5.9 shows the relative lifetime of the known position scheme compared to the optimal lifetime for the network with only static cluster heads as the transmission range increases from $40m$ to $100m$. The figure plots the relative lifetime obtained by executing rounds 1, 3 and 5 of the heuristic approach. We can see that all the three curves decline slightly as the transmission range increases. When the transmission range equals $100m$, the relative lifetime after round 3 is close to 1.05. This is because that increasing the transmission range can enhance the connectivity of the network, and the packets can be forwarded to cluster heads with fewer relays. Also, the maximum burden of bottleneck nodes can be reduced. When the transmission range is long enough, the load balancing algorithm for static cluster heads can achieve close performance to the approaches for mobile cluster heads. In practice, the transmission ranges of most existing prototypes of wireless sensors

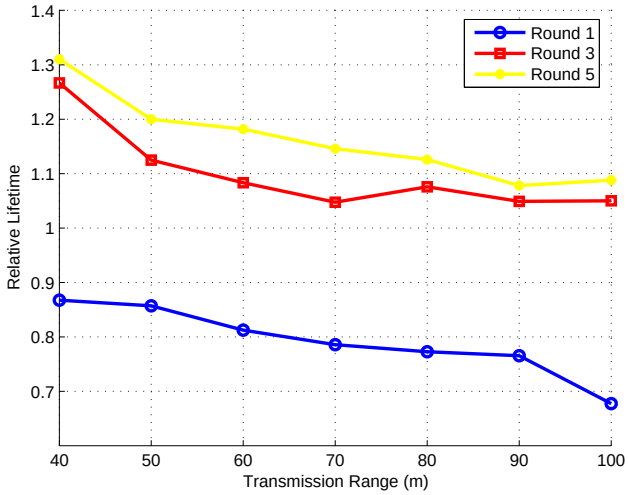


Figure 5.9: The relative lifetime of the known position scheme compared to the optimal lifetime for the network with only static cluster heads vs. the transmission range.

are $10m - 50m$. In these cases, using mobile cluster heads can greatly increase network lifetime compared to using static cluster heads.

5.4.3 Recovering from Unexpected Failures of Cluster Heads and Sensors

In this scenario, we assume that one cluster head and 100 sensor nodes are destroyed suddenly, after round 3 in the last scenario. All failed cluster heads and sensors are randomly chosen. Due to the failure of cluster heads and sensors, packets from some sensors cannot be forwarded to cluster heads through old relaying paths. In order to fix this problem, the clustering algorithm and cluster head positioning algorithm need to be executed again to determine new locations of cluster heads and clustering of sensors for the new

network layout. From Figure 5.10(a), we can see that the cluster head of the cluster located at the top-right corner is down. After re-clustering, the network is re-organized into five clusters. In Figure 5.10(b), sensor nodes in the cluster which lost its cluster head are re-allocated to two nearby clusters. Then, in Figure 5.10(b) and (c), each cluster head searches for and moves to the best location within the cluster. The simulation results of this scenario show that the proposed clustering algorithm can adaptively recover the network from unexpected failure of partial sensors and cluster heads.

5.5 Conclusions

In this paper, we have considered the problem of positioning mobile cluster heads in a two-layer hybrid sensor network to maximize network lifetime. Two-layer hybrid networks are more scalable and energy-efficient than homogeneous sensor networks. In such a network, since all sensing data goes to cluster heads, the positions of cluster heads may affect the direction of traffic flow significantly. In order to prolong the lifetime of sensors, the location of the cluster head needs to be planned to balance the traffic load. We first showed that the problem of positioning cluster heads to maximize the network lifetime is NP-hard. We then presented a heuristic algorithm for positioning the cluster heads and balancing the traffic load in the network. By moving the cluster head to a better location, the traffic load is balanced and the network lifetime is prolonged. Simulations were run on the NS-2 simulator, and the results show that our cluster head positioning algorithm can increase the network lifetime by a significant amount. In addition, the algorithm can recover the network from unexpected failure of sensors and cluster heads.

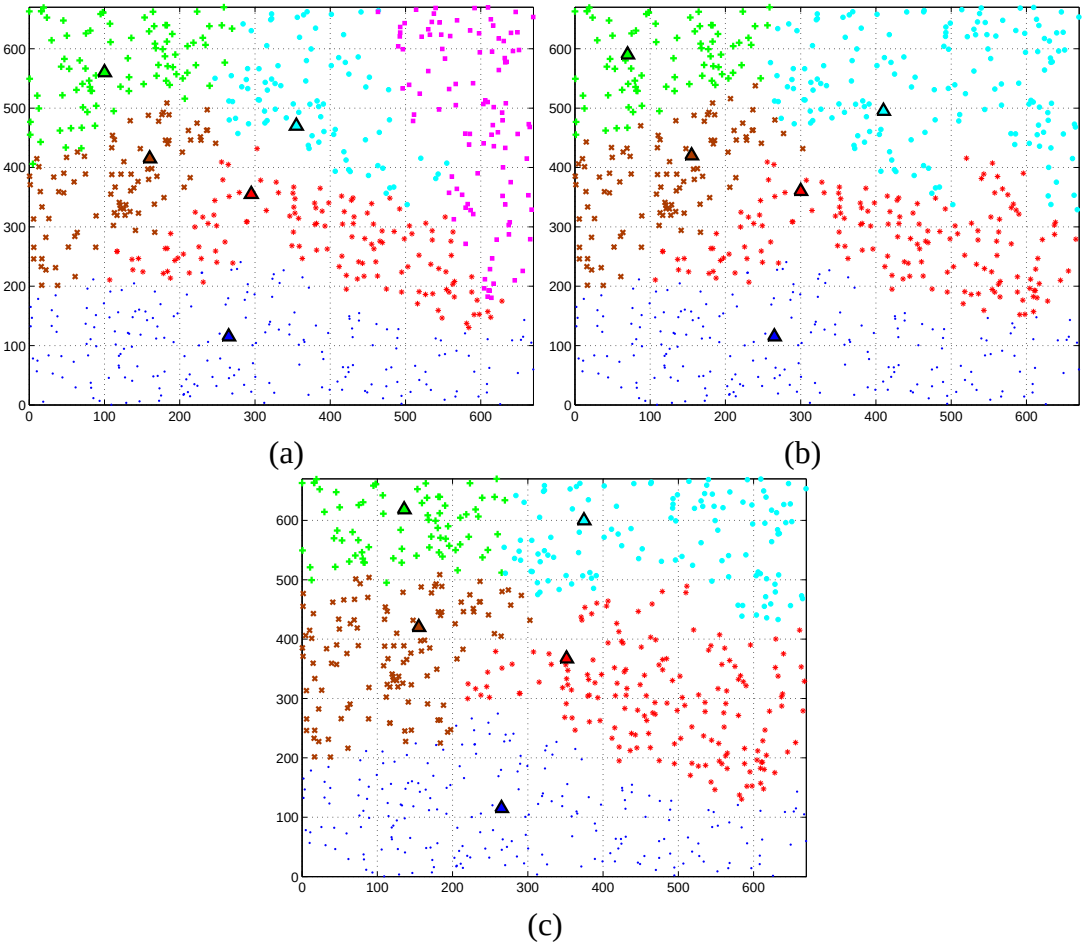


Figure 5.10: Network layout after unexpected hazards which cause 100 sensors and 1 cluster head to fail. (a) Layout after 100 sensors and 1 cluster head fail. (b) Layout after round 1. (c) Layout after round 2.

Chapter 6

Gathering Data with Mobile Data Collectors

In pervious chapter, we have seen how the hierarchical topology can improve the network lifetime. This chapter presents a new data gathering mechanism for large scale multihop sensor networks. A mobile data observer, called *SenCar*, which could be a mobile robot or a vehicle equipped with a powerful transceiver and battery, works like a mobile base station in the network. SenCar starts the data gathering tour periodically from the static data processing center, traverses the entire sensor network, gathers the data from sensors while moving, returns to the starting point, and finally uploads data to the data processing center. Unlike SenCar, sensors in the network are static, and can be made very simple and inexpensive. They upload sensing data to SenCar when SenCar moves close to them. Since sensors can only communicate with others within a very limited range, packets from some sensors may need multihop relays to reach SenCar. We first show that the moving path of SenCar can greatly affect the network lifetime. We then present heuristic algorithms for planning the moving path/circle of SenCar and balancing traffic load in the network. We

show that by driving SenCar along a better path and balancing the traffic load from sensors to SenCar, the network lifetime can be prolonged significantly.

The rest of the chapter is organized as follows. Section 6.1 discusses some related work. Section 6.2 presents our data gathering scheme for a connected network. Section 6.3 describes the data gathering scheme for a disconnected network. Section 6.4 gives performance evaluation results and some discussions. Finally, Section 6.5 concludes the chapter.

6.1 Related Work

Mobility of sensor networks has been studied in some literatures recently [79, 80, 81, 82, 83, 84, 85, 88, 90, 86, 87]. In [79] and [80], radio-tagged zebras and whales are used as mobile nodes to collect sensing data in a wild environment. These animal-based nodes wander randomly in the sensing field, and exchange sensing data only when they move close to each other. Thus, sensor nodes in such a network are not necessarily connected all the time. Moreover, the mobility of randomly moving animals is hard to predict and control, thus the maximum packet delay cannot be guaranteed. For sensor networks deployed in an urbane area, where public transportation vehicles, such as buses and trains, always move along the fixed routes. These vehicles can be mounted with transceivers to act as mobile base stations [82, 83]. Compared to the randomly moving animals, the moving path and timing are predictable in this case. However, data exchanging still depends on the existing routes and schedules of the public transportation, and thus is very restrictive. In [86], the authors exploited controlled movement to improve data delivery performance. Some mobile observers, called message ferries, are used to collect data from sensors. Two variants were studied based on whether ferries or nodes initiate proactive movement. In

[88, 89], a number of mobile observers, called *data MULEs*, pick up data directly from the sensors when in close range, buffer it, and drop off the data to wired access points. The movement of MULEs are modeled as two-dimensional random walk. The primary disadvantage of two approaches in [86, 88, 89] is increasing latency because mobile observers have to traverse transmission range of each sensor one by one to collect data. In [85], mobile observers traverse the sensing field along parallel straight lines and gather data from sensors. In order to reduce latency, packets sent by some sensor are allowed to be relayed by other sensors to reach mobile observers. This scheme works well in a large scale, uniformly distributed sensor network. However, in practice, data mules may not always be able to move along straight lines, for example, obstacles or boundaries may block the moving paths of data mules. Moreover, the performance and cost of the data mule scheme depends on the number of data mules and the distribution of sensors. When only a small number of data mules are available and not all sensors are connected, data mules may not cover all the sensors in the network if they only move along straight lines. In [87], the authors proposed a data gathering scheme to minimize the maximum average load of any sensor by jointly considering the problems of movement planning and routing. Based on the assumption that sensors are distributed as poisson process, the average load of a sensor can be estimated as a function of the node density. However, the estimation of the average load may be inaccurate in cases when sensors are not densely deployed as poisson distribution. [90] discussed several advantages and design issues for incorporating controlled mobility into the networking infrastructure, and mainly focused on motion/speed control and communication protocol design. [81, 84] also consider mobile observers in sensor networks. [81] mainly discussed hardware/software implementation of underwater mobile observers, while [84] proposed an algorithm to schedule the mobile observer, so that there

is no data loss due to the buffer overflow. In order to make a data collecting scheme suitable to various network topologies, it is more realistic and efficient to plan the moving path of the mobile observer dynamically based on the distribution of sensors. This is the motivation of our work in this paper. In the following, we first give a data collecting scheme for a connected network with an arbitrary topology and then discuss the case where the network consists of several isolated clusters.

6.2 Data Gathering Scheme for a Connected Network

Sensor networks are usually deployed in dangerous or even human-unreachable areas, such as volcano, outer-space, seabed and so on. In such environments, human beings may not move close to the sensing field. A mobile observer, or SenCar, will be sent out to gather data from sensors periodically. Since the network may contain a large number of nodes, each tour may take a long time. In order to save the energy, sensors may turn on their transceivers only when they need to send or relay packets. Except for the transmission period, transceivers of sensors could be turned off. The entire sensor network can be divided into several clusters, where sensors in each cluster must be connected to SenCar while it is moving through the cluster. When SenCar moves close to the cluster, all sensors belonging to the cluster will be waken up and prepare to send packets. Sensing data can be collected by SenCar while it is traversing the cluster. To make this scheme work, two issues must be resolved here. The first issue is how to wake up and turn off sensors only when needed. A radio wake-up scheme was proposed in [91], which allows the transceivers of sensors to be deactivated when they are idle. The second issue is how to divide sensors into clusters. As will be described later, a moving path of SenCar consists of a series of connected line segments. Sensors close to each line segment will be organized into a cluster by SenCar, such

that the entire network can be divided into a number of clusters. A straightforward way to organize sensor nodes into clusters is to assign each sensor to the "nearest" line segment in the moving path from it. The details of clustering algorithm will be given in Section 6.2.3. While moving, SenCar will poll each sensor one by one to collect data. Relaying path and transmission time of packets are determined by SenCar. Thus, packet collision can be avoided and no routing paths need to be maintained by sensors. In addition, while planning the relaying paths, traffic load needs to be balanced to prolong the lifetime of sensors. The data gathering scheme includes three related issues: load balancing, movement planning and clustering. Given a moving path of SenCar, the load balancing algorithm can be used to find the optimal relaying paths from sensors to SenCar, when SenCar move through this path, such that the network lifetime can be maximized. The movement planning algorithm accounts for how to choose the best path from a set of candidate paths. Given a set of paths, SenCar computes the maximum lifetime that each path can achieve by running the load balancing algorithm, and then picks the best one. The clustering algorithm is used to divided the network into clusters, such that the load balancing algorithm and movement planning algorithm can be run recursively. Next, we will introduce three related issues separately, and then describe how to put them together into an integrated data gathering scheme.

6.2.1 Load Balancing

As discussed above, due to the different amount of traffic each sensor node relays, some nodes may fail sooner than others. In order to maximize the network lifetime, relaying paths must be carefully planned to balance the traffic load. Load balancing problem in static sensor networks has been investigated in some existing work, such as [73, 74, 99]. Next we will describe how to formalize the problem of maximizing network lifetime in our

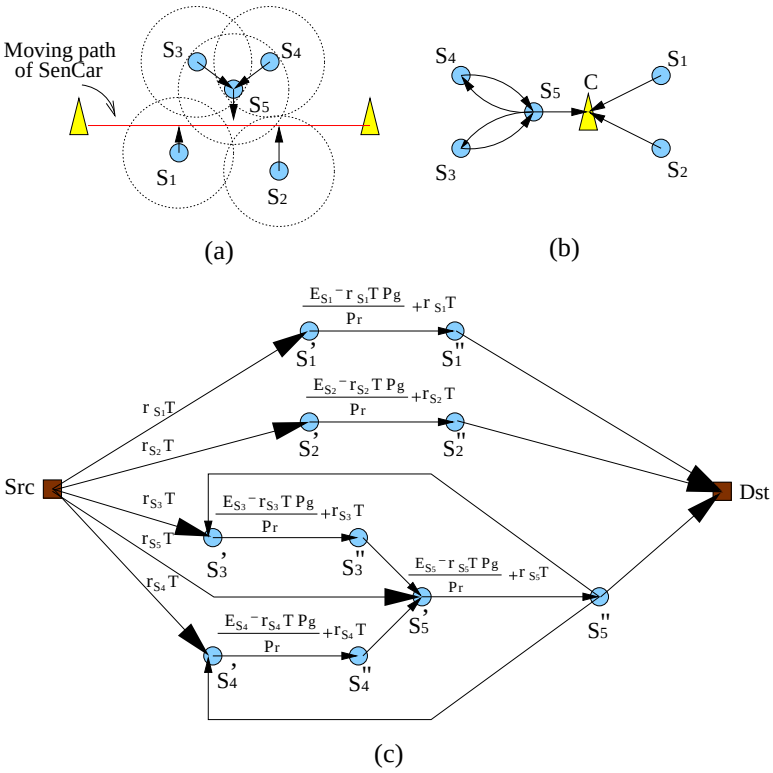


Figure 6.1: (a) Connection patterns of a sensor network. (b) Directed graph $G(S,c,A)$ corresponding to the connection patterns of the network. (c) Network flow graph $G'(S',Src,Dst,A')$ for maximizing network lifetime, where the capacities of unmarked arcs are infinity.

network into a network flow problem.

Given the connection patterns of the network and the moving path of SenCar, a sensor network can be modeled as a directed graph $G(S,c,A)$, where $S = \{s_1, s_2, \ldots, s_n\}$ is the set of all sensor nodes, c denotes SenCar and A is the set of all directed links $a(i,j)$ where $i \in S$, $j \in S \bigcup \{c\}$. For each pair of nodes $s_i, s_j \in S$, if s_i can reach s_j in one hop, arc $a(s_i,s_j)$ will be added into A. If the moving path of SenCar traverses the transmission range of s_i, or equivalently s_i can reach SenCar in one hop while SenCar is moving, add arc $a(s_i,c)$ into G. Figure 6.1(a) and (b) shows how to construct the directed graph from the connection

patterns of a network.

Given the directed graph G of a network, its corresponding flow graph $G'(S', Src, Dst, A')$ can be constructed as follows:

- For each $s_i \in S$, add two vertices s_i' and s_i'' to S', and an arc $a(s_i', s_i'')$ is added into A' with capacity $\frac{Es_i - (r_{s_i} T P_g)}{P_r} + r_{s_i} T$;
- For each arc $a(s_i, s_j) \in A$, where $s_i, s_j \in S$, add an arc $a(s_i'', s_j')$ into A' with infinity capacity;
- A pair of source and destination nodes *Src* and *Dst* are added into G', and for each $s_i' \in S'$, connect *Src* and s_i' by an arc $a(Src, s_i')$ with capacity $r_{s_i} T$;
- For each arc $a(s_i, c) \in A$, where $s_i \in S$, add an arc $a(s_i'', Dst)$ into A' with infinity capacity;

where r_{s_i} and E_{s_i} denote the data generating rate and energy limit of node s_i, P_g and P_r represent the power consumption for generating and relaying a unit of traffic, respectively, and T is the network lifetime. Since SenCar visits sensors periodically, say, every ΔT time. We can set $T = \Delta T$ at the beginning and increase T by ΔT every time. For any given T, this problem is a regular maximum flow problem [75] and can be solved by Ford-Fulkerson algorithm in polynomial time. In this construction, $(r_{s_i} T)$ limits the flow from *Src* to s_i and represents the flow generated by s_i within time T, which consumes $(r_{s_i} T P_g)$ energy. Due to the energy constraint of node s_i, the maximum flow node s_i can relay within time T is $\frac{Es_i - (r_{s_i} T P_g)}{P_r}$. Thus, the total flow a node s_i can generate and relay in time T is limited by $\frac{Es_i - (r_{s_i} T P_g)}{P_r} + r_{s_i} T$. When the maximum flow equals $\sum_{s_i \in S} r_{s_i} T$, it means until time T, all generated traffic by n sensor nodes is received by SenCar. Thus, all n sensors must be alive until T. We can keep increasing T and running Ford-Fulkerson algorithm to obtain the

maximum flow for every T value, until the maximum flow is less than $\sum_{s_i \in S} r_{s_i} T$, which indicates some nodes have failed before time T. Finally, the value of T obtained before the last run of Ford-Fulkerson algorithm is the maximum network lifetime. An example of constructing the flow graph from the connection patterns of the network is depicted in Figure 6.1.

We now analyze the time complexity of this algorithm. Let U denote the maximum units of traffic any sensor node generates and relays within time T^*, where T^* is the maximum network lifetime obtained by the algorithm. Then

$$U = \max_{s_i \in S} \left\{ \frac{Es_i - (r_{s_i} T^* P_g)}{P_r} + r_{s_i} T^* \right\}$$

According to Ford-Fulkerson algorithm [75], the maximum flow will be reached when no more flow augmenting paths can be found in the graph. A flow graph may contain $O(n^2)$ edges, where n denotes the number of nodes. The maximum flow can be added to each edge is bounded by U. Thus, the running time of this algorithm is $O(Un^2)$. Based on the connection patterns of the network and the moving path of SenCar, the optimal traffic relaying paths which maximize the network lifetime can be obtained in polynomial time by running the flow algorithm. Next we will discuss how to determine the moving path of SenCar.

6.2.2 Determining Turning Points of the Moving Path

Before formally describing the problem we consider, we first give an example to see how the moving path of the SenCar affects the network lifetime. As shown in Figure 6.2(a),

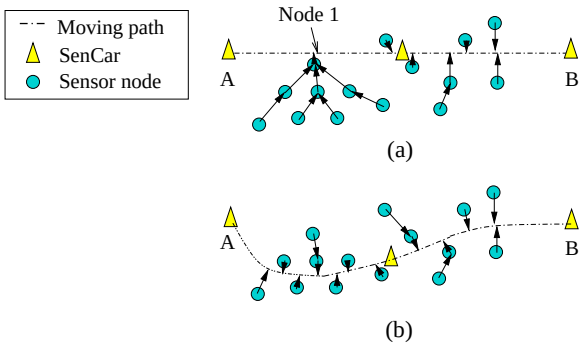


Figure 6.2: SenCar moves from A to B and collects data from nearby sensors. (a) SenCar moves along a straight path. (b) SenCar moves along a well-planned path.

SenCar traverses the sensing field from *A* to *B*, where fifteen nodes are deployed. We assume that each sensor forwards one packet to SenCar, while SenCar moves from *A* to *B*. Due to limited transmission power of sensors, packets may need multi-hop relays to reach SenCar. The sensors are organized into spanning trees to forward packets to SenCar. We can see that in Figure 6.2(a), node 1 is a bottleneck node, because it has to relay eight packets from itself and its seven child nodes to SenCar. Thus, node 1 consumes energy much faster than other nodes. After node 1 fails, the child nodes of node 1 cannot reach SenCar any more, unless SenCar changes the moving path. Figure 6.2(b) shows the relaying paths of sensors when the moving path of SenCar is well planned. We can see that each node has at most one child node and needs to send at most two packets to SenCar. In this example, if we only consider the energy consumption for transmission and roughly measure it by the number of packets transmitted, the well-planned moving path of SenCar can increase the lifetime three times compared to the straight-line moving path. From this simple example, we observe that a well-planned moving path of SenCar may minimize the maximum load of any sensor, save a lot of energy and prolong the network lifetime significantly. In addition to traffic load, the moving path of SenCar can also affect the directions of traffic flow,

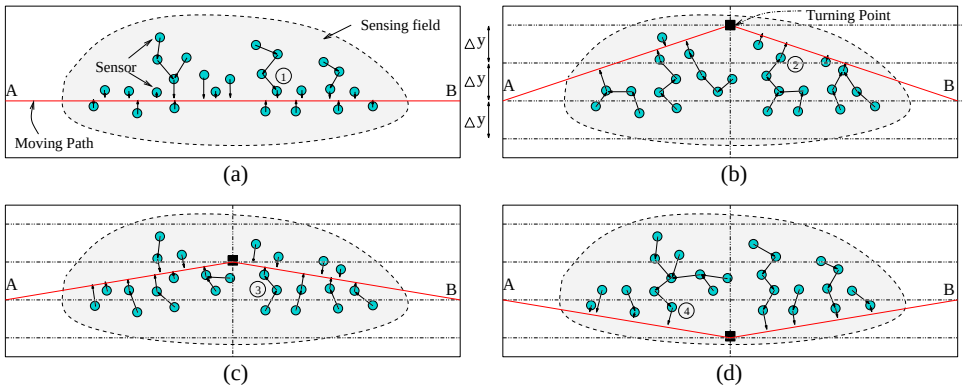


Figure 6.3: SenCar moves from A to B and collects data from nearby sensors. (a) SenCar moves from A to B through a straight path. (b) SenCar moves from A to B with turning point $(\frac{x_A+x_B}{2}, 2\Delta y)$. (c) SenCar moves from A to B with turning point $(\frac{x_A+x_B}{2}, \Delta y)$. (d) SenCar moves from A to B with turning point $(\frac{x_A+x_B}{2}, -\Delta y)$.

thereby have a significant impact on the network lifetime. Next we consider the problem of maximizing the lifetime of the network, by carefully planning the moving path of SenCar.

In practice, since it is difficult for vehicles or robots to move along any continuous curve smoothly, we simply assume that the moving path of SenCar consists of $t+1$ connected straight line segments from the starting point A to the end point B. That means SenCar needs to turn t times before it reaches the end of the path. Let $p_1, p_2, \ldots, p_t$ denote t turning points. Then, the moving path of SenCar can be represented by $A \rightarrow p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_t \rightarrow B$. Let (x_A, x_A), (x_B, y_B) and (x_{p_i}, y_{p_i}) denote the coordinates of A, B and p_i, for $i = 1, 2, \ldots, t$. We assume that the x-coordinate of any sensor is between x_A and x_B. We will use the divide and conquer strategy to find t turning points to reduce the maximum traffic load of any sensor needs to send out. Without loss of generality, let $t = 2^k - 1$, where k denotes the iterations of the path planning algorithm and $k = 1, 2, \ldots$. First, given the positions of A and B, we will find the position of the first turning point $p_{\frac{t+1}{2}}$. Since every point between A and B could become the first turning point, it makes the set of candidate

turning points an infinite set. For the sake of simplicity, we assume that the first turning point can only be chosen from a finite set of points in the bisector of the initial path. For the sake of simplicity, we assume that the first turning point can only be chosen from a finite set of points in the bisector of the initial path. Let the x-coordinate of the first turning point $x_{p_{\frac{t+1}{2}}} = \frac{x_A+x_B}{2}$, and the y-coordinate of the first turning point $y_{p_{\frac{t+1}{2}}} = m \times \Delta y$, where Δy is a fixed grid length and m can be any integer that ensures $(x_{p_{\frac{t+1}{2}}}, y_{p_{\frac{t+1}{2}}})$ to be within the range of the sensing field. After a set of eligible possible locations of the turning point are obtained, we can check each possible turning point and find the one that minimizes the maximum traffic load a sensor has to send out. For example, in Figure 6.3(a), the initial path of SenCar begins from A to B. Given the grid length Δy and the range of sensing field, there are three possible locations of the first turning point, located at $(\frac{x_A+x_B}{2}, 2\Delta y)$, $(\frac{x_A+x_B}{2}, \Delta y)$ and $(\frac{x_A+x_B}{2}, -\Delta y)$, as shown in Figure 6.3(b), (c) and (d). For each possible turning point, the load balancing algorithm introduced in the previous subsection can be used to obtain the maximum-minimum lifetime of the sensors for its corresponding moving path. Figure 6.3(a)-(d) show the connection pattern graph of four different moving paths, where nodes 1, 2, 3 and 4 are the bottleneck nodes in Figure 6.3(a)-(d), which need to send four, six, three and nine packets to SenCar, respectively. Thus, the third moving path, turned at $(\frac{x_A+x_B}{2}, \Delta y)$, provides a longer network lifetime than others. In the first step, point $(\frac{x_A+x_B}{2}, \Delta y)$ is chosen as the first turning point of the moving path. Note that sometimes better moving path may not be found by moving the turning point along the bisector of the current path. In this case, the new turning point can be simply set to the mid point between two end points of the current path.

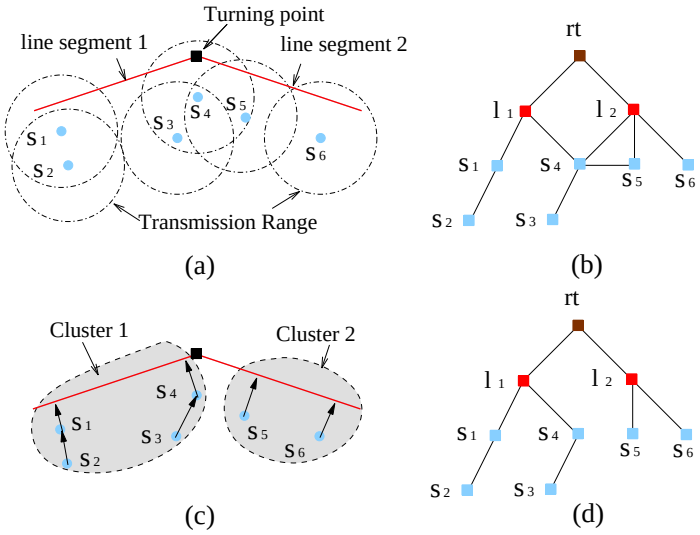


Figure 6.4: SenCar moves from A to B and collects data from nearby sensors. (a) Two line segments of the moving path cross transmission ranges of node s_1, s_4, s_5 and s_6. (b) Graph $G(S,L,E)$ of the network. (c) Clustering obtained from the shortest path tree in $G(S,L,E)$. (d) Shortest path tree obtained in $G(S,L,E)$.

6.2.3 Clustering the Network along the Segments of the Moving Path

After the first turning point is obtained, the moving path consists of two connected line segments. Then, sensors will be organized into two clusters, where each cluster corresponds to a line segment. In order to save energy, two clusters of sensors can be waked up sequentially. Sensors in one cluster forward packets to SenCar before it makes the turn, while sensors in the other cluster send data after SenCar turns. A straightforward way to organize sensor nodes into clusters is to assign each sensor to its "nearest" line segment in the moving path from it. Here, the distance from a sensor to the line segment in the routing path is measured by the hop count. Given a set of sensors S and a set of line segments L, clustering the network can be implemented by running Dijkstra shortest path algorithm in graph $G(S,L,E)$, which can be constructed as follows:

- A root vertex rt is added into V;
- For each line segment, add a vertex l_i into L and an edge $e(rt, l_i)$ into E with weight 1.
- For each sensor $s_j \in S$, add a vertex s_j into V; Connect s_j and l_i by an edge $e(s_j, l_i)$ with weight 1, if and only if sensor s_j can reach line segment l_i in one hop;
- For each pair of nodes $s_j, s_k \in S$, connect s_j and s_k by an edge $e(s_j, s_k)$ with weight 1, if and only if sensors s_j and s_k can reach each other in one hop;

As shown in Figure 6.4(a), two line segments of the moving path cross the transmission ranges of node s_1, s_4, s_5 and s_6. The corresponding graph $G(S, L, E)$ of the network is shown in Figure 6.4(b). By running Dijkstra algorithm in G, we can find the shortest path from the root vertex to all other vertices, then a shortest path tree can be obtained, which contains $|L|$ first level vertices. Figure 6.4(d) and (c) show the shortest path tree of $G(S, L, E)$ and the clustering of the network. Each first level vertex represents a line segment in the moving path. All child vertices of the first level vertex l_i in G represent a cluster of sensors corresponding to line segment l_i in the network.

6.2.4 Finding the Moving Path: Divide and Conquer

By combining the above algorithms of load balancing, finding turning points and clustering, the moving path planning algorithm can be described as follows: organizing the network into a cluster, determining the turning point from a set of possible locations of turning points, revising the path by adding the new turning point, and then dividing each cluster into two clusters. For each cluster, run the above algorithm recursively. After running k iterations of the moving path planning algorithm, $\sum_{i=1}^{k} 2^{(i-1)}$ turning points are obtained.

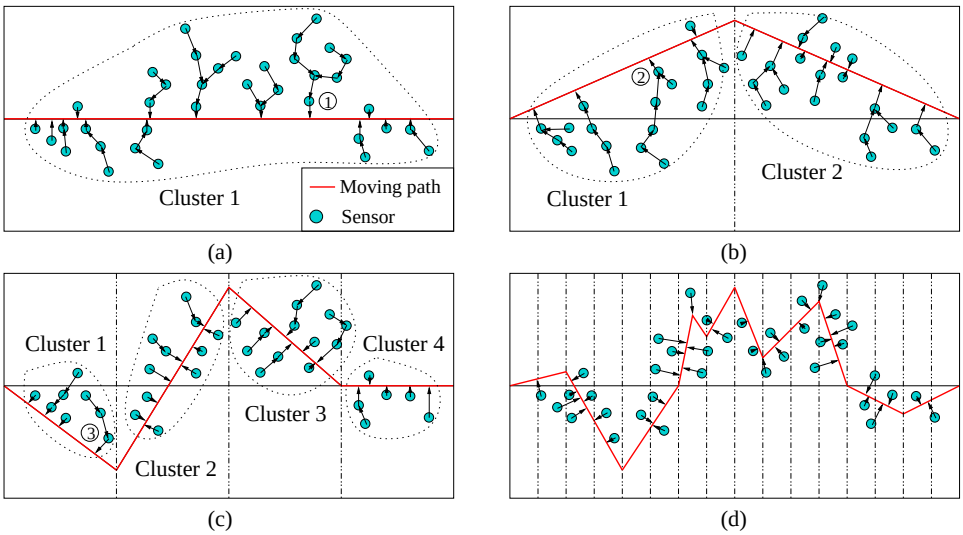


Figure 6.5: SenCar moves from A to B and collects data from nearby sensors. (a) Initial moving path is a straight line. (b) Moving path contains 1 turning point after iteration 1. (c) Moving path contains 3 turning points after iteration 2. (d) Moving path contains 15 turning points after iteration 4.

Figure 6.5 gives an example of the moving path planning algorithm. Figure 6.5(a)-(d) show the moving paths and network flows of the initial, first, second and fourth iteration, respectively. We can observe that node 1, 2 and 3 are bottleneck nodes in Figure 6.5(a), (b) and (c), which need to relay packets to SenCar from 6, 5 and 2 child nodes, respectively. These bottleneck nodes consume energy much faster than their child nodes. On the other hand, in Figure 6.2(d), SenCar traverses through the transmission range of every node. Thus, each node can send data to SenCar directly without relaying from other nodes. The moving path after iteration 4 increases the lifetime seven times compared to the initial moving path.

In the moving path planning algorithm, we can observe that adding turning points into the moving path will increase the total moving distance of SenCar, according to the triangle inequality rule. However, in practice, the total moving distance or the length of each tour may be restricted by several factors. First, the length of each tour may be determined

Table 6.1: Moving Path Planning Algorithm

Moving path planning algorithm

$i = 1$;
flag $= 1$;
while (flag $==$ 1)
  Divide the network into 2^{i-1} clusters;
  for $j = 1$ to 2^{i-1} **do**
    Find the best turning point in j_{th} cluster from all possible locations of turning points;
    Add the best turning point into the moving path;
    if the total moving distance/time cannot satisfy the constraints after the new turning point is added.
      flag $= 0$;
      Remove the new turning point from the path;
    end if
  end for
  i^{++};
end while

by the buffer size and data collecting rate of sensors as sensing data must be gathered by SenCar before the buffer overflows. If all sensors have the same memory size *mem* and data rate *rate*, the maximum length of each tour must be less than $\frac{mem}{rate}$. Second, the maximum moving distance of SenCar without recharging may be limited by its battery capacity. Third, for some delay-sensitive applications, sensing data must be uploaded to the data processing center within limited time after being collected from the environment. Thus, in many applications, the recursive moving path planning algorithm may have to terminate before the distance or time bound is reached.

By incorporating these constraints into the algorithm, we summarize the moving path planning algorithm in Table 6.1.

We now analyze the time complexity of this algorithm. Let t denote the total number of

turning points in the moving path, when the algorithm terminates. Suppose that the sensing field is divided into g grids. In order to determine a turning point, at most g possible locations of the turning point would be checked. As discussed earlier, it requires $O(Un^2)$ time to obtain the maximum network lifetime for each possible location of the turning point. Thus, the running time of moving path planning algorithm is $O(tgUn^2)$, where U and n have the same definitions as that in Section 6.2.1. Finally, we would like to point out that the moving path planning algorithm is run off-line by SenCar before the first data gathering tour. After that, only when some nodes fail or the topology of the network changes, SenCar needs to recalculate the new moving path adaptively.

6.2.5 Determining the Moving Circle of SenCar

In some applications, SenCar not only needs to traverse the sensing field, but also has to return to the starting point and upload data to the static data processing center. For such applications, moving paths become moving circles. Instead of a one-way straight line, the initial circle becomes a round-trip tour, which consists of two overlapped paths of the same shape but in opposite directions. The initial circle origins from the starting point, traverses the network, turns around and then moves back to the starting point. Both one-way paths of the initial circle pass through the network and divide the network into two parts. Sensors on each side of overlapped paths form a cluster. Each one-way path corresponds to one cluster and can be considered as the initial path of its corresponding cluster. Then, the moving path planning algorithm can be run recursively in each cluster. Finally, two separate moving paths form a moving circle.

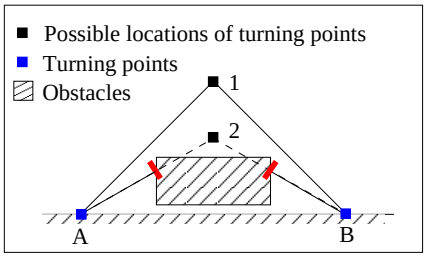


Figure 6.6: Planning the moving path in the sensing field with obstacles. The line segments from A to 2 and from 2 to B are blocked by obstacles. Thus, 2 cannot be a turning point, while 1 is eligible to be a turning point.

6.2.6 Avoiding Obstacles in the Sensing Field

We have discussed how to plan the moving path and circle of SenCar in an open sensing field. However, in most real-world applications, the working areas may be partially bounded, or have some irregular-shaped obstacles located within the sensing area. In order to make the moving path planning algorithm feasible in these situations, SenCar has to be able to avoid obstacles. Here, we assume that the complete map of the sensing field has been obtained before SenCar begins to collect data, which should include the location and shape information of obstacles in the sensing field. Then it is not difficult to adjust the basic moving path planning algorithm in Table 6.1 to avoid obstacles. For each candidate location of a turning point, SenCar will check if the line segment from the last turning point to it and the line segment from it to the next turning point are blocked by obstacles. If so, the candidate location is not eligible to be the turning point. Figure 6.6 shows an example of how to check the eligibility of each possible location of the turning point. A new path from point A to point B will be chosen from $A \rightarrow 1 \rightarrow B$ or $A \rightarrow 2 \rightarrow B$. Since the straight lines between A and 2, and between 2 and A are blocked by obstacles, 2 is not eligible to

be a tuning point. Thus, the new path from A to B can only go through point 1.

6.3 Data Gathering in a Disconnected Network

We have given the moving path planning algorithm for a connected network. However, in reality sensors are not always connected. In some applications, sensors are deployed to monitor separate areas. In each area, nodes are densely deployed and connected, while nodes that belong to different areas may be disconnected. For such applications, a mobile observer is especially suitable for data gathering. First, given the positions of sensors, a disconnected network can be divided into several connected clusters. Then, SenCar can visit connected clusters one by one, and collect data from each cluster. Thus, the entire moving circle can be divided into inter-cluster circles and inner-cluster paths. In each cluster, inner-cluster moving path can be determined by the moving path planning algorithm we gave in the previous section. Next we will describe the algorithms for clustering and inter-cluster circle planning.

6.3.1 Dividing the Network into Clusters

The objective of the clustering algorithm is to find the smallest number of connected components in the network. At the beginning, each node itself forms a connected component. If a node is connected to any node in another component, the node will be added into that component. The size of a connected component is maximized if no more nodes can be added into the component. The algorithm will not terminate until the sizes of all connected components are maximized. The clustering algorithm is shown in Table 6.2. The first part of the clustering algorithm is the neighbor discovering phase. It takes $O(n)$ time for each

node to find all its one-hop neighbors, where n is the total number of nodes in the network. Thus, the time complexity of the neighbor discovering phase is $O(n^2)$. All nodes are added into the set N at the beginning, which denotes the set of all nodes. In order to construct set C_m to represent the set of nodes in the m_{th} cluster. We can start from an empty set C_m. A node $t_j \in N$ is added into both C_m and a temporary set Tmp, and removed from N. After that all the one-hop neighbors of t_j are also added into C_m and Tmp, and removed from N. Next, remove t_j from Tmp. Thus, nodes connected to t_j can be added into C_m one by one, until Tmp is empty. When N is empty, the clustering algorithm stops. Since each element in N will be added into Tmp and removed from both N and Tmp, and finally be added into the set denoting its corresponding cluster. The complexity of this part of the algorithm is $O(n)$. Thus, the total time complexity of the clustering algorithm is $O(n^2)$.

6.3.2 Planning Inter-Cluster Circle: Touring all Clusters

In this subsection, we propose an algorithm for planning the inter-cluster moving circle. The objective of the inter-cluster circle planning is to find the shortest circle that visits all clusters and returns to the starting point of the tour. Before describing the details of the algorithm, we first introduce some terms and assumptions.

- *Left* and *Right*: We assume that all sensor nodes are in the same coordinate system. Node A is on the *left* of node B if the x-coordinate of A is less than that of node B. Similarly, we can define *Right*.

- *Most left node* and *most right node* of a cluster represent the node with the minimum and the maximum x-coordinates, respectively. If a tie exists, the most left or right node can be randomly picked. In the following, the most left and right node of cluster C_i are denoted by ln_i and rn_i respectively.

Table 6.2: Algorithm for dividing the network into clusters

Algorithm for Dividing the Network into Clusters

Add all nodes into set N;
for each element n_i in N **do**
  Find and add all one-hop neighbors of n_i into set $NB(n_i)$;
end for
$m = 0$;
while N is not empty;
  m^{++};
  Construct a new empty set C_m for the m_{th} cluster;
  Pick a node n_i from N, add it into set Tmp and C_m, and remove it from N;
  while Tmp is not empty
    for each element t_j in Tmp **do**
      for each element nb_k in $NB(t_j)$ **do**
        Remove nb_k from N, and add it into Tmp;
      end for
      Remove t_j from Tmp; Add t_j into set C_m;
    end for
  end while
end while

As shown in Figure 6.7(a), in each moving circle, SenCar traverses each cluster exactly once. Thus, the moving circle crosses the boundary of each cluster twice. In each cluster, we assume that the inner-cluster path either starts from the most left node to the most right node of the cluster or in the opposite direction. Given the most left and most right nodes of the cluster, the inner-cluster moving path can be determined separately without affecting the inter-cluster circle. For inter-cluster circle planning, we do not need to consider the path inside each cluster. We can simply find a shortest circle to connect $|C|$ pairs of most left and most right nodes, where $|C|$ denotes the number of all clusters. In order to minimize the moving distance of the inter-cluster tour, given a set of clusters, a graph $G(V,E)$ can be constructed as follows:

- For each cluster C_i in the network, add three vertices c_{i_1}, c_{i_2} and c_{i_3}, into V, where c_{i_1} and c_{i_3} denote the most left node ln_i and most right node rn_i of cluster C_i; Connect c_{i_1} and c_{i_2} by edge $e(c_{i_1}, c_{i_2})$ with distance 0; Connect c_{i_2} and c_{i_3} by edge $e(c_{i_2}, c_{i_3})$ with distance 0;

- For any two clusters C_i and C_j, add edges $e(c_{i_1}, c_{j_1})$, $e(c_{i_1}, c_{j_3})$, $e(c_{i_3}, c_{j_1})$ and $e(c_{i_3}, c_{j_3})$ into E with distance $d(ln_i, ln_j)$, $d(ln_i, rn_j)$, $d(rn_i, ln_j)$ and $d(rn_i, rn_j)$ respectively, where $d(a,b)$ denotes the distance between node a and node b, $a \in \{ln_i, rn_i\}$ and $b \in \{ln_j, rn_j\}$.

- Add a starting point s in V, for each cluster C_i in the network, connect s and c_{i_1} by edge $e(s, c_{i_1})$ with distance $d(s, ln_i)$, connect s and c_{i_3} by edge $e(s, c_{i_3})$ with distance $d(s, rn_i)$, where $d(s, ln_i)$ and $d(s, rn_i)$ denote the distance from the starting point to ln_i and rn_i.

In the above construction, in order to visit vertex c_{i_2} in G, the circle must include either segment $c_{i_1} \rightarrow c_{i_2} \rightarrow c_{i_3}$ or $c_{i_3} \rightarrow c_{i_2} \rightarrow c_{i_1}$, because c_{i_2} has only two neighbors c_{i_1} and c_{i_3}. Thus, if the shortest circle that visits all vertices and returns to the starting point can be found, the shortest inter-cluster circle visiting all clusters can be obtained. We next show that the problem of finding the shortest inter-cluster circle is NP-hard, and refers to the problem as *SICC* problem.

Lemma 2 *The SICC problem is NP-complete.*

Proof. First, it is easy to see that SICC is in NP. The shortest inter-cluster circle can be proved to achieve the shortest distance by adding up the distances of all segments in the circle. In order to show that SICC is NP-hard, we give a reduction from the well-known Traveling Salesman Problem (TSP) [92]. Given any instance of the TSP, an instance of SICC problem can be constructed as follows.

Let $G'(V',E')$ denote the graph of the TSP problem, where $V' = \{s, v'_1, v'_2, \ldots, v'_n\}$, A graph of SICC $G(V,E)$ can be constructed as follows.

- For each v'_i in G', add three vertices c_{i_1}, c_{i_2} and c_{i_3}, into V, connect c_{i_1} and c_{i_2} by edge $e(c_{i_1}, c_{i_2})$ with a distance 0, and connect c_{i_2} and c_{i_3} by edge $e(c_{i_2}, c_{i_3})$ with distance 0.

- For any edge $e(v'_i, v'_j) \in E'$, add edges $e(c_{i_1}, c_{j_1})$, $e(c_{i_1}, c_{j_3})$, $e(c_{i_3}, c_{j_1})$ and $e(c_{i_3}, c_{j_3})$ into E with the same distance $d(v'_i, v'_j)$, where $d(v'_i, v'_j)$ denotes the distance between node v'_i and node v'_j.

- Add a starting point s to V, for each edge $e(s, v'_i) \in E'$, connect s and c_{i_1} with edge $e(s, c_{i_1})$ with distance $d(s, v'_i)$, and connect s and c_{i_3} with edge $e(s, c_{i_3})$ with distance $d(s, v'_i)$.

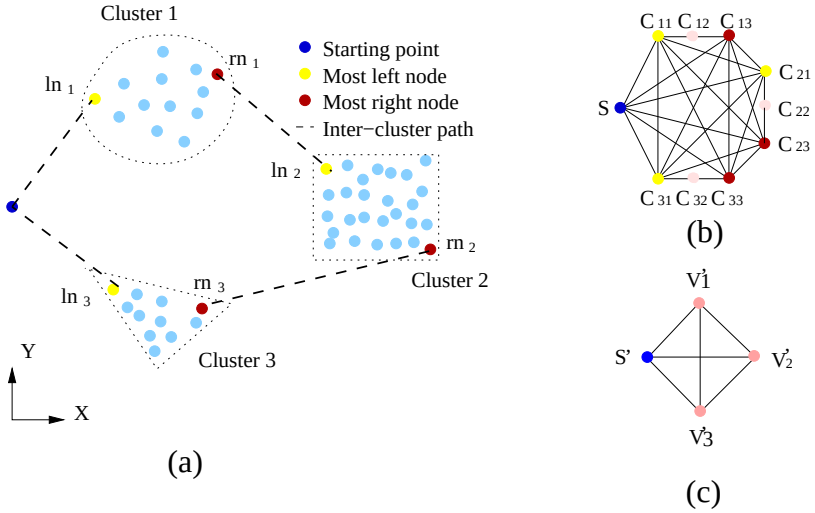


Figure 6.7: SenCar gathers data from a disconnected network. (a) The sensor network consists of three connected clusters. (b) Graph of SICC. (c) Graph of TSP.

An example of constructing G from G' is shown in Figure 6.7(b) and (c), where four vertices in Figure 6.7(c) are extended to ten vertices in 6.7(b). In the optimal solution of SICC, if the shortest circle in G contains segment $c_{i_1} \rightarrow c_{i_2} \rightarrow c_{i_3} \rightarrow c_{j_1} \rightarrow c_{j_2} \rightarrow c_{j_3}$, the shortest TSP circle in G' should include $v'_i \rightarrow v'_j$. The shortest circle that visits all vertices in SICC can achieve the shortest total distance, if and only if the optimal solution of the TSP can be found. In addition, the shortest distance of the SICC problem is equal to that of the TSP.

Though the SICC problem is NP-hard, approximate algorithms for the TSP can be adopted for the SICC. For example, a well-known 2-approximate algorithm [92] can be implemented in $O(|C|^2 \log |C|)$ time, where $|C|$ is the number of clusters. Note that in practice, $|C|$ is usually a small constant (say, less than 10). In this case, an exhaustive search may be used to find the shortest inter-cluster circle.

By combining the inner-cluster path planning algorithm, clustering algorithm and inter-cluster circle planning algorithm, SenCar can find a circle in a disconnected network.

6.4 Performance Evaluations

We have conducted extensive simulations to validate the algorithms we propose. In the simulation, we assume that a bunch of sensor nodes are densely deployed in the sensing field. Two-ray propagation model is used to describe the feature of the physical layer. With the maximum transmission power $0.858mw$, each node can communicate with other nodes as far as $40m$ away. The radio bandwidth is 250kbps. CBR traffic on the top of UDP is generated to measure the throughput. We assume each packet has a fixed size of 80 bytes, including header and payload. Each sensor has 10kBytes flash memory for storing its sensing data. Sensors collect data at a fixed rate 100 Byte/minute. Thus, in order to avoid the data overflow, sensing data has to be uploaded to SenCar every 100 minutes. After SenCar moves into the transmission range of a sensor, it stops, wakes up the sensor, collects data, and moves again after the data transmission is finished. Let the grid length in the moving path planning algorithm be $10m$. Within each cluster, multi-hop polling protocol [99] is used as the inner-cluster protocol to avoid packets collision at the MAC layer. We evaluated the moving planning algorithm for both connected networks and disconnected networks.

6.4.1 Finding the Moving Circle in an Area with Obstacles

In this scenario, suppose that 800 sensors are densely deployed into a contaminated chemistry factory building to monitor the density of leaked chemicals. The map of the

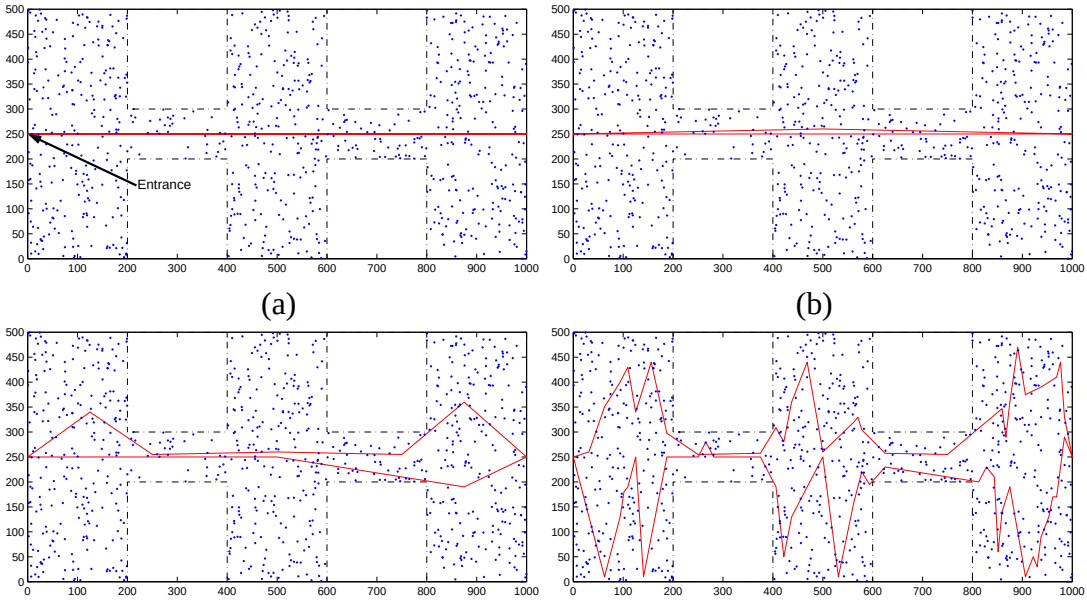


Figure 6.8: SenCar starts the data gathering circle from the entrance of the building, collects data from sensors and returns to the entrance. (a) Initial layout of the network. (b) Layout and moving circle after iteration 2. (c) Layout and moving circle after iteration 4. (d) Layout and moving circle after iteration 8.

building and the initial layout of a bunch of sensors are shown in Figure 6.8(a). The building consists of six $200m \times 200m$ large rooms, which are on both sides of a $1000m \times 100m$ aisle. The entire building is bounded by brick walls. SenCar has to move within the building. SenCar enters the building from the entrance, which is at the coordinates $(0m, 250m)$, collects data from all sensors, and returns to the entrance after a tour. We assume that the location information, connection patterns of sensors and the map of the building have been obtained during the network deployment phase. Based on this information, SenCar calculates the routes iteration by iteration by using the moving circle planning algorithm. As shown in Figure 6.8, the initial moving circle consists two overlapped, straight-line moving paths, $(0m, 250m) \rightarrow (1000m, 250m)$ and $(1000m, 250m) \rightarrow (0m, 250m)$. Figure 6.8(b), (c) and (d) show the moving circle after iterations 2, 4 and 8, respectively. From the figures, we can observe that, first, SenCar enters every room without hitting the walls of the building; second, as the number of iterations increases, SenCar moves zigzag around the building to get closer to the nodes. We next show that the movement of SenCar can balance the traffic load and prolong the network lifetime.

6.4.2 Network Lifetime

We now compare the network lifetime of the following three data gathering schemes: *Scheme 1*: A static observer placed in the center of the network (at point $(500m, 250m)$); *Scheme 2*: A mobile observer which can only move back and forth through the straight line between $(0m, 250m)$ and $(1000m, 250m)$; *Scheme 3*: SenCar which can move through a well-planned circle that starts and ends at point $(0m, 250m)$. We introduce a new metric, called *x% network lifetime*, which is defined as the network lifetime when $(100 - x)\%$ sensors either run of battery or cannot send data to the data sink due to the failure of relaying

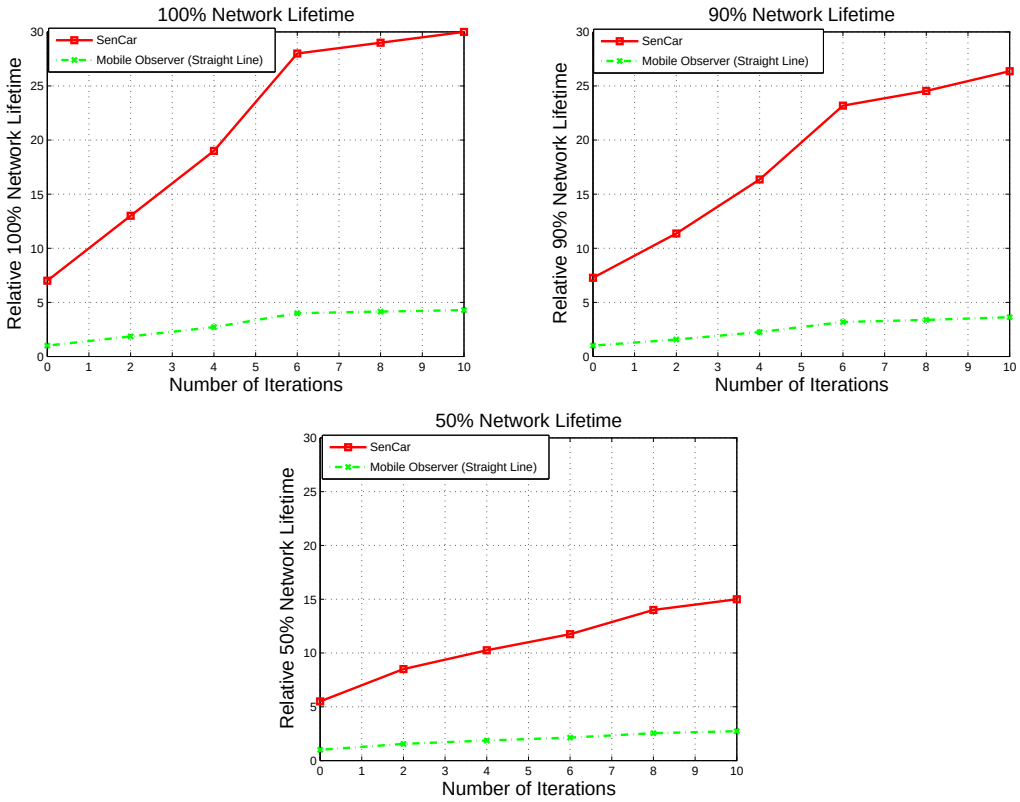


Figure 6.9: The relative network lifetime of Scheme 3 compared to Scheme 1 and Scheme 2. (a) 100% network lifetime, (b) 90% network lifetime, (c) 50% network lifetime.

nodes. In this scenario, we measure $\%x = 50\%, 90\%$ and 100% network lifetime. For the network only containing static observer, we measured the optimal x% network lifetime by using the load balancing algorithm in [99]. We also evaluated the lifetime of the network, in which a mobile observer moves through straight lines. The optimal lifetime of the first two schemes is used as the performance reference for comparison purpose. The relative x% network lifetime of *Scheme 3* compared to *Scheme 1* and *Scheme 2* are plotted in Figure 6.9. From the figure we can observe that all the relative x% network lifetime ratios of SenCar compared to Scheme 1 and Scheme 2 keep increasing from iterations 1 to 10, and reach 29.8 and 4.5 at iteration 10 when $x\% = 100\%$, 26 and 4.2 at iteration 10 when $x\% = 90\%$, 14.9 and 2.8 at iteration 10 when $x\% = 50\%$, respectively. From this experiment, we can see that a mobile observer can prolong the network lifetime significantly compared to a static observer. Moreover, a well-planned moving path performs much better than a fixed straight line path for a mobile observer.

6.4.3 Comparison with Traveling Salesman Problem (TSP) Approach

The movement planning problem can also be modeled as the well-known *Traveling Salesman Problem (TSP)*, if we force SenCar to visit the location of every sensor one by one rather than gather data remotely. The goal of TSP is to find a minimum length (cost) tour that visits every sensor exactly once, which is known to be NP-hard. Intuitively, TSP can achieve longer lifetime than our approach, since each sensor can upload data directly to SenCar without relays. On the other hand, TSP may also yield longer tour length than our approach, since SenCar needs to visit the location every single sensor. Because of the NP-hardness of *TSP*, the brutal force search method of the optimal solution in a large network becomes impossible. However, we have managed to run the optimal algorithm for

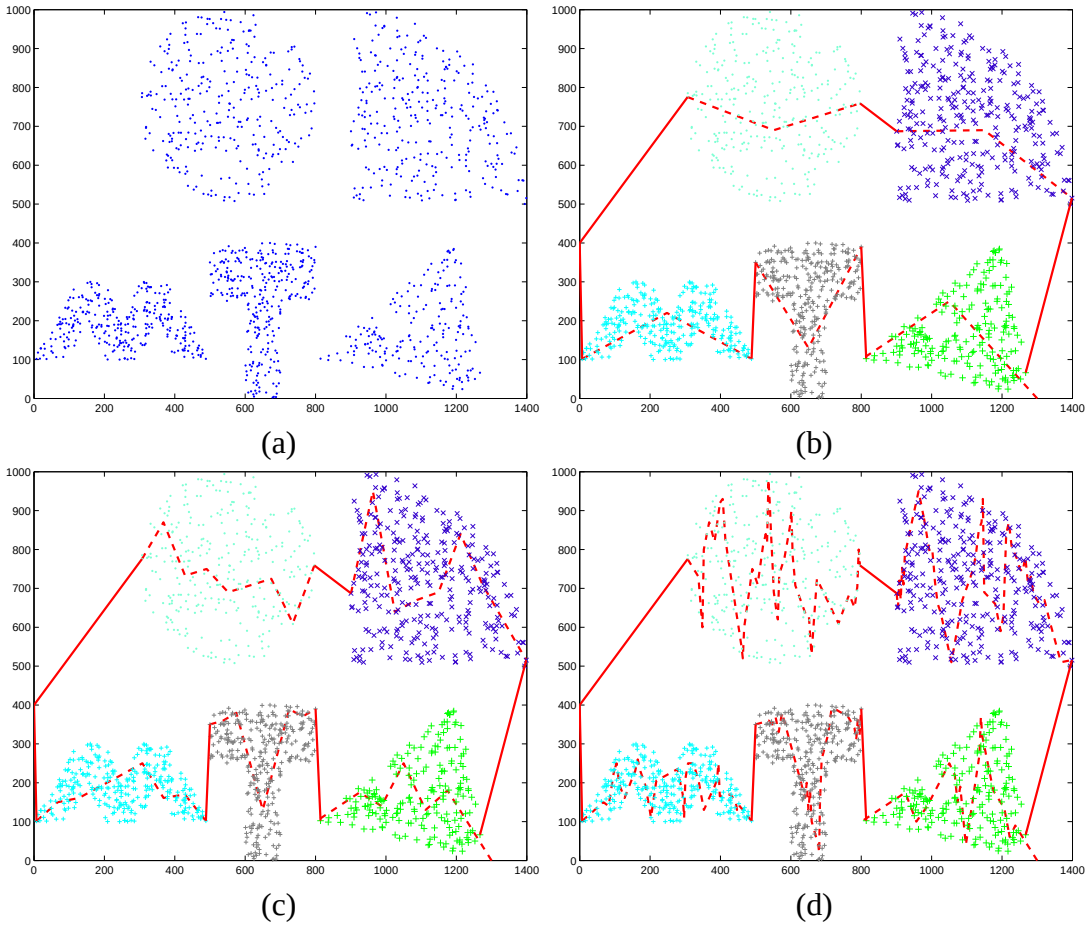


Figure 6.10: SenCar gathers sensing data from disconnected clusters. Continuous and dotted lines denote inter-cluster circles and inner-cluster paths, respectively. (a) Initial layout of the network. (b) Layout and moving circle after iteration 2. (c) Layout and moving circle after iteration 4. (d) Layout and moving circle after iteration 8.

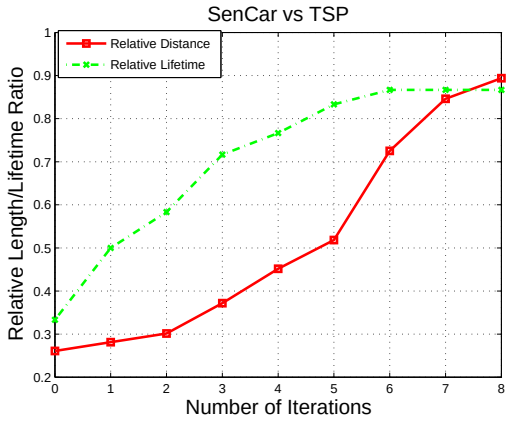


Figure 6.11: Comparison with Traveling Salesman Problem(TSP) approach

a small network for comparing with our heuristic algorithm. We use Concorde TSP solver [93] to obtain the optimal solution of TSP in a relatively small network, where 200 nodes are uniformly deployed in a $150m \times 300m$ area. We compare our approach to TSP in terms of tour length and the 100% network lifetime, and plot relative length/lifetime ratio of our approach compared to the TSP approach in Figure 6.11. We can observe in Figure 6.11, both relative length and lifetime keep increasing as the number of iterations increases. At iteration 4, our approach achieves 76% of the 100% network lifetime of TSP, with only 46% of the tour distance of TSP. After iteration 6, the relative lifetime keeps around 87%, where relative distance increases rapidly and reaches as high as 90% of the optimal solution. After iteration 8, our algorithm makes little improvement on the network lifetime, because it is hard for any polynomial algorithm to achieve the performance close to the optimal solution of a NP-hard problem. Therefore, if the tour length/delay is totally not a design concern or packet relays are not allowed, the optimal solution of TSP can achieve the longest network lifetime, tough it is NP-hard. Otherwise, our approach can provide quite good performance with relatively low cost compared to TSP.

6.4.4 Finding the Moving Circle in a Disconnected Network

In this scenario, sensor nodes are organized into five connected clusters. The moving circle of SenCar starts and ends at the same point $(0m, 400m)$. Since each cluster is disconnected from others, SenCar has to visit every cluster one by one to gather sensing data. Figure 6.10(a) depicts the initial layout of the network. We can see that sensors are deployed onto five separate areas in various shapes, including a circle, a sector, an "M"-shaped area, a "T"-shaped area, and a triangle. 300 nodes are densely deployed in each cluster. The continuous lines between clusters denote the inter-cluster circle, while the dotted lines represent inner-cluster paths. Figure 6.10(b), (c) and (d) give the moving circle after iterations 2, 4 and 8. We can observe that clusters are connected by the inter-cluster circle, and SenCar moves zigzag in each cluster to collect data.

6.5 Conclusions

In this paper, we have proposed a new data collecting mechanism by introducing a mobile data observer, SenCar, in sensor networks. SenCar works like a mobile base station, starts the data gathering tour from the outside observer, traverses the entire sensor network, collects the data from nearby sensors, and then returns to the outside observer. We have showed that the moving path of SenCar can affect the network lifetime significantly. We presented a heuristic algorithm for planning the moving path/circle of SenCar and balancing traffic load in the network. By adopting a load balancing algorithm which finds the turning points and clusters the network recursively, network lifetime can be prolonged significantly. The moving planning algorithm can be used in both connected networks and

disconnected networks. In addition, SenCar can avoid obstacles while moving. Our simulation results show that the proposed data gathering mechanism can prolong the network lifetime about 30 times compared to a network which has only a static observer, and about 4 times compared to a network whose mobile observer can only move along straight lines.

Chapter 7

Conclusions

This thesis presents a self-adaptive, scalable and energy-efficient integrated framework for large scale, unattended sensor networks, which includes: The adaptive Triangular deployment algorithm [94] can increase the non-gap coverage of mobile sensors. It also supports adaptive deployment. Without the map and information of the environment, nodes can avoid obstacles and adjust the density dynamically based upon different requirements. The CR-MAC [96] protocol can achieve much better throughput, fairness, packet delay than IEEE 802.11 RTS/CTS protocol. In particular, under saturated traffic, both the throughput and the fairness index of the CR-MAC protocol are very close to the theoretical bound. The single path flooding chain routing algorithm [95] can provide reliable end-to-end routing, and significantly save the bandwidth and power for resource limited mobile nodes, especially in large networks. The clustering and Load Balancing mechanism [99, 98, 97, 100] can increase the network lifetime by a significant amount. In addition, the algorithm can recover the network from unexpected failure of sensors and cluster heads. The data gathering mechanism [101] by using SenCar can prolong the network lifetime greatly by introducing a mobile data collector.

The proposed research combines protocol design, algorithm design, analytical, probabilistic and simulation techniques to conduct comprehensive studies on the above issues. The proposed research will have a significant impact on fundamental design principles and infrastructures for the development of future sensor networks. The outcome of this project will be applicable to a wide spectrum of applications, including space, military, environmental, health care, home and other commercial areas.

Bibliography

[1] E. Biagioni and K. Bridges, "The application of remote sensor technology to assist the recovery of rare and endangered species," Special Issue on Distributed Sensor Networks, *International Journal of High Performance Computing Applications*, Vol. 16, No. 3, Aug. 2002.

[2] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton and J. Zhao. "Habitat monitoring: Application driver for wireless communications technology," *Proc. of 2001 ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, April 2001.

[3] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler and J. Anderson, "Wireless sensor networks for habitat monitoring," *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, Atlanta, GA, Sept. 2002.

[4] S. Chessa and P. Santi, "Crash faults identification in wireless sensor networks," *Computer Communications*, Vol. 25, No. 14, pp. 1273-1282, Sept. 2002

[5] L. Schwiebert, S. K. S. Gupta and J. Weinmann, "Research challenges in wireless networks of biomedical sensors," *Proc. ACM/IEEE Conf. on Mobile Computing and Networking (MobiCom)*, pp. 151-165, 2001.

[6] http://robotics.eecs.berkeley.edu/ pister/SmartDust/, 2004.

[7] http://www-mtl.mit.edu/research/icsystems/uamps/, 2004.

[8] G. Asada, T. Dong, F. Lin, G. Pottie, W. Kaiser and H. Marcy, "Wireless integrated network sensors: low power systems on a chip," *European Solid State Circuits Conference*, The Hague, Netherlands, Oct. 1998.

[9] http://www.isi.edu/scadds/pc104testbed/guideline.html, 2004.

[10] "Temote Sky sensor nodes, http://www.moteiv.com/"

[11] "Crossbow wireless sensor nodes, http://www.xbow.com/"

[12] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, Aug. 2002, pp. 102-114.

[13] The Ultra Low Power Wireless Sensor Project, http://www-mtl.mit.edu/ jimg/project_top.html, 2004.

[14] R. Min, M. Bhardwaj, N. Ickes, A. Wang and A. Chandrakasan, "The hardware and the network: total-system strategies for power aware wireless microsensors," *IEEE CAS Workshop on Wireless Communications and Networking*, Pasadena, CA, USA, Sept. 2002.

[15] Mars Exploration Rover Project, http://mars.jpl.nasa.gov, 2004.

[16] A. Cerpa and D. Estrin, "ASCENT: Adaptive self-configuring sensor network topologies," *Proc. of the IEEE Infocom*, June 2002.

[17] C. Perkins, and P. Bhagwat, “Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computer”, *Proc. of the ACM SIGCOMM’94 Conference on Communications Architectures, Protocols and Applications*, London, UK, August 1994, pp. 234-244.

[18] P. Jacquet, P. Muhlethaler, and A. Qayyum, “Optimized link state routing (OLSR) protocol”. *Internet Draft. draft-ietf-manet-olsr-06.txt*, July 2002.

[19] R. G. Ogier, F. L. Templin, B. Bellur, and M. G. Lewis, “Topology broadcast based on reverse-path forwarding (tbrpf)”. *Internet Draft, draft-ietf-manet-tbrpf-05.txt*, March 2002

[20] D. B. Johnson, D. A. Maltz, and J. Broch, “Dynamic source routing in ad hoc wireless networks,” *Mobile Computing*, edited by Tomas Imielinski and Hank Korth, Kluwer Academic Publishers, 1996, Chapter 5, pp. 153-181.

[21] D. B. Johnson, D. A. Maltz, and J. Broch, “DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks,” *Ad Hoc Networking*, C. Perkins, Ed. Addison-Wesley, 2001, pp. 139-172.

[22] V. Park and S. Corson. “Temporally-ordered routing algorithm (TORA) version 1 functional specification,” *Internet Draft: draft-ietf-manet-tora-spec-04.txt*, July 2001.

[23] C.Perkins, E. Royer, “Ad-hoc on-demand distance vector routing”, *Proc. of WMCSA’99*, New Orleans, LA, USA, Feb. 1999, pp. 90-100.

[24] C. Perkins, E. Royer, and S. Das, “Ad hoc on demand distance vector (AODV) routing,”. *IETF Draft*, http://www.ietf.org/internet-drafts/draft-ietf-manetaodv -10.txt, Jan. 2002.

[25] M. Mauve, J. Widmer, and H. Hartenstein, “A survey on position-based routing in mobile ad-hoc networks”. *IEEE Network*, November 2001.

[26] I. Stojmenovic, “Position based routing in ad hoc networks,” *IEEE Commmunications Magazine*, Vol. 40, No. 7, July 2002, 128-134.

[27] S. Basagni, I. Chlamtac, V.R. Syrotiuk, and B.A. Woodward. “A distance routing effect algorithm for mobility (DREAM),” *Proc. Fourth Annual ACM/IEEE International Conference in Mobile Computing and Networking (MobiCom)*, pp. 76-84, 1998.

[28] Y. Ko and N. H. Vaidya, “Location-aided routing (LAR) mobile ad hoc networks,” *ACM/IEEE MobiCom’98*, Dallas, TX, 1998.

[29] I. Stojmenovic and X. Lin, “Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks,” *IEEE Trans. Parallel and Distributed Systems*, vol. 12, No. 10, pp. 1023-1032, 2001.

[30] P. Bose, P. Morin, I. Stojmenovic and J. Urrutia, “Routing with guaranteed delivery in ad hoc wireless networks,” *Wireless Networks*, Vol. 7, No. 6, pp. 609-616, 2001.

[31] B. Karp and and H.T. Kung. “Greedy perimeter stateless routing for wireless networks,” *Proc. ACM/IEEE MobiCom*, Boston, MA, Aug. 2000.

[32] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, “Highly-resilient, energy-efficient multipath routing in wireless sensor networks,” *Mobile Computing and Communications Review*, Vol. 4, No. 5, October 2001.

[33] E.D. Kaplan, ed., *Understanding GPS – Principles and Applications*, Norwood MA: Artech House, 1996.

[34] S. Capkun, M. Hamdi, J.P. Hubaux, “GPS-free positioning in mobile ad-hoc networks,” *Proc. Hawaii Int. Conf. on System Sciences*, Jan. 2001.

[35] Z.J. Haas and B. Liang, “Ad-hoc mobility management with uniform quorum systems,” *IEEE/ACM Trans. Networking*, vol. 7, no. 2, pp. 228-240, Apr. 1999.

[36] J. Li, J. Jannotti, D.S.J. De Couto, D.R. Karger and R. Morris, “A scalable location service for geographic ad hoc routing,” *Proc. ACM/IEEE MobiCom*, pp. 120-130, Boston, Aug. 2000.

[37] R. Morris, J. Jannotti, F. Kaashoek, J. Li and D. Decouto, “CarNet: A scalable ad hoc wireless network system,” *ACM SIGOPS European Workshop*, Kolding, Denmark, Sept. 2000.

[38] S. Giordano and M. Hamidi. “Mobility management: The virtual home region,” Technical Report, Oct. 1999.

[39] H. Takagi and L. Kleinrock, “Optimal transmission ranges for randomly distributed packet radio terminals,” *IEEE Trans. Communications*, vol. 32, no. 3, pp. 246-257, Mar. 1984.

[40] T.-C. Hou and V.O.K. Li, “Transmission range control in multihop packet radio networks,” *IEEE Trans. Communications*, vol. 34, no. 1, pp. 38-44, Jan. 1986.

[41] E. Kranakis, H. Singh, and J. Urrutia, “Compass routing on geometric networks,” *Proc. 11th Canadian Conference on Computational Geometry*, Vancouver, Aug. 1999.

[42] J. Ni and S. Chandler, ”Connectivity properties of a random radio network,” *Proc. IEE Communications*, vol. 141, pp 289-296, Aug. 1994.

[43] "IEEE standard for wireless LAN medium access control (MAC) and physical layer (PHY) specifications," *IEEE Standard 802.11*, June 1999.

[44] http://www.ieee802.org/11/, 2004.

[45] B.P. Crow, I. Widjaja, J. G. Kim and P. T. Sakai, "IEEE 802.11 wireless local area network," *IEEE Communications Magazine*, vol 35, Sep.pp. 116-126, 1997.

[46] A. Chandra, V. Gummalla and J.O. Limb, "Wireless medium access control protocols," *IEEE Communications Surveys and Tutorials*, pp. 2-15, Second Quarter 2000.

[47] N. Prasad and A. Prasad, *WLAN Systems and Wireless IP for Next Generation Communications*, Artech House Publishers, 2001.

[48] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE Journal on Selected Areas in Communications*, vol.18, no.3, pp.535-547, Mar. 2000.

[49] G. Bianchi and I. Tinnirello, "Kalman filter estimation of the number of competing terminals in an IEEE 802.11 network," *IEEE INFOCOM*, pp. 844-852, April 2003.

[50] F. Cali, M. Conti and E. Gregori, "Dynamic tuning of the IEEE 802.11 protocol to achieve a theoretical throughput limit," *IEEE/ACM Trans. Networking*, vol.8, no.6, pp.785-799, Dec. 2000.

[51] Y. Kwon, Y. Fang and H. Latchman, "A novel MAC protocol with fast collision resolution for wireless LANs," *IEEE INFOCOM*, pp. 853-862, April 2003.

[52] C. Wang and B. Li, "A gentle collision resolution mechanism for IEEE 802.11 DCF," *IEEE Trans. on Vehicular Technology*, 53(4): 1235-1246, July 2004.

[53] H. Wu, Y. Peng, K. Long, S. Cheng and J. Ma, "Performance of reliable transport protocol over IEEE 802.11 wireless LAN: analysis and enhancement," *IEEE INFOCOM*, pp.599- 607, June 2002.

[54] V. Bharghavan, A. Demers, S. Shenkar and L. Zhang, "MACAW: A media access protocol for wireless LAN's," *ACM SIGCOMM*, pp. 212-225, Aug. 1994.

[55] S. Xu and T. Safadawi, "Does the IEEE 802.11 MAC protocol work well in multihop wireless Ad Hoc networks?" *IEEE Communications Magazine*, vol. 39, no. 6, pp. 130-137, 2001.

[56] R. Jain, A. Durresi, G. Babic, "Throughput fairness index: an explanation," *ATM Forum Document Number: ATMForum/99-0045*.

[57] G. D. Coughlan, J. E. Dodd and B. M. Gripaios, *The ideas of particle physics: an introduction for scientists*, Cambridge University Press, 1991.

[58] M. de Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf, *Computational Geometry Algorithms and Applications*, Springer-Verlag, 1997.

[59] A. Howard, Maja J. Mataric and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed scalable solution to the area coverage problem," *Proc. of the 6th International Conference on Distributed Autonomous Robotic Systems (DARS02)*, Fukuoka, Japan, 2002, pp. 299-308.

[60] S Poduri and G. S. Sukhatme, "Constrained coverage for mobile sensor networks," *IEEE International Conference on Robotics and Automation*, New Orleans, LA, May 2004, pp. 165-172.

[61] G. Wang, G. Cao and T. La Porta, “Movement-assisted sensor deployment,” *Proc. of IEEE Infocom*, March 2004.

[62] Y. Zou and K. Chakrabarty, “Sensor deployment and target localizations based on virtual forces,” *Proc. of IEEE Infocom*, 2003.

[63] N. Bulusu, J. Heidemann and D. Estrin, “Adaptive beacon placement,” *Proc. of the 21st International Conference on Distributed Computing Systems (ICDCS-21)*, Phoenix, Arizona, April 2001.

[64] J. Borenstein, and Y. Koren, “Obstacle avoidance with ultrasonic sensors,” *IEEE Journal of Robotics and Automation*, Vol. RA-4, No. 2, 1988, pp. 213-218.

[65] Q. Li and D. Rus, “Global clock synchronization in sensor networks,” in *Proc of IEEE Infocom*, March 2004.

[66] F. A. Tobagi and L. Kleinrock, “Packet switching in radio channels: Part II-The hidden terminal problem in carrier sense multiple-access and the busy-tone solution,” *IEEE Trans. Commun.*, Vol. COM-23, 1975, pp. 1417-1433.

[67] Z. J. Haas and J. Deng, “Dual busy tone multiple access (DBTMA) - a multiple access control scheme for ad hoc networks,” *IEEE Trans. Commun.*, Vol. 50(6), 2002, pp. 975-984.

[68] Chipcon CC2420 RF transceiver, http://www.chipcon.com/

[69] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-efficient communication protocols for wireless microsensor networks,” *Proc. of HICSS*, Maui, Hawai, Jan. 2000.

[70] O. Younis and S. Fahmy, “Distributed clustering in ad-hoc sensor networks: a hybrid, energy-efficient approach,” *IEEE INFOCOM 2004*, Hong Kong, 2004.

[71] S. Bandyopadhyay and E. Coyle, “An energy efficient hierarchical clustering algorithm for wireless sensor networks,” *IEEE INFOCOM 2003*, San Francisco, CA, 2003.

[72] Alan Amis, et. al, “Max-min D-cluster formation in wireless ad hoc networks,” *IEEE INFOCOM 2000*, Tel-Aviv, Israel, 2000.

[73] J.H. Chang and L. Tassiulas, “Energy conserving routing in wireless ad-hoc networks,” *IEEE INFOCOM 2000*. Tel-Aviv, Israel, 2000.

[74] A. Bogdanov, E. Maneva and S. Riesenfeld, “Power-aware base station positioning for sensor networks,” *IEEE INFOCOM 2004*, Hong Kong, March 2004.

[75] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, 1993.

[76] R.L. Francis, L. F. McGinnis Jr., J. A. White, *Facility layout and location : an analytical approach*, Prentice-Hall, 1998.

[77] N. Bulusu, J. Heidemann and D. Estrin, “Adaptive beacon placement,” *21st International Conference on Distributed Computing Systems (ICDCS-21)*, Phoenix, Arizona, April 2001.

[78] T.S. Rappaport, *Wireless Communications: Principles and Practice*, Prentice-Hall, 1996.

[79] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet," in *Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2002.

[80] T. Small and Z. Haas, "The shared wireless infostation model - a new ad hoc networking paradigm (or where there is a whale, there is a way)," *ACM MobiHoc 2003*.

[81] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin and P. Corke, "Data collection, storage, and retrieval with an underwater sensor network," *Proc. of ACM Sensys*, 2005.

[82] A Chakrabarty, A Sabharwal and B Aazhang, "Using predictable observer mobility for power efficient design of a sensor network," *Second International Workshop on Information Processing in Sensor Networks (IPSN)*, April 2003.

[83] A. Pentland, R. Fletcher and A. Hasson, "Daknet: rethinking connectivity in developing nations," *IEEE Computer*, vol. 37, no. 1, pp. 78-83, January 2004.

[84] A.A. Somasundara, A. Ramamoorthy, M.B. Srivastava, "Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines," *IEEE Real Time Systems Symposium (RTSS)*, December 2004.

[85] D. Jea, A.A. Somasundara and M.B. Srivastava, "Multiple controlled mobile elements (data mules) for data collection in sensor networks," *2005 IEEE/ACM International Conference on Distributed Computing in Sensor Systems (DCOSS '05)*, June 2005.

[86] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," *ACM MobiHoc'04*, 2004.

[87] J. Luo and J.-P. Hubaux, “Joint Mobility and Routing for Lifetime Elongation in Wireless Sensor Networks,” *IEEE INFOCOM 2005*, 2005

[88] R. C. Shah, S. Roy, S. Jain, and W. Brunette, “Data MULEs: modeling a three-tier architecture for sparse sensor networks,” *IEEE Workshop on Sensor Network Protocols and Applications (SNPA)*, 2003.

[89] S. Jain, R. C. Shah, W. Brunette, G. Borriello and S. Roy, “Exploiting mobility for energy efficient data collection in wireless sensor networks,” in ACM/Kluwer MONET 2005.

[90] A. Kansal, A. Somasundara, D. Jea, M. Srivastava, and D. Estrin, “Intelligent fluid infrastructure for embedded networks,” *The Second International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2004.

[91] C. Guo, L.C. Zhong and J.M. Rabaey, “Low power distributed MAC for ad hoc sensor radio networks,” *IEEE GLOBECOM 2001*

[92] S.S. Skiena, *Algorithm Design Manual*, Springer-Verlag, pp. 319-322, 1997.

[93] “Concorde TSP solver, http://www.tsp.gatech.edu/concorde.html”

[94] M. Ma and Y. Yang, “Adaptive triangular deployment algorithm for unattended mobile sensor networks,” accepted for publication in *IEEE Transactions on Computers*.

[95] M. Ma, Y. Yang and C. Ma, “Single path flooding chain routing in ad hoc networks,” *International Journal of Sensor Networks*, vol. 1, no.1/2 pp. 11 - 19, 2007

[96] M. Ma, and Y. Yang, “A novel contention-based MAC protocol with channel reservation for wireless LANs,” in Proceeding of *International Conference on Broadband Networks (BROADNETS’05)*, Oct, 2005.

[97] M. Ma and Y. Yang, “Clustering and load balancing in hybrid sensor networks with mobile cluster heads,” in proceeding of *International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QSHINE’06)*, Aug, 2006.

[98] M. Ma, Z. Zhang and Y. Yang, “Multi-channel polling in multi-hop clusters of hybrid sensor networks,” in proceeding of *IEEE Global Telecommunications Conference (GLOBECOM’05)*, Nov, 2005.

[99] Z. Zhang, M. Ma, and Y. Yang, “Energy efficient multi-hop polling in clusters of two-layered heterogeneous sensor networks,” accepted for publication in *IEEE Transactions on Computers*.

[100] M. Ma, C. Ma and Y. Yang, “A cross-layer data gathering scheme for heterogeneous sensor networks based on polling and load-balancing,” in Proceeding of *IEEE Wireless Communications and Networking Conference 2007 (WCNC’07)*, Mar, 2007.

[101] M. Ma and Y. Yang, “SenCar: an energy efficient data gathering mechanism for large scale multihop sensor networks,” accepted for publication in *IEEE Transactions on Parallel and Distributed System.*