Delay-Oriented Reliable Communication and Coordination in Wireless Sensor-Actuator Networks

NGAI Cheuk Han

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Computer Science and Engineering

Supervised by

Prof. LYU Rung-Tsong Michael

Abstract of thesis entitled:

Delay-Oriented Reliable Communication and Coordination in Wireless Sensor-Actuator Networks

Submitted by NGAI Cheuk Han

for the degree of Doctor of Philosophy

at The Chinese University of Hong Kong in May 2007

Wireless sensor-actuator networks, or WSANs, greatly enhance the existing wireless sensor network architecture by introducing powerful and mobile actuators. These actuators are expected to work with the sensor nodes and perform much richer application-specific actions. For the applications which request for fast and accurate report of the environmental events, an efficient and reliable communication/coordination scheme is urged. Unfortunately, multi-hop communication in a WSAN is inherently unreliable due to frequent sensor failures and network partitions. Excessive delays, introduced by congestion or in-network data aggregation, further aggravate the problem.

In this thesis, we propose a general reliability-centric framework for event reporting in WSANs. We point out that the reliability in such a real-time system depends not only on the accuracy, but also the importance and freshness of the reported data. Our proposed design thus integrates three key modules, 1) an efficient and fault-tolerant event data aggregation algorithm, 2) a delay-aware data transmission protocol, and 3) an adaptive actuator allocation algorithm for unevenly distributed events. We further propose a latency-oriented fault tolerant data transport protocol (LOFT) and a power-controlled real-time data transport protocol (POWER-SPEED) for WSANs. LOFT balances the workload of sensors by checking their queue utilization and handles node/link failures by an adaptive replication algorithm. POWER-SPEED transmits packets in an energy-efficient manner while maintaining soft real-time packet transport. We evaluate our framework and the two proposed protocols through extensive simulations, and the results demonstrate that they achieve the desirable reliability for WSANs.

To minimize the data collection time, we propose a new routing design. We present the mathematical formulation of the route design problem, and show that it is computationally intractable. We then propose two practical algorithms to reduce the delay of the sensors. Our algorithms adaptively adjust the actuator visiting frequencies to the sensors according to their relative weights and data generation patterns. We further propose a probabilistic route design (PROUD) algorithm which adapts to network dynamics. We present the distributed implementation for PROUD and an extension which accommodates actuators with variable speeds. We also propose algorithms for load balancing among the actuators. Simulation results show that our algorithms can effectively reduce the overall data collection time. They adapt to the network dynamics and balances the energy consumption of the actuators.

Finally, we present a novel algorithm for intruder detection in a sinkhole attack of wireless sensor network. The algorithm can identify the intruder and deal with multiple malicious nodes effectively. We have evaluated the performance of the proposed algorithm through both numerical analysis and simulations, which confirmed the effectiveness and accuracy of our algorithm.

摘要

無線傳感執行器網絡(WSANs)藉著加入功能強大和可移動的執行器，大大提高了現有的無線傳感器網絡結構。結合執行器與傳感器的功能可應用在更多特殊用途上。在環境應用方面，一個高效率和可靠的通訊或協調系統能迅速和準確地報告環境事件。可是，無線隨意多跳躍的通訊在 WSAN 上並不可靠，例如頻繁傳感器失靈和網絡分割。由網絡擠塞或數據聚合而導致的過度延誤，亦會進一步加重問題。

在此論文中，我們提議一個在 WSANs 上的可靠事件報告框架。我們指出在這樣一個即時系統裏，可靠性不僅取決於準確性，亦取決於數據的重要性和即時性。依循這個論據，我們的設計結合了三個重要模塊，第一個是高效率和容錯事件數據聚合算法，第二個是即時的數據通訊協定，第三個是有效的執行器分派算法。另外，我們提出兩個 WSANs 的通訊協定，分別是針對傳輸時間的容錯數據(LOFT)通訊協定和節省能源的即時數據(POWER-SPEED)通訊協定。LOFT 透過檢查傳感器的隊列運用來平衡傳感器工作量，並且用一種能適應環境的複製算法來處理傳感器或通訊錯誤。POWER-SPEED 以節省能源的方式傳送小包，並維護小包的即時運輸。我們利用大量的模擬實驗來評估上述的事件報告框架和通訊協定，結果顯示它們在 WSANs 上的可靠性令人滿意。

爲把數據收集時間縮到最短，我們提議一個新的路線設計。我們以數學公式提出路線設計問題，並且指出這問題在一般計算上是很難處理的。接著，我們提出兩種實際高效率的算法以縮短傳感器的等候時間。我們的算法能根據傳感器的相對重要性和數據樣式來調整執行器收集傳感器數據的頻率。此外，我們提出一個能適應網絡環境變化的機率路線(PROUD)算法。我們同時提出 PROUD 的分散式方案和增設使 PROUD 可適應執行器易變速度的方案。我們也提出一些算法以平衡執行器的工作量。模擬結果顯示出我們的算法不但能有效地縮減整體數據收集的時間，而且能適應網絡環境變化和平衡執行器的能源消耗。

最後，我們提出一個爲偵查在無線傳感器網絡進行排水口攻擊(Sinkhole Attack)的入侵者的新穎算法。這算法能有效地識別入侵者及對付多個惡意結盟的傳感器。我們以數據分析和模擬實驗來評估這算法的表現，結果它的效率和準確性都被肯定了。

Acknowledgement

First and foremost, I would like to express my sincere gratitude to my supervisor Professor Michael R. Lyu for his generous guidance and patience given to me in the past six years. It took enormous time and effort to nurture a student from doing a final year project to completing a Ph.D. today. His consistent support and encouragement, as well as his inspiring advice are extremely essential and valuable in my research work. I have learned a lot from his breadth of knowledge, his enthusiasm on teaching and research, and his patience and consideration to the students.

I am also greatly indebted to Professor Jiangchuan Liu, for nourishing me to be a better researcher, and always be there for help. His insightful comments have improved my papers and this thesis tremendously, and his advice has always helped me to make the right decision. I am also very grateful for his hospitality in Simon Fraser University (SFU) for two summers, which has greatly broadened my experiences and horizon. I am so grateful to Prof. Man-Hon Wong and Prof. Moon-Chuen Lee for their precious time to serve as my thesis examiners.

I would like to thank my nice collaborator, Yangfan Zhou, for discussing research problems, working closely with me, and teaching me how to use the network simulator. I also appreciate other members in our research group, including Xia Cai, Pat Chan, Xinyu Chen, Hongbo Deng, Steven Hoi, Kaizhu Huang, Xiaoqi Li, Zhenjiang Lin, Hao Ma, Wyman Wong, Zenglin Xu, Haixuan Yang for their help and discussion in many aspects of my research work. I also express my appreciation to the faculty and staff in the Department of Computer Science and Engineering. Time spent at CUHK would not be so interesting and enjoyable without the friendship from my officemates and friends. This list is not complete: Alan Chu, Albert Lam, Chi-Kong Chan, Carmen Lam, Gordon Lam, KK Lo, Dong Ni, Jixiang Quo, Brian Tsue, Raymond Wong, and Eric Yu.

In SFU, I enjoyed two wonderful summers conducting research in the Network Modelling Laboratory, where I made a lot of friends. I would like to thank Dan Wang for his comments on my work and his help on my understanding of algorithms. I also want to thank Ming Zhang, Bin Zhou, Ming Hua, Mag Lau, Xinghuo Zeng, Roy Hu, Yong Wang, Peter Chen, Michael Letourneau, Zhengbing Bian, Yan Long, Feng Wang, Samantha Chow, and Vivian Chu for their discussions on research, their help and valuable friendship.

During my studies, I also had the good fortune to visit the Tsinghua National Laboratory for Information Science and Technology. I would like to offer my most sincere thanks to Professor Ke Xu and Professor Yong Cui for their hospitality. They introduce me to fascinating research areas and share their insights and expertise on Next Generation Internet, which have greatly broadened my knowledge and horizon.

Finally, this work is dedicated to my family. My parents teach me the value of knowledge, the joy of love, and the importance of family. My sister is my forever friend who accompanies me and brings me great happiness. I want to thank them for their love and lifelong support.

This work is dedicated to my parents and my sister for their
unconditional love and support over the years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The advancement of hardware and engineering technology has turned distributed embedded systems such as sensors, actuators, and various mobile devices [96] into reality. Wireless sensor networks (WSNs), which are formed by a group of sensors, has become extremely popular in recent years with their capability of monitoring the environments [6].

1.1 Wireless Sensor Networks

A wireless Sensor Network (WSN) is a group of sensor nodes with a wireless communications infrastructure intended to monitor and record conditions at diverse locations. It is commonly used for monitoring the environment, like temperature, humidity, pressure, wind direction and speed, illumination intensity, vibration intensity, sound intensity, power-line voltage, chemical concentrations, pollutant levels and vital body functions [66]. WSN was originally developed for military applications such as battlefield surveillance. However, it is now used in many civilian application areas, including environment and habitat monitoring, healthcare applications, home automation, and traffic control [105, 133].

A sensor node is small, lightweight and portable device that equipped with a transducer, microcomputer, transceiver and power source. It generates electrical signals based on sensed physical effects and phenomena. The microcomputer processes and stores the sensor output. The radio transceiver or other wireless communications device receives data and transmits data to the sensor nodes or computers. The power for each sensor node is derived from the electric utility or from a battery [133]. The cost of sensor nodes ranges from hundreds of dollars to a few cents, depending on the size of the sensor network and the complexity required of individual sensor nodes. Size and cost constraints on sensor nodes result in corresponding constraints on resources such as energy, memory, computational speed and bandwidth [105].

1.1.1 Characteristics

1. Self-organizing capabilities: A wireless sensor network (WSN) consists of a large number of sensor nodes. They are deployed over an area and form a wireless network. The position of sensor nodes need not be engineered or pre-determined. This allows random deployment in inaccessible terrains or disaster relief operations. On the other hand, this also means that sensor network protocols and algorithms must possess self-organizing capabilities.

2. Cooperative effort of sensor nodes: A unique feature of sensor networks is the cooperative effort of sensor nodes. Sensor nodes are fitted with an on-board processor. Instead of sending the raw data to the nodes responsible for the fusion, sensor nodes use their processing abilities to locally carry out simple computations and transmit only the required and partially processed data.

3. Short-range communication and multihop routing: Since large

number of sensor nodes are densely deployed and they are having short communication range. Hence, multihop communication in sensor networks is expected to consume less power than the traditional single hop communication. Furthermore, the transmission power levels can be kept low, which is highly desired in covert operations. Multihop communication can also effectively overcome some of the signal propagation effects experienced in long-distance wireless communication [3].

4. Limitations on energy and computation power: The sensor nodes are autonomous devices with limited battery, computational power, and memory.

5. Dynamic topology: Dynamic environmental conditions require the system to adapt over time to changing connectivity and system stimuli.

6. Operation: The complexity of wireless sensor networks generally consist of a data acquisition network and a data distribution network, monitored and controlled by a management center. The plethora of available technologies makes even the selection of components difficult, let alone the design of a consistent, reliable, robust overall system [35].

1.1.2 Applications

WSNs provide a wide variety of commercial and industrial applications to monitor data that would be difficult or expensive to monitor using wired sensors. They could be deployed in wilderness areas, where they would remain there for monitoring some environmental variables without the need to recharge or replace their power supplies. Typical applications of WSNs include monitoring, tracking, and controlling. Some of the specific applications are habitat monitoring, object tracking, nuclear reactor controlling, fire detection, traffic monitoring, etc. In general, a WSN is scattered in a region where it is meant to collect data through its sensor nodes. They can be applied in environmental monitoring, habitat monitoring, acoustic detection, seismic detection, military surveillance, inventory tracking, medical monitoring, smart spaces process monitoring, structural health monitoring, etc [4, 85, 68, 104, 134].

1.1.3 Research Challenges

It is important for sensor networks to reliably aggregate and disseminate information within a time frame that allows the controllers to take necessary actions, even in case of poor spatial distribution of sensor devices, wireless interference, and malicious destruction. Since out-of-date information is of no use, a key technical challenge in cooperative engagement is how to effectively coordinate and control sensors in real-time over an unreliable wireless network. These key challenges are described as below [116].

1. Data-centric: Sensor networks are intrinsically data-centric in which data from multiple sources related to a same physical phenomenon need to be aggregated and sent to a base station [75].

2. Location-based: Data in sensor networks usually correspond to physical locations rather than logical IDs. New data centric and location-based protocols [64, 83, 72]were developed to improve scalability and efficiency.

3. Topology: Sensors are placed in open fields for environmental applications with non-uniform node distribution and in large scale. Also, the data generation rates and traffic patterns are changing quickly and unpredictable.

4. High fault rates: Sensor networks are subject to high fault rates due to environmental noises, obstacles, power depletion, malicious destruction, node and link failures.

5. Limited resources: Sensor networks run on small devices with limited with limited battery, communication range, computation power and memory.

1.2 Wireless Sensor-Actuator Networks

Although sensors can collect data from the surroundings effectively, they are passive devices and are not interactive to the environments. Wireless sensor-actuator networks (WSANs), which include both actuators and sensors, become an extension to WSNs. Actuators are mobile devices that can make decisions and perform appropriate actions in response to the sensor measurements. They are resource-rich devices equipped with more energy, stronger computation power, longer transmission range, and are usually mobile. One example of actuators is robots, which can communicate and perform different actions. On the other hand, sensors are small and low-cost devices with limited energy, sensing, computation, and transmission capability.

Sensors and actuators collaborate together to monitor and react to the surrounding world. Sensors perform sensing and report the sensed data to actuators, while the actuators then carry out appropriate actions in response. WSANs can be applied in a variety of commercial, industrial, scientific, and military applications like environmental monitoring, sensing and maintenance in large industrial plants, military surveillance, medical sensing, attack detection, and target tracking. Apart from the above, the technologies developed can be applied to aerospace industries as well. For example, a number of sensors and actuators can be deployed on a planet for exploration. The sensors can collect data on the planet and report interesting events to the actuators. Then, actuators can go to particular locations for more detailed observations. For example, they may collect some stone samples for bringing back to the space ship, capture high-resolution pictures, or record videos for deeper investigations.

A number of applications in WSANs require a quick response from the actuators to react to the environments. For example, actuators with water sprinkler are expected to arrive at the scene of fire immediately to stop the spread of fire. Similarly, actuators are expected to react as soon as possible in applications like intruder detection or object tracking. They have to make sure the person or the object is still in the reported area when they arrive [5].

In this thesis, we propose a delay-aware reliable communication framework, which provides timely reactions to the environments upon detection of an event. We focus on event-driven applications in a self-organized network. This network has no centralized control to the sensors and actuators. Moreover, the sensors report to the actuators only when their sensed data fall in the range of interest. For example, a group of sensors will report an event when the detected temperature is over a certain degree. An event can be any incident happening in the environments being monitored, such as a fire, a leakage of gas, or an attack.

In comparing with WSNs, senor-to-actuator and actuator-to-actuator communications become a special feature in WSANs. Moreover, sensors in WSANs may have multiple potential destinations for event reporting, which is different from a single and static sink for data collection in WSNs. WSANs usually contain multiple actuators available for reaction, so a good actuator-to-actuator coordination is necessary for providing a fast and effective response. Our solution explores the different capabilities and functionalities of sensors and actuators and offers efficient communication and coordination among them. It consists of two steps: First, a delay-aware and distributed event-reporting algorithm for sensors to send the application data to the actuators; and second, an efficient coordination algorithm for designing the routes of actuators to collect data from sensors efficiently. The event-reporting algorithm allows the sensors to transmit data to actuators via the paths with minimum delay. Also, the data with more importance will be transmitted with higher priority. The actuator coordination algorithm allows the actuators to patrol on routes that minimize their average inter-arrival time to the static sensors. Since security is integral to a reliable system, we study the sinkhole attack problem in wireless sensor networks and present a novel algorithm for detecting the intruder in such an attack. Our algorithm can effectively identify the intruder through analyzing the network flow information. It is also robust in dealing with multiple malicious nodes that cooperatively hide the real intruder.

1.3 Delay-Oriented Reliable Communication

1.3.1 Delay-Aware Reliable Event Reporting

Modern hardware and software technologies for embedded systems allow real applications of micro sensors with radio transceivers [4, 38, 103, 97]. WSNs constructed by a group of sensors, have been suggested for numerous novel applications, such as monitoring for harsh environments and protecting national borders. Recently, actuator nodes, which have much stronger computation and communication power than uni-purpose micro-sensors, have also been introduced [5]. An actuator can perform diverse tasks, such as processing the data reported from the sensors and accordingly interacting with the environment; a mobile actuator (*e.g.*, a robot) could even change its location periodically to serve the application better.

The sensors and actuators can form a powerful and yet cost-effective hybrid network, that is, a WSAN. While the functionalities of the actuators are application-specific, a well-designed communication model between the two types of nodes is crucial to a WSAN. In particular, given that the actuators need accurate event data from the sensors to perform corresponding actions, reliability is an important concern in the sensor-actuator communication.

In this thesis, we design a generic framework for reliable event reporting in WSANs. We argue that the reliability in this context is closely related to the delay, or the freshness of the events, and they should be jointly optimized. We also suggest that the non-uniform importance of the events can be explored in the optimization. We therefore present a delay- and importance-aware reliability index for the WSANs. Our framework seamlessly integrates three key modules to maximize the reliability index: 1) A multi-level data aggregation scheme, which is fault-tolerant to error-prone sensors; 2) A priority-based transmission protocol, which accounts for both the importance and the delay requirements of the events; and 3) an actuator allocation algorithm, which smartly distributes the actuators to match the demands from the sensors.

Our framework is fully distributed, and is generally applicable for diverse WSANs. Within this generic framework, we present an optimized design for each of the modules, and also discuss their interactions. The performance of our framework is evaluated through extensive simulations. The results demonstrate that our framework can significantly enhance the reliability of event reporting; it also makes more effective use of the expensive actuators.

1.3.2 Latency-Oriented Fault Tolerant Transport Protocol

Actuators, which have much stronger computation and communication power, can process the data reported from the sensors and interact with the environment accordingly [47]. A well-designed communication module between the two types of nodes is crucial to WSANs [4]. In particular, given that the actuators need timely event data from the sensors to perform corresponding actions, reliability is an important concern in the sensor-actuator communication. Unfortunately, the low-power multi-hop communications in WSANs are inherently unreliable. The frequent sensor failures and the excessive delays due to node and link failures or congestions further worsen the problem.

To provide reliable communication, we focus on the design of a latency-oriented fault tolerant data transport protocol in WSANs. We contend that the reliability in this context requires successful arrivals of packets within a latency bound, and a cross-layer design is thus necessary. Specifically, the transmission delay can be jointly optimized with the path success rate of transmission to handle node and link failures. We also suggest that the non-uniform importance of the events of interest can be explored in the optimization. Our protocol consists of two steps: First, it estimates the load of the neighbors and decides the next hops which can provide on-time delivery of event data to the actuator; and second, it copes with the transmission failures by providing redundant packets adaptively. Simulation results show that it remarkably improves the reliability for data reporting from sensors to actuators with reasonable overheads.

1.3.3 Power-Controlled Real-Time Data Transport Protocol

There have been many novel potential applications of WSNs as well as WSANs , such as environmental monitoring and national-borders protection [5, 85, 124]. In WSANs, data collection on the physical phenomena of interest is still a major and critical work as in WSNs. Usually, it is required that the actuators can react with the sensing events quickly and perform interactions with the environment accordingly. For example, if the sensors in a WSAN that monitors a forest-beat report abnormal temperature in some area, the actuators should move to that area to check whether there is a fire in a timely manner. QoS guarantee in the domain of timeliness is highly desired in the sensor-to-actuator data transport protocol for delay-sensitive event-reporting of WSANs.

Unfortunately, like in WSNs, sensors in WSANs are still inherently subject to limited energy resource as recharging the batteries of sensors is impractical due to the unattended operational features of the network [5]. Therefore, the main challenge of designing a real-time data transport protocol for delay-sensitive event-reporting of WSANs is to achieve energy-efficiency. The protocol should guarantee that the actuators can receive packets on delay-sensitive events in given latency bounds and the transport scheme should cost as low energy as possible.

There are many existing real-time data transport protocols in the literature for WSNs. Lu *et al* [83] described a packet scheduling policy called Velocity Monotonic Scheduling. It accounts for both time and distance constraints. SPEED [52] by He *et al* combines feedback control and non-deterministic QoS-aware geographic forwarding. Felemban *et al* [41] proposed a Multi-path and Multi-Speed Routing Protocol (MMSPEED) for probabilistic QoS guarantee in WSNs. Multiple QoS levels are provided in the timeliness domain by using different delivery speeds, while various requirements are supported by probabilistic multipath forwarding in the reliability domain.

However, the network and traffic features of WSANs are different from those of WSNs. In WSANs, actuators are mobile sinks of the data traffic. The mobility of data sinks poses that the traffic routes must be reestablished frequently. The transport protocol for WSANs should be stateless. Also, actuators can directly communicate with each other as they can be equipped with powerful antennae. Therefore, sensor reporting packets can be delivered in an *anycast* manner. The transport protocol for WSANs can convey packets to any of the actuators. Based on this feature, we specifically tailor the solution of the real-time data transport problem for WSANs. We address this challenging problem by proposing POWER-SPEED, a real-time data transport protocol for WSANs to achieve energy-efficient data transport for delay-sensitive event reporting. With POWER-SPEED, a sensor node can estimate the downstream QoS condition to actuators based on the spatio-temporal historic data of the upstream paths.

Without requiring any control packets, sensor nodes can select the next-hop neighbor to actuators according to the estimation. Each node then efficiently adjust the power level of its wireless transmitter to a minimum value under the constraint that the packet sent by this node could just reach its intended neighboring node. In this way, POWER-SPEED reduces the energy consumption of data transport while maintaining QoS requirement in the timeliness domain.

1.4 Delay-Oriented Reliable Coordination

1.4.1 The Route Design Problem

WSNs have been applied in environment monitoring, battlefield surveillance, chemical attack detection, and target tracking [4, 38]. The asymmetric communication patterns from sensors to the sink, however, often overload the sensors close to the sinks and reduce the network lifetime. Moreover, network partitions may occur in sensor networks, which make multihop communications impossible. To alleviate these problems, mobile elements, such as mobile sinks [119] or mobile relays [84], have been suggested for collecting data in WSNs. In WSAN, an actuator (*e.g.*, a robot) can move around to cover the sensing field and interact with static sensors. Each static sensor has a limited buffer, which stores locally sensed data until some actuator approaches. It can then upload the data to the actuator with short-range communications and free its buffer.

Note that the amount and frequency of data generation in the sensors are non-uniform [114]. For example, the sensors which collect more urgent or sensitive data are of higher importance or weights, and thus they expect shorter actuator inter-arrival time to report data more frequently. Similarly, locations with significant data variations are also expected more frequent visits.

More formally, there is a Route Design Problem (RDP) for the actuators to minimize their average inter-arrival time to the static sensors. In this thesis, we present the first formal study on the RDP problem. We demonstrate that the general problem is NP-hard. We then develop two effective algorithms for route calculation. Our algorithm is based on the observation that a sensor can have shorter waiting time for data uploading if it is included in more routes and/or shorter routes. To this end, our proposed Route Design Algorithm by Varying Number of Visits (RDNV) constructs routes with similar lengths, while the sensors are visited by different number of routes. On the other hand, the Route Design Algorithm by Varying Path Length (RDPL) algorithm constructs routes with different lengths and visits the sensors by distinct routes according to their weights. Apart from the centralized approaches, we also present a distributed implementation of the route design algorithms for large-scale sensor networks. Our simulation results show that the proposed algorithms can effectively reduce the overall data collection time.

1.4.2 Adaptive Delay-Minimized Route Design

As mentioned above, the amount and frequency of data generation in the sensors are often non-uniform [114]. For example, the sensors with higher data generation rate or the locations with higher event occurring probability expect more frequent visits. In tackling RDP problems, while deterministic algorithms may work for static route calculations, in reality, the weight of sensors and event frequency are time varying, which call for an adaptive algorithm. A distributed and load-balanced implementation is also necessary for a large-scale sensor network with multiple actuators.

In this thesis, we propose an effective and adaptive algorithm, called Probabilistic Route Design (PROUD) for route calculation. Our algorithm is based on a probabilistic model, which allows sensors to have shorter expected waiting times for data uploading if they are assigned with higher weights. Specifically, PROUD constructs a priori route that consists of all the sensor locations, while the actuators visit them probabilistically and cyclically according to their weights. It can adapt to network dynamics by updating the visiting probability easily without any re-calculation on the priori route. We show that this approach works for both small-scale and large-scale sensor-actuator networks, and present a distributed implementation. We further introduce enhancements to accommodate actuators with variable speeds, which works for applications that demand bounded inter-arrival times. Finally, we devise a Multi-Route Improvement and a Task Exchange algorithm that enable load balancing. Our simulation results show that the proposed algorithm can effectively reduce the overall data collection time and evenly distribute the energy consumption among the actuators.

1.5 Intruder Detection in Sinkhole Attack

WSNs and WSANs consist of a set of geographically distributed sensor nodes, which continuously monitor their surroundings and forward the sensing data to a base station (referred to as a sink) through multi-hop routing. Given the importance of the sensed data, various attacks have targeted sensor networks [71, 122]. While some of the attacks are common in different types of networks, the many-to-one communication pattern between the sensors and the sink poses unique challenges [70, 100, 112]. A typical example is the sinkhole attack, where an intruder attracts surrounding nodes with unfaithful routing information, and then alters the data passing through it or performs selective forwarding [71, 23, 25]. The sinkhole attack prevents the base station from obtaining complete and correct sensing data, and thus leads to a serious threat. It is particularly severe for wireless sensor networks given that the wireless links are vulnerable and the sensors are often deployed in open areas with weak computation and battery power. The existing routing protocols in sensor networks are generally susceptible to the sinkhole attack [71, 64, 127, 13]. Although some secure mechanisms are proposed and make use of cryptographic techniques to protect network traffic [61, 101, 24, 26], they are often localized, or suffer from high computation overheads and require time synchronization among the nodes.

To ensure the reliability and security of the network, we propose a novel lightweight algorithm for detecting the sinkhole attack and identifying the intruder involved. We deviate from the traditional strategy of defending against an attack using cryptography; instead, we first detect the attack by observing the network flow information, and then identify the intruder and malicious nodes, which can later be isolated to protect the network.

We focus on a general many-to-one communication model, where the routes are established based on the reception of route advertisements. Our solution explores the asymmetricity between the sensor nodes and the base station, and makes effective use of the relatively-high computation and communication power of the base station [71, 111, 100]. It consists of two major parts: (1) a secure and low-overhead algorithm for the base station to collect the network flow information from the attacked area; and (2) an efficient identification algorithm that analyzes the routing pattern and locates the intruder. We also consider the complex scenario with colluding nodes that cooperatively cheat the base station about the intruder location. Specifically, we examine multiple suspicious nodes and identify the intruder with a voting method. We show that the solution is correct as long as the normal nodes are dominant.

1.6 Contribution of This Thesis

In this thesis, we propose a unified solution for providing delay-oriented reliable communication and coordination in wireless sensor-actuator networks. To achieve this, we present a real-time communication framework for delay-minimized data transmission from sensors to actuators and effective coordination for fast reactions by actuators in response to the sensed events. In particular, we focus on the delay-aware event reporting and the route design problem in WSANs. Delay-aware event reporting is a general reliability-centric framework that provides a multi-level data aggregation, priority-based data transmission, and actuator allocation. Apart from the communication, we present the route design problem and propose several solutions, which aim at designing effective routes to minimize the actuators inter-arrival time to sensors in data collection. Finally, we study the security issues in wireless sensor networks. Particularly, we propose an effective intruder detection algorithm for sinkhole attacks in WSNs. The main contributions of this thesis can be further described as follows:

1. *Real-Time Communication Framework*
   As actuators perform actions in response to the sensed events in WSANs, real-time communications and quick reaction are necessary. Two major problems are discussed: How to minimize the transmission delay from sensors to actuators, and how to improve the coordination among the actuators for fast reaction. To tackle these problems, a real-time communication framework is presented to support event detection, reporting, and actuator coordination. It explores the timely communication and coordination problems among the sensors and actuators. We propose two self-organized and distributed algorithms for event reporting and actuator coordination.

2. *Delay-Aware Reliable Event Reporting*
   A general reliability-centric framework for event reporting is proposed. We argue that the reliability in such a real-time system depends not only on the accuracy, but also the importance and freshness of the reported data. Our design follows

this argument and seamlessly integrates three key modules that process the event data, namely, an efficient and fault-tolerant event data aggregation algorithm, a delay-aware data transmission protocol, and an adaptive actuator allocation algorithm for unevenly distributed events. Our transmission protocol adopts smart priority scheduling that differentiates event data of non-uniform importance. It is followed by an actuator allocation algorithm, which smartly distributes the actuators to match the demands from the sensors.

3. *Latency-Oriented Fault Tolerant Transport Protocol*
   A latency-oriented fault tolerant data transport protocol is proposed for WSANs. We argue that reliable data transport in such a real-time system should resist to the transmission failures, and should also consider the importance and freshness of the reported data. We articulate this argument and provide a cross-layer two-step data transport protocol for on-time and fault tolerant data delivery from sensors to actuators. Our protocol adopts smart priority scheduling that differentiates the event data of non-uniform importance. It balances the workload of sensors by checking their queue utilization and copes with node and link failures by an adaptive replication algorithm.

4. *Power-Controlled Real-Time Data Transport Protocol*
   A real-time data transport protocol (POWER-SPEED) is proposed for WSANs to achieve energy-efficient data transport in delay-sensitive event reporting. In POWER-SPEED, sensor nodes select their next-hop neighbors to actuators according to the spatio-temporal historic data of the upstream QoS condition, which completely avoids control packets. With an adaptive transmitter power control scheme, POWER-SPEED conveys packets in an energy-efficient manner while maintaining soft real-time packet transport. It reduces the energy consump-

tion of data transport while ensuring the QoS requirement in the timeliness domain.

5. *The Route Design Problem*
   Multiple actuators can patrol along different routes and communicate with the static sensors. To minimize the data collection time, an effective route design is crucial for the actuators to travel in the sensed field. We present a mathematical formulation of the route design problem, and show that the general problem is computationally intractable. We then propose two practically efficient algorithms to reduce the waiting time for the sensors. Our algorithms adaptively differentiate the actuator visiting frequencies to the sensors according to their relative weights and data generation patterns.

6. *Adaptive Delay-Minimized Route Design*
   We propose PROUD, a probabilistic route design algorithm for wireless sensor-actuator networks in a stochastic and dynamically changing sensing environment. This is a departure from the previous static and deterministic mobile element scheduling problems. PROUD offers delay-minimized routes for actuators and adapts well to network dynamics and sensors with non-uniform weights. This is achieved through a probabilistic visiting scheme along pre-calculated routes. We present a distributed implementation for route calculation in PROUD and extend it to accommodate actuators with variable speeds. We also propose the Multi-Route Improvement and the Task Exchange algorithms for load balancing among actuators.

7. *Intruder Detection for Sinkhole Attack*
   A sinkhole attack forms a serious threat to sensor networks, particularly considering that the sensor nodes are often deployed in open areas and of weak computation and battery power. A novel algorithm is presented for detecting the intruder in a sinkhole attack. The algorithm first finds a list of suspected nodes

through checking data consistency, and then effectively identifies the intruder in the list through analyzing the network flow information. The algorithm is also robust to dealing with multiple malicious nodes that cooperatively hide the real intruder. It is efficient as its communication and computation overheads are reasonably low for wireless sensor networks.

1.7 Organization of this Thesis

The rest of this thesis is organized as follows: In the next chapter, we present the background on Wireless Sensor Networks (WSNs) and Wireless Sensor-Actuator Networks (WSANs), followed by the related work on delay-oriented reliable communication and coordination in WSANs and intrusion detection in WSNs. Chapter 3 presents a real-time communication framework to support event detection, reporting, and actuator coordination in WSANs. Chapter 4 proposes a delay-aware reliable event reporting scheme, which includes an efficient and fault-tolerant event data aggregation algorithm, a delay-aware data transmission protocol, and an adaptive actuator allocation algorithm. Chapter 5 focuses on a latency-oriented fault tolerant data transport protocol (LOFT), which can handle node and link failures. Chapter 6 further describes an energy-efficient data transport protocol (POWER-SPEED) for delay-sensitive event reporting. Chapter 7 formally presents the route design problem in WSANs and proposes two practically efficient algorithms to reduce the waiting time for the sensors. Chapter 8 proposes a probabilistic route design (PROUD) algorithm for wireless sensor-actuator networks in a stochastic and dynamically changing sensing environment. Chapter 9 presents an efficient algorithm for detecting the intruder in a sinkhole attack in a sensor network. Finally, Chapter 10 summarizes the thesis and discusses future directions.

□ End of chapter.

Chapter 2

Background and Literature Review

2.1 Wireless Sensor-Actuator Networks

Wireless Sensor-Actuator network (WSAN) is an extension of traditional static wireless sensor networks. It is referred to a group of sensors and actuators linked by wireless medium to perform sensing and acting tasks. WSAN is capable of observing the physical world, processing the data, making decisions based on the observations and performing appropriate actions [5]. For example, sensors are deployed in a forest for detecting a fire, while actuators are equipped with water sprinkle, such that they can easily be extinguished before being spread uncontrollably.

In WSANs, sensors are actuators work closely with each others. Sensors perform sensing from the environment and actuators react on the environment in response to the sensor readings. Since actors are resource-rich nodes equipped with better processing capabilities, higher transmission powers and longer battery life, they can perform more complicated and energy consuming activity than sensing task. They can respond rapidly to sensor input, so real-time communication is very important in WSANs since actions are performed on the environment after the sensing occurs [2].

2.1.1 Characteristics

As mentioned above, the sensing and acting are performed by sensors and actuators, respectively. WSANs have the following characteristics [5].

1. Heterogeneous property: Sensors are low-cost, low power devices with limited sensing, computation, and wireless communication capabilities, while actuators are resource rich devices equipped with better processing capability, higher transmission powers and longer battery life. They cooperate together to form a heterogeneous network that is different from traditional WSNs.

2. Real-time requirement: Many applications in WSANs require the actuators to respond quickly to the sensing data, so real-time event reporting and reaction is very important. For example, the intruder may already went away if the data is reported to the actuators too late or the actuators react too slowly.

3. Coordination: Different from WSNs, there is no centralized server (or sink) for data collection and coordination. On the contrary, sensor-actuator and actuator-actuator coordination provide transmission of event from sensor to actuators and make decisions on the most appropriate action among actuators.

2.1.2 Applications

WSANs can be an integral part of systems such as battlefield surveillance, nuclear, biological or chemical attack detection, home automation and environmental monitoring [5]. More specially, it can be applied in habitat monitoring, forest fire detection, smart environments, flood detection, damage assessment, nuclear, biological and chemical contamination detection, monitoring critical resources, vehicle tracking, car thefts detection, traffic control, etc. For example, in fire detection applications, sensors can relay the exact origin and intensity of the fire to water sprinkler actors so that the fire can easily be extinguished before it spreads. Similarly, motion and light sensors in a building can detect the presence of intruders, and command cameras or other instrumentations to track them. Furthermore, sensors for structural health monitoring in airplanes or spaceships can drive instruments to timely take countermeasures against critical mechanical stress or structural faults. As a last example, in earthquake scenarios sensors can help locate survivors and guide robots performing rescue operations [89].

2.1.3 Research Challenges

From the above characteristics, we have the following research challenges in WSANs [29].

1. Sensor-Actuator Coordination: Single or multiple actuators can receive the information from sensors about the sensed phenomenon. Single-Actuator (SA) or Multiple-Actuator (MA) should be investigated to ensure synchronization in the reporting time of the sensed data and the communication pattern is appropriate for given applications.

2. Coordination among Actuators: A communication model should be provide among the actuators for efficient data transmission. Also, algorithms are needed to coordinate and assign task to the actuators.

3. Transport Layer: The transport protocol should support both the reliability and real-time requirements.

4. Routing Layer: Cluster-based architecture with an actuator as clusterhead may be considered. The routing of data should be adaptive to the mobility of actuators. Also, it should also support the real-time and reliability requirements.

5. Medium Access Control: Effective MAC protocol is required to transmit the event information from large number of sensors to actuators and among the actuators. They should satisfy the real-time constraint on computation and communication.

6. Cross-Layering: Cross-laying can be considered to integrate the current layered protocol designs. The new approach is expected to enhance the performance and flexibility for WSANs due to constraints of low energy consumption and low latency.

2.2 Communication and Coordination Framework in WSAN

Although a number of protocols are proposed for WSN, they may not work well when applying directly to WSAN. There are additional considerations on the heterogeneous characteristics, network structure, and different operations among the sensors and actuators. Particularly, the sensor-to-actuator communications and actuator-to-actuator communications are the unique features of WSAN in comparison with WSN. Several investigations have been done on exploring the heterogeneous sensor networks [90, 126], but they do not cope with the special features and ways of operations in WSAN. For WSAN, Hu et al [58] propose an anycast communication paradigm. It constructs an anycast tree rooted at each event source and updates the tree dynamically according to the join and leave of the sinks. Their approach discovers the routes by flooding of the interest from the sinks. Also, E. Cayirci et al [16] propose a power-aware many-to-many routing protocol. Actuators register the types of the data that they are interested by broadcasting a task registration message. Then, the sensors build their routing tables accordingly. In this scheme, the sensed data generated by any sensor node are forwarded to every actuator that is interested in that type of data, which may produce a lot of network traffic. Moreover, both approaches overlook the coordination among the sensors and actuators, which can be improved to increase the efficiency of event reporting and reaction. Furthermore, Melodia et al. [89] propose a distributed coordination framework for wireless and actuator networks based on an event-driven clustering paradigm. All sensors in the event area forward their readings to the appropriate actors by the data aggregation trees. Their work assumes immobile actuators that can act on a limited area defined by their action range, and provides actuator-actuator coordination to split the event area among different actuators. Our work shares the similar event-driven hypothesis, but we propose an event-reporting algorithm which divides the event area into pieces of maps and transmits the sensed data with special ordering to reduce the response time. Moreover, our actuator coordination algorithm can support mobile actuators under sparse deployment. Apart from the above, various papers discuss the research challenges and work on diverse topics in WSAN. Dinh et al. [29] review the recent research achievements and open research issues, and evaluate the performance of three popular ad-hoc network routing protocols in handling actuator-to-actuator communications. Durresi et al. [34] present a geometric broadcast protocol for WSAN (GBSA). It is a distributed algorithm where nodes make a local decision on whether to transmit based on a geometric approach. M. Coates [21] addresses the evaluation of causal relationships in WSAN, so that the expected marginal response of a system can be estimated. Hu and Cao [56] propose a two-level re-keying/re-routing scheme and a multiple-key management scheme to provide security for WSAN. Ganeriwal et al. [44] consider a network where nodes have traction ability. They present methods for the network to be aware of its own integrity and use actuators to repair the coverage loss in the area being monitored.

2.3 Delay-Aware Reliable Event Reporting in WSAN

Wireless sensor networks (WSNs) have been extensively studied recently; see surveys in [4, 38, 103]. Efficient and reliable event reporting is also an important issue in WSANs. Some related protocols are proposed for wireless ad hoc networks [19, 113, 18, 62], but they are impractical for large scale dynamic sensor networks. For WSNs, He et al. [52] proposed a real-time communication protocol SPEED, which combines feedback control and non-deterministic quality of service (QoS) aware geographic forwarding. Lu et al. [83] described a packet scheduling policy, called Velocity Monotonic Scheduling, which inherently accounts for both time and distance constraints. Felemban et al. [40] proposed Multi-path and Multi-Speed Routing Protocol (MMSPEED) for probabilistic QoS guarantee in WSNs. Multiple QoS levels are provided in the timeliness domain by using different delivery speeds, while various requirements are supported by probabilistic multipath forwarding in the reliability domain. Huang et al. [15] proposed a spatiotemporal multicast protocol, called “mobicast”, which provides reliable and just-in-time message delivery to mobile delivery zones. Ergen et al. [37] presented a routing algorithm that maximizes the sensor network lifetime, and further incorporates delay guarantees into energy efficient routing by limiting the length of paths from each sensor to the collection node.

For reliable data transport with transmission failures, Aidemark et al. [1] presented a framework for achieving node-level fault tolerance (NLFT) using time-redundant task scheduling in the nodes. Ganesan et al. [45] described the use of multipath routing for energy-efficient recovery from node failures in wireless sensor networks, proposing and evaluating the classical node-disjoint multipath and the braided multipath designs. Djukic et al. [30] used path diversification to provide QoS, which bounds on the end-to-end delay and probability of packet loss (PPL) by transmitting packets with erasure codes and multiple paths. S. Jain et al. [65] considered the problem of routing in a delay tolerant network in the presence of path failures. Jain proposed an approach that improves the probability of successful message delivery by applying a combination of erasure coding and data replication. Wang et al. [118] studied direct transmission and flooding on delay and fault tolerant mobile sensor network (DFT-MSN) and introduced an optimized flooding scheme that minimizes the transmission overhead of flooding. There are also related works in the general embedded or delay-tolerant network settings. For example, Khanna et al. [73] suggested that the failure of any node in a path can be detected and circumvented using backup routes. Assayad et al. [8] proposed a bi-criteria scheduling heuristic in data-flow graphs to maximize the system's reliability and minimize the system's run-time. Dubois-Ferriere et al. [32] introduced a scheme for error-correction that exploits temporal and spatial diversity through packet combining.

Our work is motivated by the above studies. The key difference is that we focus on the interactions between sensors and actuators, not uniform network nodes. In this context, additional considerations are needed to address the heterogeneous characteristics and the unique interactions within the network.

There have been studies exploring heterogenous sensor networks, *e.g.*, [108, 90, 126], but they do not cope with the special features of actuators. For WSAN, Hu et al. [58] proposed an anycast communication paradigm. This constructs an anycast tree rooted at each event source and updates the tree dynamically according as the sinks join and leave the network. E. Cayirci et al. [16] offered a power-aware many-to-many routing protocol. Actuators register the data types of interest by broadcasting a task registration message; the sensors then build their routing tables accordingly. Melodia et al. [89] further presented a distributed coordination framework for WSANs based on an event-driven clustering paradigm. All sensors in the event area forward their readings to the appropriate actors by the data aggregation trees. While these works have explored the potentials of WSANs, reliability issues, and in particular, the reliability of event reporting from sensors to actuators, have yet to be addressed.

Transmission failures frequently occur in wireless communications [135, 120]. Dubois-Ferriere et al. [32] introduced a scheme for error-correction that exploits temporal and spatial diversity through packet combining. Ganesan et al. [45] described the use of multipath routing for energy-efficient recovery from node failures in wireless sensor networks, which presents and evaluates the classical node-disjoint multipath and the braided multipath designs. S. Jain et al. [65] considered the problem of routing in a delay tolerant network in the presence of path failures. It improves the probability of successful packet delivery by applying a combination of erasure coding and data replication. Wang et al. [118] investigated direct transmission and flooding on the delay and fault tolerant mobile sensor network (DFT-MSN), and introduced an optimized flooding scheme that minimizes the transmission overhead of flooding.

Our work is motivated by the above studies. The key difference is that we provide an effective protocol for latency-oriented fault tolerant data reporting, and focus on the timely and reliable interactions between sensors and actuators. Although there have been studies exploring the heterogenous sensor networks, *e.g.*,[90, 126], as well as wireless sensor-actuator networks [89], the reliability issues, in particular those for data transport from sensors to actuators, have yet to be addressed.

2.4 Delay-Minimized Route Design in WSAN

Mobile elements have been proposed to carry data in wireless networks. Shah et al. [110] presented an architecture using moving entities (data mules) to collect sensor data. There have also been studies on mobile sinks with predictable and controllable movement patterns [17, 69], and the optimal time schedule for locating sojourn points [119]. Apart from the above, Zhao et al. [137] proposed a message ferrying (MF) approach to address the network partition problem in sparse ad hoc networks. Luo et al. [84] investigated a joint mobility and routing algorithm with mobile relays to prolong the lifetime of wireless sensor networks. Gu et al. [49] proposed a partitioning-based algorithm to schedule the movement of mobile element (ME) to avoid buffer overflow in sensors and reduce the minimum required ME speed. Their solution was customized for an "eyes" topology, where the events are concentrated at certain locations. Solutions for sensor networks with general uniform distribution were left to be explored. Bisnik et al. [12] studied the problem of providing quality coverage using mobile sensors and analyzed the effect of controlled mobility on the fraction of events captured. Their focus is not on the route design problem. Recently, a route design algorithm for multiple ferries is proposed by Zhang et al. [132]. It considers a delay tolerant network scenario with point-to-point data transfer between sensors of uniform weights. Zhao et al. [136] considered the Message Ferrying (MF) scheme which exploits controlled mobility to transport data in delay-tolerant networks, where end-to-end paths may not exist between nodes. Wu et al. [123] proposed a logarithmic store-carry-forward routing (SCFR) protocol in MANETs, which exploits node mobility to assist message delivery especially in unconnected networks. Han et al. [50] addressed the problem of deploying a minimum number of relay nodes to achieve diverse levels of fault-tolerance in the context of heterogenous wireless sensor networks, where target nodes have different transmission radii. Zhang et al. [130] studied four fault-tolerant relay node placement problems, discussed their computational complexity and presented a polynomial time approximation algorithm with a small approximation ratio.

Our work is motivated by the above investigations. The key difference is that we focus on adaptive and distributed route design for multiple mobile components in WSANs, specifically, actuators moving along independent routes. Our objective is to minimize the average actuator inter-arrival time in dynamically changing environment. We also address issues regarding the non-unform weights of the static sensors. This is different from the previous mobile element scheduling problems where one seeks to minimize the waiting time in a static and deterministic environment.

A closely related problem to RDP is the vehicle routing problem (VRP), which considers scheduling vehicles stationed at a central facility to support customers with known demands, targeting at minimizing the total distance travelled [20]. There are a number of variations to VRP, including the Capacitated VRP (CPRV)[106] and VRP with time windows (VRPTW)[79]. While these investigations have studied the routes of mobile components, the unique features of the actuators and the heterogeneous sensor networks have yet to be explored.

Capacitated VRP (CPRV) fixes all vehicles to have uniform capacity, and aims at serving known customer demands for a single commodity from a common depot at minimum transit cost [106]. The VRP with time windows (VRPTW) is the same problem that VRP with the additional restriction that a time window is associated with each customer, defining an interval wherein the customer has to be supplied [77, 79]. While these works have studied the routes of mobile components, the special features of actuators and the heterogeneous sensor networks have yet to be explored.

Different from the above, our problem do not fix the visiting time to the sensors. Moreover, we target at differentiating the visiting frequency among the sensors with different weights.

Our work is also motivated by the studies on robotics. Koenig et al. [74] suggested a generic framework for auction-based multi-robot routing and analyzes a variety of bidding rules for different team objectives.

2.5 Intruder Detection in Sinkhole Attack

Intrusion detection has long been an active research topic in the Internet evolution [27]. Recently, many detection algorithms have been proposed for wireless ad hoc networks as well. Most of them assume uniform nodes and symmetric data communication patterns between the nodes [25, 131, 60]. The one-to-many communication pattern in wireless sensor networks however poses different challenges, in particular, the sinkhole attack. The weaker computation and battery power of the sensor nodes further aggravates the problem. Pirzada et al. [102] applied a trust scheme to the routing protocol to detect sinkhole and wormhole attacks in a sensor network, but it requires the nodes to operate in a promiscuous mode. Hu et al. [59] introduced packet leash, which confides the maximum transmission time and distance of each packet. It assumes that a node can obtain a key for any other node and authentication is applied to each data packet. On the contrary, our work does not require the nodes to support promiscuous mode nor authenticate every data packet. Our work is also motivated by the existing studies on filtering false reports or avoiding jammed failed nodes in sensor networks. Specifically, Doumit and Agrawal [31] showed a self-organized criticality (SOC) and Hidden Markov models based algorithm, which can effectively detect data inconsistencies in sensor networks. Wood et al. [121] proposed a mechanism for detecting and mapping jammed regions. They described a protocol for identifying the surrounding nodes of a jammer, thus avoiding the broken links and congested nodes in the region. Staddon et al. [115] demonstrated that the topology of the network could be efficiently conveyed to the base station, allowing for quick tracing of failed nodes with moderate communication overhead. Ding et al. [28] further proposed a localized algorithm for identifying faulty sensors. Ye et al. [128] presented a Statistical En-route Filtering (SEF) mechanism for detecting and dropping false reports. It integrates multiple authentication codes, probabilistic verification, and data filtering to determine the truthfulness of each report. Our work differs from them in that we focus on sinkhole attacks, where the intruder and multiple malicious nodes actively disturb the one-to-many data communications or even interfere the identification algorithm itself.

□ **End of chapter.**

Chapter 3

A Real-Time Communication Framework for WSAN

3.1 Overview

Wireless sensor-actuator network (WSAN) comprises of a group of distributed sensors and actuators that communicate through wireless links. Sensors are small and static devices with limited power, computation, and communication capabilities responsible for observing the physical world. On the other hand, actuators are equipped with richer resources, able to move and perform appropriate actions. Sensors and actuators cooperate with each other: While sensors perform sensing, actuators make decisions and react to the environment with the right actions. WSAN can be applied in a wide range of applications, like environmental monitoring, battlefield surveillance, chemical attack detection, intrusion detection, space missions, etc. Since actuators perform actions in response to the sensed events, real-time communications and quick reaction are necessary. To provide effective applications by WSAN, two major problems remain: How to minimize the transmission delay from sensors to actuators, and how to improve the coordination among the actuators for fast reaction. To tackle these problems, we designed a real-time communication framework to support event detection, reporting, and actuator coordination. This thesis explores the timely communication and coordination problems among the sensors and actuators. Moreover, we proposed two self-organized and distributed algorithms for event reporting and actuator coordination. Some preliminary results are presented to demonstrate the advantages of our approach.

3.2 Real-Time Communication Framework

As mentioned earlier, we consider a network which consists of a large number of sensors and multiple actuators. Sensors are static and resource-limited devices for monitoring the environments and reporting events to the actuators. Actuators are mobile devices with richer resources and longer-range transmission that enable them to communicate with each other directly. We assume an event-driven model in the framework, in which sensors only send application data to the actuators when they discover an event. Upon receiving the events, actuators process the data and decide upon which to perform actions.

Many applications in WSAN require real-time response to the physical world. A real-time communication framework, which provides efficient communication and coordination among sensors and actuators, is essential to achieve a timely reaction. Such a framework relies on a low latency communication in the event reporting process from sensors to actuators, and a well-organized coordination algorithm that ensures a quick move to the event area by the actuators.

Indeed, our real-time communication framework focuses on providing a low latency event-reporting algorithm for sensor-to-actuator communication and an effective coordination algorithm among the actuators. Figure 9.1 shows the workflow of our real-time communication framework. Firstly, a group of sensors detect an event in the area where they are located. The sensors, which sensed the event earlier, start clustering and aggregating the data from their surrounding nodes. The event area is divided into pieces of maps according to the clusters formed. The maps and the corresponding data are reported to their closest actuators separately. Also, the data are transmitted in a special order, such that the most important data are sent first, with the following details later. This ensures the actuators can obtain a rough image on the event within a short time. Without waiting for the arrival of a complete report, the actuators can already start their coordination. They combine the maps they received and determine how many and which actuators should perform the actions. The size of the event area, together with the distance between the event area and the actuators, are significant factors in making a decision. Intuitively, a larger event area requires more number of actuators to perform the actions. Also, actuators located closer to the event are normally a better choice for reaction as they can arrive at the scene of event faster. Finally, the assigned actuators move to the event area and perform appropriate actions with proper location update mechanism. In this thesis, we mainly focus on the event-reporting algorithm and the actuator coordination algorithm, but we also provide a comprehensive view from event detection and report regarding actuator coordination and reaction.

We assume that every sensor and actuator in the network knows its location. This is quite natural as nodes should be able to recognize the locations of the events in order to monitor, report, and react. Their locations can be obtained by equipping with a GPS receiver [87] or the position can be determined by some localization techniques [57, 109, 51]. Our framework also adopt the geographical-based protocols for routing as they scale up well and can adapt to the location changes of the actuators easily. In geographical-based routing, locations of nodes are exploited to route data in the network [80, 125]. A routing protocol can control certain system parame-

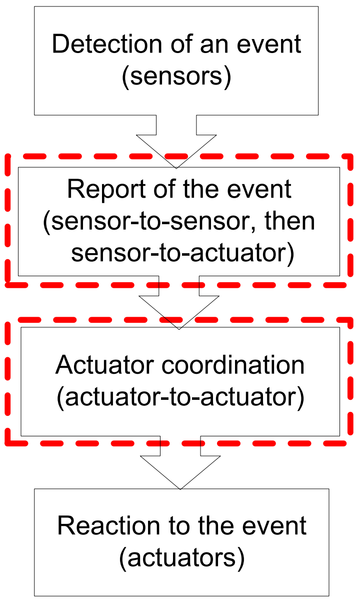


Figure 3.1: Workflow of the framework.

ters in order to adapt to the current network conditions as well as the available energy levels. For example, the transmission delay can be considered when a node is selecting a neighbor that the message will be forwarded to [52]. Finally, we impose a virtual grid structure on the network, so a simplified coordinate representation can be applied to ease the formation and combination of the pieces of maps during event reporting and actuator coordination. Although the network area can be represented by grids of equal size, the initial (x,y) coordinate system is still employed as the basic scheme in our framework.

For the ease of exposition, in Table 9.1, we list the major notations used throughout this thesis.

Table 3.1: List of Notation

v	Sensor node
r	Sensor that builds and reports a piece of map in event area
a_i	Actuator
e	Event ID
h	Hops to the r in its cluster (depth)
v_h	Sensor node with depth h from r
n_v	Neighbors of node v
(x_v, y_v)	Coordinates of node v
$data_v$	Data collected by node v
S_r	Nodes on the map being reported by r
B_r	Boundary nodes on the map being reported by r
$mean_{v_h}$	Mean from v_h and its descendants with depth $h+1$
$MEAN_{S_r}$	Overall mean among S_r
C	Center of the map S_r
l_v	Location of node v
d_v	Distance from node v to r
$R(a_i)$	Voronoi cell associated with a_i

3.3 Event Detection and Report

3.3.1 Formation of Maps

Sensor actuator networks can be applied in event-oriented applications such as fire detection, gas leakage detection, intruder detection, etc. In these systems, the sensors collect data from the environment and report special events to the actuators. As we know, sensor networks contain a lot of redundant information. To reduce the network traffic, the sensor will aggregate event reports from the neighboring nodes. The sensors r, which detected an event the earliest, start the formation of maps as shown in algorithm 14:

For building a map, node v floods the event detection messages

Algorithm 1 Formation of Map

```
for nodes $r$ detected an event do
   if data aggregation not yet started then
      Broadcast $DetectEvt(r, 0, e)$ msg. to $n_r$;
   end if
end for
for nodes $v$ receive $DetectEvt(r, 0, e)$ msg. from $v'$ do
   if $(h < max_{hop})$ and $(v.event$ and $!v.reported)$ then
      forward $DetectEvt(v, h + 1, e)$ msg. to $n_v$;
   else
      reply $ReplyEvt$(meets boundary) msg. to $v'$;
   end if
end for
for nodes $v$ receive $ReplyEvtmsg.$ do
   if msg.==meets boundary then
      reply $ReplyEvt(x_v, y_v, data_v, e)$ msg. to parent;
   else
      concat own data and reply $ReplyEvt$ msg. to parent;
   end if
end for
```

to nodes less than max_{hop} from r. Only those nodes which have detected an event and not yet reported forward the message to the next hop. We represent the nodes belonging to the cluster formed by node r as S_r. Multiple r can exist for the same event. They are usually the nodes that detected the event earlier than the others, which start constructing their own clusters. These clusters divide the event area into pieces of maps, as shown in Figure 9.2. Each map will be reported by one sensor r to one actuator. B_r is the boundary nodes on the map of S_r, where $B_r \subseteq S_r$. Nodes in B_r are either max_{hop} hops from node r, or located on the boundary between two maps. Nodes in B_r stop forwarding the $DetectEvt$ message and reply to the previous node with their coordinates, data value, and the event ID. The event ID may include the type of the event and the event discovery time, which is determined by r.

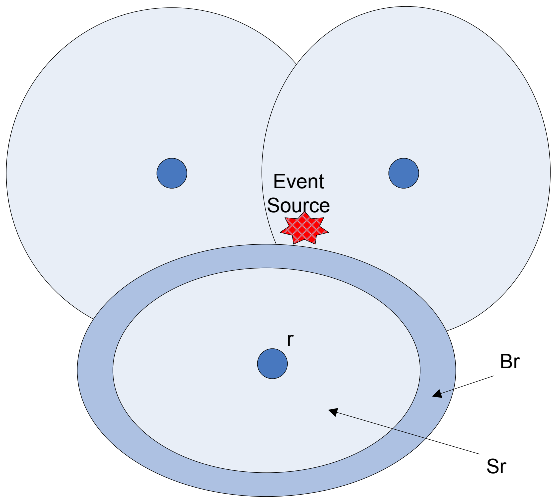


Figure 3.2: Pieces of maps formed in an event.

3.3.2 Data Aggregation

Each piece of map can be represented by a tree structure with the sensor r as the root (Figure 9.3). When a node receives the replies from its descendent nodes, it concatenates its own reply and forwards them to the previous hop. Nodes with even number of depth h concatenate the reply with its own coordinates and sensed data, while nodes with odd number of depth h aggregate the data from their immediate descendants before forwarding them. Nodes with odd number of depth calculate the mean from the data values sensed by themselves and their descendants with the depth $h+1$ (Algorithm 15). The following equation shows how $mean_{v_h}$ can be calculated by node v_h.

Let h be no. of hops from sensor r,

v_h be the node in depth h,

$data_{v_h}$ be the data collected by node v_h,

$data_{v_{hj}}$ be the data collected by the j^{th} descendent of v_h.

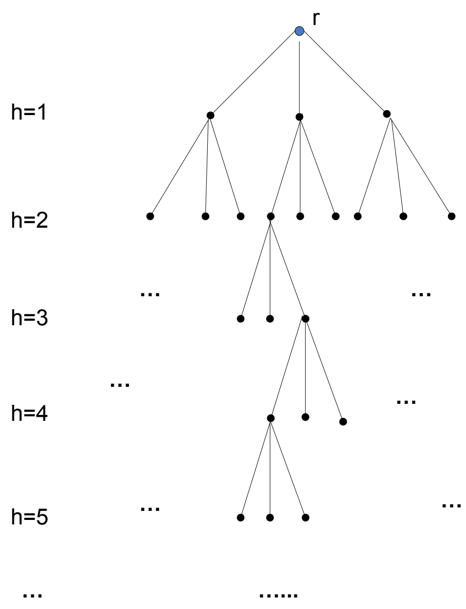


Figure 3.3: Tree representation of nodes on the map S_r.

$$mean_{v_h} = (\sum_{j=1}^{c_{v_h}} data_{v_{hj}} + data_{v_h})/(c_{v_h} + 1), \tag{3.1}$$

where c_{v_h} is the no. of immediate descendants of v_h.

Finally, the root sensor r collects all the coordinates and sensed data from the nodes in S_r. It is responsible for reporting this event to its closest actuator. The actuator, which receives this event report, is not necessary the one to perform the actions. At the same time, other actuators may receive information for the same event from other pieces of maps as well. All the informed actuators then coordinate and decide the reaction together. To speed up the coordination, sensor r divides the data on its map into different layers according to its importance. For example, the event type, location, and time are the basic information for starting actuator coordination, so they should be transmitted first.

Algorithm 2 Data Aggregation

```
for nodes $v_h \in S_r$ receive $ReplyEvt$ msg. do
  if $h == odd$ then
    gather all data from its descendants $v_{h_j}$ in $h+1$;
    $mean_{v_h} = (\sum_{j=1}^{c_{v_h}} data_{v_{hj}} + data_{v_h})/(c_{v_h}+1)$;
    remove $data_{v_{h_j}}$ from $ReplyEvt$ msg.;
    concat $mean_{v_h}, x_{v_h}, y_{v_h}, e$ to $ReplyEvt$ msg.;
    forward $ReplyEvt$ msg. to parent in depth $h-1$ ;
  else
    concat $x_{v_h}, y_{v_h}, data_{v_h}, e$ to $ReplyEvt$ msg.;
    forward $ReplyEvt$ msg. to parent in depth $h-1$;
  end if
end for
```

3.3.3 Layered Data Transmission

In our work, the data are divided into the base layer and the refinement layer. The base layer contains the type of event, the time when the event is first detected, the location of the map, and the mean value of the collected data. The mean gives the actuator a general idea on the condition of the map. It can be calculated by r with the following equation.

$$MEAN_{S_r} = (\sum_{i=1}^{N_{h=odd}} mean_i * (c_i+1)) / \sum_{i=1}^{N_{h=odd}} (c_i+1), \tag{3.2}$$

where i refer to all nodes with odd no. of depth in the S_r.

As mentioned before, virtual grids are imposed on the network area, so the location of map can be simplified as follows:

Let $a \times b$ be the size of the virtual grid, locations (x_i, y_i) of each node i in B_r can be represented by (x'_i, y'_i) in a grid coordinate, where $(x'_i, y'_i) = (x_i/a, y_i/b)$. After collecting all (x', y') of B_r and removing the redundant coordinates, a set of grid coordinates representing the location of the map S_r is obtained.

The refinement layer contains all the means calculated by nodes with odd number of depth and their corresponding locations. These values are transmitted in a special order based on its distance from the centre C on the map S_r. $< mean_{d_i}, x_{d_i}, y_{d_i} >$ represents the mean and coordinates of a node i with distance d_i from C, where $\|l_i - l_C\| = d_i$.

The sensor r first forwards the data in the base layer, and then the refinement layer, as shown in Figure 4. It sends refinement layer with the following sequence, given $d_{max} = Max\|l_j - l_C\|$ for $\forall j \in S_r$:

$< mean_0, x_0, x_0 >$: data from the node located at C

$< mean_{d_{max}}, x_{d_{max}}, y_{d_{max}} >$: data from the node with distance d_{max} at C

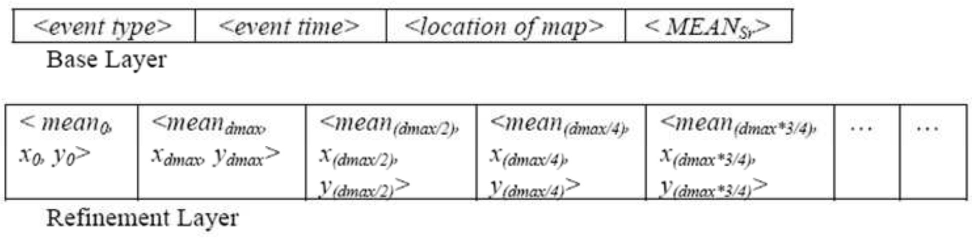
$< mean_{d_{max/2}}, x_{d_{max/2}}, y_{d_{max/2}} >$: data from the node with distance $d_{max/2}$ at C

$< mean_{d_{max/4}}, x_{d_{max/4}}, y_{d_{max/4}} >$: ...

$< mean_{d_{max*3/4}}, x_{d_{max*3/4}}, y_{d_{max*3/4}} >$: ...

......

By the above sequence, the actuators achieve the event reporting gradually with more important information first. This allows them to start the actuator coordination much faster. They can then determine how many and which actuator(s) should perform the action as soon as possible, while finer information arriving later can help the planning of detailed action strategy. We adopt a location-based routing protocol with the consideration of transmission delay [4] for event reporting. More research work can be done in the future for finding



<event type>	<event time>	<location of map>	$< MEAN_{S_r}>$

Base Layer

$< mean_0, x_0, y_0>$	$<mean_{dmax}, x_{dmax}, y_{dmax}>$	$<mean_{(dmax/2)}, x_{(dmax/2)}, y_{(dmax/2)}>$	$<mean_{(dmax/4)}, x_{(dmax/4)}, y_{(dmax/4)}>$	$<mean_{(dmax*3/4)}, x_{(dmax*3/4)}, y_{(dmax*3/4)}>$	...	...

Refinement Layer

Figure 3.4: Base layer and refinement layer.

the optimal actuator and path with minimum delay.

3.4 Actuator Coordination and Reaction

We present the algorithms for the combination of maps on the event area, actuator coordination, and location update for the actuators in this Section.

3.4.1 Combination of Maps

After an actuator receives the data in the base layer from the sensor r, it gets one piece of map in the event area. It then combines multiple maps if it receives more than one report on same type of event happening in the same area within time period t_e. Moreover, it will start communicating with other actuators located closely to the event area as well. They exchange information for combining their maps and approximating the size of the event as shown in Figure 5 and Algorithm 3. The coordination among the actuators starts before the arrival of the data in the refinement layer; therefore, it brings a quicker response from actuators.

The actuators get a general idea on the event location by finding the x_{max}, x_{min}, y_{max}, and y_{min} of the maps. They can also re-

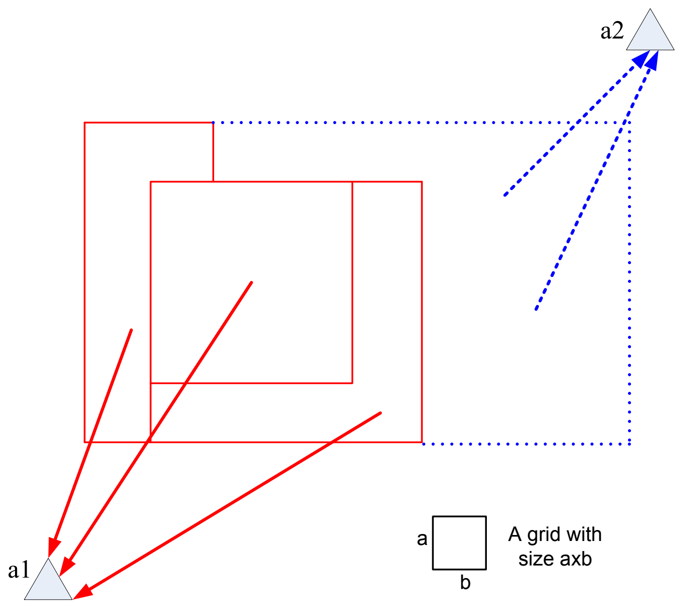


Figure 3.5: Combination of maps.

organize the event area by dividing the combined map into different rectangles. They can represent the event area by $< x_l, y_l, x_r, y_r >$, where x_l, y_l, x_r, y_r represent the $x-$ and $y-$ coordinates of the grids in lower-left and upper-right corners of a piece of rectangular map.

Next, the actuators involved can determine how many and which of them will perform the appropriate actions. For example, actuators located closer to the event should have higher priority to react. We assume the actuators possess the same speed for moving and reacting to the same event for simple analysis. Since a larger event should be assigned with more actuators to response, we estimate the number of actuators N as A/s, where s is the approximate area size to be handled by one actuator. Let $Area$ be the size of the event area, m_i be the time actuator i takes to arrive at the attack area, and w be the rate of performing appropriate actions by actuators, then the total time T required for accomplishing the appropriate actions

Algorithm 3 Data Aggregation

```
for each actuator $a$ on event $e$ do
  if received multiple $S_r$ then
    Gather the $B_r$ in grid coordinates from all $S_r$;
    Remove the redundant $B_r$;
    Remove the connected $B_r$;
    Store the remaining $B_r$ in $B_a$;
  end if
  Exchange the $B_a$ with other actuators;
  Remove the redundant $B_a$;
  Remove the connected $B_a$;
  Estimate the $B_a$ by finding lower-left and upper-right grids $< x_{min}, y_{min} >$ and $< x_{max}, y_{max} >$;
end for
```

is calculated by:

$$w * \sum_{i=1}^{N} (T - m_a) = Area, \tag{3.3}$$

where $N = Area/s$.

During the coordination, the actuators can obtain more details of the event when the data in the refinement layer arrives. The refinement layer contains additional information for analysis, such as where the event source locates and the seriousness of the event. This may help the actuator plan an appropriate action sequence or the proper strategy in responding to the event.

3.4.2 Location Update

Since the actuators will move when they carry out appropriate actions, their locations should be updated for the sensors in the corresponding areas. An actuator will broadcast messages about its departure and arrival to the surrounding sensors. The sensors will then determine their potential actuator again. The process is shown in Figure 9.7.

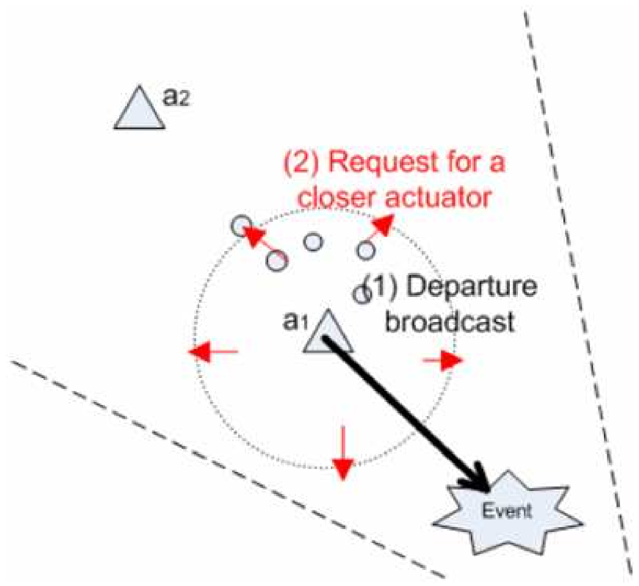


Figure 3.6: Leave of actuator a_1.

We assume that the sensors always report the event to their closest actuators. Let $A = \{a_1, a_2, \ldots, a_n\}$ be the set of actuators in the network with their various location vectors, i.e. $l_i \neq l_j, \forall i \neq j$. The region $R(a_i)$ is called the Voronoi cell associated with a_i, where

$$R(a_i) = \{l|(\|l - l_i\| \leq \|l - l_j\|), \forall i \neq j\} \tag{3.4}$$

Nodes in $R_(a_1)$ should be informed for the departure of a_1, so they will look for another actuator. Transmission range R_t is required for broadcasting the departure message to all nodes being affected.

Let N_{a1} be set of neighboring actuators of a_1,

$$R_t = max\{(l_{a_i} - l_{a+1})/2\}, \forall a_i \in N_{a1}. \tag{3.5}$$

Nodes which are located in $R(a_1)$ and received a_1's departure message will look for another actuator nearby. They will broadcast messages to neighbors for requesting the locations of the nearby actuators. Nodes located on the boundary of $R(a_1)$ will reply with the locations of their closest actuators. This information will be propagated hop-by-hop to the nodes in $R(a_1)$. Each of them can then choose the closest actuator as its new actuator for event reporting. Similarly, the actuator will broadcast its arrival and its new location to the surrounding sensors when it arrives at another place after event reaction. The affected sensors will then update their records of actuators.

3.5 Summary

In this thesis, we present a real-time communication framework for wireless sensor-actuator networks. It provides an efficient event-reporting algorithm, which reduces the network traffic and minimizes the transmission delay by dividing the event area into smaller pieces of maps. The data are aggregated and further divided into different layers according to their importance. It is then transmitted to the closest actuator in the order of significance. This approach enables the actuators to start coordination without waiting for the arrival of the complete event information. Multiple actuators can combine their pieces of maps and decide on the appropriate actuator(s) to perform the actions as soon as possible. The assigned actuators will broadcast their move to the surrounding nodes, so the affected sensors can update the actuator information dynamically for future reporting. We also consider the heterogeneous characteristics and functionalities of sensors and actuators, and offer a distributed, self-organized, and comprehensive solution for real-time communications in WSAN. Our future work will focus on formalizing the current approach, providing performance analysis, and evaluating the solution by experiments. Moreover, we are interested in enhancing the efficiency and reliability of the current approach.

□ **End of chapter.**

Chapter 4

Delay-Aware Reliable Event Reporting

4.1 Overview

WSANs greatly enhance the existing wireless sensor network architecture by introducing powerful and possibly even mobile actuators. The actuators work with the sensor nodes, but can perform much richer application-specific actions. To act responsively and accurately, an efficient and reliable reporting scheme is crucial for the sensors to inform the actuators about the environmental events.

In this chapter, we consider the transmission delay from the sensors to the actuators. We propose a general reliability-centric framework for event reporting in WSANs. We point out that the reliability in such a real-time system depends not only on the accuracy, but also the importance and freshness of the reported data. Our design follows this argument and seamlessly integrates three key modules that process the event data, namely, an efficient and fault-tolerant event data aggregation algorithm, a delay-aware data transmission protocol, and an adaptive actuator allocation algorithm for unevenly distributed events. Our transmission protocol adopts smart priority scheduling that differentiates event data of non-uniform importance. We evaluate our framework through extensive simulations; the results demonstrate that it achieves desirable reliability with minimized delay.

4.2 Network Model and Objective

In this section, we present a WSAN model and list our design objectives of the reliable event reporting framework.

4.2.1 Network Model

We consider a wireless sensor-actuator network (WSAN) that consists of a collection of sensor nodes s and actuator nodes a. The field covered by this network is divided into virtual blocks for event monitoring, as illustrated in Figure 9.1. We assume that the sensors and actuators are aware of their locations, and hence, the associated grids. The location information can be obtained either through GPS [87] or various localization techniques [57, 109, 51, 48].

Each sensor is responsible for collecting event data in its associated block. Since malfunctioned sensors may give inconsistent readings, the data in the same block will be aggregated to form a consistent mean value before reporting. A subset of the sensors in the field, referred to as *reporting nodes*, v, are responsible for forwarding the aggregated event data to the actuators for further actions. As we will show later, the aggregation occurs in a distributed manner, along with the data flow toward the reporting node v. Also note that the communications from the sensors to the actuators follow an anycast paradigm, that is, event reporting is successful if any of the actuators receives the report.

We focus on the reliable event data transmission from the sensors to the actuators. The corresponding actions that the actuators should perform are out of the scope of this chapter, and are really application specific. It is however worth noting that, for most of such applications, strict reliability as in TCP is often not necessary and even impossible given the errors/distortions arising in aggregation and transmission; on the other hand, timely delivery is much more important, as it not only enables shorter response times for the actuators, but also implies more accurate decisions since the data are fresher.

We thus propose a reliability index, which measures the probability that the event data are aggregated and received accurately within pre-defined latency bounds. Since the events may have different importance, depending on their types, urgency, and seriousness, our index and reporting framework also accommodates such differences. To realize this, each sensor in our framework maintains a priority queue, and, during transmission, important event data are scheduled with higher priorities. Beyond this differentiation in individual nodes, the queue utilization also serves as a criterion for next-hop selection in routing toward actuators.

4.2.2 Design Objective

We now give a formal description of the system parameters, and our objective is to maximize the overall reliability index, $\mathbb{R}$, across all the events, as follows:

System Parameters

e : Event

q_e : Data report of event e

Q_e : Set of data reports of event e that satisfy the end-to-end latency constraint

$Imp(e)$: Importance of event e

B_e: Latency bound for sensor-actuator reporting of event e

D_{q_e}: End-to-end delay of data report q_e

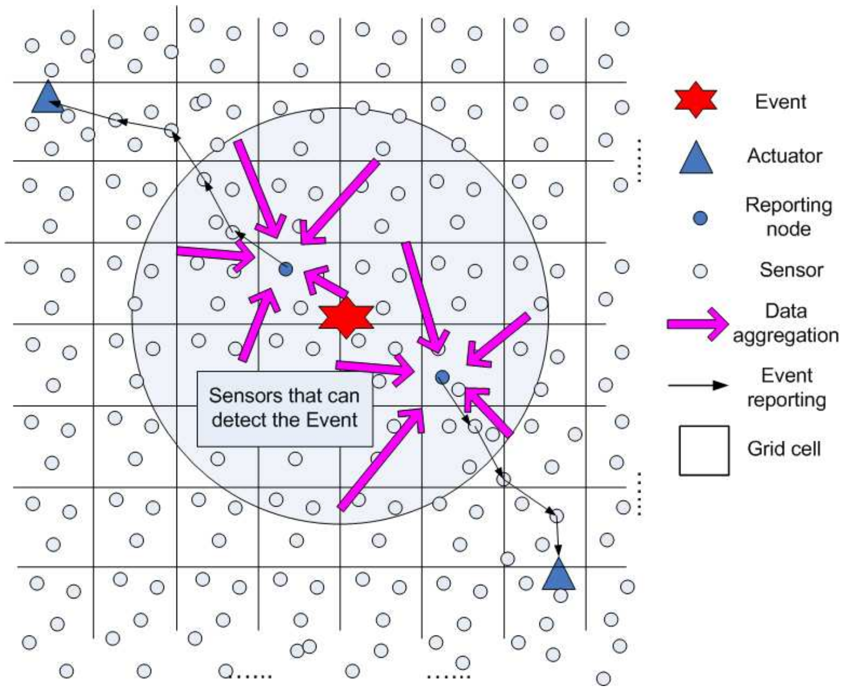


Figure 4.1: An illustration of the WSAN model and event reporting from sensors to actuators.

N_e: Number of data reports for event e
f: Probability of failures in data aggregation

Objective
Maximize

$$\mathbb{R} = \sum_{\forall e} \left(\frac{Imp(e)}{\sum_{\forall e} Imp(e)} * r_e \right), \tag{4.1}$$

where $r_e = \frac{|Q_e|(1-f)}{N_e}$.

Subject to

$$D_{q_e} \leq B_e \tag{4.2}$$

Clearly, the overall reliability of the system, $\mathbb{R}$, depends on the importance of the events and their respective reliability, r_e. The latter further depends on the reports reaching an actuator within the delay bound and without failure in aggregation. Aggregation failure happens only if malfunctioned sensors dominate a block.

4.3 The Reliable Event Reporting Framework

Our framework addresses the whole process for event reporting, and integrates three generic modules to achieve the above reliability objective. Specifically, when an event (*e.g.*, a fire) occurs, the sensors located close to the event will detect it. After aggregation, which removes redundancy and inconsistent readings, the reporting nodes will forward the reports to the actuators. Such forwarding is delay- and importance-aware, implemented through prioritized scheduling and routing in each sensor. It is further enhanced to cope with transmission failures by an adaptive replication algorithm. We also provide an actuator allocation module that determines the locations of the actuators. It ensures a balanced and delay-minimized allocation of actuators to process the unevenly distributed events in the network.

Figure 9.2 illustrates the workflow of our framework. We now offer detailed descriptions of the three modules.

4.3.1 Grid-Based Data Aggregation

In a densely deployed sensor network, multiple sensors may sense the same event with similar readings. Hence, it is preferable to aggregate the data before reporting to the actuators. Our grid-based aggregation algorithm works as follows (see Figure 9.3):

For each block, there is an aggregating node that first collects the event data, $<x_1, x_2, ..., x_n>$, and finds their median med. It will

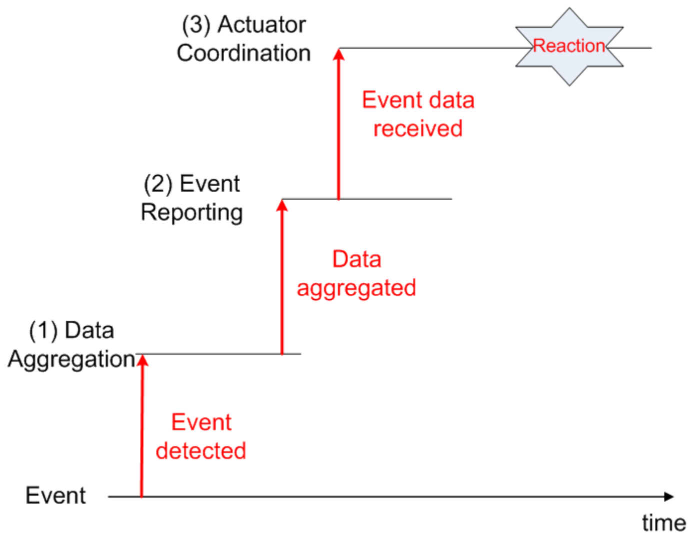


Figure 4.2: Workflow of the framework.

compare each data x_i with med and filter out those with a significant deviation (*e.g.*, greater than a predefined threshold Δd). These data could be from malfunctioned sensors, which will then be blacklisted. Then, the aggregating node will calculate the mean value $\overline{x_g}$ from the remaining data in block g (Algorithm 1). We consider the aggregated data to be reliable if more than half of the sensors in the block are normal. The reliability for the aggregated data from block g thus can be evaluated as

$$1 - f_g = 1 - \sum_{i=\lceil N_x/2 \rceil}^{N_x} \binom{N_x}{i} (f_s)^i (1 - f_s)^{N_x - i},$$

where f_g is the failure probability of block g on data aggregation, N_x is the number of nodes in block g, and f_s is the fraction of sensors that are malfunctioned.

The aggregating node may serve as the reporting node to forward the aggregated data to actuators. The aggregation, however, can be easily extended to multiple levels, where a reporting node

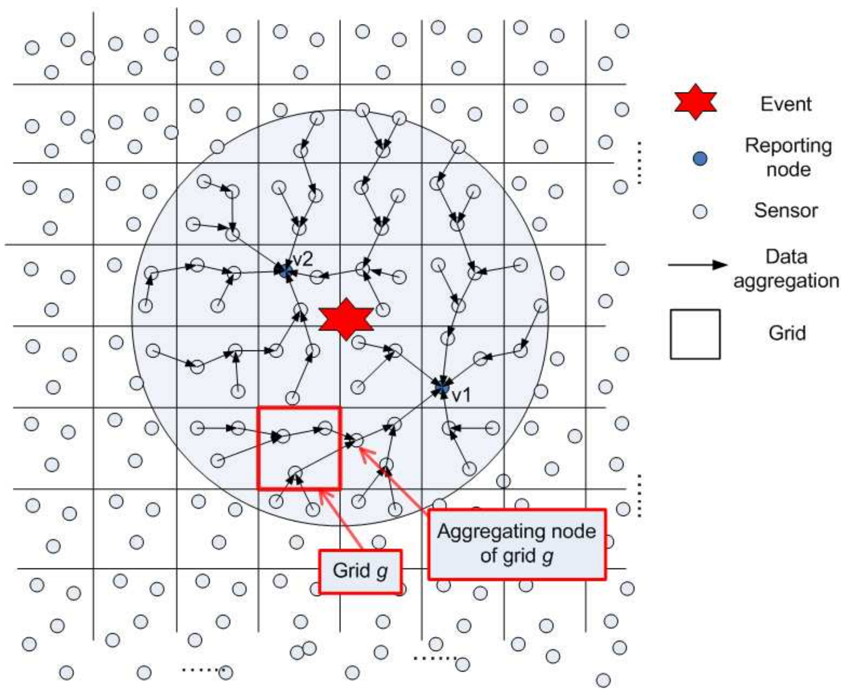


Figure 4.3: Grid-based data aggregation.

is responsible for further collecting and aggregating the data from the aggregating nodes in surrounding blocks, as shown in v (Figure 9.3). For the 2-level case, each sensor independently decides whether it will serve as a reporting node according to probability p_v. Here, $p_v = \frac{1}{N_g * N_x}$, where N_g is the number of data reports to be transmitted by a reporting node. Notice that each block has only one summarized mean data value, so N_g is also equal to the number of blocks to be reported by one reporting node. Other bidding algorithms for reporting nodes selection could be used as well in our framework, *e.g.*, those in [76].

4.3.2 Priority-Based Event Reporting

The routing and transmission protocol for event reporting from the reporting nodes to the actuators is the core module in our framework.

Algorithm 4 Data Aggregation

```
Define: $\overline{x_g}$ as aggregated data mean of block $g$;
for each sensor $s$ receive data $x_i$ do
  if multiple $x_i \in g$ and $s$ is the aggregating node then
    find the median $med$ among data $<x_1, x_2, ..., x_n>$;
    for each data $x_i \in g$ do
      if $x_i$ - $med > \Delta d$ then
        blacklist node $i$;
      end if
    end for
    $\overline{x_g}$ = mean of the un-blacklisted data $x_i \in g$;
  end if
end for
```

The key design objective here is to maximize the number of reports reaching the destination within their latency bound, and, for different event types, to give preference to important events. To this end, we adopt a priority queue in each sensor, which plays two important roles: 1) prioritized scheduling to speed up important event data transmission; and 2) queue utilization as an index for route selection to meet the latency bounds.

In our preemptive priority queue, the packets for the event data are placed according to their data importance, and each priority is served in a first-in-first-out (FIFO) discipline. Since a light-weighted sensor network with few event occurrences seldom suffers from excessive transmission delays, we focus on a network with frequent event occurrences. In such a network, the queuing delay can be the dominating factor over the processing and propagation delays.

The queueing delay of the highest priority queue is $\overline{d_{q_1}} = \overline{R} + \overline{SN_{q_1}}$, where $\overline{R} = \frac{1}{2}\sum_{k=1}^{K} \lambda_k \overline{S^2}$ is the mean residual service time in the node, $\overline{N_{q_1}}$ is the mean number of packets in the first queue, K is the number of priority queues, λ_k is the arrival rate of the packets in priority queue k, and $\overline{S}$ and $\overline{S^2}$ are the expectation and second moment of the service time of the sensor. We assume the packet arrival is Poisson. $\overline{S}$ can be obtained in each individual sensor by observing the time it takes to serve a packet.

By Little's theorem, $\overline{N_{q_1}} = \lambda_1 \overline{d_{q_1}}$, and the load of priority k is $\rho_k = \lambda_k \overline{S}$, hence the waiting time of a packet in the first priority queue is:

$$\overline{d_{q_1}} = \frac{\overline{R}}{1-\rho_1}.$$

Similarly, the waiting time of a packet in the second priority queue is:

$$\overline{d_{q_2}} = \overline{R} + \overline{SN_{q_1}} + \overline{SN_{q_2}} + \overline{S}\lambda_1 \overline{d_{q_2}} = \frac{\overline{R} + \rho_1 \overline{d_{q_1}}}{1-\rho_1-\rho_2}.$$

The mean waiting time $\overline{d_{q_k}}$ of a packet in the k^{th} priority queue is:

$$\overline{d_{q_k}} = \frac{\overline{R}}{(1-\rho_1-...-\rho_{k-1})(1-\rho_1-...-\rho_k)}.$$

Sensors periodically exchange control information with neighboring nodes through beacon messages or piggyback messages. A control message contains such information as waiting time and rate to the actuators. When routing the event data packets, a sensor should not select a next hop that is busy in forwarding important data. On the contrary, it selects a next hop that has a smaller queueing time for the corresponding priority, or it may select a next hop that it can preempt the data packets with lower importance.

More formally, consider node i that receives a new event data $data_e$. Given the control message it has received from neighbor j, node i can obtain $<a, \overline{S}, \lambda_{high}, \lambda_{low}>$, where a is the target actuator, $\overline{S}$ is the expected service time of node j, $\lambda_{high} = \sum_{\forall k, imp(data_k) \geq imp(data_e)} \lambda_k$ is the sum of all λ_k of the data that are equal or more important than $data_e$, and $\lambda_{low} = \sum_{\forall k, imp(data_k) < imp(data_e)} \lambda_k$ is the sum of all λ_k of the data that are less important than $data_e$.

Node i needs to ensure that the end-to-end latency for $data_e$ is no more than the latency bound B_e. To this end, it first estimates the advancement $h_{i,j}$ towards the actuator a from i to j, and then the maximum hop-to-hop delay from i to j, $delay_{i,j}$:

$$h_{i,j} = \frac{\parallel a, i \parallel - \parallel a, j \parallel}{\parallel a, i \parallel}.$$

So,

$$delay_{i,j} \leq B_e * h_{i,j}.$$

Since $delay_{i,j} = d_q + d_{tran} + d_{prop} + d_{proc}$, the maximum queueing delay $d_{q_{max}}$ is:

$$d_{q_{max}} = B_e * h_{i,j} - (d_{tran} + d_{prop} + d_{proc}).$$

Only neighbors with $d_{q_{max}} > 0$ will be considered as the next hop; otherwise the latency bound cannot be met. Among these candidates, node i starts inspecting the neighbors with both $\lambda_{low} = 0$ and $\lambda_{high} = 0$, followed by the remaining neighbors. Here, $\lambda_{low} = 0$ implies that it is not forwarding any event data with importance lower than that considering by node i; if node i forwards the data to this node, it will not affect the transmission time for the existing packets in that node; Similarly, $\lambda_{high} = 0$ means that it is not transmitting any data with higher importance, so the data from node i, if forwarded, can be served with the highest priority. For each of the candidates mentioned above, node i calculates the maximum data rate λ_i that it can forward while satisfying the latency bound:

$$d_{q_{max}} > \frac{\overline{R}}{(1 - \lambda_{high}\overline{S})(1 - \lambda_{high}\overline{S} - \rho_i)},$$

and

$$\rho_{i,j} < 1 - \lambda_{high}\overline{S} - \frac{\overline{R}}{(1 - \lambda_{high}\overline{S})d_{q_{max}}},$$

where $\rho_{i,j} = \lambda_{i,j}\overline{S}$ is the maximum affordable load of j for handling data from i on event e.

Then the event data packets are forwarded to the neighbor with the highest $h_{i,j}$ and satisfactory $\lambda_{i,j}$, which is the closest to the destination with enough capacity for transmission. The affordable packet arrival rate $\lambda_{i,j}$ is satisfactory when it is greater than the data rate received by i. Each intermediate node updates the latency bound B_e before forwarding the packet to the next hop, according to this equation:

$$B_e = B_e - (t_{depart} - t_{arrive}) - d_{tran} - d_{prop},$$

where $(t_{depart} - t_{arrive})$ is the elapse time of the packet in a node, d_{tran} can be computed using the transmission rate and the length of the frame containing the packets, and d_{prop} is the propagation time, which is in the order of several microseconds in wireless transmission.

After the transmission starts, the sensor will update its $\overline{S}$ and the routes regularly to make sure the transmission can be completed within the latency bound. If the latency bound is not met, the sensor has to forward the packets to another route. In the worst case, if no alternative can be found, the sensor may inform the previous node to select another route in the future.

4.3.3 Actuator Allocation

Once an actuator receives the event report, it will perform application-specific actions. Meanwhile, it will inform other actuators to suppress their potential actions in case some of them receive the same report later. Such coordination can be achieved through direct one-hop communications using another wireless channel, given that the actuators are much more powerful than the regular nodes.

In this anycast paradigm, reducing the distances from the sensors to their closest actuators clearly decreases the reporting delay. Since the reports are triggered by events, we suggest that an actuator allocation be performed according to the event occurrence frequency. Intuitively, the locations with more events should be allocated more actuators, so as to reduce the reporting distances. Such an allocation can be performed in the initial stage based on pre-estimated frequencies, or, with mobile actuators, performed periodically to accommodate event dynamics.

Algorithm 5 gives an allocation that balances the load of the actuators as well as minimizing the anycast distances. In this algorithm, first, the event frequency $freq_g$ of every block g will be summed. Then, the field A will be equally divided into two, denoted by $A1$ and $A2$, according to the frequency distribution. That is, $A1$ and $A2$ have the same event occurrence frequency and each is allocated half of the actuators. The process repeats recursively for $A1$ and $A2$, until each subfield contains only one actuator.

Figures 4.4 and 4.5 demonstrate our actuator allocation results with 6 and 10 actuators, respectively. In practice, the algorithm can be executed by one designated actuator after collecting the event frequency information. It then informs the allocation result to other actuators, which may then move to the corresponding locations.

Algorithm 5 Actuator Allocation

```
$ActuatorAllocation$(Field $A$, int $ActuatorNum$)
$TotalFreq \leftarrow \sum_{\forall g_i \in A} freq_{g_i}$;
$TmpFreq \leftarrow 0$;
$i \leftarrow 0$;
while TmpFreq < TotalFreq/2 do
    $TmpFreq \leftarrow TmpFreq + freq_{g_i}$;
    $i++$;
end while
$A1 \leftarrow \bigcup_{k=0}^{i} g_i$;
$A2 \leftarrow A - A1$;
$ActuatorAllocation(A1, ActuatorNum/2)$;
$ActuatorAllocation(A2, ActuatorNum - ActuatorNum/2)$;
end $ActuatorAllocation$
```

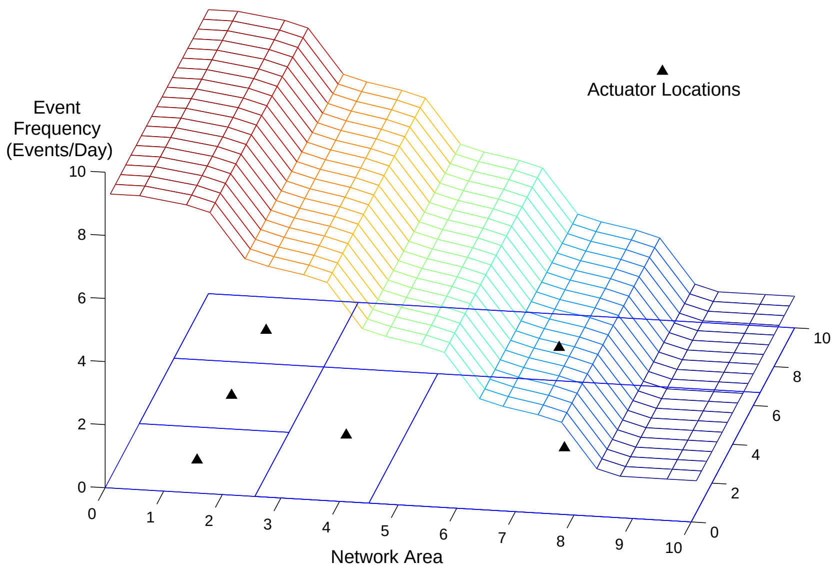


Figure 4.4: Actuator allocation with 6 actuators.

4.4 Performance Evaluation

We have conducted NS-2 [39] simulations for our proposed reliable event reporting framework. The simulation settings are mainly drawn from [52], which are summarized in Table 8.1.

4.4.1 Reliability of Event Reporting

In the first set of experiments, we evaluate the reliability of our event reporting algorithm. To this end, we generate 4 events randomly in the network and vary their data rate from 10pkt/sec to 80pkt/sec. Two of the four events are high priority events with importance 1.0 (events 2 and 4), while the two are low priority events with importance 0.3 (events 1 and 3). Each packet should be reported to the actuator within the latency bound of 2 sec.

We first assume that all the reports are routed to the same actuator. We fix the locations of the events and change the seed to generate

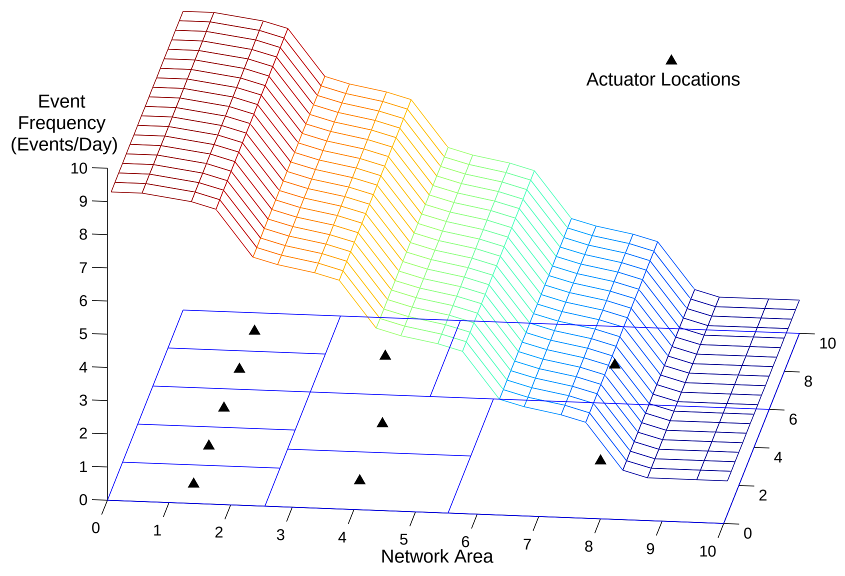


Figure 4.5: Actuator allocation with 10 actuators.

different sensor locations. Figure 4.6 shows the on-time reachability of the four events with our priority-based event reporting with event importance (PREI). For comparison, we also show the result with the geographic routing protocol (GRP) [72, 42], where greedy forwarding is employed and there is no differentiation regarding the event types. We can see that our PREI achieves much higher on-time reachability for the important events (event 2 and 4). The reachability for the low important events however is lower than that in GRP. This follows our design objective that important events will be served with higher priority and better quality routes.

Note that, even if two different events are of the same importance, their reachabilities could be different, depending on their locations. This also happens when we compare the average delay. However, our PREI generally performs better for the same event.

Figure 4.7 further shows the average delays in the PREI and GRP. It is clear that the delay in PREI is generally lower than that in GRP. This is because the PREI considers the workload of the neighbors

Table 4.1: Simulation Parameters

Network size	200m x 200m
No. of sensors	100
Node placement	Uniform
Radio range	40m
MAC layer	IEEE 802.11
Bandwidth	2Mbps
Packet size	32 bytes
No. of actuators	1-6
No. of concurrent events	3-10
B_e	2sec

when selecting the route. An interesting observation is that, in PREI, the average delays of the more important events are not necessarily lower than the less important events; e.g., the delay for Event 1 is lower than all others, even though its importance is not high. The reason is that this event is closer to the actuator than the others. We find the average per-hop delays are generally lower for important events. Also note that the actuator allocation algorithm can mitigate this delay, as will be examined later.

Finally, Figure 4.8 shows the overall reliability index, $\mathbb{R}$, of the two protocols. Again, it demonstrates that PREI outperforms GRP, and the gap increases when the data rate becomes higher.

4.4.2 Actuator Allocation

In this experiment, we show the effectiveness of our actuator allocation algorithm. To emulate the nonuniform event occurrences, we divide the whole field into three, with the event occurrence probability 0.6, 0.333, and 0.067, respectively.

Our simulator generates events according to the above probability with data rate 60pkt/s, and it allows different number of concurrent events in the network as represented in the x-axis of Figures 4.9 and

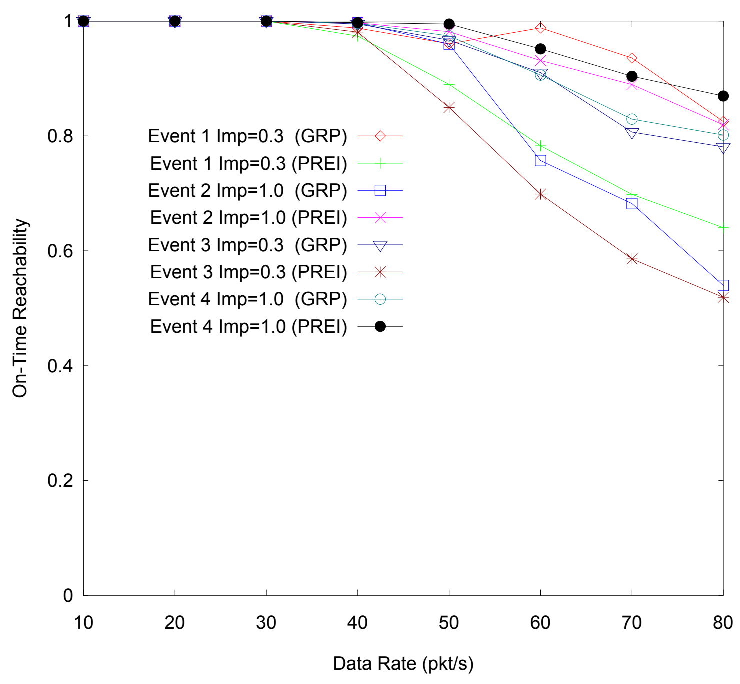


Figure 4.6: On-time reachability.

4.10.

Figure 4.9 gives the on-time reachability with different number of concurrent events. We first focus on 2 and 3 actuators only, and investigate the impact of using more actuators later. We can see from Figure 4.9 that the reliability with actuator allocation outperforms that without allocation (i.e., random distribution). While the more actuators there are, the better performance we can expect, we notice that the effect of allocation is remarkable. In fact, the performance of a 2-actuator system with allocation is very close to that of 3-actuator without allocation, and even outperforms it when there are few concurrent events.

Figure 4.10 shows the corresponding average delay. Not surprisingly, 3-actuator with allocation achieves the lowest delay. Similar to the on-time reachability, the delay for the 2-actuator with allocation is close to the case of 3-actuator without allocation. The results

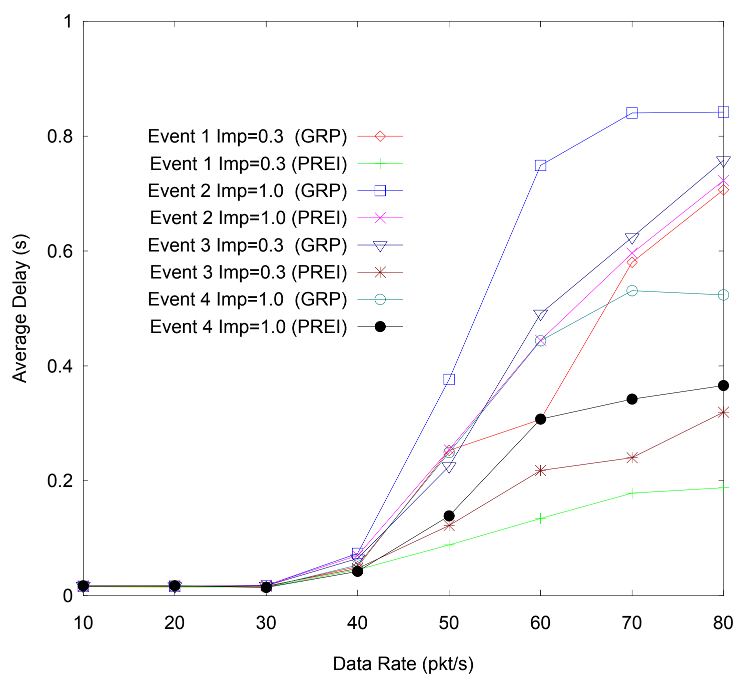


Figure 4.7: Average delay.

suggest that actuator allocation is an effective tool for improving the efficiency of event reporting.

To further investigate the impact of the number of actuators, we fix the number of concurrent events at 10 and vary the number of actuators from 1 to 6. Figure 4.11 shows the on-time-reachability as a function of the number of actuators with and without actuator allocation. Again, event reporting with actuator allocation achieves higher on-time reachability than that without actuator allocation with the same number of actuators. Intuitively, given more actuators, we can generally expect better performance, even if they are randomly deployed. This can be verified from the figure. We can see that the on-time reachability monotonically increases with more actuators, while the difference between the two schemes (with/without allocation) becomes smaller. Similar trends can also be found in Figure 4.12, which shows the average delay of event reporting as a function

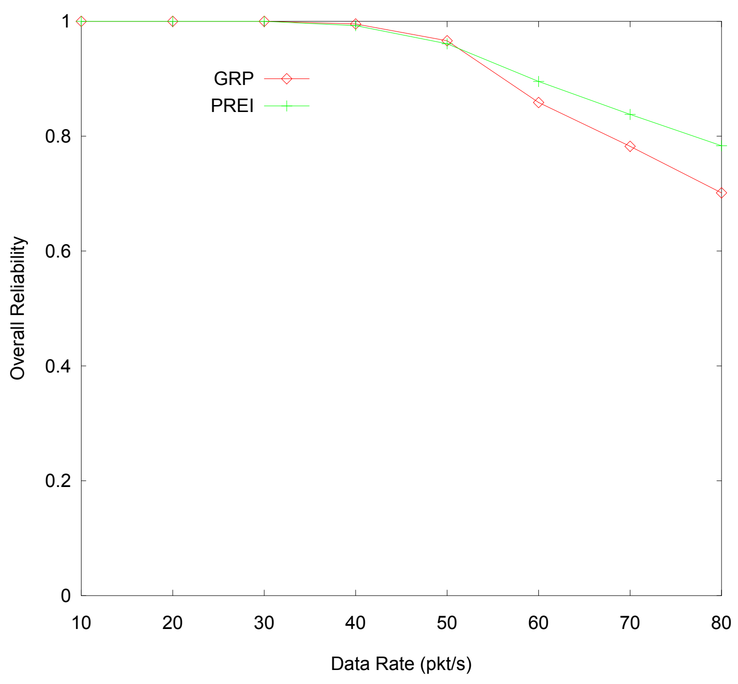


Figure 4.8: Overall reliability.

of the number of actuators.

4.5 Summary

In this chapter, we focused on reliable event reporting from sensors to actuators in a wireless sensor-actuator network (WSAN). We argued that the reliability in this context is closely related to the delay, or the freshness of the events, and they should be jointly optimized. We also suggested that the issue of non-uniform importance of the events can be explored in the optimization. Following this argument, we proposed a general delay- and importance-aware event reporting framework. Our framework seamlessly integrates three key modules to maximize the reliability index: 1) A multi-level data aggregation scheme, which is fault-tolerant with error-prone sensors; 2)

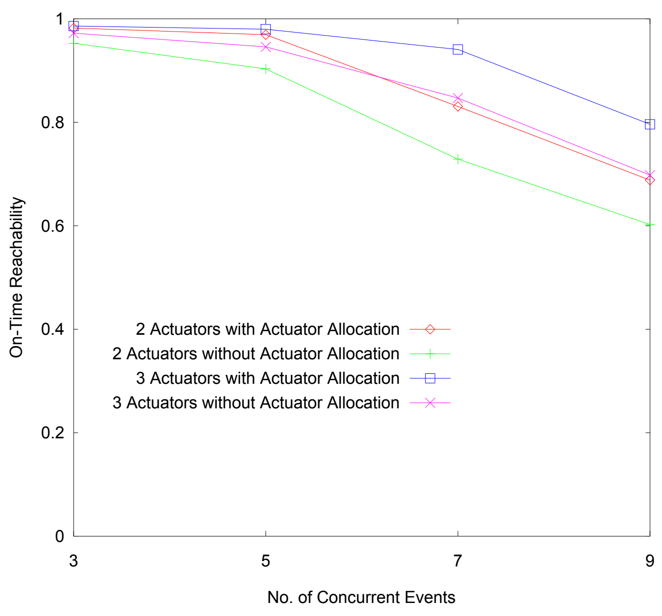


Figure 4.9: On-time reachability with actuator allocation.

A priority-based transmission protocol (PREI), which accounts for both the importance and delay requirements of the events; and 3) an actuator allocation algorithm, which smartly distributes the actuators to match the demands from the sensors.

Within this generic framework, we presented an optimized design for each of the modules, and also discussed their interactions. We also evaluated the performance of our framework through simulations. The results demonstrated that our framework makes effective use of the actuators, and can significantly enhance the reliability of event reporting.

□ **End of chapter.**

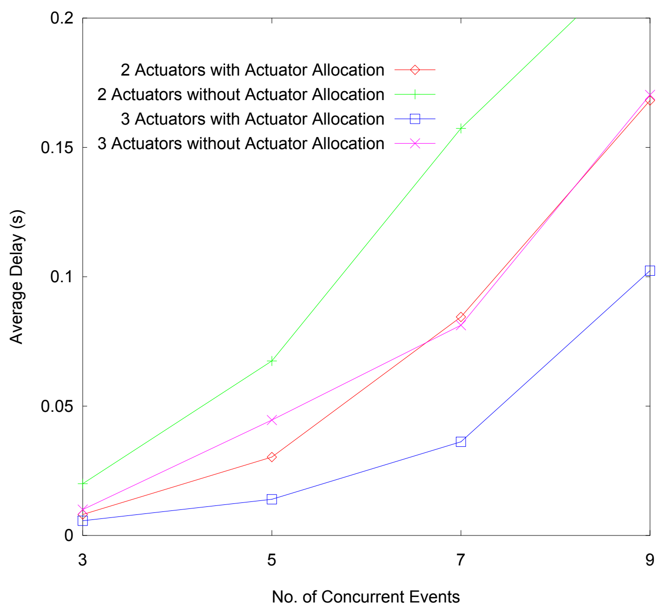


Figure 4.10: Average delay with actuator allocation.

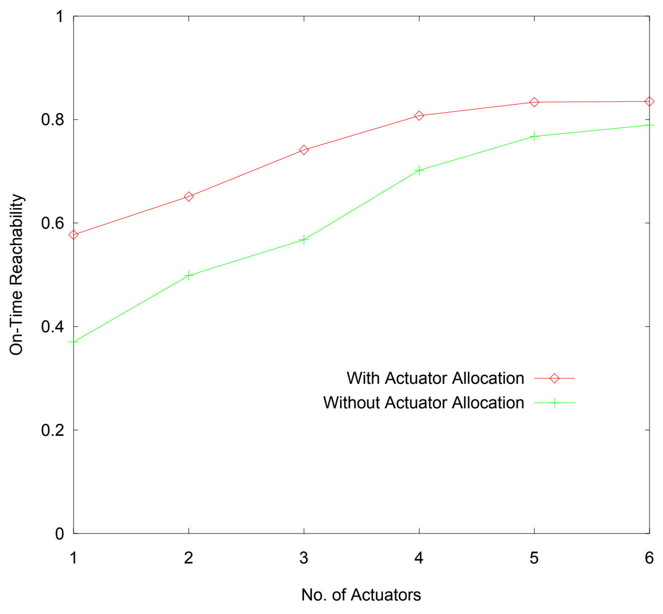


Figure 4.11: On-time reachability vs. no. of actuators.

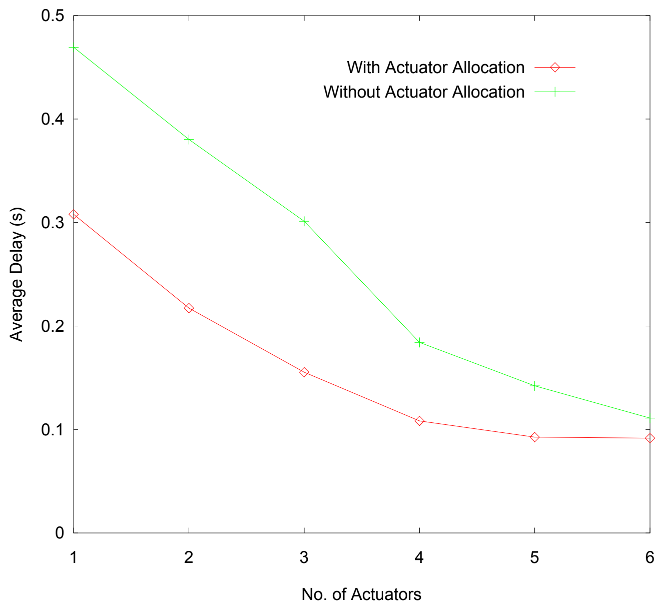


Figure 4.12: Average delay vs. no. of actuators.

Chapter 5

Latency-Oriented Fault Tolerant Transport Protocol

5.1 Overview

We considered the transmission delay on event reporting from sensors to actuators in the previous chapter. We suggest that the unreliable multi-hop communications and congestion may cause excessive delays. Then, we propose a delay-aware event reporting framework to address the problem. Apart from the transmission delay, the frequent sensor and link failures may lead to packets loss and unreliable data transmission. Thus, it is important to provide a fault tolerant protocol to ensure efficient and reliable data transmission from sensors to actuators.

In this chapter, we propose a latency-oriented fault tolerant data transport protocol in WSANs. We argue that reliable data transport in such a real-time system should resist to the transmission failures, and should also consider the importance and freshness of the reported data. We articulate this argument and provide a cross-layer two-step data transport protocol for on-time and fault tolerant data delivery from sensors to actuators. Our protocol adopts smart priority scheduling that differentiates the event data of non-uniform importance. It balances the workload of sensors by checking their queue utilization and copes with node and link failures by an adaptive replication algorithm. We evaluate our protocol through extensive simulations, and the results demonstrate that it achieves the desirable reliability for WSANs.

5.2 Network Model and Objective

In this section, we present a WSAN model and list our design objectives of the reliable data transport protocol.

5.2.1 Network Model

We consider a wireless sensor-actuator network that consists of a collection of sensor nodes s and actuator nodes a, as illustrated in Figure 9.1. We assume that the sensors and actuators are aware of their locations. This information can be obtained either through GPS [87] or various localization techniques [57, 109, 51].

All sensors are responsible for collecting event data. A subset of the sensors in the field, referred to as *reporting nodes*, v, are responsible for forwarding the aggregated event data to the actuators for further actions. The communications from the sensors to the actuators follow an anycast paradigm; that is, an data transport is successful if any of the actuators receives the report.

We focus on the reliable event data transmission from the sensors to the actuators. The corresponding actions that the actuators should perform are beyond the scope of this chapter, and is really application specific. It is worth noting that, node and link failures may cause errors or distortions in transmission and degrade the reliability. Also, timely event data delivery not only enables short response time for the actuators, but also implies more accurate decisions given

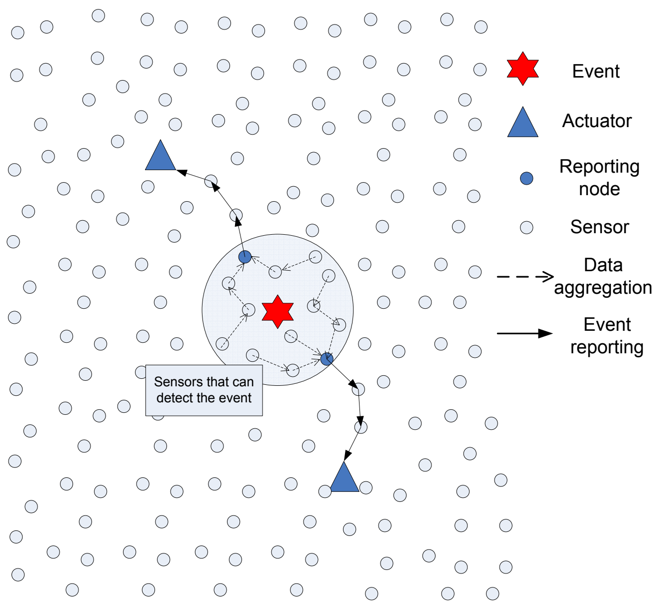


Figure 5.1: An illustration of the WSAN model and data transport from sensors to actuators.

the fresh data.

5.3 Design Objective

We consider a sensor-actuator network in which the sensors are responsible for collecting event data and a subset of them are responsible for forwarding the aggregated event data to the actuators for further actions. The communications from the sensors to the actuators follow an anycast paradigm; that is, a data transport is successful if any of the actuators receives the report. Similar to other geographic routing algorithms, the sensors and actuators are able to determine their coordinates by means of a location system like GPS or relative positioning based on signal strength estimation [54]. Sensors also exchange location information with neighbors by periodic beaconing [52, 41, 42]. In addition, they make use of the beacons for estimating the workload of the neighbors in our protocol.

We propose a reliability index, which measures the probability that the event data are transmitted to the actuator successfully within a pre-defined latency bound. Each event also has an application-specific importance level in between 0 and 1. An event with higher importance is expected to achieve higher reliability. To realize this, each sensor maintains a priority queue, and, during transmission, important event data are scheduled with higher priorities. Moreover, replication is applied adaptively depending on the event importance and the link reliability to cope with transmission failures.

We give a formal description of the system parameters as shown in Table 7.1. Our objective is to maximize the overall reliability index, $\mathbb{R}$, across all the events, as follows:

Table 5.1: System Parameters

Event	e
Data report of event e	q_e
Set of data reports of event e that reach the actuator within the latency constraint	Q_e
Importance of event	$Imp(e)$
Latency bound for sensor-actuator reporting of event e	B_e
End-to-end delay of data report q_e	D_{q_e}
Number of data reports for event e	N_e

Maximize

$$\mathbb{R} = \sum_{\forall e} (\frac{Imp(e)}{\sigma} * r_e), \tag{5.1}$$

where $r_e = \frac{|Q_e|}{N_e}$ and $\sigma = \sum_{\forall e} Imp(e)$.

Subject to

$$D_{q_e} \leq B_e. \tag{5.2}$$

Clearly, the overall reliability of the system, $\mathbb{R}$, depends on the importance of the events and their respective reliability, r_e. The latter further depends on the reports reaching an actuator successfully within the latency bound.

5.4 Latency-Oriented Fault Tolerant Data Transport Protocol

Our latency-oriented fault tolerant (LOFT) protocol addresses the problem of data transport from the event source to the actuator. It seamlessly integrates modules across different layers in achieving the above reliability objective. In this section, we first discuss the latency-oriented and importance-aware transmission through prioritized scheduling and routing for each sensor. Then, we provide a feedback algorithm to estimate the link qualities and determine the replication factor adaptively in the presence of node and link failures. It ensures a balanced, latency-oriented, and fault tolerant data transport process across different events in the network.

5.4.1 Estimating the Load of Neighbors

The key design objective here is to maximize the number of reports reaching the destination within their latency bounds, and, for different event types, to give preference to important events. Estimating the load of the neighbors allows a packet to be forwarded to a next hop with less queueing and transmission time. To this end, we adopt a priority queue in each sensor, which plays two important roles:

1) prioritized scheduling to speed up important event data transmission; and 2) queue utilization as an index for route selection to meet the latency bounds.

In our preemptive priority queue, packets for the event data are placed according to their data importance, and each priority queue is served in a first-in-first-out (FIFO) discipline. Our protocol is able to handle a network with heavy traffic, in which queueing delay can be a dominating factor over the processing and propagation delays.

Figure 9.2 shows how node i forwards packets to its neighbors j_1, j_2, and j_3. The geographical distances from j_1, j_2, and j_3 to actuator a are represented by $||j_1, a||$, $||j_2, a||$, and $||j_3, a||$, respectively. Only the neighbors, which provide satisfactory advancement from i to a, will be considered as the next hop. Furthermore, the queue utilization of the neighbors is considered in route selection. For example, the data e_1 flowing into i has the highest priority, so it will be served by the highest priority queue q_1. Among all the neighbors of i, j_3 is selected as it provides e_1 with the best service by an empty highest priority queue q_1.

The queueing delay of the highest priority queue is $\overline{d_{q_1}} = \overline{R} + \overline{SN_{q_1}}$, where $\overline{R} = \frac{1}{2}\sum_{k=1}^{K} \lambda_k \overline{S^2}$ is the mean residual service time in the node, $\overline{N_{q_1}}$ is the mean number of packets in first queue, K is the number of priority queues, λ_k is the arrival rate of the packets in priority queue k, and $\overline{S}$ and $\overline{S^2}$ are respectively the expectation and second moment of the service time of the sensor. We assume the packet arrival is a Poisson process. $\overline{S}$ can be obtained in each individual sensor by observing the time it takes to serve a packet.

More formally, consider node i that receives a new event data $data_e$. It obtains control packets from its neighbors j, $\langle a, \overline{S}, \lambda_{high}, \lambda_{low}\rangle_j$, where a is the target actuator, $\overline{S}$ is the expected service time of node j, $\lambda_{high} = \sum_{\forall k, Imp(data_k) \geq Imp(data_e)} \lambda_k$ is the sum of all arrival rates λ_k of the data that are equal to or more important than $data_e$, and $\lambda_{low} = \sum_{\forall k, Imp(data_k) < Imp(data_e)} \lambda_k$ is the sum of all λ_k of the data that are less important than $data_e$.

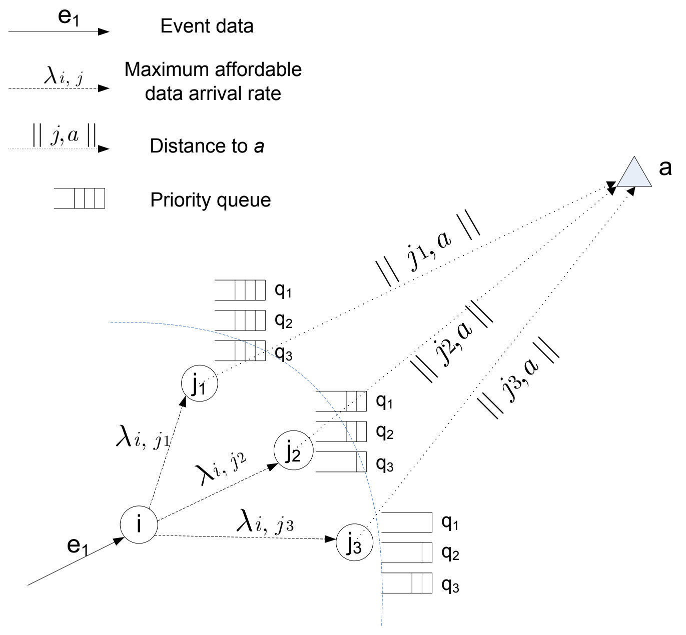


Figure 5.2: Maximum affordable arrival rate from i to j.

Node i needs to ensure that the end-to-end latency for $data_e$ is no more than the latency bound B_e. To this end, it first estimates the advancement $h_{i,j}$ towards the actuator a from i to j, and then the maximum hop-to-hop delay from i to j, $delay_{i,j}$. Note

$$h_{i,j} = \frac{\| a, i \| - \| a, j \|}{\| a, i \|}.$$

So,

$$delay_{i,j} \leq B_e * h_{i,j}.$$

Since $delay_{i,j} = d_q + d_{tran} + d_{prop} + d_{proc}$, the maximum queueing delay $d_{q_{max}}$ is:

$$d_{q_{max}} = B_e * h_{i,j} - (d_{tran} + d_{prop} + d_{proc}).$$

Only neighbors with $d_{q_{max}} > 0$ will be considered as the next hop; otherwise, the latency bound cannot be met. Among these candidates, node i starts inspecting the neighbors with both $\lambda_{low} = 0$ and $\lambda_{high} = 0$, followed by the remaining neighbors. For each candidate above, node i calculates the maximum data rate λ_i that it can forward while satisfying the latency bound:

$$d_{q_{max}} > \frac{\overline{R}}{(1 - \lambda_{high}\overline{S})(1 - \lambda_{high}\overline{S} - \rho_{i,j})},$$

and

$$\rho_{i,j} < 1 - \lambda_{high}\overline{S} - \frac{\overline{R}}{(1 - \lambda_{high}\overline{S})d_{q_{max}}},$$

where $\rho_{i,j} = \lambda_{i,j}\overline{S}$ is the maximum affordable load of j for handling data from i on event e.

Then the event data packets are forwarded to the neighbor with the highest $h_{i,j}$ and satisfactory $\lambda_{i,j}$, which is the closest to the destination with enough capacity for transmission. Each intermediate node updates the latency bound B_e before forwarding the packet to the next hop.

5.4.2 Coping with Transmission Failures

As mentioned before, packets will be dropped if they expire before reaching the actuators. Apart from that, data may be lost due to link failures, such as link transmission errors, buffer overflow, or node failures along the path. However, there exist multiple destinations (actuators) and multiple paths for anycast data transport in WSANs. Different levels of reliability can therefore be obtained based on the requirements of various event data. We adopt adaptive packet replications to handle link failures and provide reliability in terms of the success arrival of packets. In this section, we extend the above routing algorithm to cope with transmission failures in data transport.

For simplicity, we consider that the event reliability requirement R_{req} is proportional to its event importance. For example, an event with important level of 0.8 will have the reliability requirement of 0.8. We define link loss rate $L_{i,j}$ as the packet loss rate from node i to its next hop j, and path success rate P_j as the probability that a packet from node j reaches the actuator (destination) successfully. Instead of forwarding a packet to one next hop with the highest $h_{i,j}$ and satisfactory $\lambda_{i,j}$, node i forwards the packet to multiple next hops and decides the replication factor r_f of packets adaptively.

Again, consider node i and its potential next hops j_1, j_2, and j_3 in Figure 9.3. The observed link loss rate from i to j_1, j_2, and j_3 are L_{i,j_1}, L_{i,j_2}, and L_{i,j_3}. Based on these link loss rates, the corresponding path success rates P_{j_1}, P_{j_2}, and P_{j_3} from j_1, j_2, and j_3 to a are estimated. The allocation of packets from i to its neighbors is proportional to their maximum affordable arrival rates λ_{i,j_1}, λ_{i,j_2}, and λ_{i,j_3} to balance the load. After that, node i may check if the estimated path success rate can meet the event reliability requirement R_{req}. If not, it decides the replication factor r_f to meet the requirement and forwards the replicated packets to the next hops.

We now discuss the above process in detail. First, node i selects the top k neighbors with the highest $h_{i,j}$ and satisfactory $\lambda_{i,j}$, and estimates their link loss rates $L_{i,j}$. Each neighbor j periodically reports the number of packets it received from node i, so that i can calculate the loss rate $L_{i,j}$ with the number of packets it sent to j in a particular time interval. Then, it can obtain the link loss rate by an EWMA (Exponentially Weighted Moving Average) [82] approach with its previous and current estimations of the link loss rate. Then, i estimates the path success rate P_j from i to a via j as follow:

$$P_j = (1 - L_{i,j})^{1/h_{i,j}}.$$

Sensor i will allocate the packets to its neighbors according to

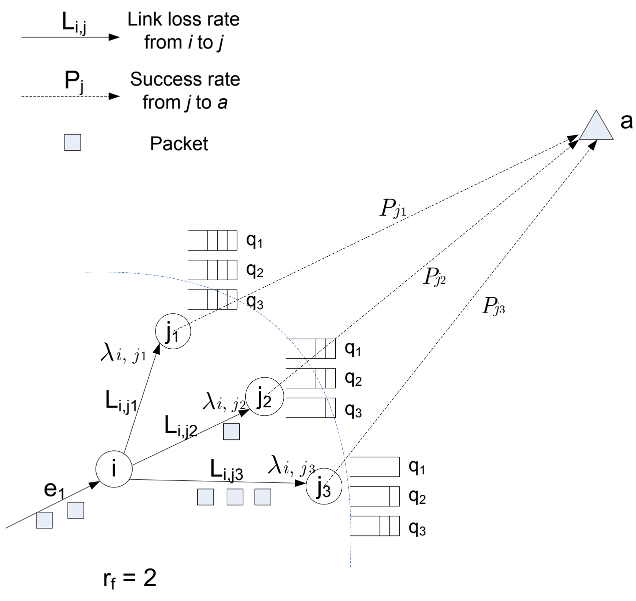


Figure 5.3: Forwarding packets with replication factor r_f=2.

their $\lambda_{i,j}$. The neighbors with higher $\lambda_{i,j}$ will be allocated with more code blocks. The proportion $prop_j$ of packets to neighbor j is:

$$prop_j = \frac{\lambda_{i,j}}{\sum_{n=1}^{k} \lambda_{i,n}}.$$

The probability that the packet can be delivered successfully from i to a by these k neighbors, P_i, can then be estimated as:

$$P_i = \sum_{j=1}^{k} \left(\frac{\lambda_{i,j}}{\sum_{n=1}^{k} \lambda_{i,n}} * P_j\right).$$

Then, node i determines the replication factor r_f with the following equation:

$$r_f = ceil(R_{req}/P_i).$$

The replication factor r_f must be greater than R_{req}/P_i, where R_{req} is initialized as the required event reliability, or the event importance in our work, by the event source. Each of the neighbors above will be allocated with proportion $prop_j$ of packets from i. The corresponding path success rate P_j will become the required reliability R_{req} of that particular path from j to the actuator.

Each node j, which received the packets, selects the next hop m' with the highest $h_{j,m}$ and satisfactory $\lambda_{j,m}$. Similarly, the path success rate obtained must be greater than R_{req}:

$$(1 - L_{j,m'})^{1/h_{j,m'}} \geq R_{req}.$$

If the link loss rate from j to m' satisfies the above equation, packet will be forwarded to m'. Since the reliability of a path is composed by a series of links on it:

$$(1 - \overline{L}_1)(1 - \overline{L}_2)(1 - \overline{L}_3)...(1 - \overline{L}_n) > R_{req},$$

and

$$(1 - \overline{L}_2)(1 - \overline{L}_3)...(1 - \overline{L}_n) > R_{req}/(1 - \overline{L}_1),$$

where the $\overline{L}_1, \overline{L}_2, ..., \overline{L}_n$ are the packet loss rates of the links on the path.

Node j updates the reliability R_{req} and forwards it with the packets to the selected neighbor m':

$$R'_{req} = R_{req}/(1 - L_{j,m'}).$$

In case that $L_{j,m'}$ does not satisfy the required reliability, node j will look for the neighbor with the next highest $h_{j,m}$ and satisfactory $\lambda_{j,m}$. The process is repeated until it goes through all the potential neighbors with high $h_{j,m}$ and $\lambda_{j,m}$. If no single neighbor can provide low enough link failure rate, j forwards packets to multiple neighbors and decides the replication factor, as shown in node i.

5.5 Evaluation

We have conducted simulations for our proposed reliable data transport protocol in NS-2 network simulator [39]. The simulation settings are mainly drawn from [52], which are summarized in Table 8.1. We consider a loss model in which packet loss occurs because of poor channel quality [65]. Every link is characterized by a failure probability, which is the probability that a packet is dropped during the transmission to the next hop.

Table 5.2: Simulation Parameters

Network size	200m x 200m
No. of sensors	100
Node placement	Uniform
Radio range	40m
MAC layer	IEEE 802.11
Bandwidth	2 Mbps
Packet size	32 bytes
Control packet rate	1 pkt/s
B_e	2 sec

We evaluate the event reliability, average delay of individual events, and overall reliability obtained in our latency-oriented fault tolerant (LOFT) data transport protocol. Two events are generated randomly in the network with the event importance 1.0 and 0.4, respectively. Each packet should be reported to the actuator within the latency bound of 2 sec. For comparison, we also show the results of the priority-based event reporting with event importance approach (PREI) [98], where priority-based forwarding is employed, but the link failures are not considered. We repeat the experiments for 10 times by changing the random seed and show the average results.

5.5.1 Protocol Performance with Varying Link Failure Probability

In this experiment, we fix the data rate at 15 pkt/s and vary the link failure probability f. It means that there is a probability f for each link to encounter transmission failure when forwarding a packet to the next hop. Figure 5.4 shows that LOFT protocol achieves much higher event reliability than PREI approach for both types of events. The reliability of the more important event (event 1) is higher than that of the less important event (event 2) in LOFT. This follows our design objective that important events should be guaranteed with higher reliability. On the contrary, the reliability of the two events are similar in the PREI, though priority-based routing is applied. It is because PREI has no mechanism to handle link failures, and hence cannot provide any differentiation on the reliability among different events.

Figure 5.5 further shows the average delay in LOFT and PREI. PREI performs a bit better than LOFT when the link failure probability is low. This is because it always selects the next hop with the lightest workload, while LOFT also estimates and considers the link packet loss rates when selecting the route. However, it is clear that the delay in LOFT is lower than that in PREI when the link failure probability increases. The reason is that replication is applied in LOFT, so packets are routed through multiple paths. Intuitively, it achieves a lower data delivery delay.

The overall reliability index, $\mathbb{R}$, of the two protocols is shown in Figure 5.6. It demonstrates that LOFT outperforms PREI, and the gap increases when the link failure probability becomes higher.

5.5.2 Protocol Performance with Varying Data Rate

We further study the effect of data rates to the performance of our protocol. We fix the link failure probability as 0.05 and vary the data

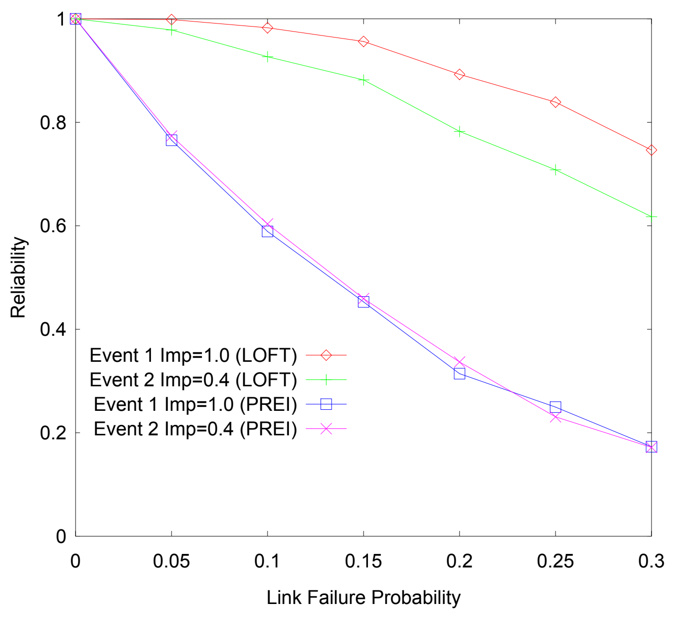


Figure 5.4: Event reliability with data rate 15pkt/s.

rates. Figure 5.7 shows that our LOFT protocol can achieve nearly perfect reliability, while PREI can only achieve reliability close to 0.8. It also indicates that the reliability achieved is independent of the data rates. Similarly, Figure 5.8 shows that LOFT achieves small and comparable average delay with PREI. Note that, the average delay of the less important event (event 2) in LOFT increases with the data rates. It is because the traffic load of the network increases with replication under a high data rate. The queuing and transmission times may then become non-negligible for the low-priority packets. Figure 5.9 again shows that the overall reliability of LOFT is higher than that of PREI.

We now increase the link failure probability to 0.3 and repeat the experiment. Figure 5.10 shows that our LOFT can still achieve satisfactory reliability even when the link failure probability is high. It is clear that the reliability in LOFT is much higher than that in PREI.

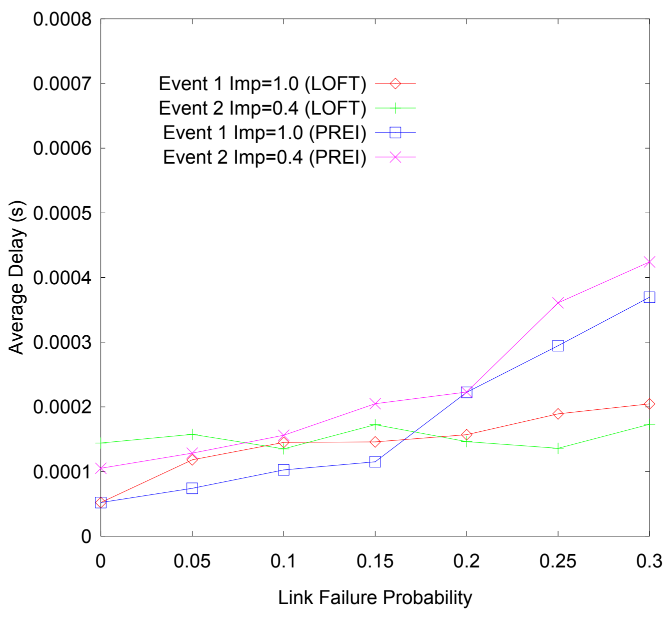


Figure 5.5: Average delay with data rate 15pkt/s.

Likewise, Figure 5.11 shows that LOFT achieves small and comparable average delay with PREI. Note that, the average delay with low data rate is surprisingly higher than that with high data rate. We believe that it is due to the inaccurate estimation on the workload and packet loss rate of the neighbor when there is only a small number of packets.

Again, the LOFT provides much higher overall reliability than PREI, as shown in Figure 5.12. In comparison with Figure 5.9, the reliability difference between the two protocols becomes larger when the link failure probability increases.

5.5.3 Links with Different Failure Probability

In real networks, links usually have different failure rates. We generate different failure probability to the links in this experiment. Each link in the network is assigned with a failure probability between 0

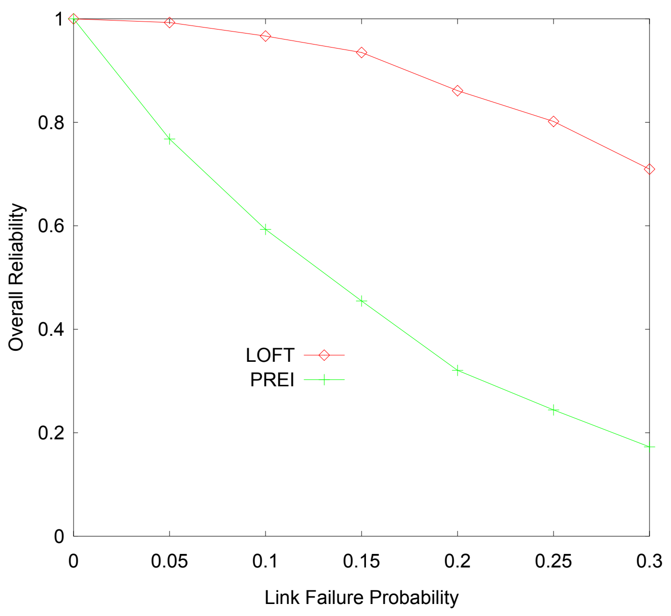


Figure 5.6: Overall reliability with data rate 15pkt/s.

and 0.3 randomly. Figure 5.13 and Figure 5.14 demonstrate similar results on the event reliability and average delay, with equal failure probability among the links. Finally, Figure 5.15 shows LOFT achieves much higher overall reliability than PREI, which agrees with the result in Figure 5.12.

5.5.4 Protocol Overhead

We discuss the overhead of the PREI and LOFT protocol here. There are basically three types of overhead, including 1) location update packets, 2) control packets for estimating the loads and link loss rates of neighbors, and 3) replicated data packets for handling link failures. Since the location update packets are inherited from traditional geographic routing protocols [52, 41, 42], we focus on the remaining two types of overhead.

The control packets for estimating the load of neighbors are re-

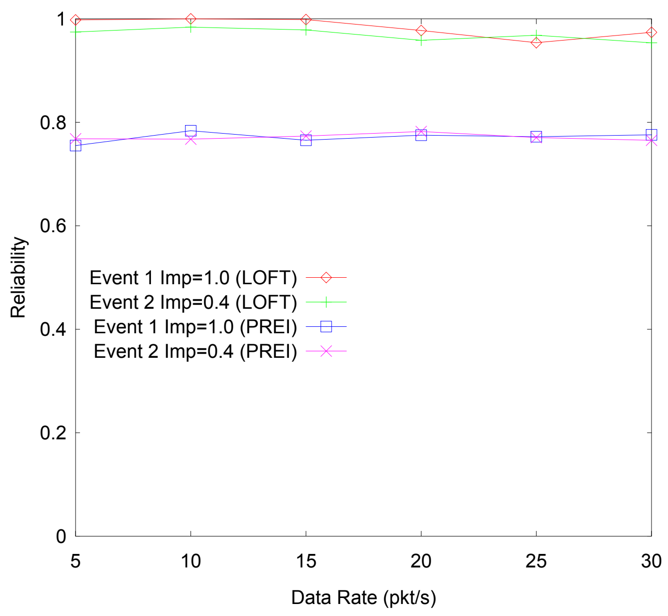


Figure 5.7: Event reliability with link failure probability 0.05.

quired for both PREI and LOFT. LOFT further utilizes these control packets for estimating the link loss rate of neighbors, so no extra packets are required. In our experiment, the control packet rate is 1 pkt/s, which is relatively low in comparison with the data rate. Regarding the replicated data packets, we evaluate the data replication factor in LOFT. Figure 5.16(a) shows that the replication factor is around 2 when the link failure probability is between 0 and 0.3 with data rate 15 pkt/s. The replication factor increases with the link failure probability to cope with the packet loss due to transmission failures. Similarly, Figure 5.16(b) also shows that the replication factor is independent of the data rates. The above results indicate that our protocol overhead is affordable in meeting the reliability objective for latency-oriented and fault tolerant data transmission.

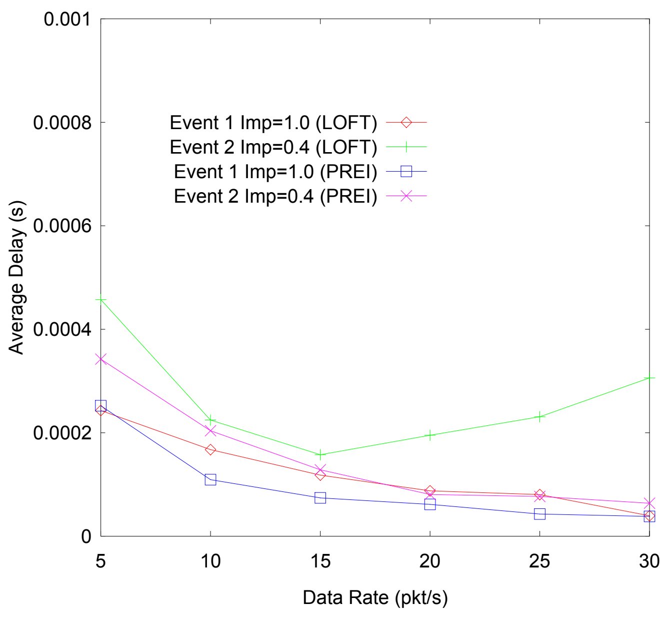


Figure 5.8: Average delay with link failure probability 0.05.

5.6 Summary

In this chapter, we proposed a reliable data transport protocol in WSANs. We considered that the system reliability in this context is closely related to the delay and the resistance to link failures, which should be jointly optimized. We also suggested that the non-uniform importance of the event data can be explored in the optimization procedure. Following these arguments, we proposed a general latency-oriented fault tolerant data transport protocol. It adopts smart priority scheduling and applies replication of packets adaptively in handling link failures. The protocol consists of two steps. It first locates a list of neighbors that provide satisfactory advancement to the actuator by checking their locations, and then estimates their maximum affordable incoming data rates through analyzing their priority queue utilization. To cope with transmission failures, packet loss rates of the links are updated regularly. Packets are al-

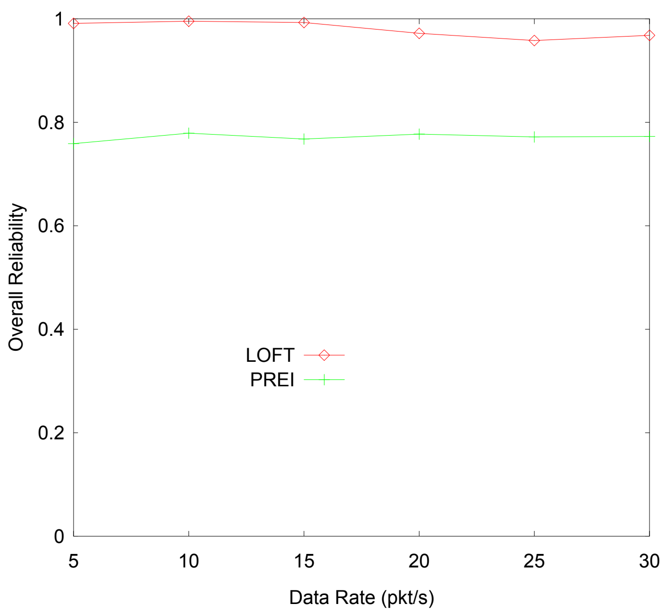


Figure 5.9: Overall reliability with link failure probability 0.05.

located to the next hops in proportion to their maximum affordable data rates. Replication of packets is applied adaptively based on the required data reliability and the estimated path success rates for delivery. The performance of the proposed protocol has been examined through extensive simulations. The results demonstrated that our protocol can significantly enhance the reliability for data transport, while the average delay for high priority packets is minimized.

☐ **End of chapter.**

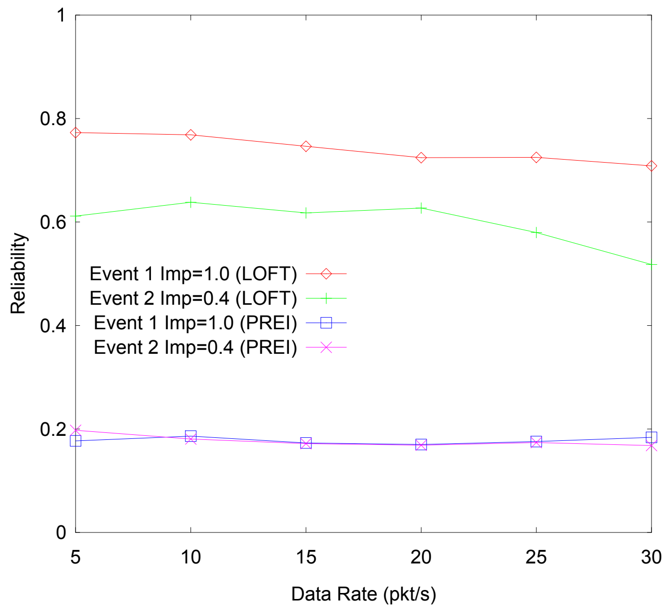


Figure 5.10: Event reliability with link failure probability 0.3.

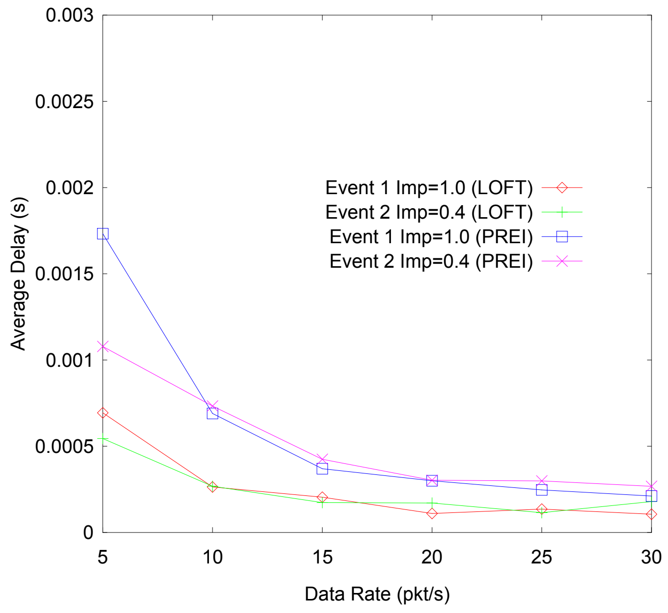


Figure 5.11: Average delay with link failure probability 0.3.

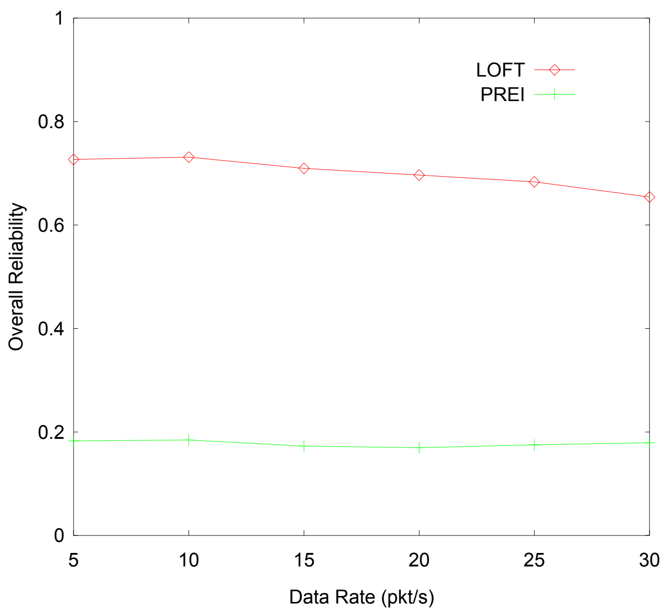


Figure 5.12: Overall reliability with link failure probability 0.3

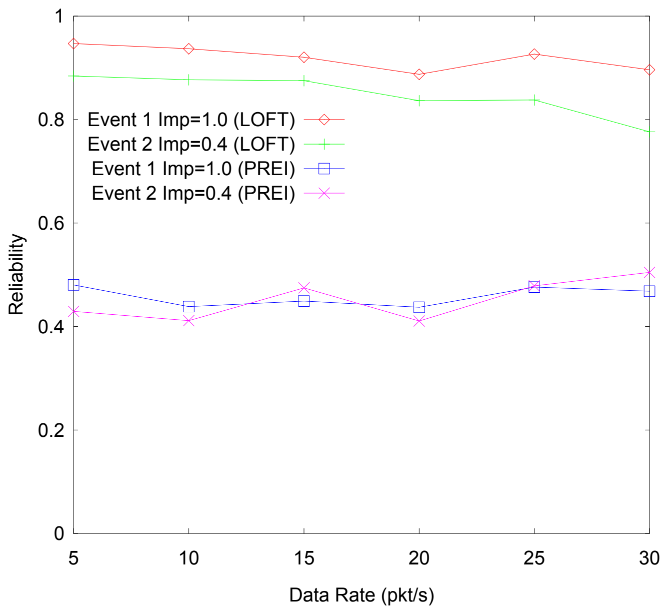


Figure 5.13: Reliability with random link failure probability between 0 and 0.3.

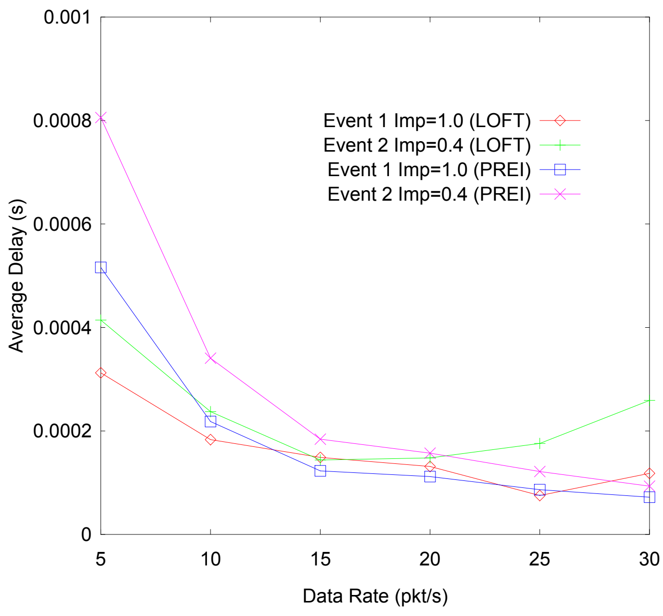


Figure 5.14: Average delay with random link failure probability between 0 and 0.3.

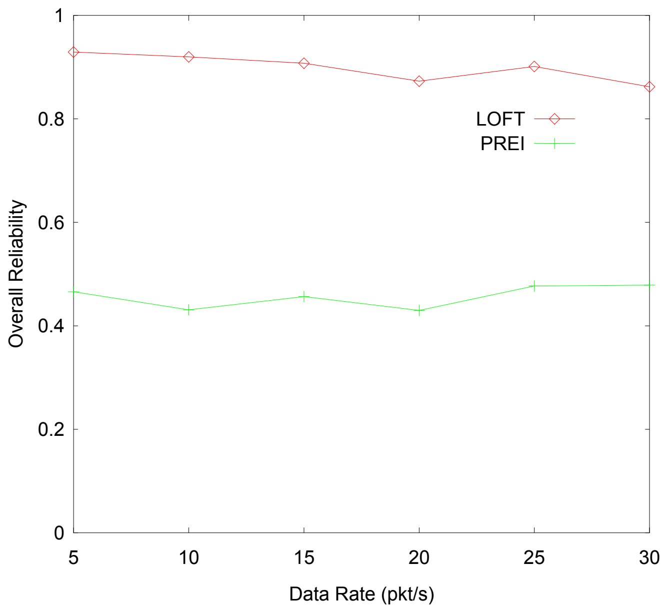
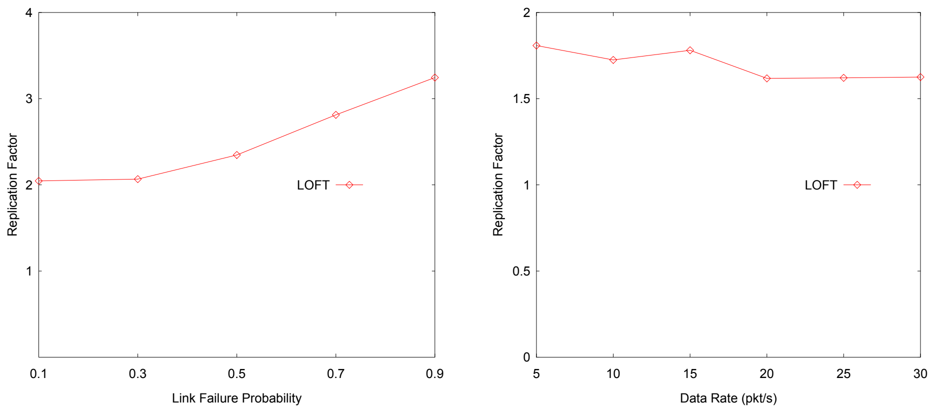


Figure 5.15: Overall reliability with random link failure probability between 0 and 0.3.



(a) (b)

Figure 5.16: Replication factor with (a) data rate 15 pkt/s (b) link failure probability 0.05.

Chapter 6

Power-Controlled Real-Time Data Transport Protocol

6.1 Overview

In the previous chapters, we discussed delay-aware and reliable event reporting from sensors to actuators. In this chapter, we consider also the energy efficiency in data transmission. We specifically tailor the data transport protocol design problem according to the features of WSANs and propose POWER-SPEED, a real-time data transport protocol for WSANs to achieve energy-efficient data transport for delay-sensitive event reporting. In POWER-SPEED, sensor nodes select the next-hop neighbor to actuators according to the spatio-temporal historic data of the upstream QoS condition, which completely avoids control packets. With an adaptive transmitter power control scheme, POWER-SPEED conveys packets in an energy-efficient manner while maintaining soft real-time packet transport. It thus reduces the energy consumption of data transport while ensuring the QoS requirement in timeliness domain. We demonstrate the effectiveness of POWER-SPEED through simulations with NS-2.

6.2 Energy-Efficient Real-Time Data Reporting of Delay-Sensitive Events

We consider a WSAN that consists of a collection of sensor nodes S and actuator nodes A. Each sensor node is denoted by s_i ($s_i \in S,\ i = 1, 2, ..., |S|$). Sensor nodes are stationary. Their locations do not change after they are deployed. Each actuator node is denoted by a_j ($a_j \in A,\ j = 1, 2, ..., |A|$). Actuator nodes are mobile.

For an event that takes place in the network area, a subset R of S can sense the event and report event data to the actuators. We call these nodes *reporting nodes*, each of which is denoted by r_k ($r_k \in R,\ k = 1, 2, ..., |R|$). Because the actuators can directly communicate with each other as they can be equipped with powerful antennae, we consider that the destination of the reporting packets can be any of the actuators, *i.e.*, reporting traffic can be routed in an anycast manner to the actuators.

For each in-network node u, $N(u)$ denotes the set of nodes which node u can communicate directly without relaying of other nodes. We call nodes in $N(u)$ the *neighbors* of node u. Each node is aware of the existence of each of its neighbors.

6.2.1 Location-Aware Networks

WSANs are employed to sense and handle environmental data. It is usually required that each in-network node is aware of its geographic location as the location information is necessary to identify and locate physical phenomena. In such WSANs, each sensor or actuator node should know its approximate geographic location. Location information is achievable if each node carries a GPS receiver or if some localization algorithms (*e.g.*, that in [14]) are employed.

Such location information can greatly facilitate the design of a data transport protocol, which will be explored in our following discussion. The location of a node is denoted by $X(u)$. We define $Dis(u, v)$ as the physical (Euclidian) distance between node u and node v,

$$Dis(u, v) = \|X(v) - X(u)\|, \tag{6.1}$$

where u and v can be s_i or a_i.

6.2.2 Stateless Data Transport via Dynamic Paths

In WSANs, the actuators, *i.e.*, the sinks of sensor-reporting traffic are mobile. As a result, the network topology changes frequently. Global shortest-path routing from reporting nodes to the actuators is not feasible because frequent reestablishment of shortest paths inevitably causes high overhead in terms of energy required for exchanging control packets. Therefore, to be energy-efficient, the sensor reporting packets have to be forwarded in a stateless manner. An in-network node does not maintain a routing table to the actuators. It should find out where to forward the sensor reporting packets by exchanging as few control packets as possible with its neighbors. The data transport protocol must convey sensor reporting packets to the actuators via dynamic paths in order to adapt to the mobility of the actuators.

6.2.3 Delay-Sensitive Data Transport

In most application cases of WSANs, reporting data are delay-sensitive. For example, in a WSAN that performs real-time environmental monitoring, it is required that environmental data are obtained by the in-network actuators within a predefined latency bound. Different events may have different latency-bound requirements. $L(e)$ denotes the latency bound of event e. We consider a packet on a particular event from a reporting node is successfully delivered if the packet can reach an actuator with latency less than the latency bound. A protocol for delay-sensitive data transport should successfully deliver as many packets as possible. It should also adapt to the different latency-bound requirements of different events.

6.2.4 Power-Controlled Packet Transmission

Since WSANs are usually location-aware networks, we notice that such location information can be utilized to conduct power control of a node's wireless transmitter. A transmitter power control scheme enables each node to set its power level to a minimum value under the constraint that the packet sent by this node could just reach its intended neighbor. The energy consumption of data transport can thus be reduced. Power-controlled packet transmission is an important technique to save the energy consumptions of sensor nodes and prolong the lifetime of a network.

The prerequisite of a transmitter power setting scheme is that each sensor node can set its own wireless transmitter power level. This is true in typical sensor node implementations. For example, the Berkeley Mica Mote [55] provides such program interfaces.

According to the wireless signal fading model [107], packets transmitted from node u can reach its intended neighbor v if the transmitter power setting of node u satisfies:

$$Pr(u) \geq c \cdot (Dis(u, v))^n \tag{6.2}$$

Here c is a constant related to the wireless system parameters. n is the signal fading factor whose value is typically in the interval $[2, 5]$. The optimal transmitter power setting for node u to send a packet to node v is therefore computed by:

$$Pr(u) = c \cdot (Dis(u, v))^n = c \cdot \|X(v) - X(u)\|^n \tag{6.3}$$

6.3 Designing the POWER-SPEED Protocol

6.3.1 Estimating the Hop-by-Hop Delays

As a protocol designed to transport real-time packets on delay-sensitive events, POWER-SPEED requires to know how long a packet has been experienced in the network. As each data packet has a *deadline* (*i.e.*, the maximum time the packet can experience before it reaches an actuator), the remaining time before the packet expires is a crucial parameter for POWER-SPEED to select where to forward the packet and thus guarantee that the packet can reach an actuator before expiration. Before discussing the details on this next-hop selection scheme in Section 6.3.3, we elaborate the mechanism we adopt to calculate the delay a packet has experienced and how to estimate traffic condition of the downstream path (*i.e.*, hop-by-hop delay a packet will experience in the future).

To facilitate the following discussion, we demonstrate part of a POWER-SPEED data packet in Figure 6.1.

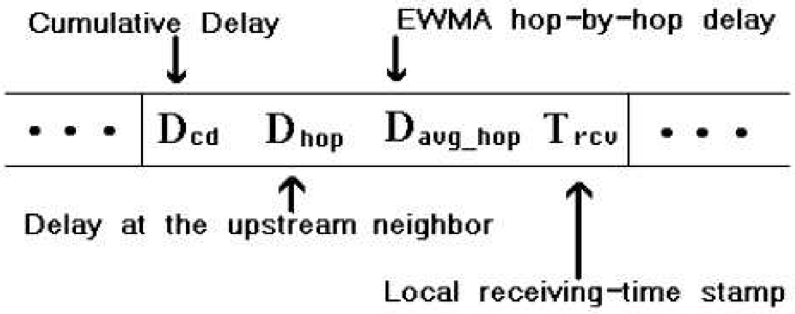


Figure 6.1: Part of a POWER-SPEED packet

The interval between the time when node u receives a packet and the time the packet is sent and received by node u's intended neighbor v, denoted by $Delay(u)$ is:

$$Delay(u) = t_{prop} + t_{proc} + t_q + t_{tran} \tag{6.4}$$

where t_{prop} is the propagation delay from u to its intended neighbor, t_{proc} is the processing delay at node u, t_q is the queuing delay at node u during which the packet is waiting to be sent out, and t_{tran} is the transmission delay which is related to the channel bandwidth and packet size. We call $Delay(u)$ the *hop-by-hop delay* between node u and its neighbor.

In POWER-SPEED protocol, node u timestamps the T_{rcv} field of a packet when the packet is received. When it sends the packet, it can encapsulate the time experienced by the packet inside node u into the D_{hop} field of the packet. This time value is $t_{proc} + t_q$ in Equation (6.4). t_{prop} is the light-speed propagation delay which can be ignored. t_{tran} can be calculated by its intended node (*i.e.*, node v) by bandwidth value and packet size. With t_{tran} and the encapsulated value of D_{hop}, node v can thus approximately calculate $Delay(u)$. Node v can calculate the *cumulative delay* (denoted by D_{cd}) that the packet has experienced in the network when it receives the packet and update the D_{cd} field of the packet (which initially is zero when the packet is generated) by adding $Delay(u)$.

The current $Delay(u)$ is also averaged by node v with an exponentially weighted moving average (EWMA) approach [63] to get the estimation of hop-by-hop delay from node u to node v. $\widetilde{D}(u)$ denotes this estimation. Specifically, $\widetilde{D}(u)$ is updated with current $Delay(u)$ according to:

$$\widetilde{D}(u) = \alpha\widetilde{D}(u) + (1-\alpha)Delay(u) \tag{6.5}$$

where α is a constant.

To describe how an intermediate relaying node s_n estimate the future hop-by-hop delay a packet will experience, denote $\{r_1, s_1, s_2, ..., s_n\}$ as the traffic path from a source node r_1 to s_n. In POWER-SPEED, node s_n estimates the future hop-by-hop delay a packet will experience based on only the hop-by-hop delay estimation of the upstream hops the packet has traveled, *i.e.*, the estimation of the hop-by-hop delay between each pair of adjacent nodes in $\{r_1, s_1, s_2, ..., s_n\}$. This means that POWER-SPEED does not require feedbacks from downstream nodes to estimate the future hop-by-hop delay a packet will experience. It thus completely avoids control packets, which featuring one of the merits of POWER-SPEED. Details are as follows.

Node s_1 can know $\widetilde{D}(r_1)$ as discussed above. It then fills the D_{avg_hop} field of the data packets with $\widetilde{D}(r_1)$. For node s_i ($1 < i \leqslant n$), it updates the D_{avg_hop} field according to:

$$D_{avg\_hop} = \beta D_{avg\_hop} + (1 - \beta)\widetilde{D}(s_{i-1}) \tag{6.6}$$

where β is a constant. Because $\widetilde{D}(s_{i-1})$ is calculated at node s_i and previous D_{avg_hop} is obtainable from packets, no control packet is required to obtain D_{avg_hop} according to Equation 6.6.

In POWER-SPEED, node s_n deems that the D_{avg_hop} calculated in Equation 6.6 as the estimation of the future hop-by-hop delay the packet will experience in its future journey. Such an estimation approach is very reasonable because it employs a 2-dimensional (spatio-temporal) EWMA estimation (*i.e.*, Equations (6.5) and (6.6)) that considers the impact of both time and space historical data.

6.3.2 Calculating Packet Transport Speed and Packet Forwarding Candidates

In POWER-SPEED, packet transport *speed* is defined as the average hops a packet can go through in one second. With the notations used in Section 6.3.1, node s_n calculates the packet speed by:

$$speed = \frac{1}{D_{avg\_hop}} \tag{6.7}$$

where D_{avg_hop} is obtained with Equation (6.6). Note that $speed$ is a positive real number.

Node s_n can calculate the maximum number of hops the packet can go through before its expiration with:

$$max\_hops = speed \cdot (deadline - D_{cd}) \tag{6.8}$$

For each actuator a_j, s_n calculates whether each node w in $N(s_n)$ is nearer than s_n to a_j[1]. If this is true, s_n estimate the number of hops a packet has to travel from s_n via w to a_j by Equation (6.9). If $hops < max\_hops$, node w is considered as one of the packet forwarding candidates. It means that the packet can go to actuator a_j via node w and it is expected to arrive at actuator a_j before it expires.

$$hops = \frac{Dis(s_n, a_j)}{Dis(s_n, a_j) - Dis(w, a_j)} \tag{6.9}$$

6.3.3 Selecting Next-Hop Neighbor

To be energy-efficient, a node (also denoted by s_n) should send packets to one of its packet forwarding candidates so that the total energy consumption required to deliver the packet from the node to an actuator is considered minimized. That is to say, node s_n should locally minimizes the total energy consumption required for a packet to reach an actuator.

Because the energy consumption for receiving a packet and processing a packet is constant, we consider the energy required for sending a packet to each packet forward candidate w for node s_n , denoted by $E(s_n, w)$. With transmitter power control, it is as follows.

$$E(s_n, w) = \gamma Pr(s_n) = \gamma c \cdot (Dis(s_n, w))^n \tag{6.10}$$

[1]This is feasible as a sensor node can easily know the locations of its neighbors. Actuators, although mobile, can broadcast their current locations to sensor nodes as they are equipped with powerful antennae

where γ is a constant related to the packet size.

Node s_n estimates the total energy consumption for the packet to reach actuator a_j via node w as follows.

$$total\_e(s_n, w) = hops \cdot E(s_n, w) \tag{6.11}$$

where $hops$ is calculated with Equation (6.9).

Among all packet forwarding candidates, node s_n selects the node that achieves the minimum value of total energy consumption calculated with Equation (6.11). It then sets its transmitter power level according to Equation (6.3) and forwards the packet to this candidate. In such a next-hop selection scheme, a packet that will expire in a longer period of time will adaptively be sent with lower transmitter power level to save energy. On the contrary, a packet that will expire sooner will be sent adaptively with higher transmitter power level, which results in fewer hop numbers between the sender to a destination actuator, and thus guarantees that the packet can reach its destination in a shorter period of time.

Note that this next-hop selection scheme in POWER-SPEED requires no control packets. And it is a fully distributed and localized algorithm which well suits WSANs.

6.4 Simulation Results

We program POWER-SPEED with the NS-2 [39] simulator and study the performance of POWER-SPEED with simulations. We investigate the performance of POWER-SPEED in terms of the total energy consumption to report data on delay-sensitive events and the proportion of the packets that can be delivered to actuators in time. For comparison, we also do simulations with the geographic routing protocol (GRP) [72], where greedy forwarding is employed and no control packet is required like POWER-SPEED. Detailed settings of the simulation network are summarized in Table 8.1.

Table 6.1: Simulation Parameters

Network size	200m x 200m
No. of sensors (N)	100
Bandwidth	2 Mbps
Radio range	40m
MAC layer	IEEE 802.11 without RTS/CTS and ACK
Wireless Communication Model	Two-Ray Ground
Packet Size	36 bytes

In order to have a direct observation on the energy-saving effect of POWER-SPEED, we first study the protocol performance in case that there is only one concurrent event in the network. We randomly deploy i sensor nodes in a uniform manner. Let the event takes place at location $(50m, 50m)$. The reporting of the event by source nodes lasts for 1000 seconds. We change i from 100 to 250, and we change the reporting rate of source node from 5 to 30 packets/second. For each setting, we select different random seeds and run simulations for 5 times. Results are averaged.

Figure 6.2 shows the total energy consumption of the whole network in case that $i = 100$ when different report rates of source node are set. The curves are in linear manner as energy consumption is linearly related to number of packets transmitted. It can be seen that POWER-SPEED effectively reduces the total energy consumption for conducting an event-reporting task. It saves about 40% energy comparing to the GRP protocol. This is not surprising due to the energy-saving technique adopted in POWER-SPEED.

Figure 6.3 further shows the impact on in-network node number on the total energy consumption in case that the reporting rate of source node is $20\ packets/s$. We can see that when the node density increases, the enhancement of POWER-SPEED is better than that

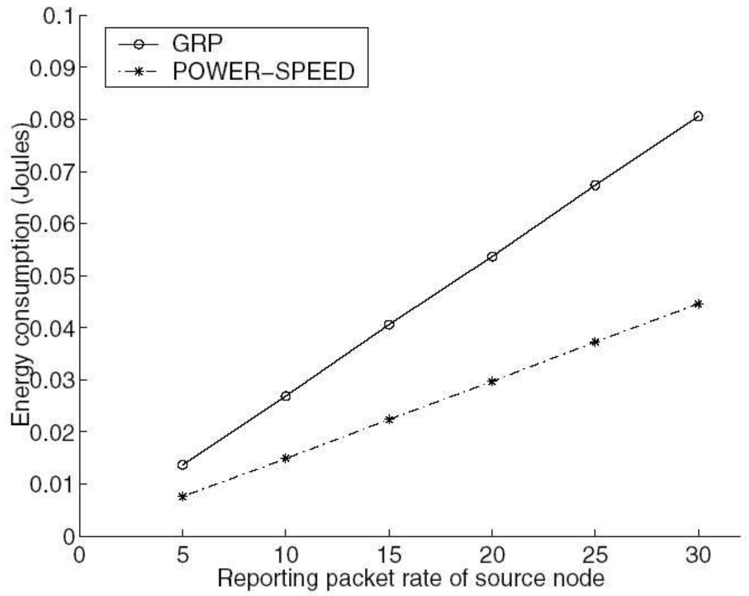


Figure 6.2: Total energy consumptions of different source reporting rates

of GRP. This is natural. GRP adopts a best-effort packet-forwarding mechanism with fix transmission range, the hop numbers between sources and actuators are nearly the same although node density increases. POWER-SPEED, on the other hand, adapts to node density and selects more energy-efficient paths as node density increases.

To study how well POWER-SPEED adapts to different latency requirements, we conduct simulations in which two concurrent delay-sensitive events with different latency requirements coexist. We randomly deploy 100 sensor nodes in a uniform manner. We let the event e_1 and e_2 takes place at locations $(50m, 50m)$ and $(80m, 80m)$. The reporting of the event by source nodes lasts for 1000 seconds. The maximum tolerable delays of packets on e_1 and e_2 are $0.025s$ and $1s$ in order to investigate how POWER-SPEED performs in reporting a more critical (in term of latency bound requirement) but farther event e_1. We set the reporting rate of source node in the range $(10, 30)$. For each setting, we also set different random seeds to run simulations, the results of which are averaged.

Figure 6.4 demonstrates the in-time packet arrival-rate of packets

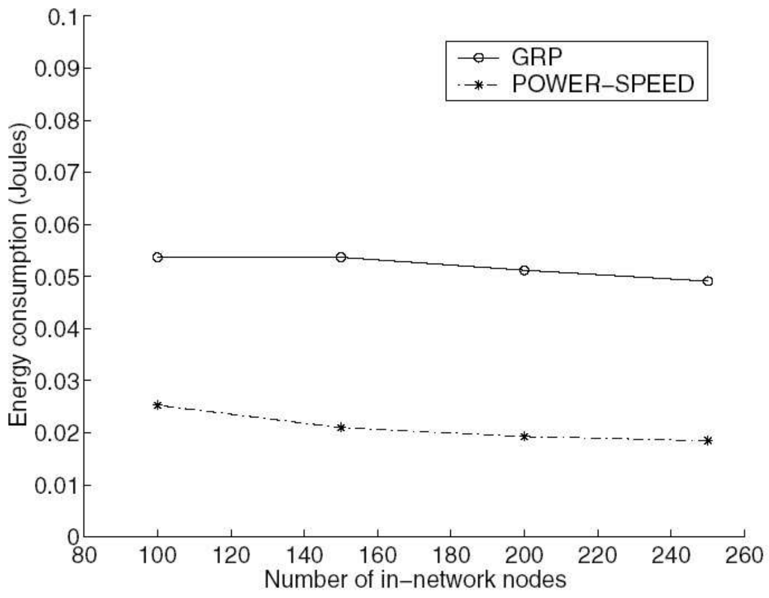


Figure 6.3: Total energy consumptions of different node number

on e_1 and Figure 6.5 demonstrates the total energy consumption. POWER-SPEED is comparable to GRP in terms of in-time packet arrival-rate of a critical event (*i.e.*, e_1). But it saves more than 50% of energy comparing to GRP. This is because POWER-SPEED can adapt well to events with different latency requirements. Packets on a less critical event (*i.e.*, e_2) can be sent with lower transmitter power level to save energy, while packets on a more critical event (*i.e.*, e_1) can be sent with higher transmitter power level to guarantee the arrival-rate.

6.5 Summary

This chapter studies the problem of real-time data transport for reporting delay-sensitive events in WSANs. We carefully tailor our protocol design by examining the features of WSANs, and propose POWER-SPEED, a power-controlled real-time data transport protocol for WSANs. POWER-SPEED avoids the overhead of control

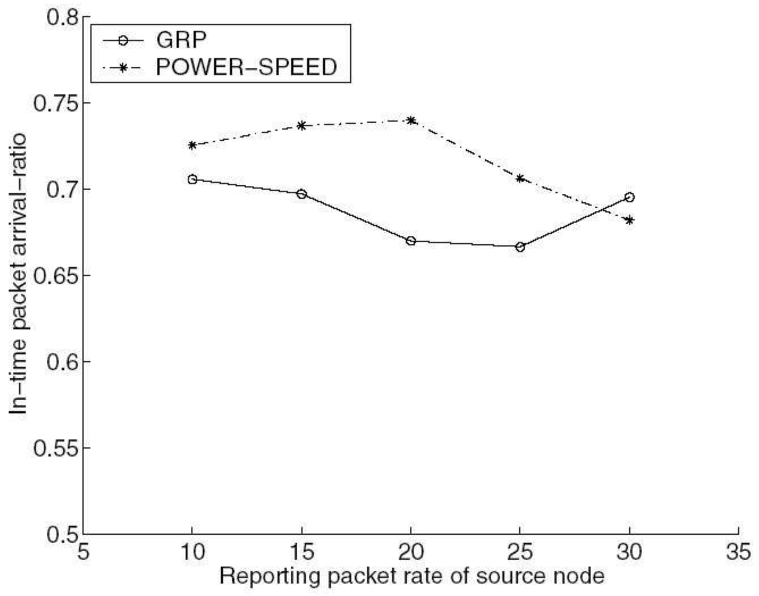


Figure 6.4: Comparison of in-time packet arrival-rates

packets by estimating the downstream path quality (in terms of delay) with only spatio-temporal historic data of the upstream path quality, which is obtainable without exchanging additional packets. POWER-SPEED selects the next-hop neighbor based on the downstream path quality and the latency-bound requirement of packets. Adaptively, it sends a packet that will expire in a longer period of time with lower transmitter power level to save energy. And it sends a packet that will expire sooner with higher transmitter power level, which results in fewer hop numbers between senders to destination actuators, and thus guarantees that the packet can reach its destination in a shorter period of time. In this way, POWER-SPEED achieves energy efficiency while maintaining the QoS requirement in timeliness domain. We perform extensive simulations with the popular network simulation tool NS-2 and the results demonstrate the effectiveness of our protocol.

□ **End of chapter.**

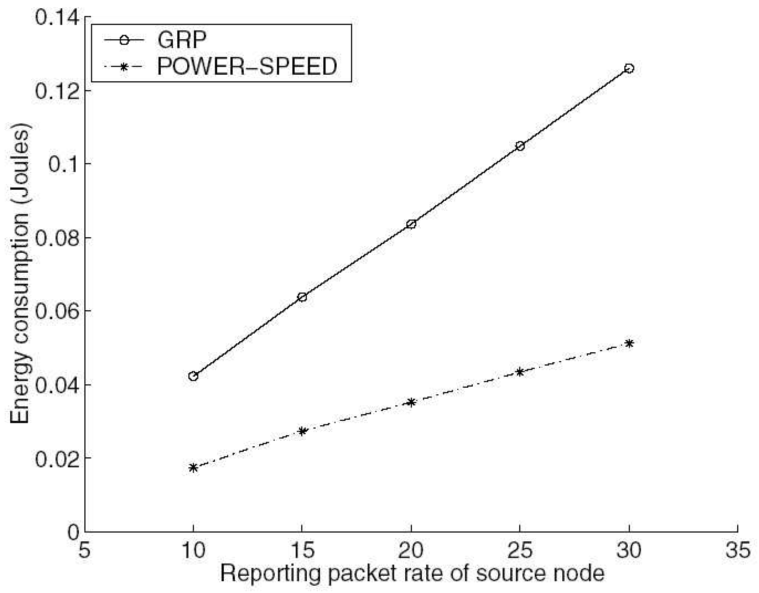


Figure 6.5: Comparison of total energy consumptions

Chapter 7

The Route Design Problem

7.1 Overview

After studying delay-oriented reliable communication in WSANs, we move on to discuss the coordination problem among the actuators. We consider the mobility of the actuators and work on the route design problem for efficient and cooperative data collection.

In WSANs, multiple actuators can patrol along different routes and communicate with the static sensors. To minimize the data collection time, an effective route design is crucial for the actuators to travel in the sensing field. In this chapter, we present a mathematical formulation of the route design problem, and show that the general problem is computationally intractable. We then propose two practically efficient algorithms to reduce the waiting time for the sensors. Our algorithms adaptively differentiate the actuator visiting frequencies to the sensors according to their relative weights and data generation patterns. The first algorithm builds routes by constructing minimum spanning trees with equal size, where sensors with higher weights will be included by more routes. Conversely, the second algorithm builds routes with different lengths, so the sensors with higher weights can join shorter routes. We then discuss a distributed implementation for route design with a simplified actuator allocation algorithm. Simulation results demonstrate that our algorithms can effectively reduce the overall data collection time in wireless sensor-actuator networks.

7.2 Problem Formulation

We consider a Wireless Sensor-Actuator Network consisting of multiple mobile actuators and a set of static sensors. The actuators move in the sensing field along independent routes. Each static sensor has a limited buffer to accommodate locally sensed data. When an actuator approaches, the sensor can upload the data to the actuator and free its buffer.

The parameters of this system are summarized in Table 7.1. Our objective is to minimize the overall length of routes, while guaranteeing locations with higher weights can achieve lower actuator inter-arrival time.

Table 7.1: System Parameters

s	Sensor node
R	Communication range of sensor
c_{ij}	Cost from sensor location i to j
x_{ijk}	Boolean indicating whether link (i, j) is on route k
W_j	Weight of sensor j (a value between 0.0 and 1.0)
N_j	Number of sensors with weight W_j
A_j	Average actuator inter-arrival time for sensor j
T_k	Period of route k
N	Number of sensors
M	Number of actuators

Route Design Problem (RDP):

$$Minimize \sum_{\forall j} A_j * W_j * N_j, \tag{7.1}$$

where $T_k = \sum_{i=1}^{N} \sum_{j=1}^{N} x_{ijk} * c_{ij}$, $A_j = \mathbb{F}(T_1, T_2, ..., T_{M'})$ is the function of all T_k that pass through the sensor j.

Subject to:

$$\sum_{k=1}^{M} \sum_{i=1}^{N} x_{ijk} \geq 1, \forall j = 1, ..., N \tag{7.2}$$

$$||s, k|| \leq R_s, \forall s, \exists k \tag{7.3}$$

$$\sum_{i=1}^{N} x_{ipk} - \sum_{j=1}^{N} x_{pjk} = 0, \forall k = 1, ..., M, p = 1, ..., N \tag{7.4}$$

$$y_j - y_i + N \sum_{k=1}^{M} x_{ijk} \leq N - 1, i \neq j = 1, ..., N \tag{7.5}$$

$$x_{ijk} \in \{1, 0\}, \forall i, j, k;\ y_i \text{ arbitrary}, \tag{7.6}$$

where our objective (Eq. 1) is to minimize weighted average actuator inter-arrival time among all sensors. $\mathbb{F}(T_1, T_2, ..., T_{M'})$ is a function for calculating the A_j of a sensor that is passed by the routes with period $T_1, T_2, ..., T_{M'}$. Constraint (2) suggests that each sensor should be on at least one route. Constraint (3) ensures every sensor s can communicate with the actuators, provided that at least one point on some routes k is within the communication range R of the sensor. Constraint (4) states that if an actuator visits a sensor, it must also depart from it. Constraint (5) is the subtour-elimination condition derived for the travelling salesman problem [91]. The route design problem differs from the traditional vehicle routing problem in the following aspects:

1) The sensors are of different weights, according to their data

generation rates and importance. Sensors with higher weights expect lower average actuator inter-arrival times.

2) Sensors upload data to actuators through wireless communications. Data transmission is possible only when the distance between the sensor and actuator is within a communication range R.

3) It is not necessary for each route to pass through the depot (or the base station), as the actuators generally can communicate with the base station directly with their stronger power.

We now offer some observations on the general route design problem.

Definition 1. A route is a tour walked through repeatly by an actuator.

Property 1. *Route with a Hamiltonian cycle achieves shorter maximum inter-arrival time A_{max} than that without.*

Proof. Let $p_1, p_2, .., p_n$ be the sensors on a route, and $c_{12}, c_{23}, ..., c_{n1}$ be the cost for an actuator to move between consecutive sensors. We have $T = \sum_{\forall c} c/v$ is the period of the route, where v is the moving speed of the actuators. If the route is a Hamiltonian cycle, A_{max} of any sensor p is always equal to T. On the contrary, considering a route that is not a Hamiltonian cycle, an actuator will need to walk from one end to another end, then back to the beginning sensor to complete a cycle. The A_{max} of the sensors at two ends will then be $2T$, which is much longer than T. $\square$

Property 2. *The average actuator inter-arrival time A_j of a sensor j on multiple routes can be calculated as $\mathbb{F}(T_1, T_2, ..., T_{M'}) = \frac{\prod_{k=1}^{M'} T_k}{\sum_{k=1}^{M'}(\prod_{i=1,i\neq k}^{M'} T_i)}$.*

Proof. Find a common multiple $\mathbb{Q}$ among the period $T_1, T_2, ..., T_{M'}$ of all routes that pass through sensor location j. One of the simplest common multiple is $\mathbb{Q} = T_1 * T_2 * ... * T_{M'}$. If $\mathbb{Q}$ is a time interval, the number of times that the actuator on route k passes through sensor j is $\prod_{i=1,i\neq k}^{M'} T_i$. The total number of times that the actuators visit j is the summation of the number of visits by each actuator, $\sum_{k=1}^{M'}(\prod_{i=1,i\neq k}^{M'} T_i)$. The average inter-arrival time A_j is then equal to the total time over the total number of visits, that is, $A_j = \frac{\prod_{k=1}^{M'} T_k}{\sum_{k=1}^{M'}(\prod_{i=1,i\neq k}^{M'} T_i)}$. □

Theorem 1. *The route design problem (RDP) is NP-hard.*

Proof. To prove the route design problem (RDP) is NP-hard, we show that HAM-CYCLE $\leq_p$ RDP. Let $G = (V, E)$ be an instance of HAM-CYCLE. We construct an instance of RDP as follows. We form a complete graph $G_k = (V_k, E_k)$ for each actuator k, where $E_k = (i, j) : i, j \in V_k, i \neq j, G' = (V, E') = \bigcup_{\forall k} G_k, V = \bigcup_{\forall k} V_k, E' =$

$\bigcup_{\forall k} E_k$, for $k = 1, ..., M$, and we define the cost function c by

$$c_{i,j} = \begin{cases} 0 & \text{if } (i,j) \in E, \\ 1 & \text{if } (i,j) \not\in E. \end{cases}$$

Considering W_j = constant for all $j \in V$, all sensors j will be visited once by any of the actuators, such that the instances of RDP is $(G', c, C[1, ...k, ...M] = 1)$, where $C[k]$ is the sum of $c_{i,j}$ for all $(i,j) \in E_k$.

We now show that the graph G has a Hamiltonian cycle if and only if G' has M tours, where the cost $C[k]$ at each cycle k is at most 1. Suppose that graph G has a Hamiltonian cycle h. Each edge in h belongs to E and thus has cost 0 in G'. Select M edge in h, $(i_1, j_1), (i_2, j_2), ...(i_M, j_M)$ and remove them. Then, reconnect these sensors in sequence $(j_1, i_2), (j_2, i_3), ...(j_M, i_1)$. Since the edges $(j_1, i_2), (j_2, i_3), ...(j_M, i_1)$ do not belong to E, each of them has cost 1. As a result, h is broken into M cycles, where the cost $C[k]$ of each cycle k is at most 1.

Conversely, suppose that G' has M cycles $h_1, ...h_k, ...h_M$ of cost at most 1. We can easily merge these M cycles into one Hamiltonian cycle. Since the cost of each edge in E_k are 0 and 1, at most one edge of each cycle is 1, and all other edges are 0. Remove the edge in each cycle that is 1, $(i_1, j_1), (i_2, j_2), ...(i_M, j_M)$. Then, reconnect the sensors involved in a new order, $(j_1, i_2), (j_2, i_3), ...(j_M, i_1)$, such that a single hamiltonian cycle can be formed. $\Box$

7.3 Route Design Algorithms for Multiple Actuators

We now propose two route design algorithms, RDNV and RDPL, for multiple actuators under a centralized scenario. It is practical for one of the actuators or the base station to execute the algorithm in a small-scale network. We will then discuss a distributed implementation of the route design algorithm for large sensor networks in the next section.

Our route design algorithms calculate independent route for each individual actuator. We will first present the RDNV algorithm, which suggests the actuators to have independent routes with similar lengths and guarantees highly weighted sensors to be visited by more actuators. Then, we will present the RDPL algorithm, which forms independent routes with different lengths and ensures the highly weighted sensors to be visited by actuator in shorter route.

7.3.1 Route Design Algorithm by Varying Number of Visits (RDNV)

The RDNV algorithm utilizes multiple minimum spanning trees (MSTs) for designing the routes of multiple actuators. Intuitively, we want to construct M routes with equal period T, in which sensors with higher weights W_i will be visited more frequently. Notice that each actuator is in charge of one route. A sensor with weight W_i will be passed by $W_i * M$ actuators. In other words, it will be found on $W_i * M$ routes. For example, a sensor with $W_i = 0.75$ in a network with 4 actuators will be visited by 3 different actuators. If all actuators have the same period T, from Property (2), its average actuator inter-arrival time A_{avg} will be $T/3$.

RDNV allows sensors to join different number of routes according to their weights. Each time, a sensor selects the route with the shortest length, such that its waiting time can be minimized. This strategy will also equalize the lengths of various routes. The algorithm provides great flexibility for a sensor to join any routes. It works well especially for networks with uniform random sensor distribution, where the sensors spread evenly and utilize this flexibility beneficially.

In the following, we describe the details of the MST-based route design algorithm.

Clustering with Spanning Trees

Sensors are assigned with a specific number of routes N_i according to their weights W_i, where $N_i = \lceil W_i * M \rceil$. The locations with the same N_i belong to the set S_i. Our algorithm builds M spanning trees T_k, which include the sensors on the M routes respectively, where $k = 1, ..., M$. Firstly, the sensors with the highest N_i, say $N_i = M$, will be assigned to all trees. Then, the locations with the next highest N_i will be assigned to N_i trees with the lowest costs (Figure 9.1). The process is repeated until there is no remaining locations, as shown in Algorithm 6. Figure 9.2 shows the final routes formed by three trees, which have similar lengths. It demonstrates that sensors with higher weights will be visited by more actuators. Since the routes have similar lengths, they also achieve lower actuator inter-arrival time A_{avg}.

Forming the TSP Solution

The M trees above result in M groups of sensors that should be walked through by actuators on distinct routes. The route design

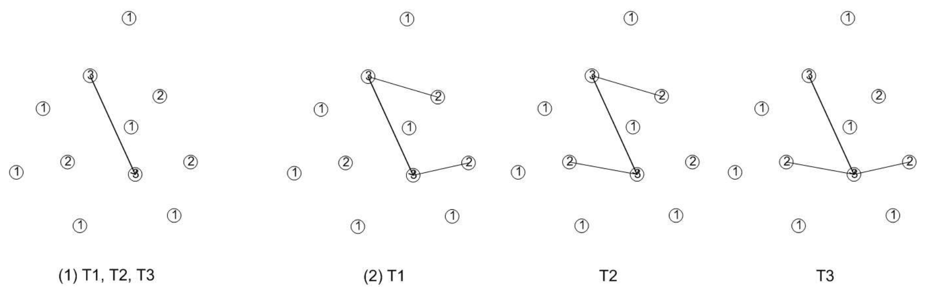


Figure 7.1: Step 1: Sensors with N_i = 3 are involved in all trees; Step 2: Sensors with N_i = 2 are involved in any two of the trees in RDNV.

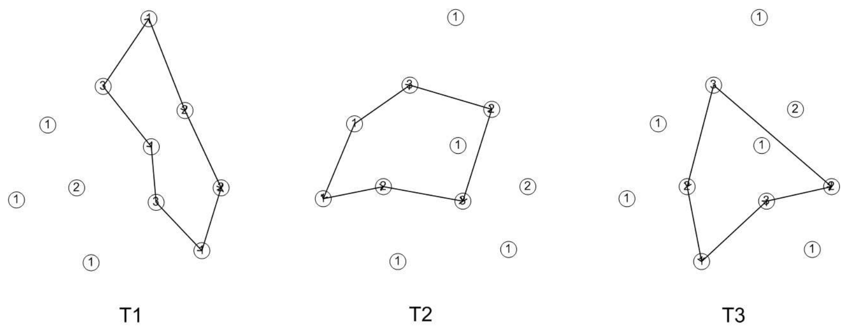


Figure 7.2: Final routes formed in RDNV.

problem can then be reduced to the Travelling Salesman Problem (TSP) for each group of sensors. In the literature, several algorithms to calculate the TSP paths are proposed, such as the nearest neighbor, LKH [11], and some polynomial-time approximation schemes [7]. Among different approximation algorithms, the Approx-TSP-Tour algorithm [22] uses the minimum spanning tree to create a tour whose weight is a lower bound on the length of an optimal traveling-salesman tour. Its cost is no more than twice of the minimum spanning tree's weight. After computing the MST, then it performs a preorder traversal on the tree to obtain a Hamiltonian cycle [49]. In our solution, the MST is created using Prim's algorithm [22], which is in polynomial-time.

Algorithm 6 Spanning tree construction in RDNV

```
for $k = 1$ to $M$ do
  $T_k = \phi$;
end for
for $i = M$ down to 1 do
  for each node $p \in S_i$ do
    for each spanning tree $T_k$ do
      $T'_k = T_k \bigcup p$;
      Calculate the new cost of the tree $T'_k$, $C(T'_k)$
    end for
    sort the list of $C(T'_k)$
    for the top $i$ trees with the lowest new cost $C(T'_k)$ do
      $T_k = T'_k$;
      $C(T_k) = C(T'_k)$;
    end for
  end for
end for
```

Determining the Locations of Actuators

It is more efficient for a sensor to have short waiting time to the actuators, so the maximum inter-arrival time A_{max} may also be an important consideration other than A_{avg}. We focus on the sensors with the highest W_i and provide a simple method for deciding the starting point of the actuator on its route. A location with the highest W_i can be selected as a reference point p_r. Then, each actuator k will be assigned to the point after travelling for time $T * k/M$ from p_r on its own route. This is to encourage more even inter-arrival time of the actuators. More advanced methods can be studied as the future work.

Time Complexity Analysis:
Assume that N and M denote the number of sensors and the number of actuators, respectively. Then, the complexity for each step of the RDNV algorithm is as follow:

- Step 1: The spanning tree construction in algorithm 7 is dominated by the sorting with respect to the size of the trees when the sensors are joining the trees. Since the number of trees is equal to M, the sorting results in $O(NMlogM)$ complexity. Also, the sensors in S_i join N_i trees according to their weights, which result in $O(NM)$ complexity, given that $N_i \leq M$.

- Step 2: The running time of the Appro-TSP-Tour algorithm is $O(E) = O(N^2)$, since the input is a complete graph.
- Step 3: Time complexity of step 3 is $O(M)$.

In summary, the RDNV algorithm has an overall time complexity of $O(NMlogM + N^2)$.

7.3.2 Route Design Algorithm by Varying Path Length (RDPL)

We focus on the sensor network with non-uniform distribution in this algorithm. In some networks, the sensors may be concentrated in different clusters and network partitions are possible. Unlike uniform random distribution, sensors in the same cluster may have similar capability and weights as well. In RDPL, sensors with similar weights will be assigned to the same route with a particular length. Sensors with higher W_i will be put on shorter routes, and vice versa. The following are the steps of this algorithm.

Constructing the MSTs with Different W_i

We divide the weight $W_i \in [0, 1]$ into w ranges. Sensors are grouped into the appropriate sets of sensors, $S_1, S_2, ..., S_w$, according to their weight ranges. A minimum spanning tree T_k is then constructed for each set of sensors and its corresponding cost $C(T_k)$ is calculated. The cost $C(T_k)$ is a good approximation to the length of route k.

Estimating the Number of Actuators for each T_k

Each of the tree will be assigned with an appropriate number of actuators n_k serving it. Since higher weighted sensors should achieve shorter actuator inter-arrival time, the ideal inter-arrival time for sensors $S_1, S_2, ..., S_w$ could be $w * T, (w-1) * T, ..., 2 * T, T$, respectively, where T is the inter-arrival time for the sensors in the highest weight range. The cost $C(T_k)$ represents the length of the route, and $C(T_k)/a_k$ should then be proportional to the inter-arrival time $(w + 1 - k) * T$, such that

$$\begin{aligned} C(T_1)/a_1 &= w * T * v \\ C(T_2)/a_2 &= (w-1) * T * v \\ &\dots \\ C(T_w)/a_w &= T * v, \end{aligned} \tag{7.7}$$

where $a_1, a_2, ..., a_w$ are the number of actuators that should be assigned to the corresponding set of sensors, and v is the speed of actuator. The sum of the assigned number of actuators in all routes should be equal to the total number of actuators M in the network, such that

$$a_1 + a_2 + ... + a_w = M. \tag{7.8}$$

From the above equations, we achieve

$$\frac{C(T_1)}{w * T * v} + \frac{C(T_2)}{(w-1) * T * v} + ... + \frac{C(T_w)}{T * v} = M, \tag{7.9}$$

hence,

$$T * v = \{\sum_{k=1}^{w} C(T_k)/(w + 1 - k)\}/M \tag{7.10}$$

and

$$\begin{aligned} a_k &= C(T_k)/(T * v * (w + 1 - k)) \\ &= \frac{C(T_k) * M}{(w + 1 - k)\{\sum_{k=1}^{w} C(T_k)/(w + 1 - k)\}} \end{aligned} \tag{7.11}$$

Allocating the Actuators

The a_k achieved above is an ideal value, which may not be an integer, so we need to determine the practical number of actuators n_k to be assigned to a route. If a_k is smaller than 1, the sensors in S_k will be accumulated in $accum_S$ with other sensors in the following weight ranges, until the accumulated number of actuator $accum_a$ is greater than 1. Then, a new route R_q can be formed with n_q = $round(accum_a)$, including all the sensors in $accum_S$. The details of route formation and actuator allocation are shown in Algorithm 7.

Note that a route may have n_q greater than $accum_a$, which means the assigned actuators are more than the required actuators. In this case, more sensors can be added to the route, without degrading the expected actuator inter-arrival time. The route may prolong its length $Cost(R_q)$ by adding higher weighted sensors that are located closely until $Cost(R_q)$ is no longer smaller than $Cost_{exp}(R_q)$, where $Cost_{exp}(R_q)$ is the expected length of R_q. The length difference $Cost_{exp}(R_q) - Cost(R_q)$ can be calculated by $(n_q - accum_a) * (w + 1 - w_{max}) * T$, where w_{max} is the maximum weight among the sensors in $accum_S$.

Forming the TSP Solution

Up to now, a number of routes R_q are designed with the corresponding set of sensors S_q and actuators. Similar to RDNV, the route design problem can then be reduced to the Travelling Salesman Problem (TSP) for each route. Some R_q may be assigned with more than one actuator, say n_q. These actuators can divide the tree T_q into n_q sub-trees with similar cost before computing the TSP solution.

Time Complexity Analysis:

In addition to N and M, w denotes the number of weight ranges in RDPL, where $w \leq M$. Then, the complexity for each step of the RDPL algorithm is as follow:

Algorithm 7 Route construction and actuator allocation in RDPL

```
$q = 1; remain_a = M; accum_a = 0; accum_S = \emptyset;$
for $k = w$ down to 1 do
  if $k > 1$ then
    if $a_k \geq 1$ then
      $n_q = floor(a_k + accum_a);$
      $S_q = S_k \bigcup accum_S;$
      $remain_a - = n_q;$
      $q + +;$
      $accum_S = \emptyset;$
      $accum_a = 0;$
    else
      $accum_a + = a_k;$
      $accum_S; \bigcup = S_k;$
      if $accum_a \geq 1$ then
        $n_q = round(accum_a);$
        $S_q = accum_S;$
        $remain_a - = n_q;$
        while $Cost(R_q) < Cost_{exp}(R_q)$ do
          Route prolongation to $R_q$;
        end while
        $q + +;$
        $accum_S = \emptyset;$
        $accum_a = 0;$
      end if
    end if
  else
    $n_q = remain_a;$
    $S_q = accum_S \bigcup S_k;$
    while $Cost(R_q) < Cost_{exp}(R_q)$ do
      Route prolongation to $R_q$;
    end while
  end if
end for
```

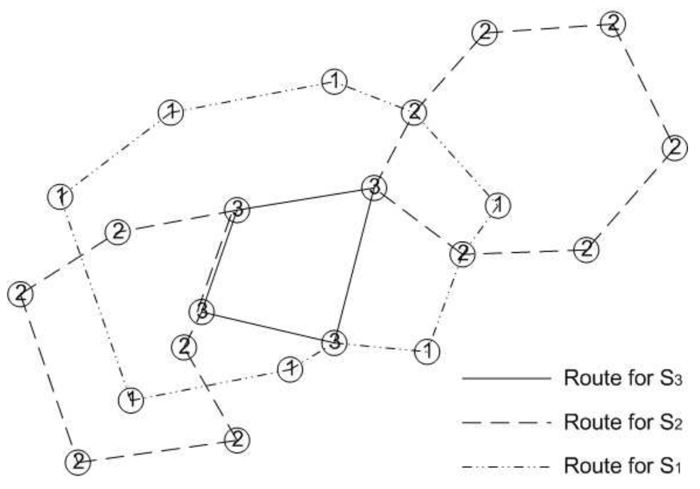


Figure 7.3: Final routes formed for four actuators in RDPL.

- Step 1: Forming MSTs for nodes in different weight ranges results in $O(N^2)$ complexity.
- Step 2: The complexity of calculating the a_k for all the weight ranges is $O(w^2)$.
- Step 3: The route construction and actuator allocation in Algorithm 7 is dominated by the union of disjoint set operations. It happens when the nodes in different S_k are accumulated in $accum_S$. We assume the disjoint-set-forest implementation with the union-by-rank and path-compression heuristics, since it is asymptotically the fastest implementation known [22]. There are $O(w)$ operations on the disjoint-set forest, which in total takes $O(w\alpha(E,V))$ time, where α is the functional inverse of Ackermann's function. Since $\alpha(E,V) = O(logE)$ and $E = N^2$ in a complete graph, the total running time of this step is $O(wlogN)$.
- Step 4: Similar to RDNV, the running time of the Appro-TSP-Tour algorithm is $O(E) = O(N^2)$.

As a result, the RDPL algorithm has an overall time complexity of $O(N^2) + O(w^2) + O(wlogN) + O(N^2) \approx O(N^2)$, given that $w \leq M < N$.

7.4 Distributed Implementation

In practice, in a large sensor network, it could be difficult for a base station or an actuator to know exactly the locations of all the sensors and accordingly make decisions on route design. Hence, we develop a distributed implementation, D-RDPL, which is based on the RDPL algorithm. Although RDNV can also be extended in a distributed manner, it requires more messages for the actuators to broadcast the sizes of their trees when the sensors are joining different routes.

In D-RDPL, the sensors and actuators identify the locations of other sensors together in a fully distributed manner. Then, the actuators can perform route design cooperatively without any centralized server. We now present the details of D-RDPL. Note that some new concepts are introduced in D-RDPL beyond the original RDPL to accommodate the distributed and lightweight design requirements.

Forming the R-clusters

Sensors construct MSTs with the same weight by local communications with their neighbors. If the communication range of sensors is R, the weight of each edge e in an MST must be smaller than or equal to R, that is $w(e) \leq R$. We define such an MST as an R-Cluster, $F_i(V', E')$, which includes all the sensors that are in the same weight range w_i and within R to some sensors in $F_i(V', E')$. We can apply various distributed MST construction algorithms [43, 9, 36] to form the R-clusters.

From Property 3, an R-cluster $F_i(V', E')$ of weight range w_i is part of the MST T_i, which includes all the sensors with weight range w_i in the network.

Definition 2. R-Cluster is an MST, $F(V', E')$, formed by a group of sensors by wireless communications in the network $G(V, E)$. The weight of edge $w(e') \leq R$ for all $e' \in E'$. The distance between two sensors u and v, $\|u, v\| > R$, for all $u \in V'$, $v \in V - V'$, where R is the communication range of sensor, $V' \subseteq V$.

Property 3. *If $F_i(V', E')$ is an R-cluster for some sensors in weight range w_i, it is a sub-tree of $T_i(V, E)$, $F_i \subseteq T_i$, where T_i is the MST that contains all the sensors with weight range w_i in the network.*

Proof. T_i is an MST containing all the sensors V with weight range w_i, where $V' \subseteq V$. At least one edge (u, v) in E, but not in E', connects some sensors $u \in V'$ to some sensors $v \in V - V'$, such that T_i is an MST including all V.
We now prove the property by contradiction. Suppose, to the contrary, there is an edge e' in E', but not in T_i. Removing e' breaks F_i into two components X and Y. We assume X is the component that contains u. There must be another edge x' in E connecting the component Y to any sensors in $V - Y$, such that $w(x') \leq w(e')$. Since F_i is an MST, $w(e')$ is already the shortest edge connecting X and Y. Edge x' must be connecting component Y to some sensors other than V', that is $V - V'$. Hence, the weight of edge x' is equal to the distance between two sensors u' and v', $w(x') = \|u', v'\|$, where $u' \in Y$, $v' \in V - V'$. Since $w(x') \leq w(e')$ and $w(e') \leq R$, thus, $\|u', v'\| \leq R$, yielding a contradiction. $\square$

Connecting the R-clusters

An R-cluster forest with the same weight can be connected together to form a bigger MST, T_i, which contains all the sensors of the same weight in the network. From Property 4, the edge with minimum weight that connects an R-cluster to another R-cluster in the forest can construct T_i.

Actuators will be assigned to different areas in the network, so they can explore the R-clusters and construct the MST cooperately to increase network efficiency. Each actuator looks for the R-clusters in its area, connects them according to their weight ranges, and stores the costs of trees. The cost of the MST portion in w_i managed by actuator A_j is represented by C_i^j, where $w_i = 1, 2, ..., w$.

Note that some R-clusters or some edges connecting them may be crossing two areas, so communications are required among the actuators in the neighboring areas. If an edge is crossing two areas, its cost will be counted by the actuator with a lower ID. At the end, sensors in various weight ranges form independent MSTs. The total number of MSTs is equal to w.

Property 4. *Let A be a subset of E that is included in some MST for $G = (V, E)$, and let $F = (V', E')$ be an R-cluster in the forest $G_a = (V, A)$. If (u, v) is an edge with the minimum weight connecting F to another R-cluster in G_a, then $A \bigcup \{(u, v)\}$ is also a subset of some MST.*

Proof. Let T be the MST that includes A, and assume that T does not contain the edge (u, v). We shall construct another tree T' that includes $A \bigcup \{(u, v)\}$ and show that T' is an MST.
Since T is an MST, there must be an edge that connects F and $V - F$ in T. Let (x, y) be any such edge and (x, y) does not belong to A. The edge (u, v) also connects the sensors u and v which are on the opposite sides of the cut $(F, V - F)$, so (u, v) forms a cycle with the path p that contains (x, y). Removing (x, y) breaks T into two components. Adding (u, v) reconnects them to form a new spanning tree $T' = T - \{(x, y)\} \bigcup \{(u, v)\}$. Since (u, v) is an edge with the minimum weight crossing $(F, V - F)$ and (x, y) also crosses this cut, $w(u, v) \leq w(x, y)$. Therefore, $w(T') = w(T) - w(x, y) + w(u, v) \leq w(T)$. Based on T is an MST and $w(T') \leq w(T)$, thus, T' must be an MST also. We have $A \subseteq T'$, since $A \subseteq T$ and $(x, y) \notin A$; thus, $A \bigcup \{(u, v)\} \subseteq T'$. Consequently, since T' is an MST, $A \bigcup \{(u, v)\} \subseteq T'$ is also a subset of some MST. □

Allocating the Actuators to the Routes

The cost of each MST is calculated by summing up the cost of its sub-trees, $C_i = \sum_{j=1}^{M} C_i^j$. It also represents the length of the route R_i for sensors in a particular weight range w_i. As shown in Equation (7), the number of actuators for sensors in weight range w_i is proportional to $C_i/(w + 1 - i)$. From Equation (10), the number of actuators a_i for sensors in different weight ranges is obtained. Since the computation of this equation is pretty lightweight, it is possible for each actuator to make its own calculation. Alternately, one actuator can perform the calculation and broadcast to all other actuators, but it may consume more energy and time for messaging.

Then, the actuators can form routes and decide n_i with a simplified approach, as shown in Algorithm 8. Routes with $a_i > 0$ are assigned with at least one actuator. The remaining actuators are allocated to the route with the maximum a_i repeatly, until all actuators are assigned. As a result, the number of actuators to the sensors with different weights is determined.

Next, the actuators can select the route that they would walk through, starting from the actuator with the lowest ID. An actuator A_j usually selects the route R_i, which has the highest C_i^j and is not selected by other actuators.

Algorithm 8 Simplified actuator allocation in D-RDPL

```
for $i = 1$ to $w$ do
  $n_i = 0$;
end for
$remain_a = M$;
for $i = 1$ to $w$ do
  if $a_i > 0$ then
    $n_i = 1$;
    $remain_a - -$;
    $a_i - -$;
  end if
end for
while $remain_a > 0$ do
  Find the maximum $a_i*$;
  $n_i * ++$;
  $remain_a - -$;
  $a_i * --$;
end while
```

Forming the TSP Solution

Similar to RDPL, route prolongation is required for route R_i if $n_i > a_i$. It implies that excessive actuators are assigned to R_i, so they are expected to prolong the length and visit more sensors. Then, partition of routes can be performed if there are more than one actuator serving a route, for example $n_i > 1$. Finally, the TSP solution can be computed by each actuator to form its own route.

Complexity Analysis:
We now analyze the communication complexity and time complexity of D-RDPL. The communication complexity is the worst case analysis for the total number of elementary messages sent during the algorithm. The time complexity is the maximum possible number of time units from start to the completion of the algorithm, assuming that the inter-message delay and the propagation delay of an edge is at most one time unit of the global clock. The complexity for each step of the D-RDPL algorithm is as follow:

- Step 1: The sensors broadcast “hello” messages to their neighbors to obtain the weights of their adjacent edges, which take $O(N)$ messages. We adopted a distributed MST construction algorithm [43] to form the R-clusters. It requires $O(E_r+NlogN)$ messages and $O(N)$ time, where E_r is the number of node pairs that are within the communication range R in N. Since R is small, $E_r \ll N^2$.
- Step 2: The actuators exchange messages for finding the outgoing edges with the minimum weight among the R-clusters in different areas. Assume the actuators can communicate directly with long range communication, they require $O(N_R)$ messages, where N_R is the number of R-clusters, and $N_R < N$. Afterwards, they can connect the R-clusters with the Kruskal’s algorithm [22], which takes $O(E_R log N_R)$ time, where $E_R = N_R^2$.
- Step 3: The actuator allocation algorithm in Algorithm 8 can be computed locally in $O(Mwlogw)$ time. Then, the actuators can select their routes by exchanging $O(M)$ messages.

- Step 4: Similar to RDPL, the running time of the Appro-TSP-Tour algorithm is $O(E) = O(N^2)$.

As a result, the D-RDPL algorithm requires $O(NlogN)$ messages in total and the convergence time is $O(N)$. Overall, D-RDPL takes $O(N^2)$ computation time, given that $w \leq M < N$.

7.5 Performance Evaluation

We have conducted extensive simulations for our proposed route design algorithms with multiple actuators. The simulation settings are mainly drawn from [52], which are summarized in Table 8.1.

Table 7.2: Simulation Parameters

Network size	200m x 200m
Sensor distribution	Uniform random or Cluster-based uniform or Cluster-based non-uniform
No. of sensors (N)	100
Weight of sensors (W_j)	0.0-1.0
No. of actuators (M)	5 or 8
Speed of actuators	v
Radio range	40m
MAC layer	IEEE 802.11

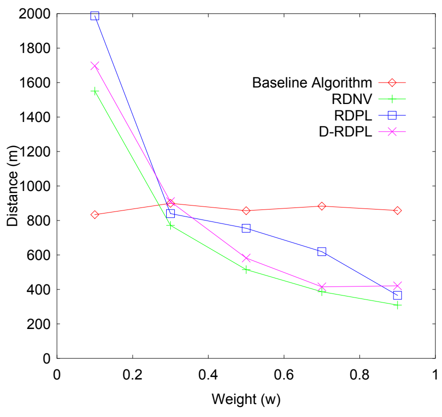
7.5.1 Average Actuator Inter-Arrival Time

In the first set of experiments, we evaluate the average actuator inter-arrival time A_{avg} with various kinds of sensor distributions.

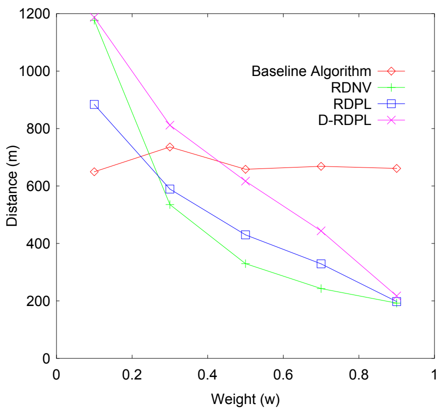
Note that the average inter-arrival distance D_{avg} is shown in our results, instead of the inter-arrival time A_{avg}. It is because the moving speed v of actuators may vary in different environments. Indeed, the average inter-arrival time of actuator A_{avg} can be readily calculated by D_{avg}/v, where v is the moving speed of the actuators. We also compare our results with the baseline algorithm, which divides the sensors into different groups according to their weights, and provides a route for each group to be walked through by an actuator.

Uniform Random Sensor Distribution

Figure 7.4(a) shows the average inter-arrival distance D_{avg} for an actuator to visit the sensors periodically under uniform random sensor distribution with N = 100 and M = 5. It evaluates distances D_{avg} to the sensors with weights in the ranges 0.0-0.2, 0.2-0.4, 0.4-0.6, 0.6-0.8, and 0.8-1.0, respectively. The result of a baseline algorithm, in which the sensors of the same weight range are walked through by the same actuator, is shown for comparison. The experimental result demonstrates that the sensors with higher weights achieve shorter inter-arrival distances D_{avg}, and hence shorter inter-arrival time A_{avg} in our algorithms. Also, all the three algorithms perform better than the baseline algorithm in terms of achieving shorter inter-arrival times. Figure 7.4(b) shows the same experiment with M=8. The average inter-arrival distances become much shorter when the number of actuator increases. It is because more sensors can be visited in the same period of time with more actuators. The lengths of the routes can be shorter as well. From the two figures, RDNV performs better than RDPL and D-RDPL with both M=5 and M=8. The reason is that RDNV provides greater flexibility in building the routes. Since the routes have similar lengths in RDNV, sensors in different weight ranges may join any routes with the least cost. On the contrary, the routes have different lengths in both RDPL and D-RDPL. The sensors with higher weights must join shorter routes, so their choices are limited.



(a)



(b)

Figure 7.4: Average inter-arrival distance under uniform random sensor distribution with N=100 (a) M=5 (b) M=8.

Cluster-Based Uniform Sensor Distribution

We next evaluate our algorithms under cluster-based and uniform sensor distribution. Sensors are deployed unevenly, which may result in clusters and network partitions. Specifically, we place the sensors into three clusters and generate the weights of sensors uniformly and randomly in this experiment.

Figures 7.5(a) and (b) show the average actuator inter-arrival distance D_{avg} with M=5 and M=8, respectively. Under a cluster-based sensor deployment, our algorithms again achieve better performance than the baseline algorithm. An interesting observation is that the D_{avg} under cluster-based sensor deployment is generally shorter than that under uniform random deployment for all the algorithms. The reason is that the sensors are more concentrated under cluster-based deployment, so they have shorter Euclidean distances and lead to the routes with shorter lengths. Similarly, RDNV usually performs better than RDPL and D-RDPL. However, the difference between its D_{avg} and that of RDPL and D-RDPL becomes smaller under cluster-based deployment, especially with M=8. It is because

RDPL and D-RDPL can form routes among the sensors within the same cluster when there are enough actuators. This greatly reduces the lengths of the routes, and hence the average inter-arrival time. The result of Algorithm 8 also provides satisfactory performance with more lightweight computation.

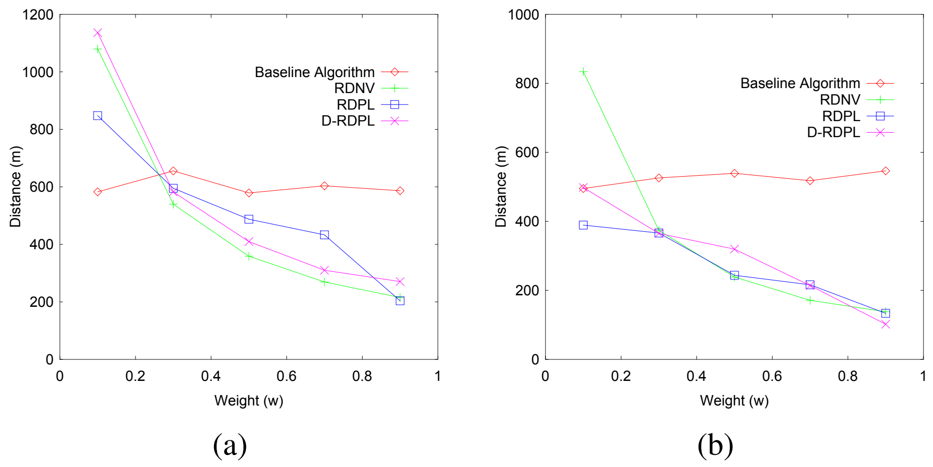


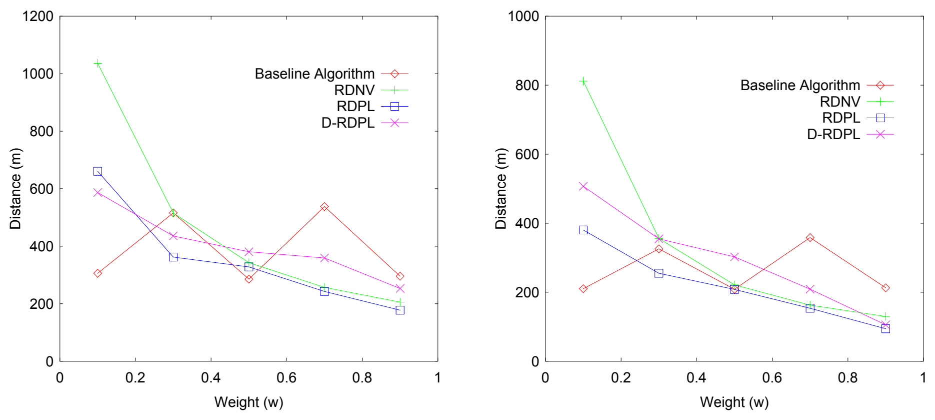
Figure 7.5: Average inter-arrival distance under cluster-based uniform sensor distribution with N=100 (a) M=5 (b) M=8.

Cluster-Based Non-Uniform Sensor Distribution

We further evaluate our algorithms under a cluster-based and non-uniform sensor distribution. Apart from deploying the sensors into three clusters, we also put the sensors with similar weights into one cluster here. The weights of the sensors in the three clusters fall in the ranges 0-0.33, 0.34-0.66, 0.67-1.0, respectively.

Similarly, Figures 7.6(a) and (b) show the results for the same network with M=5 and M=8. It is interesting that the inter-arrival distance of the baseline algorithm is not a flat line, but an “M” shape. Remember that we divided the weight of sensor into five ranges 0-0.2, 0.2-0.4, 0.4-0.6, 0.6-0.8, and 0.8-1.0 in our study. The sensors in weight range 0-0.2 are included in cluster 1. Likewise, the sensors in weight ranges 0.4-0.6 and 0.8-1.0 are included in cluster 2 and 3 separately. On the other hand, sensors in weight range 0.2-0.4 are involved in both clusters 1 and 2, and the same case in 0.6-0.8. The route formed across two clusters is naturally longer than that within only one cluster Therefore, the D_{avg} of the sensors in weight ranges 0.2-0.4 and 0.6-0.8 is longer than that in other three weight ranges.

From Figure 7.6, we observe that RDPL performs generally better than RDNV. It shows that forming routes with various lengths can reduce the actuator inter-arrival time under cluster-based and non-uniform sensor deployment. The algorithm performs especially well for the sensors in lower weight ranges. In RDNV, all routes have similar lengths and the sensors in the lowest weight range is served by only one actuator. Differently, RDPL and D-RDPL do not restrict the number of actuators serving the sensors in the lowest weight range, so multiple actuators can serve them cooperatively to reduce the inter-arrival time.



(a) (b)

Figure 7.6: Average inter-arrival distance under cluster-based non-uniform sensor distribution with N=100 (a) M=5 (b) M=8.

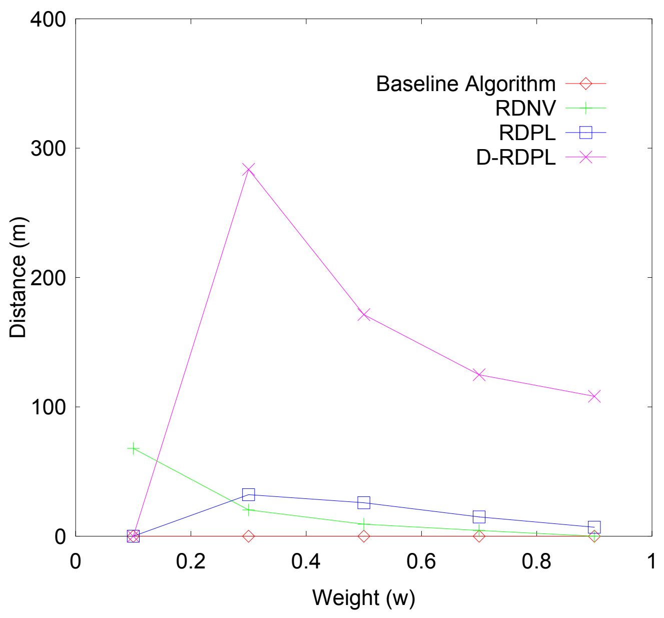


Figure 7.7: S.D. of inter-arrival distance with N=100 and M=5 under uniform random distribution.

7.5.2 Standard Deviation of Actuator Inter-Arrival Time

We have observed D_{avg} for the sensors with the same weight in the previous experiments. We are also interested in knowing whether a sensor can achieve similar D_{avg} to the other sensors with the same weight. In this experiment, the standard deviation (S.D.) of inter-arrival time among the sensors in different weight ranges will be evaluated. Again, S.D. of inter-arrival distance will be shown, instead of the time.

Figure 7.7 shows the S.D. of actuator inter-arrival distance in a network with N=100 and M=5 under uniform random sensor distribution. Note that the baseline algorithm has S.D.=0 for all weight ranges as sensors in the same weight range are included in the same route. It is clear that the S.D. in RDNV is monotonically decreasing when the weight of a sensor increases. The reason is that the sensors with lower weights are visited by fewer actuators, so the difference

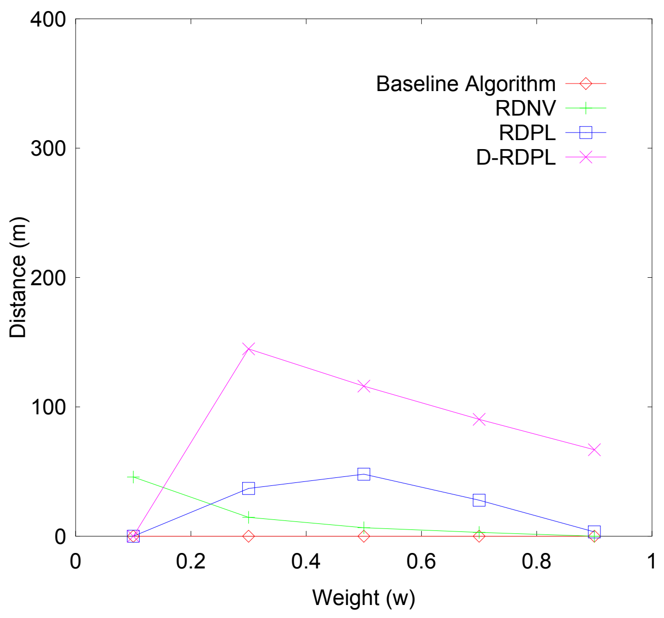


Figure 7.8: S.D. of inter-arrival distance with N=100 and M=5 under cluster-based uniform distribution.

on the route lengths may lead to more significant variations on their final inter-arrival distances D_{avg}.

RDPL achieves slightly higher S.D. than RDNV, while D-RDPL achieves much higher S.D. than both RDNV and RDPL. The S.D. of RDPL and D-RDPL is mainly due to route prolongations. Since the number of actuators is a discrete value, some routes may be assigned with excessive actuators. For example, a route which expects 1.6 actuators, may be assigned with 2 actuators. In this case, the route that is assigned with excessive actuators will prolong itself by adding more sensors. D-RDPL is relatively simple in forming routes and allocating actuators. Basically, sensors in the same weight range will be put on the same route and served by at least one actuator. The algorithm provides a differentiation of inter-arrival time relying heavily on route prolongations, thus, leading to the higher S.D. Note that both RDPL and D-RDPL obtain S.D.=0 for the sensors in the lowest weight range, as those sensors will not be affected by route

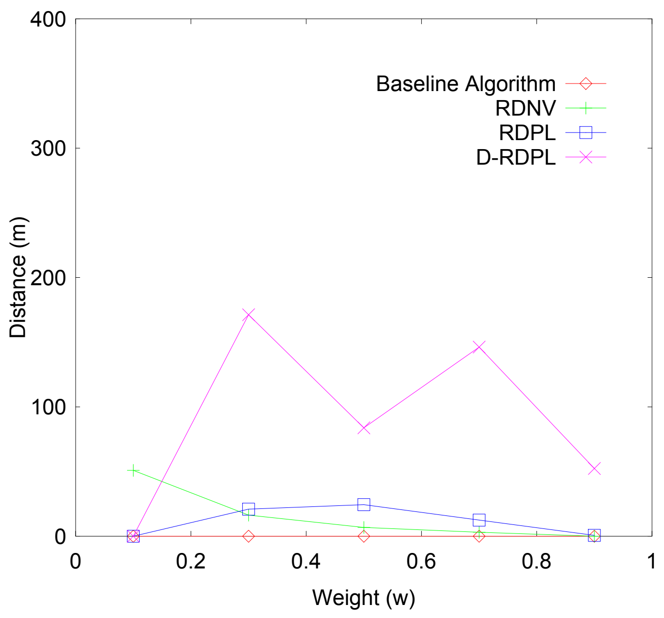


Figure 7.9: S.D. of inter-arrival distance with N=100 and M=5 under cluster-based non-uniform distribution.

prolongation.

Similarly, Figure 7.8 shows the results of the same experiment under cluster-based uniform sensor distribution. RDNV and RDPL achieve similar S.D., but D-RDPL achieves lower S.D. in comparison with that under uniform random sensor distribution. Since D-RDPL relies heavily on route prolongations, the effect of the reduced distances among the sensors becomes more obvious under cluster-based deployment.

Finally, Figure 7.9 shows the results of a network under cluster-based non-uniform sensor distribution. An interesting observation is that the S.D. in D-RDPL is not monotonically decreasing, but appears to be in "M" shape. Recall the "M" shape in the baseline algorithm, the increased Euclidean distances among the sensors also affect the S.D., so the sensors in weight ranges 0.2-0.4 and 0.6-0.8 bring higher S.D. than the others.

7.5.3 Message Overhead and Convergence Time in D-RDPL

Apart from evaluating the inter-arrival times for the proposed algorithms, we also study the communication overheads and convergence times, especially for the distributed algorithm. D-RDPL relies on the communications among the sensors and actuators when forming the R-clusters and connecting them. The number of messages and the convergence time are shown in Figure 7.10(a) and (b), respectively.

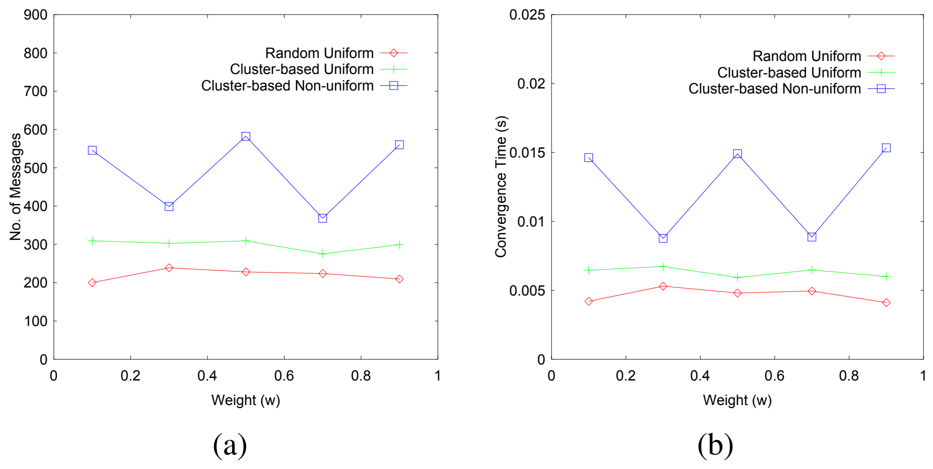


Figure 7.10: (a) Message overhead (b) Convergence time of D-RDPL with N=100 and M=5.

In Figure 7.10(a), it shows that D-RDPL achieves the fewest number of messages under uniform random sensor distribution than cluster-based sensor distribution. The sensors form an R-cluster if they are within a communication range R, so sensors under cluster-based distribution usually construct R-clusters in larger sizes. A greater set of nodes requires more messages to form an MST in a distributed algorithm. Since the message overhead is dominated by the formation of R-clusters, sensors under cluster-based distribution require more messages than those under uniform random distribution. Moreover, sensors form R-clusters only with the other sensors in the same weight range. Since the sensors in the same weight range are usually located in the same cluster under cluster-based non-uniform distribution, we can imagine that their R-clusters are more concentrated than those under cluster-based uniform distribution. Consequently, they construct larger R-clusters, and hence require more messages. Note that the message overhead appears to be “W” shape under cluster-based non-uniform distribution. The reason is that the sensors in weight ranges 0.2-0.4 and 0.6-0.8 are both separated into two clusters, so they usually form R-clusters with smaller sizes.

Figure 7.10(b) shows the convergence time in D-RDPL. The convergence time is dominated by the size of the largest R-cluster in the network as forming the greatest MST usually takes the longest time. Again, sensors under uniform random distribution achieve the shortest convergence time, while the sensors under cluster-based non-uniform distribution perform the worst. The same reasons on the size of the R-clusters apply here to explain the convergence time of D-RDPL, as well as its message overhead.

7.6 Summary

In this chapter, we focused on wireless sensor networks with multiple actuators and their route design. We demonstrated that the problem is NP-hard and proposed two effective MST-based algorithms for route design. Our algorithms aim at minimizing the overall inter-arrival time of actuators, while differentiating the visiting frequency to the sensors with different weights. The two algorithms, RDNV and RDPL, adopt two different approaches. In RDNV, sensors are visited by various number of routes with the same length. Sensors with higher weights will be visited by more routes. On the other hand, sensors are visited by routes with different lengths in RDPL. Sensors with higher weights will be visited by shorter routes. Afterwards, a distributed implementation of the route design algorithm D-RDPL is also provided as an extension of RDPL. Simulation results suggested that our algorithms can greatly reduce the average inter-arrival times in wireless sensor-actuator networks.

□ **End of chapter.**

Chapter 8

Adaptive Delay-Minimized Route Design

8.1 Overview

In WSANs, the powerful and mobile actuators can patrol along different routes and communicate with the static sensor nodes. This chapter is motivated by applications in which the objective is to minimize the data collection time in a stochastic and dynamically changing sensing environment. This is a departure from the previous static and deterministic mobile element scheduling problems.

In this chapter, we propose PROUD, a probabilistic route design algorithm for wireless sensor-actuator networks. PROUD offers delay-minimized routes for actuators and adapts well to network dynamics and sensors with non-uniform weights. This is achieved through a probabilistic visiting scheme along pre-calculated routes. We present a distributed implementation for route calculation in PROUD and extend it to accommodate actuators with variable speeds. We also propose the Multi-Route Improvement and the Task Exchange algorithms for load balancing among actuators. Simulation results demonstrate that our algorithms can effectively reduce the overall data collection time in wireless sensor-actuator networks. It well adapts to network dynamics and evenly distributes the energy consumption of the actuators.

8.2 Overview of The Route Design Problem

In this section, we describe the network model and give an overview to the route design problem.

8.2.1 Network Model

We consider a Wireless Sensor-Actuator Network (WSAN) consisting of M mobile actuators and N static sensors. Each of the sensors and actuators is equipped with a wireless transceiver. The actuators move in the sensing field along independent routes, at constant or variable speeds. Each static sensor has a limited buffer to accommodate locally sensed data. When an actuator approaches, the sensor can upload the data to the actuator and free its buffer. We also assume that the sensors have different weights, according to their data generation rates or event frequencies. Intuitively, sensors with higher weights expect shorter average actuator inter-arrival times. The weight of sensors may change dynamically according to the varying data generation rate and event frequency in the network.

8.2.2 The Route Design Problem

We consider the design of routes for multiple actuators in WSANs, and the objective is to minimize the waiting time for the sensors to upload data to the actuators. Specifically, we try to minimize the weighted average actuator inter-arrival time to sensors, which is defined as

$$Minimize \sum_{\forall i} A_i * w_i * N_i, \tag{8.1}$$

where A_i and N_i are the actuator inter-arrival time and the total number of sensors with weight w_i. We consider periodical routes, i.e., each actuator visits along a route cyclically. It is easy to show that this periodic route design minimized the expected inter-arrival time if the route is optimized.

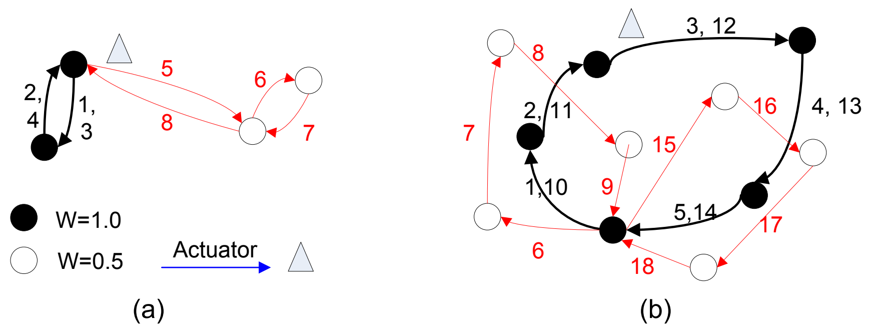


Figure 8.1: Two examples of route design.

Figure 8.1 illustrates two examples of route design in a single actuator case. The set of black nodes S_b and white nodes S_w have weights of $W_b = 1.0$ and $W_w = 0.5$ respectively. Let A_b and A_w be the actuator inter-arrival time of all black and white nodes. We expect the inter-arrival time of S_b to be half of S_w, such that $A_w = 2 * A_b$. The periodic route of the actuator is marked with numbers which indicate the visiting sequence. In this simple example, we assume that actuators move at a constant speed. In Figure 8.1(a), the actuator would visit the black nodes twice and the white nodes once every cycle. The average inter-arrival time of white nodes is thus,

$$A_w = \frac{2|TSP(S_b)| + |TSP(S_w)| + 2 * \|S_w, S_b\|}{v}, \tag{8.2}$$

where $|TSP(S_b)|$ is length of the shortest path in the traveling salesman problem (TSP) that contains the set of nodes S_b, $\|S_b, S_w\|$ is the closest distance between the set S_w and S_b, and v is the moving speed of the actuator. Note that the TSP itself is an NP-complete problem, but there are many approximation algorithms available [33].

Figure 8.1(b) shows a more complicated example with one actuator. Again, it is easy to see that the route design problem is NP-hard even in this single actuator case, so will be the multi-actuator case.

8.3 The Probabilistic Route Design (PROUD) Algorithm

We now describe the probabilistic route design (PROUD). In this algorithm, a priori route is calculated at the very beginning. Then, the sensors are visited along the route probabilistically according to their weights. For instance, sensors with a visiting probability 1.0 are visited in every cycle, while sensors with a visiting probability 0.5 only have half chance to be visited in each cycle. In this scheme, an actuator can easily update the visiting probability of the sensors based on its observed data generation rate or event frequency. In other words, the priori route can be re-used without re-calculation when adapting to network dynamics.

In the following, we first give a centralized design that depends on one of the actuators or the base station to execute the algorithm. We then extend it to a distributed implementation in the next section.

8.3.1 Small-Scale Network

Forming a Priori Route

A priori route is formed by constructing a TSP path which contains all locations to be visited. Many polynomial-time approximation algorithms have been proposed for the NP-hard TSP problem [11, 7, 22]. We adopt the well-known Approx-TSP-Tour algorithm [22] here for its low cost and bounded performance. This algorithm first creates a minimum spanning tree (MST) whose weight is a lower bound on the length of an optimal traveling-salesman tour. It then creates a tour based on the MST, and the cost of that tour is not more than twice of the optimal. The MST can be created using a polynomial-time, e.g., Prim's algorithm [22].

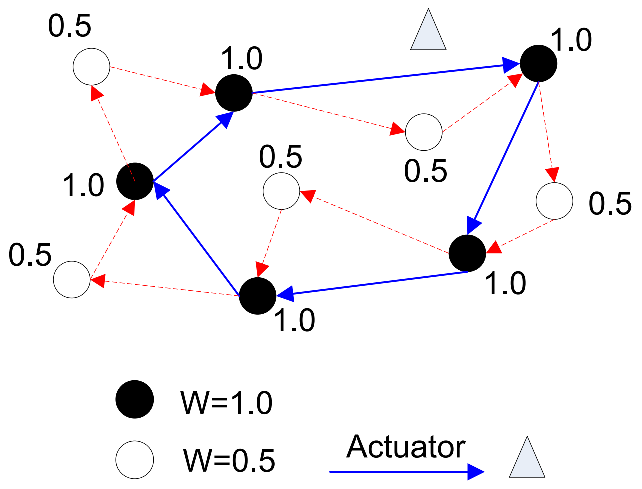


Figure 8.2: Visiting nodes probabilistically according to their weights.

Visiting Sensors Probabilistically

We then apply a probabilistic visiting model, in which actuators visit the sensors on the priori route in sequence, but selectively. Let $s_1, s_2, ..., s_i, s_{i+1}, ..., s_n$ be the sequence of sensor locations along the priori route. After visiting location s_i, the actuator determines whether to visit s_{i+1} by generating a random number between 0.0 and 1.0. If the random number is smaller than the visiting probability of s_{i+1}, i.e., p_{i+1}, then it visits s_{i+1} in the next step. If not, the actuator skips s_{i+1} and determines whether to visit the next location s_{i+2}. This process repeats in every cycle.

Intuitively, the sensors with higher weights should be assigned with a higher probability, such that they are visited more frequently. Hence, we set the visiting probability p_i of a location i to be w_i, where w_i is the (normalized) weight of the sensors. Figure 8.2 shows an example of probabilistic route design with two types of sensor nodes. The black nodes have visiting probability 1.0, which indicates that they will be visited in every cycle. On the other hand, each white node is visited only with a probability 0.5 in every cycle.

Allocating the Actuators

For a small network, the actuators can be placed evenly on the priori route during initialization. The actuators then work along the priori route and visit the sensor locations probabilistically according to the weights.

The expected route length with probabilistic visiting can be calculated as

$$E[R] = \sum_{r=0}^{n-2} \sum_{i=1}^{n} \|i, i+r\| * p_i * p_{i+r+1} \Pi_{k=1}^{r} (1 - p_{i+k}), \qquad (8.3)$$

where $1, ..., n$ is a sequence of nodes on the route R of actuator, $\|i, j\|$ is the distance between i and j, and p_i is the visiting probability of i.

For a sensor i with a visiting probability p_i, its average actuator inter-arrival time A_i is thus

$$A_i = \frac{E[R]}{p_i * v * M}, \qquad (8.4)$$

where v is the moving speed of the actuators.

In a dynamic environment, the visiting probability of the sensors can be updated according to their data generation rate or event frequency, but the route does not have to be re-calculated for each individual change.

Time Complexity Analysis:

Recall that N and M denote the number of sensors and the number of actuators, respectively. Then, the complexity for each step of the PROUD algorithm is as follow:

- Step 1: The running time of the Approx-TSP-Tour algorithm is $O(E) = O(N^2)$, since the input is a complete graph.
- Step 2: The time complexity is $O(N)$ for an actuator to select

the next locations according to the visiting probability in every cycle.

- Step 3: The time complexity of actuator allocation is $O(M)$.

In summary, the PROUD algorithm has an overall time complexity of $O(N^2 + M)$.

Bound Analysis:

Since the inter-arrival time A_i is proportional to the weights of sensors, we can focus on analyzing A_i of the locations in the lowest weight range. Let A_i and A_i* be the average actuator inter-arrival time for sensors S_i in the lowest weight range w_i in PROUD and the optimal algorithm, respectively. The optimal algorithm would visit all locations in the lowest weight range at least once in one cycle. Thus the actuator will walk on a route with a length at least $|TSP(S_i)|$. Since there are M actuators in the network, we have

$$A_i^* \geq \frac{|TSP(S_i)|}{v * M}. \tag{8.5}$$

The ratio of A_i/A_i^* is thus equal to

$$\frac{A_i}{A_i^*} \leq \frac{E[R]}{p_i * |TSP(S_i)|}. \tag{8.6}$$

8.3.2 Large-Scale Network with Partitions

In large-scale sensor networks, network partitions may happen, which divide the sensors into different clusters. In this case, actuators sharing the same route may not be as efficient as walking on distinct routes. Consider the network in Figure 8.3, route designs with two actuators walking on the same route and distinct routes are in (a) and (b) respectively. Clearly, the routes in (b) can achieve shorter inter-arrival time than (a) if

$$\|q1, q4\| + \|q2, q3\| \leq \|q1, q2\| + \|q3, q4\|. \tag{8.7}$$

This suggests that the sensor distribution should be an important consideration on route design. In particular, sensors in different clusters should be visited by actuators on independent routes to minimize the inter-arrival time.

Forming Clusters

We use a simple algorithm for clustering the sensors, as shown in Algorithm 9. It divides an MST into two sub-trees by removing its longest edge e, provided that $w(e)/w(m) \geq \delta$, where $w(e)$ is the length of edge e. By doing this, the sensors which are geographically far apart will be involved in different sub-trees, and later, distinct routes. Note that δ is set to ensure the number of clusters is smaller than the number of actuators.

Algorithm 9 Clustering the sensors

```
$Function$ Cluster($MST(S)$)
Find the edge $m$ with the median length;
Find the longest edge $e$;
**if** $w(e)/w(m) \geq \delta$ **then**
    delete edge $e$;
    Cluster($MST(S_1)$);
    Cluster($MST(S_2)$);
**end if**
```

Forming Priori Routes and Visiting Sensors Probabilistically

After clustering the sensors, the probabilistic route design algorithm can be applied in each cluster following the simple case in a small-scale network.

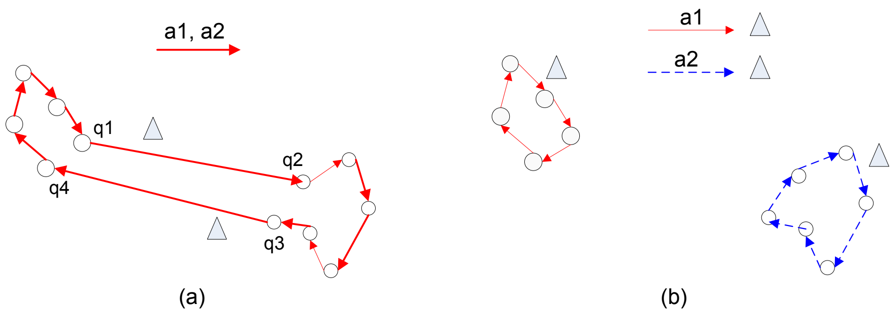


Figure 8.3: Two actuators walking on (a) the same route (b) distinct routes.

Allocating the Actuators

Multiple routes are formed from the above. They may have different expected route lengths due to the various sensor locations and visiting probabilities in the clusters. The uneven expected route lengths may cause unequal inter-arrival times for the sensors with the same weight. To address this problem we can allocate different number of actuators to the routes. Intuitively, routes with longer expected lengths should be allocated with more actuators. This is illustrated in Algorithm 10, where N_R is the total number of routes, $remain_a$ is the number of remaining unassigned actuators, and n_j is the number of actuators assigned to route R_j.

Algorithm 10 Actuator allocation for distinct routes

```
for $j = 1$ to $N_R$ do
    $n_j = 1$;
end for
$remain_a = M - N_R$;
while $remain_a > 0$ do
    Find the maximum $E[R_j^*]$;
    $E[R_j^*] = E[R_j^*] * n_j^* / (n_j^* + 1)$;
    $n_j^* + +$;
    $remain_a - -$;
end while
```

8.4 Distributed Implementation

For large network, it can be difficult for a single node to collect the information and execute the route design algorithm in a centralized manner. In this section, we present a practical distributed implementation for PROUD, in which sensors and actuators form clusters by constructing MSTs cooperatively, then the actuators construct the priori routes by traversing the MSTs independently.

Forming R-clusters

First, the sensors construct MSTs locally by communicating with their neighbors. Given the communication range of sensors is R_s, the weight of each edge e in the MST must be smaller than or equal to R_s; that is, $w(e) \leq R_s$. We refer to such an MST as an *R-Cluster*, $RC(V, E)$, which includes all the sensors that are within R_s to some sensors in $RC(V, E)$. The cost of the R-cluster is denoted by $Cost(RC)$, which is the sum of $w(e)$, $\forall e \in E$. It will be stored by the sensors in $RC(V, E)$. There are many existing distributed algorithms for forming an MST [43, 9], and we apply a fast algorithm from [36] for this purpose.

Connecting R-clusters

An R-cluster forest is formed by the sensors as above. These R-clusters can be connected together to form MSTs that contain more sensor locations. We divide the network into M sub-areas, each of which is explored by one actuator. Each actuator looks for the R-clusters in its area and connects them if they are within a certain distance, say $\|RC_1, RC_2\| \leq \delta$. Then, a new cluster is formed with cost $Cost(RC_1) + Cost(RC_2) + \|RC_1, RC_2\|$.

Similarly, the actuators also connect their R-clusters/clusters with those in their neighboring areas. Algorithm 11 shows how two actuators $A1$ and $A2$ connect their R-clusters RC_1 and RC_2, where BD is the boundary of the two corresponding areas.

Algorithm 11 Connecting the R-Clusters

```
$Function$ Connect-Cluster($RC_1(V_1, E_1), RC_2(V_2, E_2)$)
if ($\|RC_1, BD\| \leq \delta$) and ($\|RC_2, BD\| \leq \delta$) then
  Actuators $A_1$ and $A_2$ exchange locations close to $BD$;
  Find the shortest edge $e$ that connects $RC_1$ and $RC_2$;
  if $e \leq \delta$ then
    Form new cluster $\mathcal{C}_{new}(V, E)$;
    $V = V_1 \bigcup V_2$;
    $E = E_1 \bigcup E_2 \bigcup \{e\}$;
    $Cost(\mathcal{C}_{new}) = Cost(RC_1) + Cost(RC_2) + w(e)$;
  end if
end if
```

Allocating Actuators

Actuators are then allocated to the clusters, such that each cluster is assigned to at least one actuator and no clusters are unassigned. This can be achieved by running Algorithm 12 by individual actuators. Each actuator associates itself to any unassigned clusters in its area. If the associated cluster is crossing two or more areas, the actuator has to inform the actuators in those areas. It is possible that the number of clusters is greater than the number of actuators. The unassigned clusters can be connected with some assigned clusters to ensure they are served by at least one actuator. On the contrary, a remaining actuator can associate itself with a nearby cluster with the highest cost. If multiple actuators are serving one cluster, they can divide it equally and serve the sensors involved independently.

Finally, a priori route is computed by the actuator in each cluster using the Approx-TSP-Tour algorithm [22].

8.5 Enhancements to PROUD

In this section, we discuss the integration of actuators with variable speeds and also show two enhancements for load balancing among actuators.

Algorithm 12 Allocating actuators to clusters

```
$Function$ Allocate-Actuator (Actuator $A$)
if $\exists$ unassigned $\mathcal{C}_i$ in $A$'s area then
    $A$ associates with any $\mathcal{C}_i*$;
    if $\mathcal{C}_i*$ across other areas $\chi$ then
        $A$ alerts the actuators in $\chi$;
    end if
    for $\forall$ remaining unassigned $\mathcal{C}_i$ do
        Find closest assigned cluster $\mathcal{C}'$ to $\mathcal{C}_i$;
        Connect-Cluster($\mathcal{C}_i, \mathcal{C}'$);
    end for
else
    Find $\mathcal{C}*$ with the highest cost in neighboring areas;
    $A$ associates with $\mathcal{C}*$;
    Divide $\mathcal{C}*$ equally with other associated actuators;
end if
```

8.5.1 Actuators with Variable Speeds

So far we have considered actuators with constant speeds only. Actuators with variable speeds however could achieve even shorter inter-arrival time for heterogeneous networks.

Let o_i be the expected average actuator inter-arrival time for the sensors with weight w_i. For simplicity, we assume the visiting probability p_1 of the sensors with the shortest expected average actuator inter-arrival time o_1 to be 1. The visiting probability of the remaining sensors with the expected average actuator inter-arrival time, say o_i, can be calculated by o_1/o_i. The visiting probability to sensors can be updated adaptively by the actuators according to the dynamic change of the expected average actuator inter-arrival time. By adjusting the speeds of the actuators, we can ensure sensors with the same visiting probability can achieve similar inter-arrival times, even they are visited by different actuators on distinct routes.

Assume that node i on R_j has a probability p_i of being visited by actuator j every cycle. Its average actuator inter-arrival time A_i can be calculated as

$$A_i = E[R_j]/(p_i * v_j), \tag{8.8}$$

where v_j is the moving speed of actuator j.

Since A_i must be shorter than o_i,

$$v_j \geq E[R_j]/(p_i * o_i). \tag{8.9}$$

Without loss of generality, v_j can be determined easily by assuming $p_i = 1$, that is, $v_j \geq E[R_j]/o_1$.

8.5.2 Load Balancing in Route Design

For mobile actuators, since its energy consumption is also increasing with its speed [88], the unequal moving speeds might cause imbalanced energy consumption. To tackle this problem, we propose two algorithms to balance the workload of the actuators, while guaranteeing the routes designed are energy-efficient.

Multi-Route Improvement Algorithm

Since the actuator on a route with a longer expected length consumes more energy, the loads of actuators can be balanced by forming routes with identical expected lengths. A loaded actuator may assign some of its sensor locations to its neighboring actuator with the minimum expected route length.

Consider two routes R_1 and R_2 involved in multi-route improvement. Their new expected route lengths become ideal if $E[R_1'] = E[R_2'] = (E[R_1] + E[R_2])/2$. In other words, R_1 should transfer a length of $(E[R_1] - E[R_2])/2$ to R_2. Although sensor locations can be transferred one by one from $R1$ to $R2$, until the expected lengths of the two routes become equal, but this kind of node-by-node consideration is inefficient. Therefore, we provide an approximation method to find the proportion of sensor locations ξ to be transferred from MST_1 to MST_2:

$$\frac{cost(\xi)}{cost(MST_1)} = \frac{(E[R_1] - E[R_2])/2}{E[R_1]}, \tag{8.10}$$

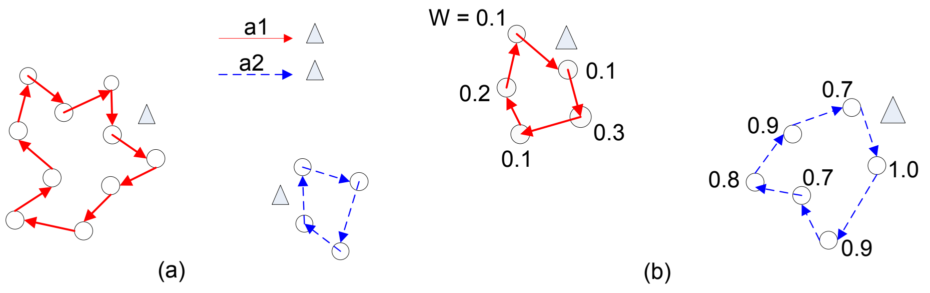


Figure 8.4: Routes involve (a) different amount of sensors (b) sensors with different weights.

where $cost(\xi)$ and $cost(MST_1)$ represent the costs of the minimum spanning trees that contain the sensor locations in ξ and R_1, respectively.

Task Exchange Algorithm

In certain scenarios, it is more energy efficient for one actuator to take up more load than another in the overall energy consumption point of view. For example, the two actuators walking on two distinct routes with unequal lengths (see Figure 8.4) achieve better performance than those on routes with identical lengths. It may happen in a network that involves clusters with different sizes or weights. It is unfavorable to enforce load-balancing by equalizing the lengths of the two routes as it will increase the lengths of both routes.

In this scenario, load balancing among the actuators can be achieved by exchanging their routes. Intuitively, an overloaded actuator may exchange its route with another actuator traveling at a lower speed. More formally, we define $Energy_{A1}$ and $Energy_{A2}$ to be the remaining energy of actuator $A1$ and $A2$, and v_1 and v_2 to be the minimum actuator speeds on routes $R1$ and $R2$. A task exchange algorithm is executed when one of the actuator $A1$ has less remaining energy than the other actuator $A2$, but it is required with a higher moving speed. As shown in Algorithm 13, tasks of the two actuators are exchanged by swapping their routes.

Algorithm 13 Task exchanges among actuators

```
if $(Energy_{A1} << Energy_{A2}) and (v1 >> v2)$ then
    $A1$ moves to $R2$;
    $A2$ moves to $R1$;
end if
```

8.6 Performance Evaluation

We have conducted extensive simulations for our proposed route design algorithms with multiple actuators. The simulation settings are summarized in Table 8.1, which are drawn from existing works [99, 52, 40].

Table 8.1: Simulation Parameters

Network size	200m x 200m
Sensor distribution	Uniform random or Cluster-based uniform or Cluster-based non-uniform
No. of sensors (N)	100
Weight of sensors (W_j)	0.0-1.0
No. of actuators	M
Speed of actuators	v
Radio range	40m
MAC layer	IEEE 802.11

8.6.1 Average Actuator Inter-Arrival Time

In the first set of experiments, we evaluate the average actuator inter-arrival time A_{avg} with various kinds of sensor distributions. Note that the average inter-arrival distance D_{avg} is shown in our results,

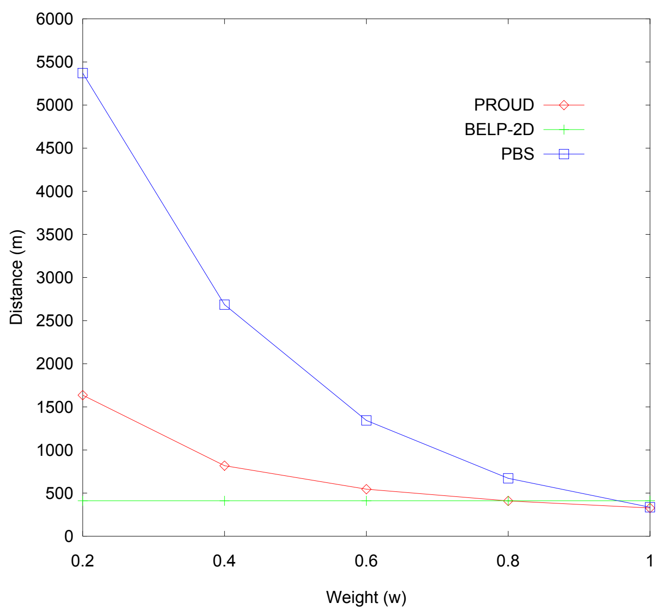


Figure 8.5: Actuator inter-arrival time under uniform random sensor distribution.

instead of the exact A_{avg}. Given the moving speed v for a specific actuator hardware, the average inter-arrival time of actuator A_{avg} can be readily calculated as D_{avg}/v.

We compare our results with that of two state-of-the-art algorithms: the partitioning based scheduling algorithm (PBS) [49] and the bounded event loss probability (BELP-2D) algorithm in the 2D case [12]. The PBS algorithm partitions all nodes into several groups (called bins) and forms a schedule that concatenates them, such that buffer overflow can be avoided in sensors with different data generation rates. The BELP-2D algorithm deals with the bounded event loss problem in a 2D space, which ensures that time elapsed between two consecutive visits is less than a critical time. It uses the solutions of the Traveling Salesman with Neighborhoods (TSPN) to find routes. To achieve a fair comparison, we adapt the Approx-TSP-Tour algorithm [22] to approximate the TSP paths in all the three algorithms.

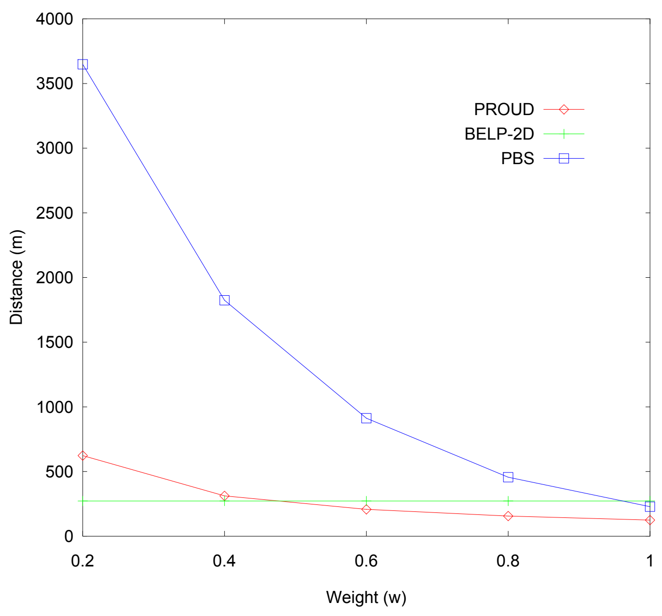


Figure 8.6: Actuator inter-arrival time under cluster-based uniform sensor distribution.

Uniform Random Sensor Distribution

Figure 8.5 shows the average inter-arrival distance D_{avg} for an actuator to visit the sensors periodically under uniform random sensor distribution with $N = 100$ and $M = 5$. It evaluates distances D_{avg} to the sensors with weights in the ranges 0.0-0.2, 0.2-0.4, 0.4-0.6, 0.6-0.8, and 0.8-1.0, respectively. The results of PROUD, PBS, and BELP-2D are shown for comparisons.

The experimental results demonstrate that PROUD, PBS, and BELP-2D have comparable inter-arrival distance D_{avg} for sensors with $w_i = 1$. Both PROUD and PBS differentiate the actuator inter-arrival times according to the weights of sensors. Sensors with higher weights achieve shorter inter-arrival distances D_{avg}, and hence shorter inter-arrival times A_{avg}. However, the D_{avg} of PBS is impractically long for most lower weighted sensors. PBS does not work well here as the locations of bins are widely spread.

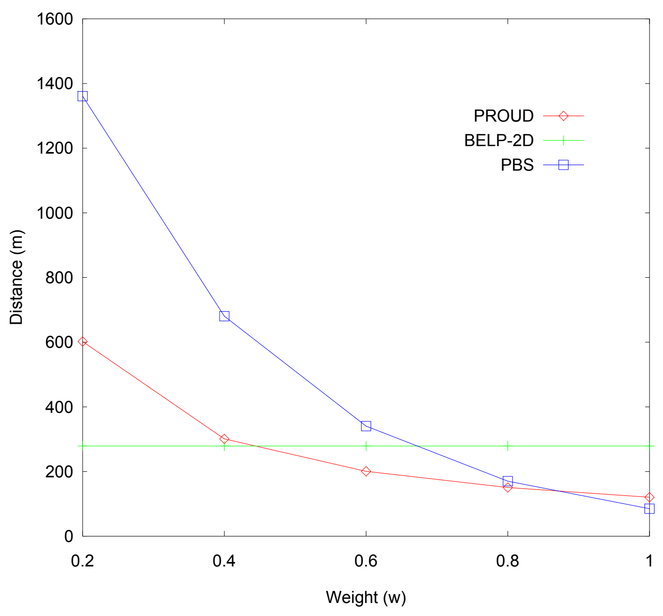


Figure 8.7: Actuator inter-arrival time under cluster-based non-uniform sensor distribution.

On the other hand, BELP-2D achieves constantly low D_{avg} for all sensors, though it does not differentiate the inter-arrival times at all. This is because the route in BELP-2D is the shortest TSP path that contains all the sensor locations. Nevertheless, PROUD is still more suitable for sensor networks with different weights. Recall that the minimum required moving speed of the actuators is determined by o_1 in both PROUD and BELP-2D. Since the A_{avg} of sensors with $w_i = 1$ in PROUD is lower than that in BELP-2D, by $A_{avg} = D_{avg}/v$, the minimum required speed of actuators in PROUD is actually lower than that in BELP-2D.

Cluster-Based Uniform Sensor Distribution

We next evaluate our algorithm under cluster-based sensor distribution. Specifically, we place the sensors into three clusters and generate the weights of sensors uniformly and randomly in this ex-

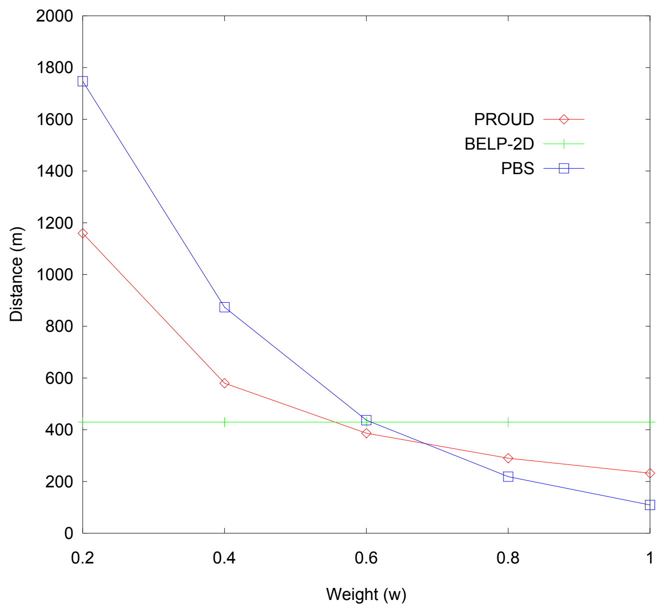


Figure 8.8: Actuator inter-arrival time under Eye Topology.

periment.

Similarly, Figure 8.6 shows the average actuator inter-arrival distance D_{avg} of the three algorithms. Under this cluster-based sensor deployment, PROUD achieves shorter D_{avg} than both BELP-2D and PBS algorithms for sensors with high and median weights. PROUD is able to differentiate the sensor visiting frequency and provide the shortest D_{avg} to highly weighted sensors, which satisfies our main objective. An interesting observation is that the D_{avg} under cluster-based sensor deployment is generally shorter than that under uniform random deployment in all the algorithms. The reason is that the sensors are more concentrated under cluster-based deployment, so they have shorter Euclidean distances, which leads to shorter routes.

Cluster-Based Non-Uniform Sensor Distribution

We further evaluate our algorithm under a cluster-based and non-uniform sensor distribution. Apart from deploying the sensors into three clusters, we also put the sensors with similar weights into one cluster here. The weights of the sensors in the three clusters fall in the ranges 0-0.33, 0.33-0.66, and 0.66-1.0, respectively.

Again, Figure 8.7 shows the results for the same network with $M = 5$. We observe that PROUD performs generally better than BELP-2D. It achieves relatively short D_{avg} for sensors with high and median weights. Again, it differentiates the D_{avg}, and hence A_{avg}, among sensors according to their weights. PROUD also achieves comparable D_{avg} with PBS for $w_i = 1$ and much lower D_{avg} for all the remaining sensors.

Overall, PROUD performs generally better than BELP-2D and PBS under various sensor and weight distributions. It always achieves shorter D_{avg} than BELP-2D for highly weighted sensors, and much shorter D_{avg} than PBS for most sensors in all cases.

8.6.2 Minimum Speed of Actuators

We investigated the average inter-arrival time (or distance) for a network that involves actuators with constant speed in the previous experiment. Now, we extend the experiment to investigate behaviors of actuators with variable speeds in PROUD. Different from the previous settings, sensors have a bound on the average actuator inter-arrival times. The aim of this experiment is to show the minimum required speeds of actuators for satisfying the required average inter-arrival times. We evaluate our algorithm in a network with $M = 8$ and $N = 100$, and set the expected average actuator inter-arrival A_i for the sensors with the highest weights $w_i = 1$ to be 2 min.

Figure 8.9 shows the minimum moving speeds of actuators in a network under uniform random sensor deployment. The eight actuators have similar minimum moving speeds as they are walking in the sub-areas where the sensor locations and weights are generated randomly. The figure also shows that the minimum speeds increase with the number of sensors. It is because the actuators need to walk on longer routes in order to visit more sensors.

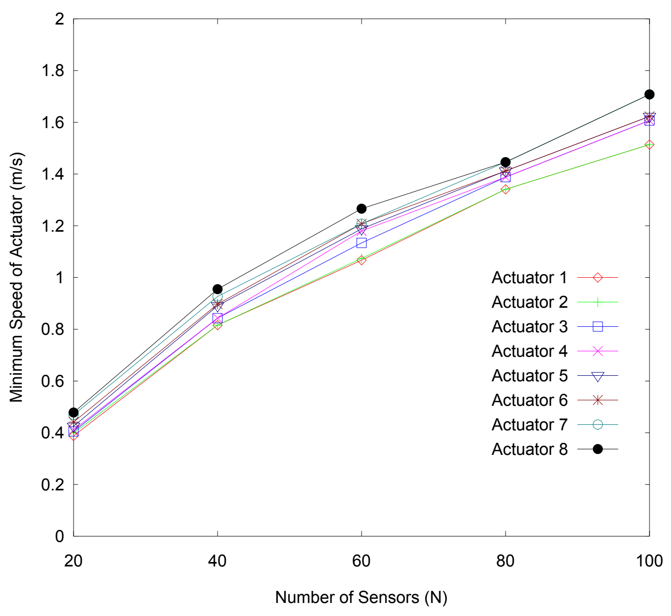


Figure 8.9: Minimum speed of actuators under under uniform random sensor distribution.

Figure 8.10 and 8.11 show the result in a network under cluster-based sensor distribution. Similar to the previous experiment, the sensors are deployed into three clusters. Again, the weights of sensors are random in Figure 8.10, while the weights in the three clusters fall in the ranges 0-0.33, 0.33-0.66, and 0.66-1.0 in Figure 8.11. The experiment results show that the required moving speeds of actuators under cluster-based sensor deployment are lower than those under uniform random deployment. The reason is that the sensors are concentrated in smaller areas, therefore, can be walked through with shorter routes. We also observe that the actuator speeds converge to three lines. The effect is especially obvious in Figure 8.11 due to its special distribution pattern of sensor weights. The three clusters are walked through by three routes with constant number of actuators on them consistently.

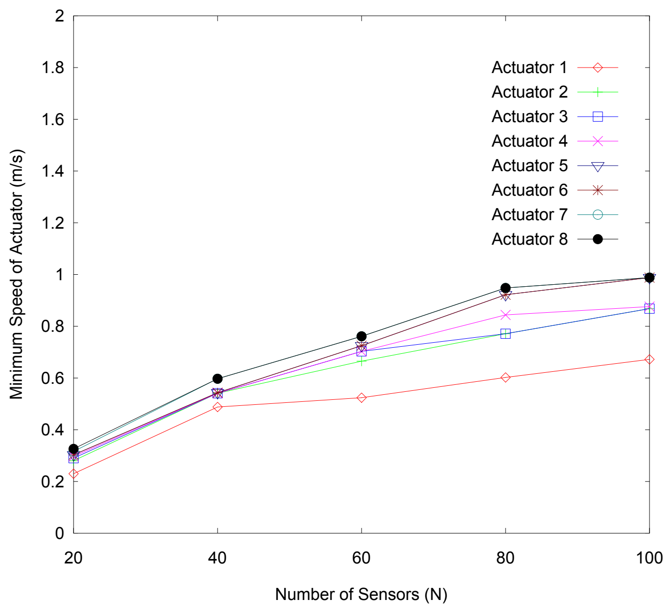


Figure 8.10: Minimum speed of actuators under cluster-based uniform sensor distribution.

8.6.3 Multi-Route Improvement

We next evaluate the performance of PROUD with multi-route improvement in this experiment. A network with 100 sensors is deployed with uniform random distribution, together with two actuators. The actuators are assigned to two sub-areas at initialization and form distinct routes separately. Since the weights of sensors change dynamically, the two actuators have to update their routes accordingly.

We let the actuators update their routes every 10 mins. The speeds of the actuators with and without multi-route improvement are compared. Figure 8.12 shows that the two actuators with multi-route improvement walk at closer speeds than that without. It is clear that the multi-route improvement balances the expected lengths of the two routes and reduces the speed difference effectively. This translates into a balanced energy consumption.

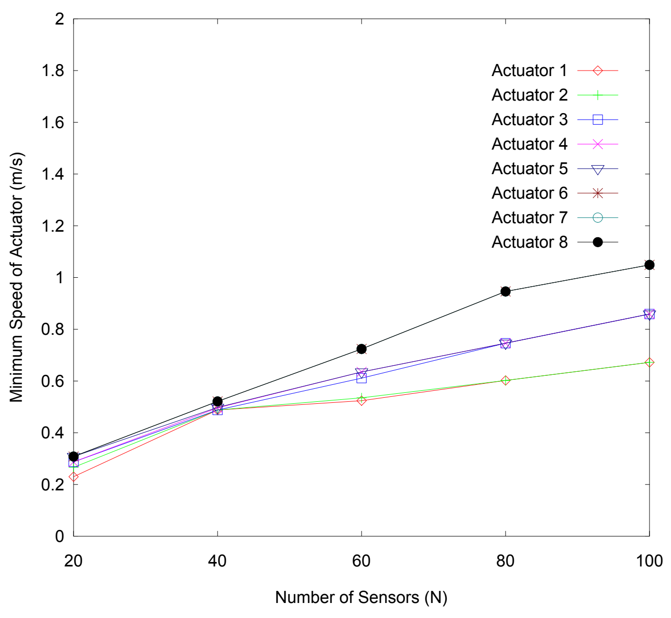


Figure 8.11: Minimum speed of actuators under cluster-based non-uniform sensor distribution.

8.6.4 Task Exchange Among Actuators

Finally, we evaluate the task exchange algorithm and focus on the energy consumptions of actuators directly. We consider a network with $M = 5$ and $N = 100$ under cluster-based distribution. Again, Clusters I, II, and III are formed, which involve sensors with low, medium, and high weights, respectively.

Figure 8.13 shows that the actuators without task exchange have significantly different energy consumptions. Particularly, Actuator 1 and Actuator 5 have a significantly different energy consumption (lower and higher, respectively) than the others. Cluster III is assigned with three actuators (Actuators 2,3,4) due to the high weights of its sensors, while Clusters I and II are both assigned with only 1 actuator. Since Cluster II has a longer expected route length than Cluster I, its actuator (Actuator 5) will move faster and consume energy more quickly. With the task exchange algorithm, Actuator 1

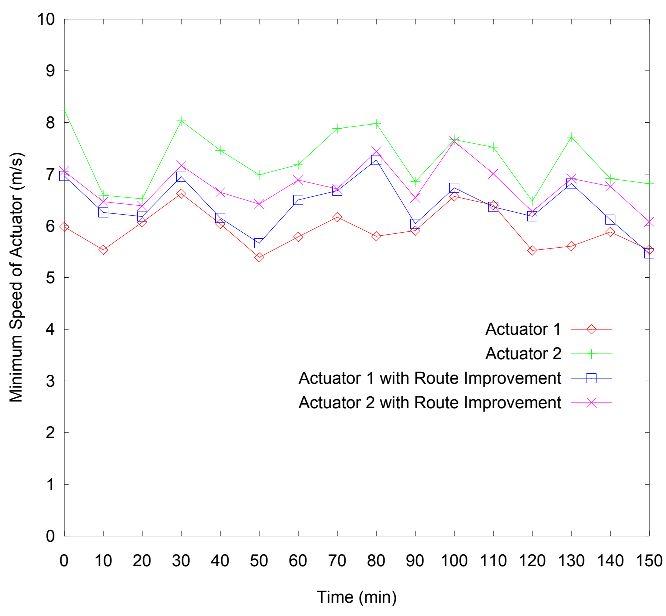


Figure 8.12: Speed of actuators with multi-route improvement.

and Actuator 5 walk on the routes of Clusters I and II interchangeably to balance their workloads, hence, achieve comparable energy consumptions.

8.7 Summary

In this chapter, we focused on wireless sensor networks with multiple actuators and their route design. We proposed an adaptive Probabilistic Route Design (PROUD) algorithm, which aims at minimizing the overall inter-arrival time of actuators with non-uniform sensor weights in a dynamically changing environment. It constitutes a significant departure from traditional static and deterministic mobile element scheduling. In PROUD, sensors are visited by actuators probabilistically along a priori route. Sensors with higher weights are visited with higher probabilities, enabling shorter actuator inter-arrival times. Most importantly, the visiting frequency to sensors

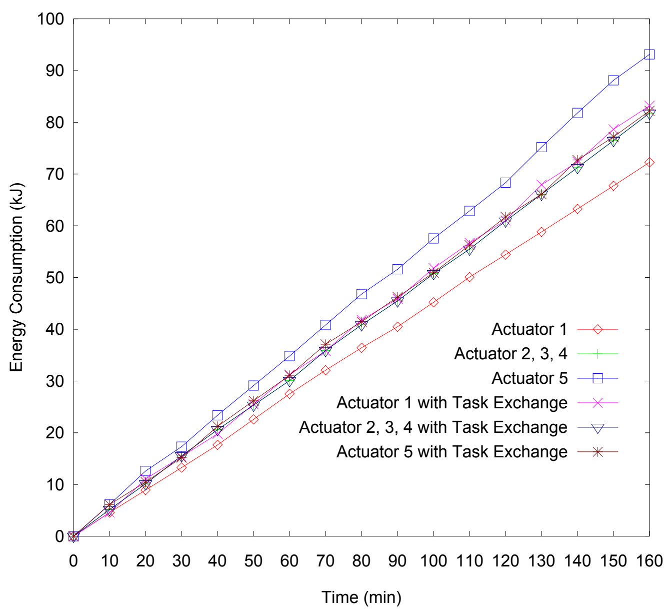


Figure 8.13: Energy consumption of actuators with task exchange.

can be updated easily by adjusting their visiting probability without complicated route re-calculations. We studied the proposed algorithm for actuators with constant velocity in both small-scale and large-scale networks. We also discussed a distributed implementation and extended the approach to accommodate actuators with variable speeds. We further proposed the Multi-Route Improvement and the Task Exchange algorithms for evenly distributing workload among the actuators. Simulation results suggested that the proposed algorithm can greatly reduce the average inter-arrival times in wireless sensor-actuator networks for highly weighted sensors. The approach also adapts well to the dynamic change of the network and effectively balances the energy consumption of the actuators.

□ **End of chapter.**

Chapter 9

Intruder Detection for Sinkhole Attack

9.1 Overview

We have disccused several aspects in delay-oriented reliable communication and coordination for WSANs. Now, we also consider the security issues in data collection for sensor networks. In particular, we propose an efficient algorithm to protect the network against a common and destruction attack on data collection.

In a WSN or WSAN, multiple nodes would send sensor readings to a base station for further processing. It is known that such a many-to-one communication is highly vulnerable to a sinkhole attack, where an intruder attracts surrounding nodes with unfaithful routing information, and then performs selective forwarding or alters the data passing through it. A sinkhole attack forms a serious threat to sensor networks, particularly considering that the sensor nodes are often deployed in open areas and of weak computation and battery power. In this chapter, we present a novel algorithm for detecting the intruder in a sinkhole attack. The algorithm first finds a list of suspected nodes through checking data consistency, and then effectively identifies the intruder in the list through analyzing the network flow information. The algorithm is also robust to deal with multiple malicious nodes that cooperatively hide the real intruder. We have evaluated the performance of the proposed algorithm through both numerical analysis and simulations, which confirmed the effectiveness and accuracy of the algorithm. Our results also suggest that its communication and computation overheads are reasonably low for wireless sensor networks.

9.2 Network Model and Problem Statement

We consider a wireless sensor network that consists of a base station (BS) and a collection of geographically distributed sensor nodes, each denoted by a unique identifier ID_v. The sensor nodes continuously collect and forward the sensed environmental data to the base station in a multi-hop fashion. As mentioned earlier, this commonly used many-to-one communication pattern is vulnerable to sinkhole attacks. In this type of attack, an intruder usually network traffic by advertising itself as having the shortest path to the base station. For example, as shown in Fig. 9.1a, an intruder, which is equipped with much higher computation and communication power than a normal sensor node, creates a high-quality single-hop link to the BS. It can then advertise imitated routing messages about the high quality route, spoofing the surrounding nodes to create a sinkhole (SH). A sinkhole can also be performed using a wormhole [78], which creates a metaphorical sinkhole with the intruder being the center; the intruder then relays the messages received in one part of the network toward the sink using a tunnel (see Fig. 9.1b).

We assume the sensor nodes are either normal or malicious. The center of a sinkhole attack is a malicious node compromised by the intruder. Note that, even if there is only one compromised node, it can affect many surrounding normal nodes by creating a high quality route to the base station. Furthermore, this intruder may also collude with some other malicious nodes. They could even collaboratively cheat the detection algorithm by suggesting a normal node as the intruder (the victim, denoted as SH').

The focus of our work is to effectively identify the real intruder

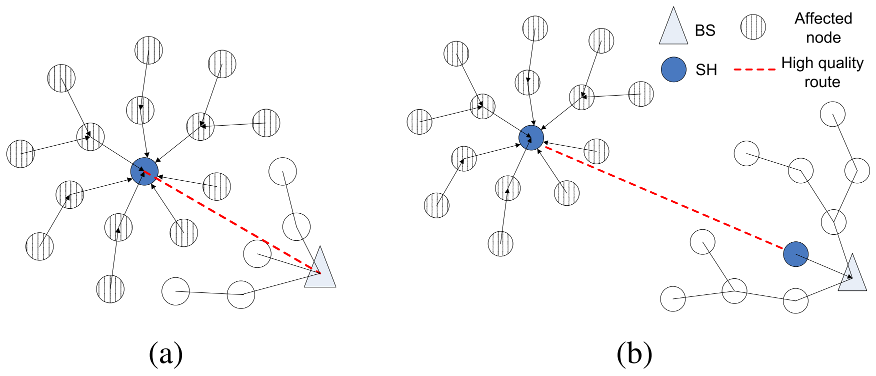


Figure 9.1: Two examples of sinkhole attack in wireless sensor networks. (a) Using an artificial high quality route; (b) Using a wormhole.

(SH) in the sinkhole attack. Once it is identified, a routing protocol or a higher-layer application can easily isolate the intruder from the network to avoid further loss. We assume that the base station is physically protected or has tamper-robust hardware [111]; hence, it acts as a central trusted authority in our algorithm design. The base station also has a rough understanding on the location of nodes, which could be available after the node deployment stage or obtained by various localization mechanisms [57]. For ease of exposition, in Table 9.1, we list the major notations used throughout this chapter.

9.3 Intruder Detection for Sinkhole Attack

We now describe our algorithm for detecting a sinkhole attack, and then efficiently identifying the intruder. We first focus on the case of a single malicious node only, i.e., the intruder (SH). We assume that, in this simple scenario, the SH will affect sensor data collection, but will not interfere the detection algorithm through dropping or altering the control messages. In the next section, we will present enhancements dealing with multiple malicious nodes that collude

Table 9.1: List of Notation

BS	Base station
SH	Real intruder in the sinkhole attack
SH'	False intruder (victim) in the sinkhole attack
ID_v	Identity of sensor node v
p	Probability of a node being malicious
d	Packet drop rate
k	No. of neighbors to which a message will be forwarded
h_{max}	Hops from the farthest node to the BS
h_{rc}	Hops from the BS where root correction takes place
l	Levels from the BS in the tree of network flow
t_l	Number of nodes at level l
N	Total number of nodes in the attacked area
F_r	Number of correct network flow information collected
F_m	Number of incorrect network flow information collected
F_s	Number of missing network flow information
F_n	Total number of network flow information collected

and even interfere the detection algorithm.

9.3.1 Estimating the Attacked Area

In a sinkhole attack, the intruder can drop, alter, or selectively forward the sensing data. The BS can suspect the existence of an attack through various statistical or application-specific data analysis. The sensors, which are closely located, are expected to have similar readings from the environment. We divide the network into a number of sub-areas and compare the data within each of them. The BS can detect the attack by finding the inconsistent data between the normal sensors and attacked sensors in the sub-areas. It can also detect the missing data from the attacked sensors and identify the attacked area. Since the area affected by the attack is limited in size, it is impossible for an intruder to alter the data from all sensors in the network. Thus, the attack must be detected in some of the sub-areas.

For illustration, consider a monitoring application in which sensor nodes submit data to the BS periodically. Let $X_1, ..., X_n$ be the sensing data collected in a sliding window, and $\bar{X}$ be their mean. Define $f(X_j)$ as,

$$f(X_j) = \sqrt{\frac{(X_j - \bar{X})^2}{\bar{X}}}.$$

A simple measure for identifying a suspected node is whether $f(X_j)$ is greater than a certain threshold, for the data from this node is noticeably inconsistent with others in the same area. More advanced statistical methods can be found in [117, 129]. After identifying a list of suspected nodes, the BS can estimate where the sinkhole locates. Specifically, it can circle a potential $attacked\ area$, which contains all the suspected nodes. An example is shown in Fig. 9.2, where the shaded nodes are found to contain missing or inconsistent data. Note that all the nodes in the circle could be attracted by the sinkhole sooner or later, and we thus refer to them as $affected\ nodes$.

9.3.2 Identifying the Intruder

Since the attacked area may contain many nodes, and the sinkhole is not necessarily the center of the area in a multi-hop sensor network, it is necessary to further locate the exact intruder and isolate it from the network. This can be achieved through analyzing the routing pattern in the affected area.

We now demonstrate a method for collecting the network flow information, which facilitates the routing pattern analysis. First, the BS sends a request message to the network. The message contains the IDs of the affected nodes, and is flooded hop by hop. For each

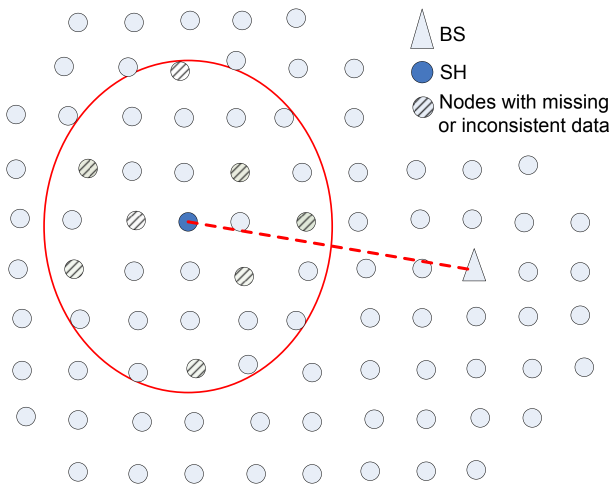


Figure 9.2: Estimate the attacked area.

node receiving the request, if its ID is there, it should respond to the BS with a message, which includes its own ID, the ID of the next-hop node, and the cost for routing, e.g, hop-count to the BS. Note that the next-hop and the cost could already be affected by the attack; hence, the response message should be transmitted along the reverse path in the flooding, which corresponds to the original route with no intruder.

At the BS, each piece of network flow information can be represented by a directed edge, $a \overrightarrow{cost} b$, where a denotes an affected node, b denotes the next hop of a, and $cost$ is the cost from a to the BS. The BS can then visualize the routing pattern by constructing a tree using the collected next hop information. Note that the area invaded by a sinkhole attack has a special routing pattern, where all network traffic flows toward the same destination, that is, the intruder SH. As shown in Fig. 9.3, once the tree is constructed, the BS can easily identify the SH, which is exactly the root of the tree in this single malicious node case.

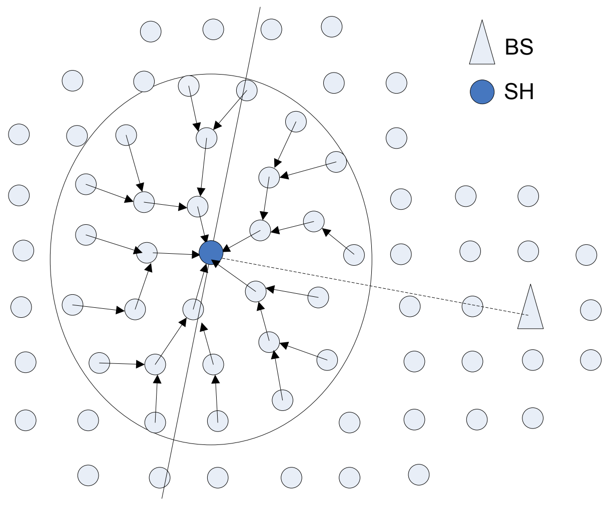


Figure 9.3: Network flow in the attacked area.

9.4 Enhancements Against Multiple Malicious Nodes

As mentioned before, there could be multiple malicious nodes that prevent the BS from obtaining correct and complete flow information for intruder detection. Specifically, they may cooperate with the intruder to perform the following misbehavior:
1. Forward the response messages selectively or even drop all (denial of service);
2. Modify the response messages passing through; and
3. Respond with false network flow information of itself.

In this section, we present effective enhancements that address these problems.

9.4.1 Dealing with Dropped Flow Information

Malicious nodes may drop the response messages of network flow information, as shown in Fig. 9.5a. To mitigate the problem, the sensors can forward the information to the BS through multiple redundant paths. Specifically, a node can forward reply messages to k neighbors, $k \geq 1$. Let p be the probability that a node is malicious, h_{max} be the number of hops from the farthest node to the BS, and the number of nodes in level l (i.e., hop count of l to the BS) be t_l. For uniformly distributed sensors, t_l can be estimated as

$$t^l = [(lR)^2\pi - (l-1)^2R^2\pi] * D = (2l-1)R^2\pi * D, \qquad (9.1)$$

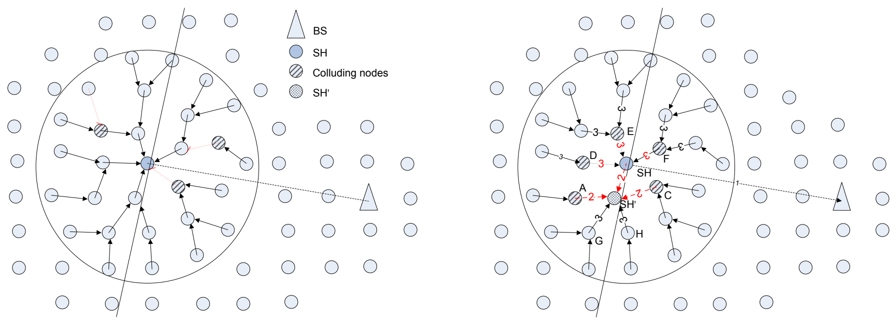
where R is the transmission range, and D is the sensor distribution density. Consider the extreme case in which a malicious node does not generate a response and drops any responses passing by, the probability that the response message from a node at level l reaches the BS is $(a-p)(a-p^k)^{l-1}$. Thus, the expected number of responses reaching the BS is

$$n = \sum_{l=1}^{h_{max}} (1-p)(a-p^k)^{l-1}t^l. \qquad (9.2)$$

As an example, for p=0.1, k=2, h_{max}=5, R=10m, D=0.01node/m^2, we have n=67.73. That is, on average, 67.73 responses will reach the BS. Such a result is reasonably good, given that the total number of normal nodes is around 78 in this setting.

Since there are still losses of the responses, the tree to be constructed based on the network flow information might be broken into several subtrees. Algorithm 14 shows a procedure to construct these trees, and the intruder is clearly in the tree with direct connection to the BS.

In some extreme cases, the malicious nodes may perform denial of service attacks. They may refuse to reply and drop all the messages passing through. However, this kind of attacks can be easily detected, as the information from the same area is completely lost.



(a) (b)

Figure 9.4: Attacked area with colluding nodes. (a) Dropping responses; (b) Providing false responses.

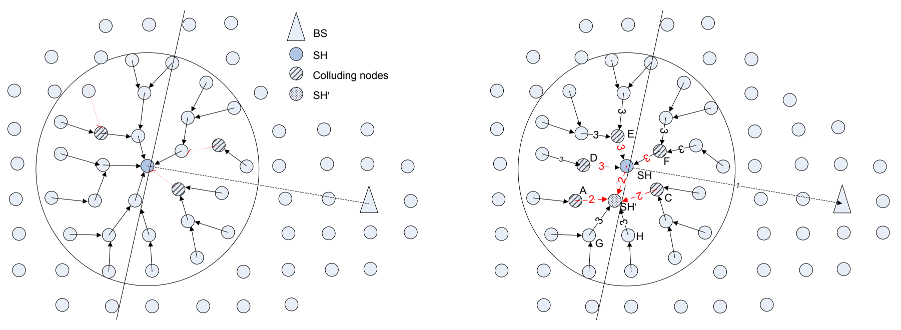
Algorithm 14 Identify multiple subtrees

```
$R = \emptyset$;
for each $v \in S$ do
    if $v$ has no incoming edge then
        $R = R \bigcup FindSubtree(v)$;
    end if
end for
$subroutine FindSubtree$(node $u$);
$R = \emptyset$;
if $u$ is not yet visited then
    mark $u$ is visited;
else
    return $\emptyset$;
end if
if $u$ has no outgoing edge then
    return $u$;
end if
for each $e(u, v)$ do
    $R' = R' \bigcup FindSubtree(v)$;
end for
return $R'$
end $FindSubtree$
```

9.4.2 Dealing with Tampered or False Flow Information

In the process for collecting the network flow information, the malicious nodes could even tamper the responses passing by or generate false responses. The receiver thus has to protect the reply message, so as to prevent an attacker from forging the network flow information. To this end, we assume every node v shares a secret key K_v with the BS, which they use in conjunction with a message authentication code (MAC) function (for example HMAC [10]) to authenticate control messages. This key can be loaded to the node through a pre-distribution protocol, e.g., that in [81]. To send a reply message R, v actually sends $< R, MAC_{K_v}(R) >$ to BS, where the notation $MAC_{K_v}(R)$ is the MAC message authentication code computed over message R with key K_v. MAC message authentication code is a short piece of information used to authenticate a message. An MAC algorithm accepts a secret key and an arbitrary-length message to be authenticated as input, and outputs an MAC (sometimes also known as a tag). Although the encryption process for generating the MAC imposes additional calculations to the sensors and the base station, the overhead is affordable with the existing lightweight symmetric encryption algorithms. Recent studies have shown that symmetric encryption and hashing function schemes can be efficiently implemented in various small sensing devices [46, 139]. The code sizes in some of these algorithms, like RC4 and RC5, are very small. They are suitable for the low-cost processors in sensors which lack large amounts of program memory. When BS receives this message, it can verify the authenticity of the message by comparing the received MAC value to the MAC that it computes for itself over the received message with K_v. More importantly, the encryption applies to the responses, whose volume is much smaller than that of the normal data traffic.

The encryption, however, cannot solve the problem that the malicious nodes themselves provide false responses to hide the real SH.



(a) (b)

Figure 9.5: Attacked area with colluding nodes. (a) Dropping responses; (b) Providing false responses.

As shown in Fig. 9.5b, two colluding nodes A and C, together with the real SH, suggest outgoing edges to a victim node SH'. To deal with this problem, the BS can detect the inconsistency among the hop count information. For instance, we can see that the incoming edges of the SH' have different number of hop counts, which is suspicious because the nodes sending messages via the same next hop should have the same hop counts to the BS. Also note that the malicious nodes D, E, and F have identical hop counts in their incoming and outgoing edges, which is again abnormal. In our algorithm, we calculate the difference between the hop counts provided by a node and the number of edges from the node to the current root. We then identify the SH and other suspicious nodes by spotting the inconsistency of the hop counts.

To this end, we maintain an array $Count$, where the i-th entry stores the total number of nodes having hop count difference i. Note that index i can be negative, which indicates that the hop count provided by a node is smaller than its actual distance from the current root. Intuitively, if $Count[0]$ is not the dominated one in the array, it means the current root is unlikely the real intruder. Through analyzing the array $Count$, we can estimate the hop counts from the SH' to the SH. For example, if most non-zero entries of Count fall in the index range [-2, 2], we suspect that the SH is two hops away from the SH', which is the current root. Given this estimation, the BS can make root correction and re-calculate the entries of Count for those nodes within two hops from the SH'. After several iterations, it can conclude the intruder if a majority of the nodes agree with a consistent result. A formal description of the above procedure can be found in Algorithm 15.

As an example, consider the attacked area in Fig. 9.6a, where the node SH' is the original root of the network flow tree, and its array $Count$ is as follow,

i	-2	**-1**	**0**	1	2
Count[i]	0	**14**	**8**	6	0

It shows that only 8 nodes agree that the SH' is the intruder. However, 14 nodes do not agree with it. In-stead, they suggest that the SH should be one hop closer to the BS than the node SH'. Since they are the majority, our correction algorithm runs again to look for a new root. After that, the node SH becomes the new root (Fig. 9.6b), and the corresponding $Count[]$ becomes,

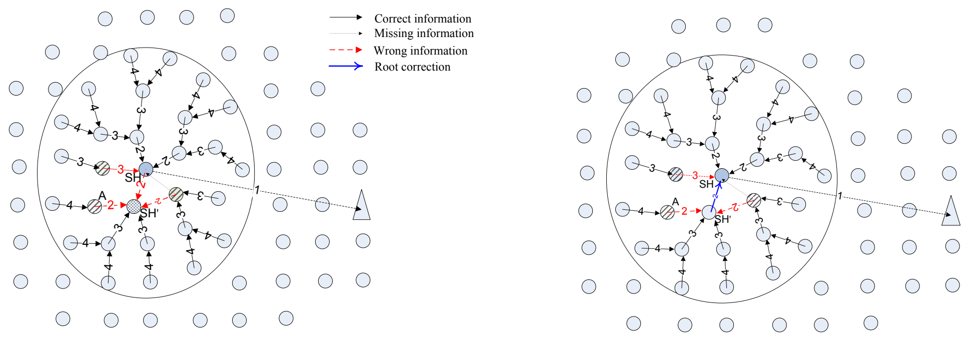
i	-2	-1	**0**	1	2
Count[i]	0	1	**21**	6	0

We can see that 21 nodes provide consistent information about the current root SH. Since the value of $Count[0]$ is the majority, the SH is concluded as the intruder.

The time complexity for calculating array $Count$ is $O(N)$, and that for correcting the roots is $\sum_{l=1}^{h_{rc}} t^l * N = O(t^{h_{rc}} * N)$. Here, h_{rc} is the average number of hops where a root correction will take place,

Algorithm 15 Find the real intruder with root corrections

```
for each root $r$ do
    initialize a new Array $count[]$;
    initialize a new Path $correctPath$;
    $checkRootByCount(r, count, 1)$;
    $S = x > 0|\forall y > 0, count[x] + count[-x] > count[y] + count[-y]$;
    $x = min(S)$;
    $correctRoot(r, r, x, 0, corretPath, count[0])$;
    apply $correctPath$ on Network $G$;
end for
$subroutinecheckRootByCount$ (Node $r$, Array $count[]$, int $depth$);
$depth = depth + 1$;
for each precedent Node $c$ of $r$ do
    increase $count[hop_count(c) - depth]$ by 1;
    $checkRootByCount(c, count, depth)$;
end for
end $checkRootByCount$;
$subroutinecorrectRoot$ (Node $r$, path $p$, int $totalLevel$, int $currentLevel$, Path $correctPath$, int $bestCount$);
if $currentLevel \geq totalLevel$ then
    return;
end if
$currentLevel = currentLevel + 1$;
for each precedent node $c$ of $r$ do
    initlaize a new array $count[]$;
    reverse $edge(c, r)$;
    if $checkRootByCount(c, count, 1)$ then
        $correctPath = p \rightarrow c$;
        $bestCount = count[0]$;
    end if
    $correctRoot(c, p \rightarrow c, totalLevel, currentLevel, correctPath, bestCount)$;
    reverse $edge(c, r)$;
end for
end $correctRoot$;
```



(a) (b)

Figure 9.6: Example of intruder identification with multiple malicious nodes.

which can be estimated from $count[]$ and is in general quite small. The total time is thus $O(N) + O(N) + O(t^{h_{rc}} * N) = O(t^{h_{rc}} * N)$, which is relatively low.

9.4.3 Proof of Correctness

We now give a simple analysis on the correctness of the above algorithm. We assume that N is the number of nodes in the attacked area, namely, $N = F_n + F_s = (F_m + F_r) + F_s$.

Property: For any F_m, our sinkhole detection algorithm works if there are more than $2F_m$ sensors reporting their network flow information successfully to the BS and at most Fm are malicious among them.

Proof: Since there are more than $2F_m$ sensors successfully reporting their network flow information, we have $F_n > 2F_m$, and consequently $F_r > F_m$. In the worst case, all the F_m malicious sensors are colluding and suggesting victim SH' as the intruder. Yet, F_r normal sensors will suggest SH, the real intruder, and the number of these nodes (F_r) is greater than the malicious nodes (F_m). Hence, our algorithm can correctly identify the real intruder through the majority vote.

Our algorithm might not work if $F_n \leq 2F_m$ and the malicious nodes are all colluding. Nevertheless, it unlikely happens in most sensor networks, where a majority of sensors should be in normal condition.

9.5 Performance Evaluation

We further evaluate the performance of our sinkhole detection algorithm through simulations. We simulate a wireless sensor network with a 200 meter by 200 meter field in which 400 nodes are placed with uniform random distribution. The sensors adopt IEEE 802.11 MAC protocol with radio range of 10 meter. A base station is placed at the center of the network to collect data from the sensors. Moreover, a sinkhole is added to the network at x- and y-coordinates (50, 50) for emulating a sinkhole attack. We are interested in evaluating the accuracy on intruder identification, communication overhead, and energy consumption of our intruder detection algorithm. Table 9.3 shows the default environment settings of our implementation, mostly adapted from [23, 26].

9.5.1 Accuracy of Intruder Identification

In the first set of experiments, we investigate the accuracy of our intruder detection algorithm for sinkhole attacks. We focus on the following three measures: 1) success rate, which is the percentage that our algorithm can correctly identify the SH; 2) false-positive rate, which is the percentage that our algorithm incorrectly identifies the SH; and 3) false-negative rate, which is the percentage that our algorithm is not able to identify any intruder but it does exist.

We first consider a mildly hostile environment in which 20% nodes are malicious. For those networks with less malicious nodes or even one (the intruder itself) only, the results are even better. We

Table 9.2: Simulation Parameters

Network size	200m x 200m
No. of sensors	400
Transmission range	10m
Location of BS	(100,100)
Location of sinkhole	(50,50)
Percentage of malicious nodes	m
Percentage of colluding nodes	m_c
Message drop rate (d)	0-80%
No. of neighbors which a message is forwarded to (k)	1-2
Packet size	30 bytes
Max. number of reply messages per packet	5

vary the ratio of colluding nodes from 0% to 20% of the total nodes in the network. When the ratio is 20%, all the malicious nodes are colluding with the intruder; for the ratios lower than 20%, a malicious node might randomly drop response messages only but would not collaborate with the intruder to provide misleading routing information, if it is not a colluding node. Fig. 9.7 shows that the success rates for dropping rates of 0, 0.2, and 0.4, respectively. For the zero dropping rate at the malicious nodes, we can see that the success rates are 100%. Also the corresponding false-positive and false-negative rates are zero, as shown in Figs. 9.8 and 9.9, respectively. This is because the number of correct routing information pieces is much more than that of the incorrect ones in this scenario, and the intruder can always be identified through routing pattern analysis and majority vote. When the drop rate increases, the success rate slightly decreases, however. Intuitively, when the malicious nodes randomly drop a lot of responses, it is possible that the correct information pieces become less than the incorrect ones, which leads to lower success rate, and, not surprisingly, higher false-positive and

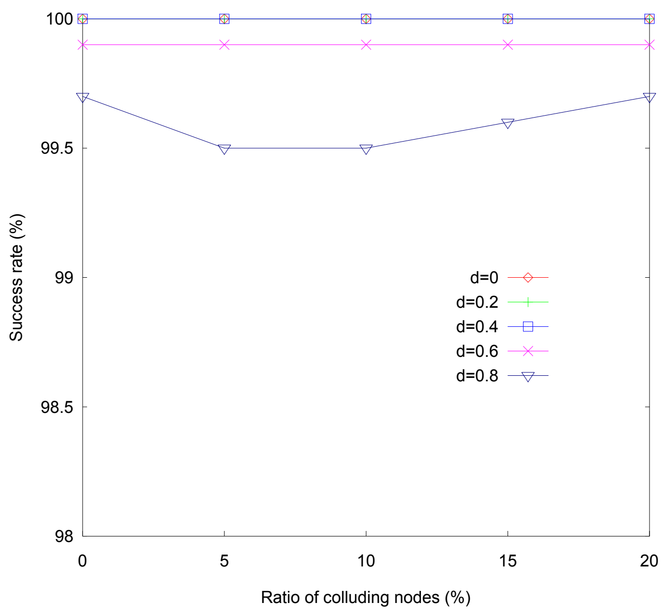


Figure 9.7: Success rate in intruder identification (m=20%).

false-negative rates. Nevertheless, the change is quite minor, even with a dropping rate of 80%. Such results suggest that our algorithm works well under a normally hostile environment.

Given the above results, it is clear that for less than 20% malicious nodes, our intrusion detection algorithm can be even more effective. We are thus more interested in its performance with other extreme hostile environments. To this end, we repeat the above experiments for m=50% and 80%, that is, more than half of the nodes are malicious. We again vary the ratio of the colluding nodes (with upper bounds of 50% and 80%, respectively). The corresponding results are shown in Figs. 9.10 through 9.15. For m=50%, the success, false-positive, and false-negative rates have similar trends as those with m=20%. Though the results are generally worse, they are still acceptable. It is worth noting that some of the curves are not monotonic, e.g., Fig. 9.12 (false-negative rates), when the ratio of colluding nodes is around 40%. This is mainly due to the random

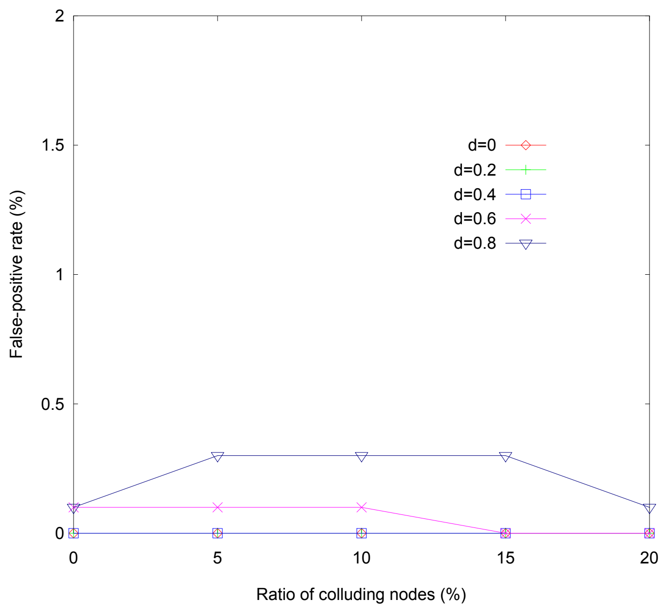


Figure 9.8: False-positive rate in intruder identification (m=20%).

drops by the malicious nodes.

For m=80%, the performance of our detection is generally unacceptable, because the malicious nodes become dominating in the network. Also note that the trends of the curves in this case are often the inverse of that in the previous two cases. For example, the success rate is the highest with a dropping rate of 0.8, and the false-negative rate decreases with increasing the ratio of colluding nodes. The reason again is because the misbehaved nodes dominate the network. Nevertheless, we do not expect any detection algorithms would work well in such an extreme environment.

9.5.2 Communication Cost

In this experiment, we evaluated the communication overhead of our algorithm. Fig. 9.16 shows the number of packets sent or received by nodes of different hops to the BS. We can see that the nodes closer to the BS have higher overheads. This is because most of

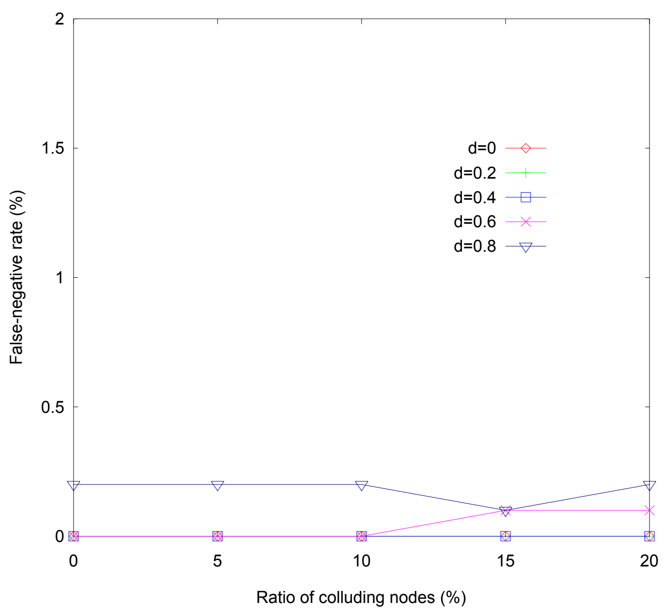


Figure 9.9: False-negative rate in intruder identification (m=20%).

the communication overhead is incurred in collecting the network flow information, and, in this process, a node closer to the BS has to relay more messages. Note that, however, the BS (0 hop) sends one request messages only, and does not have to relay response messages. The figure also shows the overhead when path redundancy is introduced for responses, where k represents the number of neighbors to which a message will be forwarded. Clearly, the larger the value of k is, the less the network flow information will be dropped by malicious nodes. Yet, our experiment shows that, when k=2, the overhead is reasonably low while a good delivering ratio can be expected. In addition, for the nodes that are far away from the attacked area (in Fig. 9.16, those of six or more hop counts), their communication overhead is almost independent of k because they do not have to respond to the request from the BS.

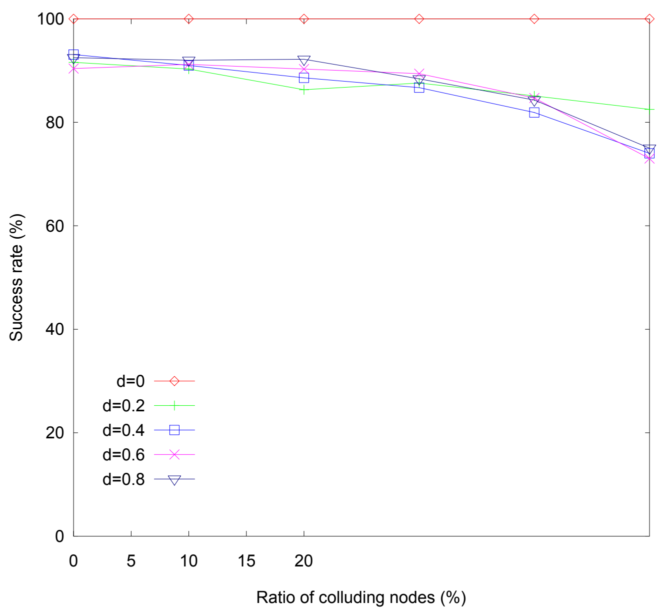


Figure 9.10: Success rate in intruder identification (m=50%).

9.5.3 Energy Consumption

Finally, we study the energy consumption for our intruder identification algorithm. As mentioned, we assume that the BS has high enough computation and battery power; hence, we mainly focus on the energy consumption in the individual sensors. It is related to both the transmission cost for request and response messages and the computation cost for encrypting messages. Table III shows a typical energy consumption for a sensor node, as adapted from [100, 53].

Table 9.3: Parameters of Energy Consumption

Communication circuit power	$5 \times 10^{-8} J/bit$
Communication antenna power	$1 \times 10^{-10} J/bit/m^2$
Encryption and MAC computation	$3 \times 10^{-9} J/bit$

Fig. 9.17 shows the average energy consumption for our algo-

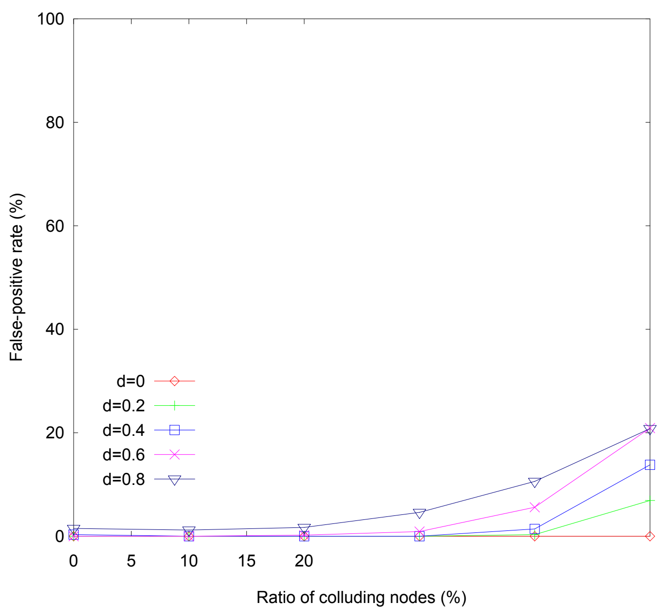


Figure 9.11: False-positive rate in intruder identification (m=50%).

rithm at each single node as a function of its hop counts to the BS. It can be seen that the consumption monotonically decreases with increasing the hop count. This is consistent with the data in Table III and the result from the previous section; basically, the communication overhead is the dominated one, and the computation part is less than 5% in the total consumption. In addition, similar to the previous experiments, applying redundant paths consumes more energy, as seen in the curve for k=2. Nonetheless, the energy consumption for our intruder detection algorithm is indeed lightweighted. For example, consider a sensor node equipped with two $3V$ 1.2 $Amp - Hour$ batteries [86, 67]. The total energy available at this node is $E_t = 7.2V * A * Hour$, which translates into $2000uJ$. Hence, the node needs to spend only a minor portion of the available energy for intruder identification throughout its lifetime.

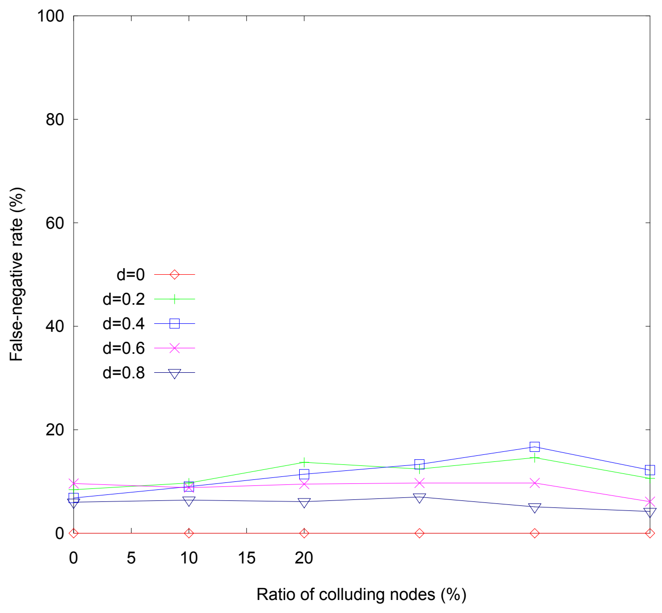


Figure 9.12: False-negative rate in intruder identification (m=50%).

9.6 Summary

In this chapter, we presented an effective method for identifying sinkhole attacks in a wireless sensor network. The algorithm consists of two steps: It first locates a list of suspected nodes by checking data consistency, and then identifies the intruder in the list through analyzing the network flow information. We also presented a series of enhancements to deal with cooperative malicious nodes that interfere the detection algorithm and attempt to hide the real intruder.

The performance of the proposed algorithm was examined through simulations. The results demonstrated the effectiveness and accuracy of the algorithm. They also suggested that its communication and computation overheads are reasonably low for wireless sensor networks. There could be many future directions toward enhancing this work; in particular, we are working on more effective statistical methods for identifying data inconsistency, which will facilitates our

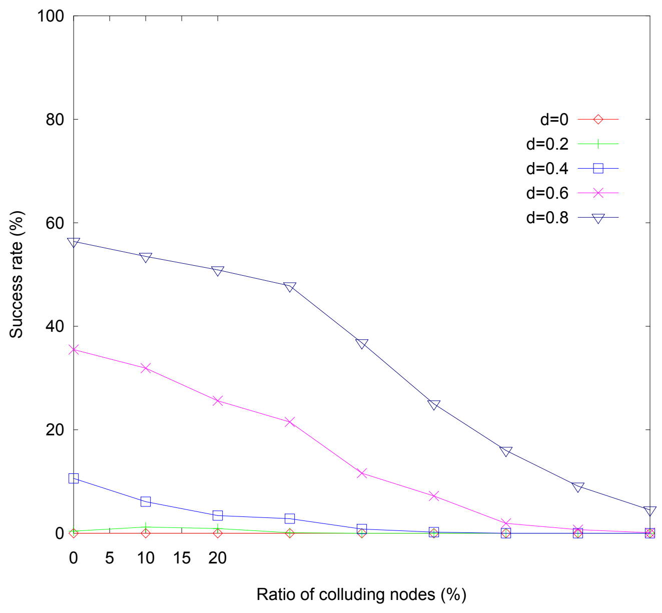


Figure 9.13: Success rate in intruder identification (m=80%).

algorithm to precisely locate the suspected nodes in sinkhole attacks.

□ **End of chapter.**

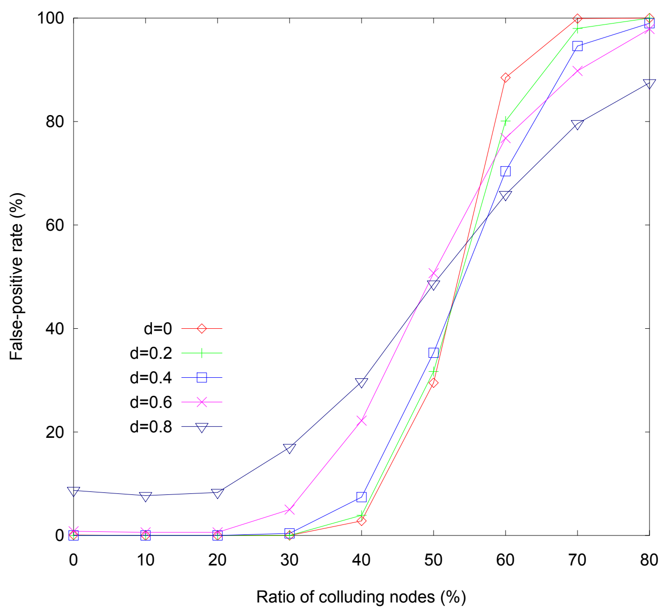


Figure 9.14: False-positive rate in intruder identification (m=80%).

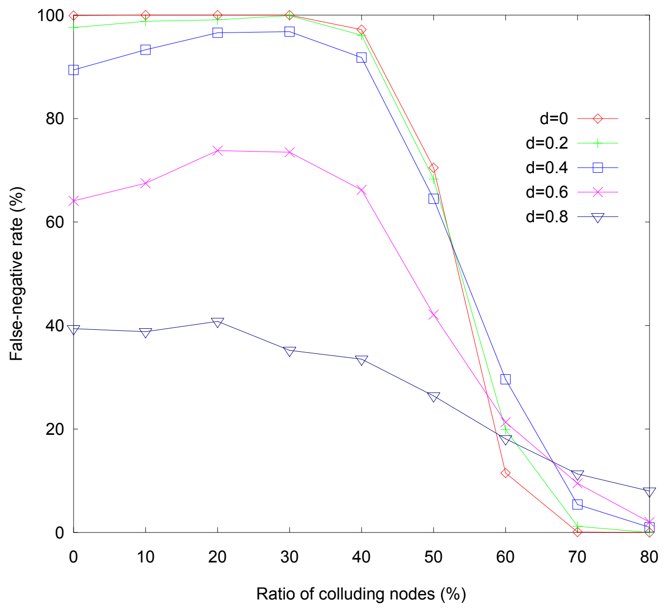


Figure 9.15: False-negative rate in intruder identification (m=80%).

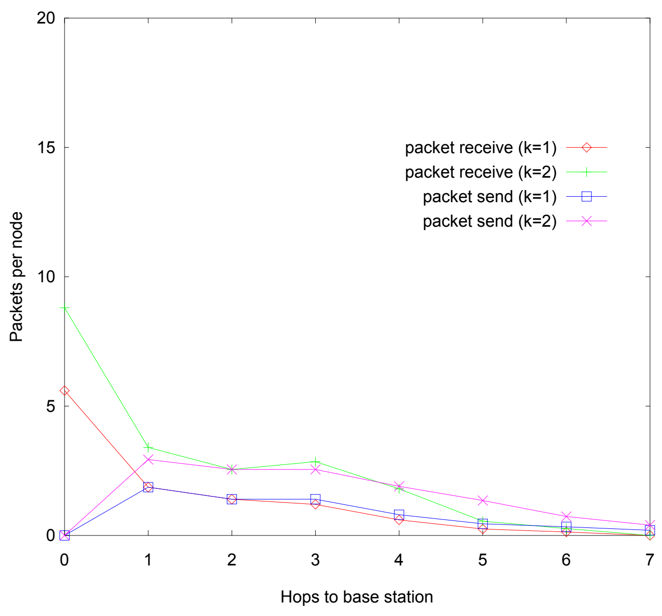


Figure 9.16: Communication cost for intrusion detection.

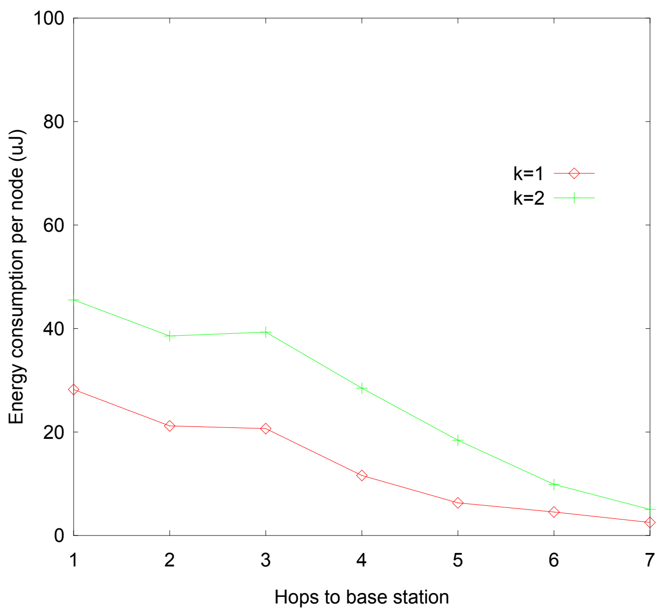


Figure 9.17: Energy consumption for intrusion identification.

Chapter 10

Conclusion

In this chapter, we investigate delay-oriented reliable communication and coordination in wireless sensor-actuator networks (WSANs). We propose a general reliability-centric framework for event reporting and data collection in a real-time system that consider the importance and freshness of the reported data. We focus on the communication from the sensors to the actuators for reporting the sensing data and the coordination among the actuators on independent routes for more efficient data collection.

Firstly, we present a real-time communication framework for wireless sensor-actuator networks. It provides an efficient event-reporting algorithm, which reduces the network traffic and minimizes the transmission delay by dividing the event area into smaller pieces of maps. The data are aggregated and further divided into different layers according to their importance. It is then transmitted to the closest actuator in the order of significance. This approach enables the actuators to start coordination without waiting for the arrival of the complete event information. Multiple actuators can combine their pieces of maps and decide on the appropriate actuator(s) to perform the actions as soon as possible. The assigned actuators will broadcast their move to the surrounding nodes, so the affected sensors can update the actuator information dynamically for future reporting. We also consider the heterogeneous characteristics and functionalities of sensors and actuators, and offer a distributed, self-organized, and comprehensive solution for real-time communications in WSANs.

A key part of this chapter is on reliable event reporting from sensors to actuators in a WSAN. We pointed out that the reliability in this context is closely related to the delay, or the freshness of the events, and they should be jointly optimized. We also suggest that the issue of non-uniform importance of the events can be explored in the optimization. Following this argument, we propose a general delay- and importance-aware event reporting framework. Our framework seamlessly integrates three key modules to maximize the reliability index: 1) A multi-level data aggregation scheme, which is fault-tolerant with error-prone sensors; 2) A priority-based transmission protocol (PREI), which accounts for both the importance and delay requirements of the events; and 3) an actuator allocation algorithm, which smartly distributes the actuators to match the demands from the sensors.

Based on reliable event reporting, we further propose a latency-oriented fault tolerant (LOFT) data transport protocol in WSANs. We provide a cross-layer two-step data transport protocol for on-time and fault tolerant data delivery from sensors to actuators. Our protocol adopts smart priority scheduling that differentiates the event data of non-uniform importance and copes with node and link failures by an adaptive replication algorithm. Apart from that, a power-controlled real-time (POWER-SPEED) data transport protocol is proposed for WSANs. POWER-SPEED achieves energy-efficiency while maintaining the QoS requirement in the timeliness domain. It sends packets that will expire later with lower transmission power to save energy. On the other hand, it sends packets that will expire sooner with higher transmitter power level in fewer hops, so it guarantees these packets can reach the destination earlier.

Another important part of this chapter is on the route design problem in wireless sensor-actuator networks. We demonstrated that the problem is NP-hard and proposed two effective MST-based algorithms for route design. Our algorithms aim at minimizing the overall inter-arrival time of actuators, while differentiating the visiting frequency to the sensors with different weights. The two algorithms, RDNV and RDPL, adopt two different approaches. In RDNV, sensors are visited by various number of routes with the same length. Sensors with higher weights will be visited by more routes. On the other hand, sensors are visited by routes with different lengths in RDPL. Sensors with higher weights will be visited by shorter routes. Afterwards, a distributed implementation of the route design algorithm D-RDPL is also provided as an extension of RDPL.

In addition, we propose an adaptive Probabilistic Route Design (PROUD) algorithm for WSANs in a dynamically changing environment. It constitutes a significant departure from traditional static and deterministic mobile element scheduling. In PROUD, sensors are visited by actuators probabilistically along a priori route. Sensors with higher weights are visited with higher probabilities, enabling shorter actuator inter-arrival times. Most importantly, the visiting frequency to sensors can be updated easily by adjusting their visiting probability without complicated route re-calculations. We studied the proposed algorithm for actuators with constant velocity in both small-scale and large-scale networks. We also discussed a distributed implementation and extended the approach to accommodate actuators with variable speeds. We further proposed the Multi-Route Improvement and the Task Exchange algorithms for evenly distributing workload among the actuators.

Finally, we study the security in wireless sensor networks to guarantee reliable collection of sensing data. More specifically, we present an effective method for identifying sinkhole attacks in a wireless sensor network. The algorithm consists of two steps: It first locates a list of suspected nodes by checking data consistency, and then identifies the intruder in the list through analyzing the network flow information. We also present a series of enhancements to deal with cooperative malicious nodes that interfere the detection algorithm and attempt to hide the real intruder. The simulation results demonstrate the effectiveness and accuracy of the algorithm with low communication and computation overheads.

In the future, communications among the actuators can be further investigated. Although the actuators have longer communication range than the sensors, this range is still limited for direct communication in a large network. Since communications among the actuators are crucial for exchanging information and coordinating the actuators to perform the required actions, more advanced actuator communication mechanisms can be explored. For instance, an actuator may spread the event information and assign tasks to other actuators. Similarly, the routes of the actuators will be updated according to the dynamic change of the sensing environment. The change of one route may also affect the others. Distributed route design, which includes more sophisticated communications and cooperation among the actuators, can be further examined. Particularly, communication points may be setup for exchanging information among the actuators on different routes.

Apart from the above, different hardware facilities and MAC layer protocols of sensors can be studied. Our proposed protocols for delay-orient event reporting can be evaluated under different hardware settings. More experiments can be conducted to understand the performance of our protocols in terms of transmission delay and energy consumption on different sensor network platforms.

Furthermore, this work may be extended to handle the existence of malicious peers in the networks. More advanced security measures can be studied to protect the networks against multiple colluding nodes.

□ **End of chapter.**

Bibliography

[1] J. Aidemark, P. Folkesson, and J. Karlsson. A framework for node-level fault tolerance in distributed real-time systems. In *Proc. of IEEE DSN*, Yokohama, Japan, Jun 28 - Jul 1, 2005.

[2] I. Akyildiz and I. Kasimoglu. A protocol suite for wireless sensor and actor networks. In *Proc. of IEEE Radio and Wireless Conference*, pages 11–14, Sep 2004.

[3] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communication Magazine*, Aug 2002.

[4] I. F. Akyildiz, W. Su, and T. Sandarasubramaniam. Wireless sensor networks: a survey. *Computer Networks*, 38(5):393–422, 2002.

[5] I. F. Akyldiz and I. Kasimoglu. Wireless sensor and actor networks: research challenges. *Elsevier Ad Hoc Networks Journal*, Octocber 2004.

[6] J. N. Al-Karaki and A. E. Kamal. Routing techniques in wireless sensor networks: a survey. *Elsevier Ad Hoc Networks Journal*, pages 325–349, 2005.

[7] S. Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *Journal of ACM*, 45(5):753–782, Sep 1998.

[8] I. Assayad, A. Girault, and H. Kalla. A bi-criteria scheduling heuristic for distributed embedded systems under reliability and real-time constraints. In *Proc. of IEEE DSN*, Florence, Italy, Jun 28 - Jul 1, 2004.

[9] B. Awerbuch. Optimal distributed algorithms for minimum weight spanning tree, counting, leader election, and related problems. In *Proc. of ACM STOC*, pages 230–240, 1987.

[10] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. *Lecture Notes in Computer Science*, 1109:1–15, 1996.

[11] J. Bentley. Fast algorithms for geometric traveling salesman problem. *ORSA Journal on Computing*, 4:387–411, 1992.

[12] N. Bisnik, A. Abouzeid, and V. Isler. Stochastic event capture using mobile sensors subject to a quality metric. In *Proc. of ACM MobiCom*, Sep 2006.

[13] D. Braginsky and D. Estri. Rumour routing algorithm for sensor networks. In *Proc. of WSNA '02*, pages 23–31, Sep 2002.

[14] N. Bulusu, J. Heidenmann, and D. Estrin. GPS-less low-cost outdoor localization for very small devices. In *IEEE Personal Communication*, Oct 2000.

[15] M. Caccamo, L. Y. Zhang, L. Sha, and G. Buttazzo. An implicit prioritized access protocol for wireless sensor networks. In *Proc. of the 23rd IEEE Real-Time Systems Symposium (RTSS)*, pages 39–48, Austin, TX, U.S., Dec 2002.

[16] E. Cayirci, T. Coplu, and O. Emiroglu. Power aware many to many routing in wireless sensor and actuator networks. In *Proc. of the 2nd European Workshop on Wireless Sensor Networks (EWSN)*, pages 236–245, Istanbul, Turkey, 31 Jan - 2 Feb 2005.

[17] A. Chakrabarti, A. Sabharwal, and B. Aazhang. Using predictable observer mobility for power efficient design fo sensor networks. In *Proc. of the 2nd International Workshop on Information Processing in Sensor Networks (IPSN)*, Apr 2003.

[18] S. Chen and K. Nahrstedt. Distributed quality-of-service routing in ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1488–1505, 1999.

[19] T. Chen, J. Tsai, and M. Gerla. Qos routing performance in multihop multimedia wireless networks. In *Proc. of the 6th IEEE International Conference on Universal Personal Communications*, pages 557–561, 1997.

[20] N. Christofides, A. Mingozzi, and P. Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20(1):255–282, Dec 1981.

[21] M. Coates. Evaluating cau.s.l relationships in wireless sensor/actor nteworks. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 937–940, Mar 2005.

[22] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and et al. *Introduction to Algorithms*. The MIT Press, 2002.

[23] B. Culpepper and H. C. Tseng. Sinkhole intrusion indicators in dsr manets. In *Proc. of BroadNets '04*, pages 681–688, Oct 2004.

[24] F. Delgosha and F. Fekri. Key pre-distribution on wireless sensor networks using multivariate polynomials. In *Proc. of SECON*, pages 118–129, Sep 2005.

[25] H. Deng, W. Li, and D. P. Agrawal. Routing security in wireless ad hoc networks. *IEEE Communications Magazine*, 40:70–75, 2002.

[26] J. Deng, R. Han, and S. Mishra. Insens: Intrusion-tolerant routing for wireless sensor networks. *Elsevier Computer Communications*, 29:216–230, 2006.

[27] D. E. Denning. An intrusion detection model. In *Proc. of IEEE Symposium on Security and Privacy*, pages 118–131, 1986.

[28] M. Ding, D. Chen, K. Xing, and X. Cheng. Localized fault-tolerant event boundary detection in sensor networks. In *Proc. of IEEE Infocom*, pages 902–913, Mar 2005.

[29] D. V. Dinh, M. D. Vuong, H. P. Nguyen, and H. X. Nguyen. Wireless sensor actor networks and routing performance analysis. In *Proc. of International Workshop on Wireless Ad-hoc Ntework*, May 2005.

[30] P. Djukic and S. Valaee. Reliable packet transmissions in multipath routed wireless networks. *IEEE Transactions on Mobile Computing*, 5(5):548–559, May 2006.

[31] S. S. Doumit and D. P. Agrawal. Self-organized critically and stochastic learning based intrusion detection system for wireless sensor networks. In *Proc. of MILCOM '03*, pages 609–614, Oct 2003.

[32] H. Dubois-Ferriere, D. Estrin, and M. Vetterli. Packet combining in sensor networks. In *Proc. of ACM Sensys*, San Diego, California, U.S., Nov 2005.

[33] A. Dumitrescu and J. S. B. Mitchell. Approximation algorithms for TSP with neighborhoods in the plane. In *Proc. of SODA'01*, pages 38–46, 2001.

[34] A. Durresi and V. Paruchuri. Geometric broadcast protocol for sensor and actor networks. In *International Conference on*

Advanced Information Networking and Applications, pages 343–348, Mar 2005.

[35] ed. D.J. Cook and S. Das. *Smart Environments: Technologies, Protocols, and Applications*. John Wiley, 2004.

[36] M. Elkin. A faster distributed protocol for constructing a minimum spanning tree. In *Proc. of ACM-SIAM SODA*, pages 359–368, 2004.

[37] S. C. Ergen and P. Varaiya. Energy efficient routing with delay guarantee for sensor networks. *ACM Wireless Networks*, 2006.

[38] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proc. of ACM MobiCom*, Seattle, Washington, U.S., 1999.

[39] K. Fall and K. Varadhan. *The ns manual*, Dec 2003. http://www.isi.edu/nsnam/ns.

[40] E. Felemban, C.-G. Lee, and E. Ekici. MMSPEED: Multipath multi-SPEED protocol for QoS guarantee of reliability and timeliness in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 5(6):738–754, 2006.

[41] E. Felemban, C.-G. Lee, E. Ekici, R. Boder, and S. Vural. Probabilistic QoS guarantee in reliability and timeliness domains in wireless sensor networks. In *Proc. of IEEE Infocom*, Miami, FL, U.S., Mar 2005.

[42] H. Frey and I. Stojmenovic. On delivery guarantees of face and combined greedy-face routing algorithms in ad hoc and sensor networks. In *Proc. of ACM MobiCom*, pages 390 – 401, Los Angeles, U.S., Sep 2006.

[43] R. G. Gallager, P. A. Humblet, and P. M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Transactions Program. Lang. Syst.*, 5(1):66–77, 1983.

[44] S. Ganeriwal, A. Kansal, and M. B. Srivastava. Self aware actuation for fault repair in sensor networks. In *IEEE International Conference on Robotics and Automation*, pages 5244–5249, Apr 2004.

[45] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *Mobile Computing and Communication Review*, 1(2), 2001.

[46] P. Ganesan, R. Venugopalan, P. Peddabachagari, A. Dean, F. Mueller, and M. Sichitiu. Analyzing and modeling encryption overhead for sensor network nodes. In *WSNA '03: Proc. of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 151–159, New York, NY, U.S., 2003. ACM Press.

[47] L. Girod, N. Ramanathan, J. Elson, T. Stathopoulos, M. Lukac, and D. Estrin. Emstar: a software environment for developing and deploying heterogeneous sensor actuator networks. *ACM Transactions on Sensor Networks*, 2007.

[48] D. K. Goldenberg, P. Bihler, Y. R. Yang, M. Cao, J. Fang, A. S. Morse, and B. D. O. Anderson. Localization in sparse networks using sweeps. In *Proc. of ACM MobiCom*, pages 110–121, 2006.

[49] Y. Gu, D. Bozdag, E. Ekici, F. Ozguner, and C.-G. Lee. Partitioning-based mobile element scheduling in wireless sensor networks. In *Proc. of SECON*, pages 386–395, Santa Clara, U.S., Sep 2005.

[50] X. Han, X. Cao, E. Lloyd, and C.-C. Shen. Fault-tolerant route relay node placement in heterogeneous wireless sensor networks. In *Proc. of IEEE Infocom*, May 2007.

[51] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher. Range-free localization schemes for large scale sensor networks. In *Proc. of ACM MobiCom*, pages 81–95, San Diego, CA, U.S., 2003.

[52] T. He, J. Stankovic, C. Lu, and T. Abdelzaher. SPEED: a real-time routing protocol for sensor networks. In *Proc. of IEEE ICDCS*, pages 46–55, Providence, RI, U.S., May 2003.

[53] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1(4), 2002.

[54] J. Hightower and G. Borriella. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, 2001.

[55] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Cullar, and K. Pister. System architecture directions for networked sensors. In *Proc. of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, Nov 2000.

[56] F. Hu and X. Cao. Security in wireless actor and sensor networks (WASN): towards a hierarchical re-keying design. In *Proc. of International Conference on Information Technology Coding and Computing (ITCC'05)*, pages 528–533, Apr 2005.

[57] L. Hu and D. Evans. Localization for mobile sensor networks. In *Proc. of ACM MobiCom*, pages 99–110, Philadelphia, PA, U.S., 26 Sep - 1 Oct 2004.

[58] W. Hu, S. Jha, and N. Bulusu. A communication paradigm for hybrid sensor/actuator networks. In *Proc. of the 15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Bacelona, Spain, Sep 2004.

[59] Y.-C. Hu, A. Perrig, and D. Johnson. Packet leashes: A defense against wormhole attacks. In *Proc. of IEEE Infocom*, pages 1976–1986, Mar 2005.

[60] Y. Huang and W. Lee. A cooperative intrusion detection system for ad hoc networks. In *Proc. of SASN '03*, pages 135–147, Oct 2003.

[61] J. Hubaux, L. Buttyan, and S. Capkun. The quest for security in mobile ad hoc networks. In *Proc. of ACM MobiHoc*, pages 146–155, Oct 2001.

[62] B. Hughes and V. Cahill. Achieving real-time guarantees in mobile ad hoc wireless networks. In *Proc. of the 24th IEEE Real-Time Systems Symposium (RTSS)*, Dec 2003.

[63] J. S. Hunter. The exponentially weighted moving average. *Journal of Quality Technology*, 18:203–210, 1986.

[64] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proc. of ACM MobiCom*, pages 56–67, Aug 2000.

[65] S. Jain, M. Demmer, R. Patra, and K. Fall. Using redundancy to cope with failures in delay tolerant network. In *Proc. of the ACM SIGCOMM*, Pennsylvania, U.S., Aug 2005.

[66] E. H. C. Jr. *Wireless Sensor Networks: Architectures and Protocols*. CRC Press, 2003.

[67] R. Jurdak, C. V. Lopes, and P. Baldi. Battery lifetime estimation and optimization for underwater sensor networks. *IEEE Sensor Network Operations*, 2004.

[68] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: Mobile networking for smart dust. In *Proc. of 5th Ann. International Conf. on Mobile Computing and Networking*, page 271V278, Aug 1999.

[69] A. Kansal, A. Somasundara, D. Jea, M. Srivastava, and D. Estrin. Intelligent fluid infrastructure for embedded networks. In *Proc. of the 2nd ACM MobiSys*, 2004.

[70] C. Karlof, N. Sastry, and D. Wagner. Tinysec: A link layer security architecture for wireless sensor networks. In *Proc. of the 2nd International Conference on Embedded Networked Sensor Systems*, pages 162–175, 2004.

[71] C. Karlof and D. Wagner. Secure routing in sensro networks: attacks and countermeasures. In *Proc. of the 1st IEEE Workshop on Sensor Network Protocols and Applications*, pages 1–15, May 2003.

[72] B. Karp and H. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proc. of ACM MobiCom*, Boston, MA, U.S., 2000.

[73] G. Khanna, S. Bagchi, and Y.-S. Wu. Fault tolerant energy aware data dissemination protocol in sensor networks. In *Proc. of IEEE DSN*, Florence, Italy, Jun 28 - Jul 1, 2004.

[74] S. Koenig, M. Lagoudakis, V. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, C. Tovey, A. Meyerson, and S. Jain. Auction-based multi-robot routing. In *Robotics: Science and Systems*, 2005.

[75] B. Krishnamachari, D. Estrin, and S. Wicker. The impact of data aggregation in wireless sensor networks. In *Proc. of International Workshop on Distributed Event-Based System*, Jul 2002.

[76] B. Krishnamachari, D. Estrin, and S. Wicker. Modelling data-centric routing in wireless sensor networks. In *Proc. of IEEE Infocom*, 2002.

[77] H. Lau, M. Sim, and K. Te. Vehicle routing problem with time windows and a limited number of vehicles. *European Journal of Operational Research*, 148:559–569, 2003.

[78] L. Lazos, R. Poovendran, C. Meadows, P. Syverson, and L. Chang. Preventing wormhole attacks on wireless ad hoc networks: a graph theoretic approach. In *Proc. of WCNC '05*, pages 1193–1199, Mar 2005.

[79] L. Lee, K. Tan, K. Ou, and Y. Chew. Vehicle capacity planning system: A case study on vehicle routing problem with time windows. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 33:169–178, 2003.

[80] J. Li, J. Jannotti, D. D. Couto, D. Karger, and R. Morris. A scalable location service for geographic ad hoc routing. In *Proc. of ACM MobiCom*, pages 120–30, Boston, Massachussets, U.S., 2000.

[81] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 52–61, New York, NY, U.S., 2003. ACM Press.

[82] C. A. Lowry, W. H. Woodall, C. W. Champ, and S. E. Rigdon. A multivariate exponentially weighted moving average chart. *Technometrics*, 34:46–53, 1992.

[83] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He. RAP: a real-time communication architecture for large-scale wireless sensor networks. In *Proc. of IEEE RTAS*, San Jose, CA, U.S., Sep 2002.

[84] J. Luo and J. Hubaux. Joint mobility and routing for lifetime elongation in wireless sensor networks. In *Proc. of the 24th IEEE Infocom*, Mar 2005.

[85] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. anderson. Wireless sensor networks for habitat monitoring. In *Proc. of ACM WSNA*, 2002.

[86] M. U. Manual. An application-specific protocol architecture for wireless microsensor networks. *Document 7430-0021-06, Rev. B*, Apr 2005.

[87] J. G. McNeff. The global positioning system. *IEEE Transactions on Microwave Theory and Techniques*, 50:645–652, Mar 2002.

[88] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. G. Lee. A case study of mobile robot's energy consumption and conservation techniques. In *Proc. of IEEE International Conference on Advanced Robotics*, pages 492–497, 2005.

[89] T. Melodia, D. Pompili, V. C. Gungor, and I. F. Akyildiz. A distributed coordination framework for wireless sensor and actor networks. In *Proc. of ACM Mobihoc*, pages 99–110, Urbana-Champaign, IL, U.S., 2005.

[90] V. P. Mhatre, C. Rosenberg, D. Kofman, R. Mazumdar, and N. Shroff. A minimum cost heterogeneous sensor network with a lifetime constraint. *IEEE Transactions on Mobile Computing*, 4(1), Jan/Feb 2005.

[91] C. Miller, A. Tucker, and R. Zemlin. Integer programming formlation of the travelling salesman problem. *Journal of the Association for computing machinery 7*, 1960.

[92] E. C.-H. Ngai, J. Liu, and M. R. Lyu. On the intruder detection for sinkhole attack in wireless sensor networks. In *Proc. of IEEE International Conference on Communications (ICC'06)*, Jun 2006.

[93] E. C.-H. Ngai, J. Liu, and M. R. Lyu. Delay-minimized route design for wireless sensor-actuator networks. In *Proc. of IEEE Wireless Communications and Networking Conference (WCNC'07)*, Mar 2007.

[94] E. C.-H. Ngai, J. Liu, and M. R. Lyu. An efficient intruder detection algorithm against sinkhole attacks in wireless sensor networks. *Computer Communications*, 2007.

[95] E. C.-H. Ngai and M. R. Lyu. An authentication service based on trust and clustering in wireless ad hoc networks: Description and security evaluation. In *Proc. of IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC'06)*, Jun 2006.

[96] E. C.-H. Ngai, M. R. Lyu, and R. T. Chin. An authentication service against dishonest users in mobile ad hoc networks. In *Proc. of IEEE Aerospace Conference*, Big Sky, Montana, U.S., Mar 2004.

[97] E. C.-H. Ngai, M. R. Lyu, and J. Liu. A real-time communication framework for wireless sensor-actuator networks. In *Proc. of IEEE Aerospace Conference*, Big Sky, Montana, U.S., Mar 2006.

[98] E. C.-H. Ngai, Y. Zhou, M. R. Lyu, and J. Liu. A delay-aware reliable event reporting framework for wireless sensor-

actuator networks. *CSE Technical Report CS-TR-2006-04, The Chinese University of Hong Kong*, Mar 2006.

[99] E. C.-H. Ngai, Y. Zhou, M. R. Lyu, and J. Liu. Reliable reporting of delay-sensitive events in wireless sensor-actuator networks. In *Proc. of the 3rd IEEE MASS*, Vancouver, Canada, Oct 2006.

[100] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler. Spins: Security protocols for sensor networks. *Wireless Network Journal*, 8:521–534, 2002.

[101] A. A. Pirzada and C. McDonald. Secure routing protocols for mobile ad-hoc wireless networks. In *T. A. Wysocki, A. Dadej, and B. J. Wysocki (Eds.), Advanced Wired and Wireless Networks*. Springer, 2004.

[102] A. A. Pirzada and C. Mcdonald. Circumventing sinkholes and wormholes in ad-hoc wireless networks. In *Proc. of International Workshop on Wireless Ad-hoc Networks*, 2005.

[103] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Communications ACM*, 43(5):551–558, 2000.

[104] S. S. Pradhan, J. Kusuma, and K. Ramchandra. Distributed compression in a dense microsensor network. *IEEE Signal Processing Magazine*, Mar 2002.

[105] C. S. Raghavendra, K. M. Sivalingam, and T. Znati. *Wireless Sensor Networks*. Kluwer Academic Publishers, 2004.

[106] T. Ralphs, L. Kopman, W. Pulleyblank, and L. Trotter. On the capacitated vehicle routing problem. *Mathematical Programming*, 94(2-3):343–359, Jan 2003.

[107] T. Rappaporrt. *Wireless Communication: Principles and Practices (2nd Edition)*. Upper Saddle River: Prentice Hall, 2002.

[108] R. K. Sahoo, A. Sivasubramaniam, M. S. Squillante, and Y. Zhang. Failure data analysis of a large-scale heterogeneous server environment. In *Proc. of IEEE DSN*, Florence, Italy, Jun 28 - Jul 1, 2004.

[109] A. Savvides, C. C. Han, and M. B. Srivastava. Dynamic fine-grained location in ad hoc networks of sensors. In *Proc. of ACM MobiCom*, pages 166–179, Philadelphia, PA, U.S., 2001.

[110] R. Shah, S. Roy, S. Jain, and W. Brunette. Data mules: Modeling a three-tier architecture for sparse sensor networks. In *Proc. of IEEE Workshop on Senosr Network Protocols and Applications (SNPA)*, 2003.

[111] E. Shi and A. Perrig. Designing secure sensor networks. *IEEE Wireless Communications*, 11:38–43, 2004.

[112] A. Silva, M. Martins, B. Rocha, A. Loureiro, L. Ruiz, and H. Wong. Decentralized intrusion detection in wireless sensor networks. In *Proc. of the 1st ACM international workshop on Quality of service and security in wireless and mobile networks*, pages 16–23, 2005.

[113] R. Sivakumar, P. Sinha, and V. Bharghavan. Cedar: Core extraction distributed ad hoc routing algorithm. *IEEE Journal on Selected Areas in Communications*, 17(8):1454–1465, 1999.

[114] A. A. Somasundara, A. Ramamoorthy, and M. B. Srivastava. Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines. In *Proc. of the 25th IEEE Real-Time Systems Symposium (RTSS)*, pages 296–305, 2004.

[115] J. Staddon, D. Balfanz, and G. Durfee. Efficient tracing of failed nodes in sensor networks. In *Proc. of WSNA '02*, pages 122–130, Sep 2002.

[116] J. A. Stankovic, T. F. Abdelzaher, C. Lu, L. Sha, and J. C. Hou. Realtime communication and coordination in embedded sensor networks. *Proceedings of The IEEE*, 91(7):1002V1022, Jul 2003.

[117] D. Wagner. Resilient aggregation in sensor networks. In *Proc. of ACM SASN*, Washington, DC, U.S., 2004.

[118] Y. Wang and H. Wu. DFT-MSN: The delay fault tolerant mobile sensor network for pervasive information gathering. In *Proc. of IEEE Infocom*, 2006.

[119] Z. M. Wang, S. Basagni, E. Melachrinoudis, and C. Petrioli. Exploiting sink mobility for maximizing sensor networks lifetime. In *Proc. of the 38th Hawaii International Conference on System Sciences (HICSS)*, 2005.

[120] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of relable multiple routing in sensor networks. In *Proc. of ACM Sensys*, Nov 2003.

[121] A. D. Wood, J. A. Standovic, and S. H. Son. Jam: A jammed-area mapping service for sensor networks. In *Proc. of the 23th IEEE Real-Time Systems Symposium (RTSS)*, pages 286–297, Dec 2003.

[122] A. D. Wood and J. A. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35:54–62, 2002.

[123] J. Wu, S. Yang, and F. Dai. Logarithmic store-carry-forward routing mobile ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, 18(6), Jun 2007.

[124] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. In *Proc. of ACM SenSys*, Nov 2002.

[125] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad-hoc routing. In *Proc. of ACM MobiCom*, pages 70–84, Rome, Italy, 2001.

[126] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh. Exploiting heterogeneity in sensor networks. In *Proc. of IEEE Infocom*, Miami, FL, U.S., Mar 2005.

[127] F. Ye, A. Chen, S. Lu, and L. Zhang. A scalable solution to minimum cost forwarding in large sensor networks. In *Proc. of ICCCN '01*, pages 304–309, Oct 2001.

[128] F. Ye, H. Luo, S. Lu, and L. Zhang. Statistical en-route filtering of injected false data in sensor networks. In *Proc. of IEEE Infocom*, pages 2446–2457, Mar 2004.

[129] N. Ye and Q. Chen. An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. *Quality and Reliability Engineering International*, 17:105–112, 2001.

[130] W. Zhang, G. Xue, and S. Mistra. Fault-tolerant route relay node placement in sensor networks: Problems and algorithms. In *Proc. of IEEE Infocom*, May 2007.

[131] Y. Zhang and W. Lee. Intrusion detection in wireless ad-hoc networks. In *Proc. of ACM MobiCom*, pages 275–283, Aug 2000.

[132] Z. Zhang and Z. Fei. Route design for multiple ferries in delay tolerant networks. In *Proc. of IEEE WCNC*, Mar 2007.

[133] F. Zhao and L. Guibas. *Wireless Sensor Networks: An Information Processing Approach.* Morgan Kaufmann, 2004.

[134] F. Zhao, J. Shin, and J. Reich. Information-driven dynamic sensor collaboration for tracking applications. *IEEE Signal Processing Magazine*, Mar 2002.

[135] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proc. of ACM Sensys*, Nov 2003.

[136] W. Zhao, M. Ammar, and E. Zegura. Controlling the mobility of multiple data transport ferries in a delay-tolerant network. In *Proc. of IEEE Infocom*, Miami, FL, U.S., Mar 2005.

[137] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *Proc. of ACM MobiHoc*, Mar 2005.

[138] Y. Zhou, E. C.-H. Ngai, M. R. Lyu, and J. Liu. Power-speed: A power-controlled real-time data transport protocol for wireless sensor-actuator networks. In *Proc. of IEEE Wireless Communications and Networking Conference (WCNC'07)*, Mar 2007.

[139] S. Zhu, S. Setia, and S. Jajodia. Leap: Efficient security mechanisms for large-scale distributed sensor networks. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 62–72, New York, NY, U.S., 2003. ACM Press.