RELIABLE AND ENERGY-EFFICIENT DATA COLLECTION IN WIRELESS SENSOR NETWORKS

by

Feng Wang

B.E., Tsinghua University, 2002

M.E., Tsinghua University, 2005

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Sciences

APPROVAL

Name:	Feng Wang
Degree:	Doctor of Philosophy
Title of Thesis:	Reliable and Energy-Efficient Data Collection in Wireless Sensor Networks
Examining Committee:	Dr. Qianping Gu Chair
	Dr. Jiangchuan Liu, Senior Supervisor
	Dr. Arthur L. Liestman, Supervisor
	Dr. Mohamed Hefeeda, SFU Examiner
	Dr. Xue Liu, External Examiner, Associate Professor of Computer Science, McGill University
Date Approved:	31 May 2012

Abstract

In recent years, wireless sensor networks (WSNs) have attracted significant attentions from both academia and industry, and are widely proposed for a broad range of applications, where data collection is often a core service to facilitate sensed data being forwarded to a central base station for further processing. Powered by batteries and using wireless communications, a WSN is more flexible than its wired counterpart. However, wireless losses/collisions may be prevalent when nodes communicate with each other. Moreover, the lifetime of a WSN largely depends on that of individual nodes, which is further constrained by the generally non-replenishable batteries. Energy efficiency thus becomes a critical concern that must be addressed.

In practice, using WSNs for data collection can be broken into three major stages, namely, deployment, message dissemination and data delivery. The deployment stage focuses on best deploying the network in the sensing field. In the message dissemination stage, network setup/management and/or collection command messages are disseminated from the base station to all sensor nodes, where the challenges lie in how to disseminate messages with low transmission costs and latencies in the presence of error-prone wireless communications. The data delivery stage fulfills the main task of data collection. Based on the information indicated by the message dissemination stage, sensed data are gathered at different nodes and delivered to the base station, with application-specified Quality-of-Service requirements to be fulfilled.

In this thesis, we tackle the design issues for each of these stages. For the deployment stage, we propose to use relay nodes for traffic relaying and suggest that the deployment should be aware of the data traffic to be collected by specific applications. We then discuss how to efficiently disseminate message in a low duty-cycle scenario, where nodes alternate between active and dormant states to conserve energy and thus extend the network lifetime. To address the traffic accumulation problem, particularly in the data delivery stage, we take a case study on high-rise structure monitoring application and propose to use elevators to facilitate data collection. Our analysis and experimental results offer systematic solutions toward reliable and energy-efficient data collection by WSNs.

To my wife and my parents

Acknowledgments

I would first like to thank my senior supervisor Prof. Jiangchuan Liu. What I learned from him is enormous. His attitude towards research has greatly influenced me. His support and encourage during all the past years are invaluable for the success of my Ph.D studies.

I want to thank Prof. Arthur L. Liestman. His support has made the road to completing this thesis smoother. Prof. Dan Wang and Dr. Yongqiang Xiong have provided constant support and encouraging during my research. The collaborations with them are precious. I also want to thank Prof. Mohamed Hefeeda and Prof. Xue Liu for serving as my thesis examiners. Their suggestions have substantially enriched this thesis.

Many colleagues of mine not only provided help in my studies but also in my everyday life. The time with them is unforgettable.

Finally, nothing would happen without your great supports, my wife and my parents. I love you all.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Wireless sensor networks (WSNs) have been applied to many applications since emerging [9, 71]. Among these applications, one common critical service is data collection, where sensed data are continuously collected at some or all of the sensor nodes and forwarded through wireless communications to a central base station for further processing. In a WSN, each sensor node is powered by a battery and uses wireless communications. This results in the small size of a sensor node and makes it easy to attach at any location with little disturbance to the surrounding environment. Such flexibility greatly eases the cost and effort for deployment and maintenance and makes wireless sensor network a competitive approach for sensor data collection compared to its wired counterpart. In fact, a wide range of such systems have been deployed in the past few years for such applications as wildlife habitat monitoring [70], environmental research [13, 64], volcano monitoring [66, 81], water monitoring [35], civil engineering [16, 34] and wildland fire forecast/detection [26].

The unique features of WSNs, however, bring many new challenges. For instance, the lifetime of a sensor node is constrained by the battery attached to it, and the network lifetime in turn depends on the lifetime of sensor nodes. Hence, to further reduce the costs of maintenance and redeployment, we often prefer to save energy in a WSN design [55]. Moreover, these challenges are complicated by wireless losses and collisions that occur when sensor nodes communicate with each other.

On the other hand, the requirements specified by sensor data collection applications raise issues that need to be considered in the network design. First of all, in some systems the deployed sensors need to cover the full area and to acquire data accurately, sensors may be required at specific locations. Also different types of data (temperature, light, vibration) may be obtained by different sensors with different sampling rates. These issues may cause unbalanced energy consumption over a WSN and significantly shorten the network lifetime if not handled carefully. In addition, since data are required to be delivered to the base station without any information loss, the data aggregation/fusion operations [59] are difficult to apply, calling for novel solutions for enhancing the network performance.

In this thesis, we will investigate a series of design issues related to wireless sensor networks for data collection, striving to identify the demands and challenges, as well as possible solutions.

1.1 Overview

1.1.1 Wireless Sensor Network

As a new type of network, WSN has many special features that do not appear in those traditional networks such as Internet, wireless mesh network and wireless mobile ad hoc network. After deployment a sensor node is expected to work for days, weeks or even years without further intervention. Since it is powered by an attached battery, energy must be conserved. This differs from Internet in which constant power sources are available. This also differs from wireless mesh and mobile ad hoc network in which the expected lifetime is several orders of magnitude lower than it is for WSN[1].

Although a sensor node is expected to work for a long time, it is often not required to work continuously, i.e., it senses ambient environment, processes and transmits the collected data; it then idles for a while until the next sensing-processing-transmitting cycle. To support fault tolerance, a location is often covered by several sensor nodes. To avoid duplicate sensing, while one node is performing the sensing-processing-transmitting cycle, other nodes are kept in the idle state. In these cases, energy consumption can be further reduced by letting the idle nodes turn to a *dormant* state in which most of the components (e.g., the wireless radio, sensing component and processing unit) in a sensor node are turned off (rather than keeping in operation as in the idle state). When the next cycle begins (indicated by some mechanism such as an internal timer), these components are set to the normal (*active*) state. *Duty-cycle* is the ratio between the active period and the full active/dormant period. A low duty-cycle WSN can expect a much longer lifetime of operation. This feature has been exploited in several systems [23, 80]. However, as will be shown later in this thesis, working in low duty-cycle also brings challenges to the network design.

Another way to conserve energy is to limit the transmission range of a sensor node. Previous

[1]For example, the expected lifetime of mobile ad hoc network is often hours or days, while that of WSN can be weeks, months or even years.

research has shown that one of the major energy costs in a sensor node comes from the wireless communication and that the cost increases with the transmission distance [27, 53]. As a result, adjustable transmission range is often preferred and dynamical adjustment may allow better performance and lower energy consumption.

1.1.2 Data Collection

In data collection service, sensors are often deployed at locations specified by the application to collect data at specific locations. The collected data are then forwarded to a central base station for further processing. Traditionally, these sensors are connected by wires which are used for both data transmission and power supply. However, the wired approach is difficult to deploy and maintain. To avoid disturbing the ambient environment, the deployment of the wires has to be carefully planned. A breakdown in any wire may disrupt the whole network and enormous time and effort may be required to find and replace a broken wire. In addition, the environment itself may make the wired deployment and its maintenance very difficult, if not impossible. For example, near a volcano [66, 81] or a wildfire scene [26], hot gases and steam can easily damage a wire. Even in a less harsh environment such as a wild habitat [13, 64, 70] or a building [16, 35, 84], the threat from rodents are still critical and make the protection of wires much more difficult than that of sensors. All these issues make wireless sensor network a preferred choice.

Although much research has been done on WSNs and quite a few prototype or preliminary systems have been deployed, sensor data collection in WSNs is still in an early stage. Its special features call for novel approaches and solutions that differ from other applications.

For example, in many other applications such as target tracking [24], sensed data or information is locally processed and stored at certain nodes and may be queried later by other nodes [29]. Sensor data collection, nevertheless, requires that all sensed data are correctly and accurately collected and forwarded to the base station, since the processing of this data needs the global knowledge and is much more complex than applications such as target tracking. This requirement also prevents using data aggregation/fusion techniques to enhance network performance. As a result, the major traffic in sensor data collection occurs when sending the reported data from each sensor to the base station. Such “many-to-one” traffic patterns, if not carefully handled, may cause a traffic accumulation problem with highly unbalanced and inefficient energy consumption across the network. As a concrete example, in the energy hole problem reported and discussed by Olariu and Stojmenovic [53], sensor nodes close to the base station are depleted quickly due to traffic relays. This creates a hole that leaves the remaining network disconnected from the base station. One possible solution to alleviate such issues is to use mobile entities that proactively move around and collect data in the sensing field [48, 65]. However, due to the harshness of the sensing environment as well as to minimize the disturbances, such a solution is often infeasible in the context of sensor data collection.

The sensors used in sensor data collection are often in greater number and of different types [13, 16, 34, 70] than in other WSNs. They may range from traditional thermometers and hygrometers to very specialized accelerometers and strain sensors. These sensors work at their own sample rates specified by the applications. The rates may differ widely from one to another. For example, a typical sampling rate of an accelerometer is $100Hz$, while the temperature is often measured in minutes. Such differences lead to different transmission rates to relay data from different type of sensors[2] which may further aggravate the imbalance of the traffic pattern and energy consumption and thus result in performance inefficiencies.

1.2 Generic Data Collection Framework for Wireless Sensor Networks

In practice, using WSNs for sensor data collection can be broken into three major stages, namely, the *deployment* stage, the *message dissemination* stage and the *data delivery* stage. Each stage has its own challenges and issues. Fig. 1.1 illustrates the three stages. The deployment stage includes deploying the WSN in the sensing field, so that the requirements from both the data collection application and the network itself are considered. To enforce connectivity as well as to prolong the network lifetime, nodes dedicated to relaying traffic may also be introduced. In the message dissemination stage, network setup/management and/or collection command messages are disseminated from the base station to all sensor nodes. Here, the challenges lie in how to disseminate messages to all sensor nodes with small transmission costs and low latencies in the presence of error-prone wireless transmissions. The data delivery stage fulfills the main task of sensor data collection. Based on the information sent in stage 2, sensed data are gathered at different sensor nodes and delivered to the base station, where different QoS requirements from the applications will dictate various gathering and delivery approaches. Stage 2 and stage 3 may alternate, so that after one round of data collection, new setup/command messages are disseminated to start a new round. The sensors may work in low duty-cycle between two collection rounds or even within stage 2 and stage 3 so as to extend

[2]Given that data aggregation/fusion is hard to apply and sensor nodes only have limited storage, sensed data are often buffered and then sent out as soon as the total size can fill a packet, which makes the transmission rate often closely related to the sample rate.

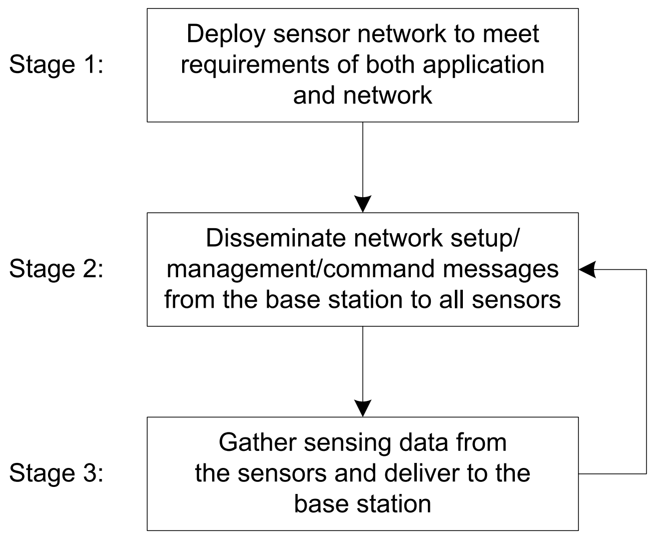

Stage 1:
Deploy sensor network to meet requirements of both application and network
Stage 2:
Disseminate network setup/ management/command messages from the base station to all sensors
Stage 3:
Gather sensing data from the sensors and deliver to the base station

Figure 1.1: Major stages of using wireless sensor networks for sensor data collection.

the network lifetime.

1.3 Related Work

A general overview was given by Akyildiz *et al.* [9] and a dedicated survey was given by Wang and Liu [74]. Here we categorize and review previous research based on the main focuses in the generic data collection framework.

1.3.1 Deployment

Many solutions have been proposed to address the problems that arise in wireless sensor network deployment. Depending on specific application requirements, different deployment strategies may be used. For sensor data collection, one typical requirement is *area-coverage* in which each location within the sensing field must be covered by at least k ($k \geq 1$) sensor nodes [49]. (Note that $k > 1$ allows for fault tolerance.) Another typical requirement is *location-coverage* in which sensor nodes must be placed at some specific locations that are chosen carefully by the applications [2, 36].

For area coverage, one solution is to use random deployment, widely adopted in other WSN applications such as target tracking [24]. An advantage of random deployment is that sensor nodes can be deployed by spraying from airplanes or simply scattering with moderate human effort. One question to be addressed here is to determine how many sensor nodes are required to achieve the necessary coverage. Given that sensor nodes are distributed by a Poisson point process, Zhang and Hou [88] derived the required density to achieve k-coverage. Later, incorporating boundary effects, Zhang and Hou [89] further proposed to deploy sensor nodes on regular grids, as grid deployment renders asymptotically lower node density than random deployment. This conclusion implies an advantage of manual deployment towards reducing equipment cost, since currently the price of a sensor node is still important. Another important issue for WSN deployment is connectivity. If the network is partitioned, an entire portion of the network becomes useless, as some sensing data can not reach the base station. If the communication range R_c is at least twice the sensing range R_s ($R_c \geq 2R_s$), then full coverage of a region also implies full connectivity [82]. Otherwise, connectivity must be explicitly considered. Assuming $R_c = R_s$, Iyengar *et al.* [30] proposed a strip-based deployment pattern to reduce the required number of sensor nodes. Bai *et al.* [10] showed that this strip-based deployment is asymptotically optimal for both full coverage and 1-connectivity when $R_c/R_s < \sqrt{3}$. In addition, Bai *et al.* [10] extended the strip-based deployment a step further by adding another vertical line of sensors and showed that this extension is optimal for both full coverage and 2-connectivity. Later, a diamond pattern was proposed by Bai *et al.* [12] and was shown to be asymptotically optimal to achieve full coverage and 4-connectivity. The optimal deployment patterns for full coverage and k-connectivity with $k \leq 6$ was finally completed by Bai *et al.* [11] who proposed a universally element pattern that unifies the previous results for 1-, 2- and 4-connectivity with their results for 3-, 5- and 6-connectivity.

Besides area coverage, another typical coverage requirement is that sensors must be manually attached to specified locations that are chosen by applications. One example is the project conducted on TsingMa Bridge in Hong Kong [36], in which the bridge is monitored by a large number of accelerometers, thermometers and strain sensors. In another recent project on the Guangzhou New TV Tower [2] in Guangzhou, China, similar sensors were further attached to the tower for real-time monitoring and analyzing. In these systems, sensors are deployed at specified locations to fulfill the civil engineering requirements. Since the locations selected by applications do not necessarily consider networking requirements such as connectivity and energy efficiency, additional relay nodes must often be placed in the sensing field to allow sensing data deliveries from sensor nodes to the base station. One question is how many relay nodes are required and where to deploy them. Cheng *et al.* [17] first modeled the relay node placement problem for connectivity as the Steiner Minimum Tree with Minimum number of Steiner Points and bounded length (SMT-MSP) problem [43] and proposed a 3-approximation algorithm. To enable fault-tolerance, a series of approximation algorithms [14, 32, 91] have been proposed to place the minimum number of relay nodes required to achieve k-connectivity for $k \geq 2$. The core idea is to compute a k-connected spanning subgraph from the full connected graph containing all sensor nodes as vertices where the edge between each pair of vertices is assigned a weight equal to the minimum number of relay nodes required to make any two neighboring nodes on the edge within each other's wireless communication range.

For data collection, basing the deployment of relay nodes solely on connectivity may not always lead to the best performance in terms of the energy efficiency and network lifetime. Motivated by this, we propose to also consider data traffic patterns during deployment design.

1.3.2 Message Dissemination

In data collection, control messages such as network setup/management or collection commands are usually needed to be sent from the base station to all sensor nodes. Although these messages are small in traffic amount, their reliable and efficient deliveries are still critical to the overall network performance. There have been numerous studies on broadcast in wired networks and in wireless ad hoc networks [18, 20, 52]. While the commonly used flooding and gossiping remain basic approaches for wireless sensor networks, substantial revisions are needed to accommodate the challenges from this new network environment [9]. For example, Smart Gossip [37] extends the basic gossip to minimize forwarding overhead. To determine the forwarding probability for each sensor node, the algorithm keeps track of previous broadcasts and adaptively adjusts the probability to match the topological properties among the sensor nodes. A timing heuristic named FDL (Forwarding-node Declaration Latency) is proposed by Guo [25] to reduce redundant message forwardings in basic flooding. The idea is to let a node defer a forwarding with a latency proportional to its residual energy. Robust Broadcast Propagation (RBP) [68] extends the flooding-based approach to achieve reliable broadcast. It first lets each node forward the received broadcast message once. Then by overhearing and explicit ACKs, a node can know the number of its neighbors that have received the message. If the number is below a threshold, the node will perform retransmissions for local repairs. Both the retransmission thresholds and the number of retries depend on the node density and topology information gathered from previous rounds of broadcast.

Trickle [38] and other research have studied code redistribution and update propagation in WSNs. Their emphasis is on the distribution of the latest version of the code in which the update frequency is much lower than that of a generic broadcast service working for many higher-level operations.

In this thesis, we consider a more realistic scenario with sensor nodes alternating between active and dormant states so as to save energy and extend the network lifetime. In this scenario, since nodes do not remain active throughout the dissemination process, previous solutions may fail or suffer from poor performance.

There have been recent studies of low duty-cycle wireless sensor networks [22, 50]. Among them, Probability-Based Broadcast Forwarding (PBBF) [50] offers an ultra-low duty-cycle MAC protocol with schedule channel polling. The MAC layer, however, considers operations of much shorter time-scales and does not involve many network-wide interactions. Dynamic Switch-based Forwarding (DSF) [22] considers data forwarding in low duty-cycles, but focuses on unicast from a data source to a sink. Our research complements these solutions by providing network-wide message dissemination, calling for novel solutions to ensure that messages are successfully delivered to all destinations.

1.3.3 Data Delivery

The core task of a WSN for data collection is to forward sensing data from each sensor to the base station. Due to the "many-to-one" feature of the sensor data collection applications, the network topology is often considered to be a tree topology rooted at the base station. This tree topology needs to be either pre-defined or dynamically formed so that data packets can use it for routing. On the other hand, the existence of wireless interference and collisions makes the scheduling of data packet transmissions a challenging problem that needs to be carefully addressed to achieve effective and efficient accesses to the wireless medium. Since sensor nodes operate autonomously, the transmission scheduling algorithm must be designed to work in a distributed manner while achieving various QoS requirements from different data collection applications.

Since all data are required to be delivered to the base station, reliability is often a mandatory requirement. Xu *et al.* [84] designed a data collection WSN system named Wisden that adopted a data delivery approach with an enhanced reliability. In this system, a hybrid recovery scheme uses both hop-by-hop and end-to-end recoveries to counter packet losses due to wireless interference and topology changes, respectively.

Energy consumption (and thus the network lifetime) is often another concern in WSNs. MAC protocols have been proposed to reduce idle listenings and to turn the radio of the sensor node to sleep mode to save energy. Such general designs, however, if being used for sensor data collection without careful consideration, may introduce additional latencies and even more energy cost. To address this issue, Lu *et al.* [47] proposed DMAC to enhance sensor data collection. In DMAC, if there are more packets to send, a flag is piggy-backed with each previous packet to indicate the next packet transmission. Instead of turning to sleep mode, the receiver keeps listening until the following transmission. Song *et al.* [67] proposed STREE, which also attempts to minimize latency and reduce energy cost. STREE adopted a wave-like forwarding pattern in which the even-level and odd-level links be active alternately along a route to the base station. It then proposed a series of solutions that coordinate different branches of the data collection tree to do wave-like forwarding while avoiding wireless collisions to achieve minimum time cost.

As data aggregation/fusion techniques are hard to apply in data collection applications, the closer a sensor node is to the base station, the more packets it needs to relay. This would cause wireless interference and numerous collisions in the region adjacent to the base station if a CSMA-based protocol is used at the MAC layer, resulting in significant degradation on the network throughput. To solve this problem, Funneling-MAC [5] is proposed to adopt a TDMA protocol within the traffic intensity region around the base station. On the other hand, Ee and Bajcsy [19] focused on the transport layer and proposed solutions using a per-hop mechanism to address congestion control and fairness.

Recently, cross-layer design has been proposed to improve the performance of wireless networks [90]. A pioneering solution in data collection WSNs was proposed by Burri *et al.* [15] to achieve ultra low power data gathering. In this solution, the MAC layer, topology control and routing are carefully designed to minimize the energy consumption of the communication subsystem.

In this thesis, we undertake a case study of the high-rise structural health monitoring application and propose to use elevators to facilitate data delivery while introducing marginal disturbance to the ambient environment. Unlike previous proposals, our approach formally models the data collection problem as a cross-layer optimization problem that jointly considers the link scheduling, packet routing and end-to-end delivery issues.

1.4 Contributions of this Thesis

The main contributions of this thesis are:

- For the deployment stage, we present an in-depth study of the traffic-aware relay node deployment problem. We develop optimal solutions for the simple case of one source node, both with single and multiple traffic flows. We show, however, that the general form of the deployment problem is difficult and that the existing connectivity-guaranteed solutions cannot be directly applied here. We then transform our problem into a generalized version of the

Euclidean Steiner Minimum Tree problem (ESMT) [86]. We face further challenges as its solution is in continuous space and may yield non-integral numbers of relay nodes. Simple rounding of the solution can lead to poor performance. We thus develop algorithms for discrete relay node assignment together with local adjustments that yield high-quality practical solutions. Our solution has been evaluated through both numerical analysis and ns-2 simulations and compared with state-of-the-art approaches. The results show that for all test cases where the continuous space optimal solution can be computed within acceptable timeframes, the network lifetime achieved by our solution is very close to the upper bound of the optimal solution (the difference is less than 13.5%). Moreover, it achieves up to 6 to 14 times improvement over the existing traffic-oblivious strategies.

- For the message dissemination stage, we revisit the message dissemination problem with active/dormant cycles. We show strong evidence that conventional dissemination approaches suffer from severe performance degradation and that, under low duty-cycles, they could easily fail to cover the whole network in an acceptable timeframe. To this end, we remodel the message dissemination problem with the consideration of duty-cycles, seeking a balance between efficiency and latency with coverage guarantees. We demonstrate that this problem can be translated into a graph equivalence and develop a centralized optimal solution. This solution provides a valuable benchmark for assessing diverse duty-cycle-aware broadcast strategies. We then extend it to an efficient and scalable distributed implementation which relies only on local information and operations. The implementation has built-in loss compensation mechanisms. The performance of our solution is evaluated under diverse network configurations. The results suggest that our distributed solution is close to the lower bounds for both time and forwarding costs. It is resistant to wireless loss and has good scalability with the network size and density. In addition, it allows flexible control toward the quality of broadcast coverage.

- For the delivery stage, we propose *EleSense*, a novel high-rise structure monitoring framework that uses elevators to assist data collection. In EleSense, an elevator is associated with the base station and collects data when it moves to serve passengers; as such, the communication distance can be effectively reduced. To maximize the benefit, we formulate the problem as a cross-layer optimization problem and propose a centralized algorithm to solve it optimally. We further propose a distributed implementation to accommodate the hardware capability of sensor nodes and address other practical issues. Through extensive simulations, we show that

EleSense achieves a significant throughput gain over the case without elevators and a straightforward 802.11 MAC scheme without the cross-layer optimization. Moreover, EleSense can greatly reduce the communication cost while maintaining fairness and reliability. We also conduct a case study with real experiments and data sets on the Guangzhou New TV Tower which further validates the effectiveness of our EleSense.

Chapter 2

Traffic-Aware Deployment

There have been various studies of relay node deployment for wireless networks. Most of these have focused on maintaining network coverage and/or connectivity. Directly applying these previous algorithms in location-coverage deployment for data collection applications may give suboptimal results due to the heterogeneous traffic flows and the many-to-one traffic pattern. For an illustration, consider a set of sensor nodes and a data sink (base station) with given locations and traffic volumes as shown in Fig. 2.1. If only connectivity is considered, the deployment scheme in Fig. 2.1a maximizes the network lifetime. This scheme evenly distributes relay nodes along the Steiner minimum tree topology. However, given the sensing data traffic from each sensor node to the base station, a better solution moves some relay nodes from the low traffic path to the high traffic one (Fig. 2.1b) can further extend the network lifetime. Motivated by this, we propose that such traffic-awareness should be incorporated during relay node deployment design.

2.1 System Model and Problem Statement

We consider a wireless sensor network that consists of *source nodes* (*S-nodes*) and *relay nodes* (*R-nodes*). S-nodes sense the ambient environment and forward the data through R-nodes to a remote base station for further processing. The locations of S-nodes and the base station are specified by the application. The average data rates of S-nodes are also known, but may differ for different S-nodes depending on the specific type of data sensed.

Given these application-specific conditions, the network lifetime closely depends on the geographical deployment of the R-nodes, as illustrated in Fig. 2.1. Let $S = \{s_1, s_2, ..., s_M\}$ denote the set of locations of M S-nodes and s_0 be the location of the base station. Let the data rate from s_i be

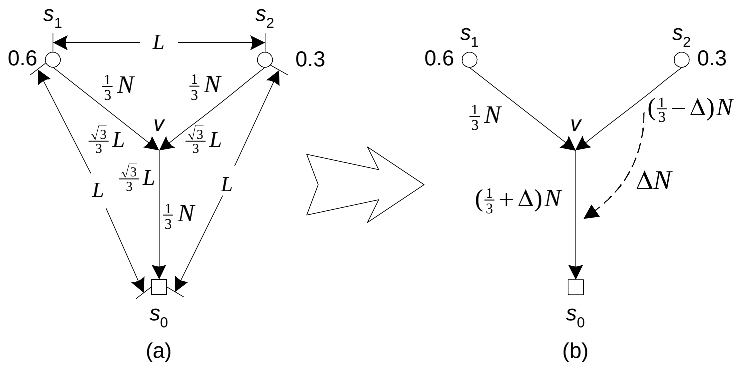


Figure 2.1: An example of relay node deployment: (a) connectivity-based deployment; (b) traffic-aware deployment. s_1, s_2 are sources with data rate of 0.6 and 0.3. s_0 is the base station. Given N relay nodes, by scheme (a), nodes relaying the traffic from v to s_0 will run out of energy much earlier than those relaying from s_1 and s_2 to v. Strategically moving some nodes (ΔN) to section (v, s_0) from the less busy section (s_2, v), the network lifetime is prolonged.

γ_i. We define traffic path $p_i = x_0 x_1 \ldots x_{l_i}$ as a sequence of R-nodes which participate in relaying the traffic flow from s_i. Similar to the solutions of Hou et al. [28] and Xu *et al.* [83], we consider how to deploy a given number of R-nodes so as to maximize the network lifetime. The problem can be formulated as follows:

Traffic-Aware Deployment Problem: Given the total number of R-nodes to be deployed N, where $N \geq M$, find geographical locations for R-nodes $F = \{f_1, f_2, \ldots, f_N\}$, so as to minimize the energy consumption of the R-nodes. Specifically, since the network lifetime is bounded by the nodes with the highest energy costs, we are interested in minimizing the maximum energy consumption among the R-nodes, i.e.,

$$\min \max_{1 \leq i \leq N} \sum_{i \in p_j, j=1..M} \gamma_j [\mathbb{E}_{recv} + \mathbb{E}_{send}(r_i)] ,$$

where $R = \{r_1, r_2, \ldots, r_N\}$ are the communication ranges of R-nodes and $P = \{p_1, p_2, \ldots, p_M\}$ are the traffic paths for S-nodes. Notice that the summation here indicates that an R-node can undertake combined traffic flows of multiple sources if it is chosen in these paths. For ease of exposition, we summarize the notation used in this chapter in Tab. 2.1. Denote R_{max} as the maximum communication range of an R-node. The deployment must satisfy the following constraints:
(1) Communication range,

$$\forall r \in R, r \leq R_{max};$$

Notation	Description
M	Number of given S-nodes
N	Number of given R-nodes
R_{max}	Maximum communication range of R-nodes
α	Exponent parameter in communication model
c	Small constant specified by physical features of transceiver in communication model
s_0	Location of remote data sink
s_i	Location of S-node i
γ_i	Average data rate of S-node i
f_i	Location of R-node i
$\overline{xy}$	Distance between node x and y
r_i	Communication range of R-node i
$\mathbb{E}_{subscript}$	Energy cost of an R-node under the situation described by the $subscript$
V	Vertex set in constructed graph $G(V, E)$
E	Edge set in constructed graph $G(V, E)$
v_i	Vertex i in V. $v_i = s_i$ for $0 \leq i \leq M$; otherwise v_i is a merge vertex
e_i	Directed edge i in E, traffics can only flow along its direction
L_{e_i}	Length of e_i
λ_{e_i}	Total data rate of traffic flows passing e_i
n_{e_i}	Number of R-nodes assigned on e_i
$\mathbb{E}_{e_i}$	Maximum energy cost of one R-node on e_i
$\mathbb{E}$	Initial energy budget on one R-node
T	Network lifetime

Table 2.1: List of notation.

(2) Forwarding path connectivity,

$$\forall p = x_0 x_1 \dots x_l \in P, \overline{f_{x_{i-1}} f_{x_i}} \leq r_{x_{i-1}}, i = 1 \dots l;$$

(3) S-nodes and sink connectivity,

$$\forall s \in S, \exists p = x_0 x_1 \dots x_l \in P, f_{x_0} = s, \overline{f_{x_l} s_0} \leq r_{x_l}.$$

To simplify exposition, we associate each S-node with an R-node at the same location (as shown in Constraint 3). This guarantees S-nodes are only involved in local communications and the network lifetime thus depends on R-nodes.

Our formulation is not restricted by specific energy models for wireless communications. For illustration purpose, the following popular energy consumption model for packet transmission [53] will be used in this chapter:

$$\mathbb{E}_{send}(r) = ar^{\alpha} + b\ ,$$

which can be normalized as

$$\mathbb{E}_{send}(r) = r^{\alpha} + c\ ,$$

where α is generally greater than 1 with typical values between 2 and 6. c is a small constant compared to r^{α} which represents the basic cost when the transceiver circuit is working. Since receiving does not include the extra cost of generating signals, the energy consumption for packet receiving is given by

$$\mathbb{E}_{recv} = c\ .$$

Finally, it is worth noting that our network model can be easily extended to a hierarchial structure where each S-node represents a cluster of geographically-close sources [26, 28, 64, 69, 81]. Our analysis and optimization below will still apply as long as the many-to-one pattern holds and the inter-cluster communications dominate the energy consumption, as is the case for most applications.

2.2 Traffic-Aware R-node Deployment: The Single Source Case

We first study the relay deployment problem of two basic cases with a single source and derive optimal solutions. These results will serve as building blocks for solving the general problem in the next section.

2.2.1 The Single Source Single Traffic Flow Case

We begin with the basic case of single source single traffic flow. An illustration is shown in Fig. 2.2, where L is the distance between the S-node s_1 (with traffic rate γ_1) and the sink s_0. We need to deploy N R-nodes between them. Obviously, N should satisfy $\frac{L}{N} \leq R_{max}$ for a feasible solution. Let the distance between the i-th R-node and the $(i+1)$-th R-node (or sink) be r_i, $i = 1, \ldots, N$. The energy cost for the i-th R-node is

$$\gamma_1[\mathbb{E}_{recv} + \mathbb{E}_{send}(r_i)] = 2\gamma_1 c + \gamma_1 r_i^{\alpha}.$$

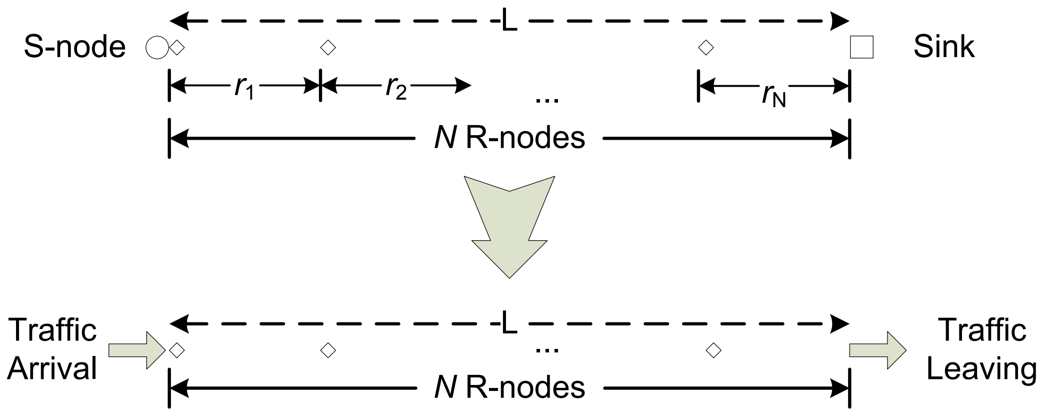


Figure 2.2: Deployment for single source single flow and its generalization.

Since $\sum_{i=1}^{N} r_i = L$, it is easy to see that the solution to $\min \max_{1 \le i \le N} (2\gamma_1 c + \gamma_1 r_i^\alpha)$ is $r_i = \frac{L}{N}$, for $i = 1, 2, \ldots, N$ and that the minimum of the maximum energy consumption among the R-nodes is $2\gamma_1 c + \gamma_1 (\frac{L}{N})^\alpha$. This result can be generalized as follows:

Theorem 2.2.1 *The optimal solution for single source single traffic flow is to start from the source and evenly deploy the R-nodes with a distance of $\frac{L}{N}$ between adjacent R-nodes. The energy consumption for each R-node is $\mathbb{E}_{single}(L, N, \gamma) = \gamma[2c + (\frac{L}{N})^\alpha]$.*

Proof: It is easy to prove the theorem by contradiction. Assume that in an optimal solution, $\exists r_i \neq \frac{L}{N}$, then there must exist one, say r_j, which is greater than $\frac{L}{N}$. Then this R-node has energy cost greater than the value given by the theorem. This contradicts with the assumption and thus proves the theorem. □

2.2.2 The Single Source Multi Traffic Flow Case

Next, we consider the case where multiple traffic flows arrive at one location and need to be relayed to another. Given N R-nodes and K traffic flows, we need to decide whether to merge these flows or to relay them separately by assigning n_i R-nodes to the i-th flow with $\sum_{i=1}^{K} n_i = N$. We first consider the case of two flows, as illustrated in Fig. 2.3.

If the traffic flows are relayed separately, according to Theorem 2.2.1 the energy consumption of one R-node for the i-th traffic flow is $\mathbb{E}_{single}(L, n_i, \gamma_i)$, for $i = 1, 2$. As in the previous subsection,

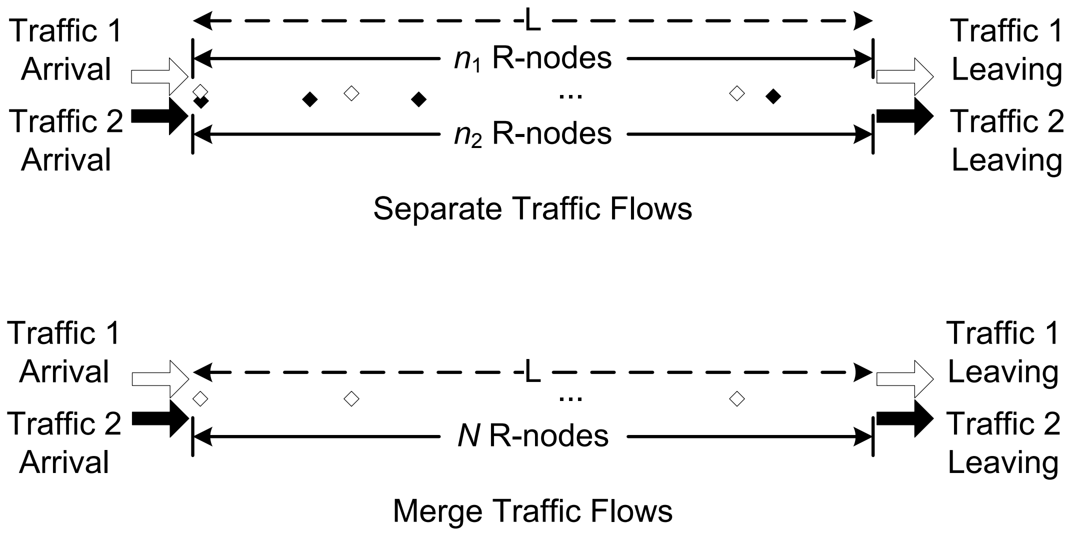


Figure 2.3: Two deployment schemes for single source two traffic flows.

it is easy to see that the R-nodes should be assigned such that

$$\mathbb{E}_{single}(L, n_1, \gamma_1) = \mathbb{E}_{single}(L, n_2, \gamma_2) \, .$$

Consequently,

$$\gamma_1[2c + (\frac{L}{n_1})^\alpha] = \gamma_2[2c + (\frac{L}{n_2})^\alpha] \, .$$

Typically, we have $(\frac{L}{n_i})^\alpha \gg c$ for $i = 1, 2$ (see [31]) and thus

$$\gamma_1(\frac{L}{n_1})^\alpha \approx \gamma_2(\frac{L}{n_2})^\alpha \, ,$$

which gives

$$\frac{\sqrt[\alpha]{\gamma_1}L}{n_1} = \frac{\sqrt[\alpha]{\gamma_2}L}{n_2} = \frac{L(\sqrt[\alpha]{\gamma_1} + \sqrt[\alpha]{\gamma_2})}{n_1 + n_2} = \frac{L}{N}(\sqrt[\alpha]{\gamma_1} + \sqrt[\alpha]{\gamma_2}) \, .$$

We then have the energy consumption of an R-node as

$$\mathbb{E}_{separarte} \approx (\frac{L}{N})^\alpha(\sqrt[\alpha]{\gamma_1} + \sqrt[\alpha]{\gamma_2})^\alpha \, .$$

On the other hand, if the traffic flows are merged, the energy consumption of one R-node becomes

$$\mathbb{E}_{merge} = \mathbb{E}_{single}(L, N, (\gamma_1 + \gamma_2)) \approx (\gamma_1 + \gamma_2)(\frac{L}{N})^\alpha \, .$$

Clearly, we have

$$\mathbb{E}_{separate} \;=\; (\frac{L}{N})^{\alpha}(\sqrt[\alpha]{\gamma_1}+\sqrt[\alpha]{\gamma_2})^{\alpha} \geq (\frac{L}{N})^{\alpha}[(\sqrt[\alpha]{\gamma_1})^{\alpha}+(\sqrt[\alpha]{\gamma_2})^{\alpha}] = \mathbb{E}_{merge}\;,$$

which shows that merging these two flows leads to the minimum energy cost on an R-node. This result can be easily generalized as follows:

Theorem 2.2.2 *The optimal solution to single source multi traffic flow is to merge all flows into one and apply the optimal scheme for single source single traffic flow.*

Proof: As discussed above, the problem is easily solved for two traffic flows that can be merged or kept separate completely.

For other situations of two traffic flows (i.e., two flows are partially merged), the theorem can be proved by contradiction. Assume that an optimal solution exists that is better than merging two flows completely. Then it is easy to divide the whole flow paths into segments such that within any segment, the two flows are either completely merged or separate. By the above discussion for the problem of completely merging or separating two traffic flows, in each segment where the two flows separate completely, the two flows can be merged together without sacrificing the network lifetime. We can continue applying this until the two flows in all segments are merged completely. This contradicts the assumption and completes the proof for two traffic flows.

For the case of K traffic flows, the proof can be done by induction. The base case of two traffic flows has been proved above. For K flows, we assume that the theorem holds for all cases from 2 to $K-1$ flows and show that it also holds for the case of K flows. The proof is similar to the proof of the two flow case. First, it is easy to prove that merging all K traffic flows completely is better than leaving them completely separated. For the remaining situations, we use contradiction and assume an optimal solution exists that is better than merging K flows completely. Then it is easy to divide the whole flow paths into segments such that within any segment each pair of flows are either completely merged or separated. Now in every section, we consider each pair of completely merged flows as one new flow until all remaining flows are completely separated. Clearly the resulting flows in each section can be merged into one without sacrificing the network lifetime (either due to the induction hypothesis or since merging all K traffic flows completely is better than leaving them completely separated) until all sections have only one flow left. This contradicts the assumption and finishes the proof. $\square$

It is worth noting that Theorem 2.2.2 and its proof also indicate that in the optimal solution there is no need to split one single flow into multiple flows since, as shown in the proof of Theorem 2.2.2,

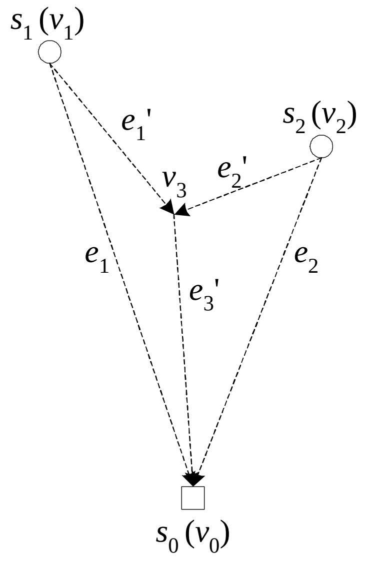


Figure 2.4: An example on deployment for multi-source with multi-traffic flows.

merging the (split) multiple flows back into one flow achieves the optimal result.

2.3 Traffic-Aware R-node Deployment: The General Case

We now address the general form of the traffic-aware deployment problem, i.e., the multi source multi traffic flow case.

2.3.1 Theoretical Solution in Continuous Space

We first translate the problem into a graph equivalence. Define directed graph $G = (V, E)$, where $V = \{v_0, v_1, \ldots, v_M, v_{M+1}, \ldots, v_{|V|-1}\}$, $E = \{e_1, e_2, \ldots, e_{|E|}\}$. Let $v_i = s_i$ for $i = 0, 1, \ldots, M$. Here, vertices v_j, $j \geq M + 1$, are called *merge vertices*. Their function will be explained later. Let $e_1, e_2, \ldots$ denote the edges that connect the vertices in V. Recall that traffics can only flow in the direction of the edge. The choice of v_j, $j \geq M + 1$ and e_i are to be determined later. Let λ_{e_i} be the sum of average data rates of the traffic flows passing through edge e_i. Let L_{e_i} be the length of the edge e_i, n_{e_i} be the number of the R-nodes assigned to edge e_i and $\mathbb{E}_{e_i}$ be the maximum energy consumption of any R-node on edge e_i.

As an example, Fig. 2.4 shows a simple case of two sources s_1 and s_2 with the base station s_0. By definition, we have $v_0 = s_0$, $v_1 = s_1$ and $v_2 = s_2$. One deployment strategy is to place the R-nodes along e_1 and e_2, and the traffic flows can then be relayed to s_0 along these two edges separately. Alternatively, we can also use a *merge vertex* v_3 and deploy R-nodes along e'_1, e'_2 and e'_3; the traffic flows then can be relayed from s_1 and s_2 via e'_1 and e'_2, merged at v_3, and arrive at s_0 via e'_3. Surely there can be other relay node deployment schemes with merge vertices placed at different locations or using different graph topologies, but the network lifetime of each scheme is always bounded by the edge containing the R-node with the maximum energy cost. Note that each edge is directed from a start point to an end point. This is exactly the cases we have discussed in last section. Thus, depending on whether one or multiple flows are relayed by an edge, we can apply Theorems 2.2.1 or 2.2.2 and obtain

$$\mathbb{E}_{e_i} = \mathbb{E}_{single}(L_{e_i}, n_{e_i}, \lambda_{e_i}) = \lambda_{e_i}[2c + (\frac{L_{e_i}}{n_{e_i}})^\alpha] \approx \lambda_{e_i}(\frac{L_{e_i}}{n_{e_i}})^\alpha .$$

To achieve $\min \max_{e_i \in E} \mathbb{E}_{e_i}$, we need $\mathbb{E}_{e_1} = \mathbb{E}_{e_2} = \ldots$, which gives

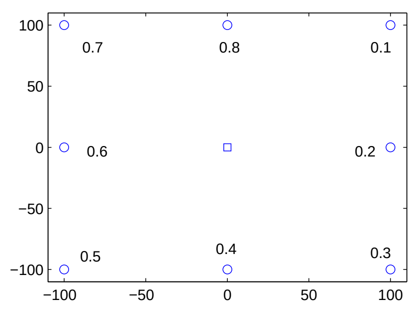
$$\frac{\sqrt[\alpha]{\lambda_{e_1}} L_{e_1}}{n_{e_1}} = \frac{\sqrt[\alpha]{\lambda_{e_2}} L_{e_2}}{n_{e_2}} = \ldots = \frac{\sum_{e_i \in E} (\sqrt[\alpha]{\lambda_{e_i}} L_{e_i})}{\sum_{e_i \in E} n_i} \tag{2.1}$$

Given that $\sum_{e_i \in E} n_i = N$, the remaining task thus becomes finding the appropriate graph topology that minimizes $\sum_{e_i \in E} (\sqrt[\alpha]{\lambda_{e_i}} L_{e_i})$. Once this is found, the edge directions and data rates can be easily determined. (A detailed discussion on deriving the optimal edge directions and data rates of a given graph topology was given by Xue *et al.* [85].) The number of R-nodes on each edge can be computed by Eq. (2.1) and the deployment then follows Theorem 2.2.1. We thus have the following observation:

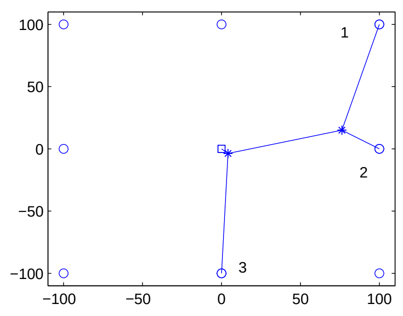
Observation 1 *The optimal solution to the general problem of multi source multi traffic flow is equivalent to minimizing the total weighted length of the edges that connect all the sources and the sink (allowing a set of merge vertices), where the weight on an edge e_i is $\sqrt[\alpha]{\lambda_{e_i}}$.*

The above problem is a generalized version of the Euclidian Steiner Minimum Tree problem[1] which is known to be NP-hard [86]. Xue *et al.* [85] proposed a heuristic which first constructs a

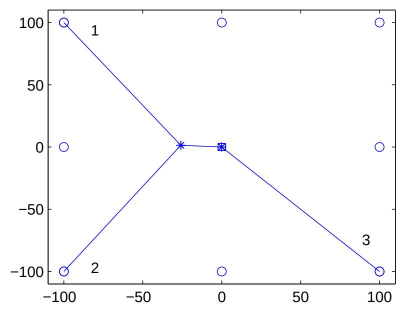
[1]Note that although the edge weights are determined by the interconnection of the graph topology, the graph topology and the edge lengths are in turn determined by the locations of the merge points, which may be chosen from anywhere within the sensing field.



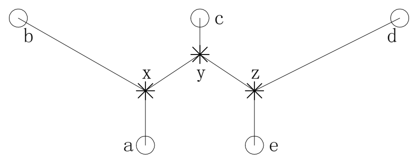
(a) Input



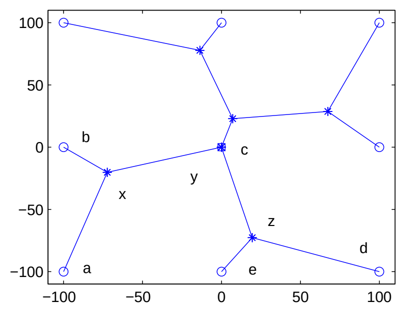
(b) Min-Min ordering



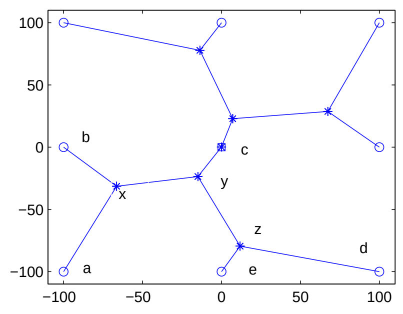
(c) Max-Min ordering



(d) Size-5 component



(e) Graph topology before 5-optimization



(f) Graph topology after 5-optimization

Figure 2.5: Different orderings and 5-optimization used in the hybrid algorithm. The sink is denoted by the small square at the center. S-nodes are denoted by small circles. Merge vertices are denoted by small stars. (a) shows the input of the illustration, where the numerical label beside each S-node shows the data rate. (b) shows the first three steps by using Min-Min ordering, where the digit beside an S-node shows in which step the node is added. (c) shows the first three steps by using Max-Min ordering, where the digit beside an S-node shows in which step the node is added. (d) shows the pattern of the size-5 component used for optimization. (e) shows an intermediate graph topology before a 5-optimization and (f) shows the result after the 5-optimization, where the positions of x, y and z have been optimized within the size-5 component.

graph topology by adding non-merge vertices one by one and then uses a backtrack algorithm to optimize each size-5 component on the constructed graph topology. Fig. 2.5 shows an example of the size-5 component and how to use it to conduct 5-optimization. The size-5 component is a Steiner-tree-like structure containing 5 outer vertices (a, b, c, d and e as illustrated in Fig. 2.5) and 3 inner vertices (x, y and z as illustrated in Fig. 2.5). During a 5-optimization, the pattern of the size-5 component is matched iteratively on a graph topology. For each match, 4 of the 5 outer vertices are deemed as non-merge vertices with the remaining one serving as the sink. The 3 inner vertices are then optimized within the component as they are merge vertices.

In the construction, non-merge vertices can be added by two ordering schemes as illustrated in Fig. 2.5: In Min-Min ordering, each added vertex minimizes the increased total weighted edge length. This is similar to the minimum spanning tree construction but is complicated due to creating a merge vertex at each step. In Max-Min ordering, each added vertex maximizes the minimum of the increased total weighted edge length. For each of these orderings an algorithm has been designed [85]. Unfortunately, no bounds on approximation performance were found for these two algorithms and when $M > 10$, either one may return sub-optimal results.

Interestingly enough, our analysis shows that the sub-optimal results obtained by different orderings are often stuck at different local optimums, even though they are designed to avoid being stuck too early before the size-5 component optimization stage. This motivates us to implement a hybrid algorithm that uses both orderings complementarily to bypass local optimums. Specifically, we start by adding non-merge vertices in one ordering, then switch to the other after k vertices have been added, where k is enumerated from 1 to $M - 1$. Fig. 2.6 shows the details of the hybrid algorithm. During our performance evaluation, we find that this hybrid algorithm successfully returns optimal results on all those test cases ($M \leq 15$) that can be verified within acceptable timeframes.

2.3.2 Practical Solution on Discrete Node Deployment

So far we have a solution for finding the graph topology, i.e., the location of the merge vertices, which minimizes the maximum energy cost on an R-node. However, directly solving Eq. (2.1) may yield a non-integral number of R-nodes being assigned to an edge. Our experience shows that a naive rounding to the closest integers can result in up to 40% performance degradation. To build a practical solution, we develop algorithms for optimal discrete R-node assignment and merge vertices adjustments.

```
Algorithm MinWeightedLength()
1:  $L_{min} \leftarrow \infty$;
2:  $V_+ \leftarrow \{v_0\}$; $E_+ \leftarrow \{\}$; $V_- \leftarrow \{v_0\}$; $E_- \leftarrow \{\}$;
3:  for $k = 1 \ldots M-1$, do
4:    for $i = 1 \ldots M$, do
5:      if $i \leq k$,
6:        update $V_+$, $E_+$ by adding one non-merge vertex that minimizes the increased
            total weighted length;
7:        update $V_-$, $E_-$ by adding one non-merge vertex that maximizes the minimum
            increased total weighted length;
8:      else
9:        update $V_+$, $E_+$ by adding one non-merge vertex that maximizes the minimum
            increased total weighted length;
10:       update $V_-$, $E_-$ by adding one non-merge vertex that minimizes the increased
            total weighted length;
11:     end if
12:   end for
13:   conduct 5-optimization on $(V_+, E_+)$ with each size-5 component;
14:   if the total weighted length $< L_{min}$,
15:     $L_{min} \leftarrow$ the total weighted length;
16:     $V_{min} \leftarrow V_+$; $E_{min} \leftarrow E_+$;
17:   end if
18:   conduct 5-optimization on $(V_-, E_-)$ with each size-5 component;
19:   if the total weighted length $< L_{min}$,
20:     $L_{min} \leftarrow$ the total weighted length;
21:     $V_{min} \leftarrow V_-$; $E_{min} \leftarrow E_-$;
22:   end if
23: end for
24: return $V_{min}$ and $E_{min}$;
```

Figure 2.6: The hybrid algorithm for computing the graph topology that minimizes the total weighted edge length.

```
Algorithm RnodeAssignment()
1: for $e_i \in E$, do $n_{e_i} \leftarrow 1$;
2: $N \leftarrow N - |E|$;
3: while $N > 0$, do
4:     find $e_i \in E$ such that $e_i$ has the largest energy cost $\mathbb{E}_{e_i} = \mathbb{E}_{single}(L_{e_i}, n_{e_i}, \lambda_{e_i})$;
5:     $n_{e_i} \leftarrow n_{e_i} + 1$;
6:     $N \leftarrow N - 1$;
7: end while;
8: return $n_{e_i}$ for all $e_i \in E$;
```

Figure 2.7: The algorithm for discrete R-node assignment on edges.

2.3.2.1 Optimal Discrete R-node Assignment

We develop a greedy algorithm (see Fig. 2.7) for the discrete R-node assignment problem, which assigns each edge an integer number of R-nodes. It starts from the assignment with one R-node on each edge (line 1-2) which, by Theorem 2.2.1, should be placed at the start point of each edge. Then we add other R-nodes one by one to the edge with the maximum energy consumption (line 4-7). This algorithm is optimal for the discrete R-node assignment as shown by the following:

Theorem 2.3.1 *Given a graph topology and any feasible R-node number, the RnodeAssignment() algorithm generates the optimal R-node assignment to the edges of the graph topology such that the maximum energy cost among the edges is minimized.*

Proof: We prove the result by induction on the number of given R-nodes N. For base case, we have $N = |E|$. The optimal and only feasible assignment is $n_{e_i} = 1$ for all $e_i \in E$.

Now we assume that the assignment achieved by our algorithm is optimal for $N = k$ with $k \geq |E|$ and consider the case $N = k + 1$. We show the optimality by contradiction.

Assume that there is a better assignment A' which has lower maximum energy cost among the edges than our assignment A. Let e_i be the edge with maximum energy cost $\mathbb{E}_{e_i}$ in our assignment and e_j be the edge with maximum energy cost $\mathbb{E}'_{e_j}$ in A'. Then we have $\mathbb{E}_{e_i} > \mathbb{E}'_{e_j}$. Since $\mathbb{E}'_{e_j}$ is maximum among all edges in A', $\mathbb{E}'_{e_j} \geq \mathbb{E}'_{e_i}$. This implies that $n_{e_i} < n'_{e_i}$. This means that to make our assignment better, at least 1 R-node needs to be moved from some other edge, say e_x to e_i while maintaining $\mathbb{E}^-_{e_x} \leq \mathbb{E}'_{e_j} < \mathbb{E}_{e_i}$. Note we use "$\mathbb{E}^-/\mathbb{E}^+$" to denote the energy cost after an R-node is removed/added.

Now we consider which edge has been assigned the $(k+1)$-th R-node from case $N = k$ to case $N = k + 1$ by our R-node assignment algorithm. We argue it must be e_i in the above situation.

```
Algorithm EnergyBalance()
1:  $V_{min} \leftarrow V$;
2:  $\mathbb{E}_{min} \leftarrow \max_{e_i \in E} \mathbb{E}_{e_i}$;
3:  while true, do
4:     for each merge vertex $v \in V$, do
5:        adjust $v$ so as to balance $\mathbb{E}_{e_i}$ among all $e_i$ connecting to $v$;
6:     end for
7:     if $\max_{e_i \in E} \mathbb{E}_{e_i} < \mathbb{E}_{min}$,
8:        $V_{min} \leftarrow V$;
9:        $\mathbb{E}_{min} \leftarrow \max_{e_i \in E} \mathbb{E}_{e_i}$;
10:    else break;
11: end while;
12: return $V_{min}$;
```

Figure 2.8: The algorithm for balancing energy consumption among different edges.

Otherwise, if it is some edge e_y other than e_i, then instead of adding the $(k+1)$-th R-node to e_y, we can move the extra R-node from e_x to e_y. By doing this, e_i now has maximum energy cost other than e_y for the case $N = k$ (note $\mathbb{E}_{e_i} > \mathbb{E}^-_{e_x}$ and $\mathbb{E}_{e_i} \geq \mathbb{E}^+_{e_y}$ due to case $N = k+1$), which means case $N = k$ can be further improved and contradicts that the case $N = k$ is optimal. Thus the $(k+1)$-th R-node must be assigned to e_i. However, if now we move the extra R-node from e_x to e_i instead of assigning the $(k+1)$-th R-node, we get a better assignment for case $N = k$ which also contradicts that the case $N = k$ is optimal. This shows that the assumption that an assignment better than ours exists for case $N = k+1$ is not correct. Together with the base case, the theorem is proved. □

2.3.2.2 Merge Vertex Adjustment

Next we adjust the merge vertices to further balance the energy consumption between different edges. For example, if there is very short edge, then even deploying one R-node to this edge can lead to waste, i.e., when the network is depleted, this R-node still has significant energy. To this end, we develop two algorithms to balance the energy consumption on different edges in order to avoid such situations. Fig. 2.8 and Fig. 2.9 show the details.

EnergyBalance() (Fig. 2.8) proceeds iteratively (the **while** loop) to balance energy consumption among edges connected to each merge vertex. In each iteration (the **for** loop), it tries to adjust the

```
Algorithm AdjustMergeVertex()
1:   $n_{e_i}$, $e_i \in E \leftarrow$ RnodeAssignment();
2:   $V_{min} \leftarrow$ EnergyBalance();
3:   $E_{min} \leftarrow E$;
4:   $\mathbb{E}_{min} \leftarrow \max_{e_i \in E} \mathbb{E}_{e_i}$;
5:   while true, do
6:      $V_{temp} \leftarrow V_{min}$; $E_{temp} \leftarrow E_{min}$;
7:      $\mathbb{E}_{temp} \leftarrow \mathbb{E}_{min}$;
8:      for each merge vertex $v \in V_{min}$, do
9 :        $V \leftarrow V_{min}$; $E \leftarrow E_{min}$;
10:        combine $v$ with closest vertex for $V$ and $E$;
11:        $n_{e_i}$, $e_i \in E \leftarrow$ RnodeAssignment();
12:        $V \leftarrow$ EnergyBalance();
13:        if $\max_{e_i \in E} \mathbb{E}_{e_i} < \mathbb{E}_{temp}$,
14:           $V_{temp} \leftarrow V$; $E_{temp} \leftarrow E$;
15:           $\mathbb{E}_{temp} \leftarrow \max_{e_i \in E} \mathbb{E}_{e_i}$;
16:        end if
17:     end for
18:     if $\mathbb{E}_{temp} < \mathbb{E}_{min}$,
19:        $V_{min} \leftarrow V_{temp}$; $E_{min} \leftarrow E_{temp}$;
20:        $\mathbb{E}_{min} \leftarrow \mathbb{E}_{temp}$;
21:     else break;
22:  end while;
23:  return $V_{min}$ and $E_{min}$;
```

Figure 2.9: The algorithm for merge vertex adjustment.

location of a merge vertex v by solving equations

$$\lambda_{e_1}(\frac{\overline{v_1 v}}{n_{e_1}})^{\alpha} = \lambda_{e_2}(\frac{\overline{v_2 v}}{n_{e_2}})^{\alpha} = \lambda_{e_3}(\frac{\overline{v_3 v}}{n_{e_3}})^{\alpha} ,$$

where $e_1 = (v, v_1)$, $e_2 = (v, v_2)$ and $e_3 = (v, v_3)$. It is possible that v has more than three edges connected to it. In this case, we explore all 3-combinations that contain the edge with the maximum energy consumption and use the solution that minimizes the maximum energy consumption among these edges. Note that $|E|$ is bounded by $(2 \times M - 1)$ [85], thus the computational complexity is polynomial. Our experience shows that the algorithm is fast in practice.

AdjustMergeVertex() (Fig. 2.9) takes the graph topology generated by the theoretical solution as an input. It first assigns R-nodes and does energy balancing (line 1-4). Then, in each iteration of the

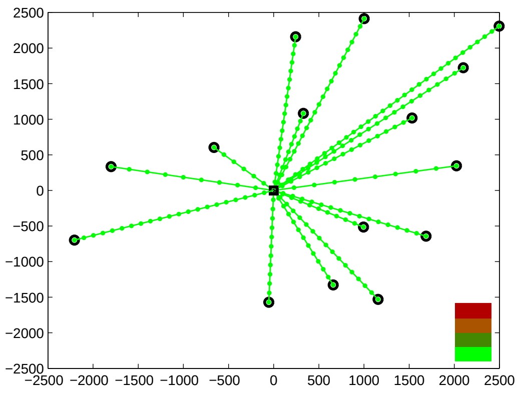
while loop, it tries to combine each merge vertex with its closest vertex and keeps the combination that yields the largest reduction to the maximum energy cost among all edges. Also during each iteration, it reassigns R-nodes and re-balances the energy consumption globally (line 11-12) so as to bypass local optimums. In next section, we will show that our solution, which considers both theoretical optimality and practical issues, has achieved excellent performance with good efficiency and balanced energy consumption.

2.4 Performance Evaluation

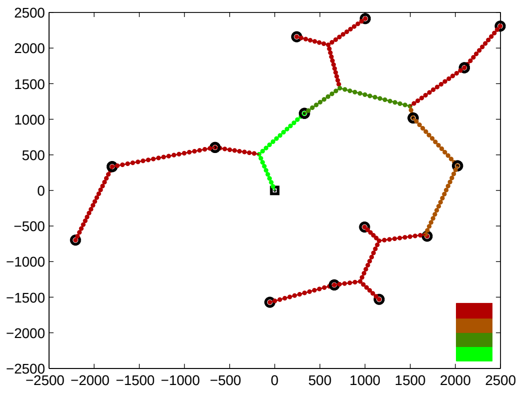
We evaluate our solution by both numerical analysis and ns-2 simulations [3]. We adopt similar configurations as those used by Kashyap *et al.* [33], Misra *et al.* [51] and Zhang *et al.* [91] in our evaluation. Specifically, we deploy 5 to 25 S-nodes uniform distributed in a field of $5000m \times 5000m$ with the sink positioned at the center. The normalized data rate of each S-node is randomly picked from $(0, 1]$. For each number of S-nodes, 10 topologies are generated. Each point in the figures thus represents the average of 10 topologies with an error bar showing the standard deviation.

For comparison, we implemented three deployment approaches: *Direct-Connection*, *Connectivity-Only*, and *Half-Traffic-Aware*. Direct-Connection connects each S-node with the sink by a dedicated data path (an edge) where R-nodes are deployed by our algorithm in Section 2.3.2. This is the most straightforward approach and serves as a base-line. Connectivity-Only is one of a series of state-of-the-art schemes proposed by Lloyd and Xue [46] and Zhang *et al.* [91] which optimize the system performance by considering connectivity only. For better performance, we use the 1-connectivity version (i.e., there is at least one data path from each S-node to the sink) and further enhance it with a better approximation for Euclidean Steiner minimum tree [85] (instead of minimum spanning tree) to construct the graph topology. The Half-Traffic-Aware approach uses the same graph topology as Connectivity-Only but assigns R-nodes using the algorithm proposed in Section 2.3.2. It is used as a reference to help understand the impact of the graph topology (by comparing with our solution) and of the discrete R-node assignment algorithm (by comparing with Connectivity-Only). Fig. 2.10 illustrates how these three approaches and our new solution deploy R-nodes on a test case of 15 S-nodes used in our evaluation. Our solution is called *Full-Traffic-Aware*.

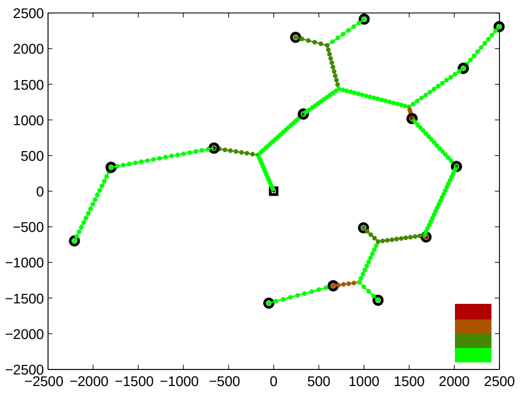
Three metrics are used for evaluation. The first is *network lifetime*, defined to be the lifetime of the first depleted R-node. In practice, this usually results in a need to dispatch a technician to replace the battery of the depleted R-node. Since sending a technician is costly, it is usually preferable to replace all batteries at the same time. Thus, the first node depletion serves as a good indicator of



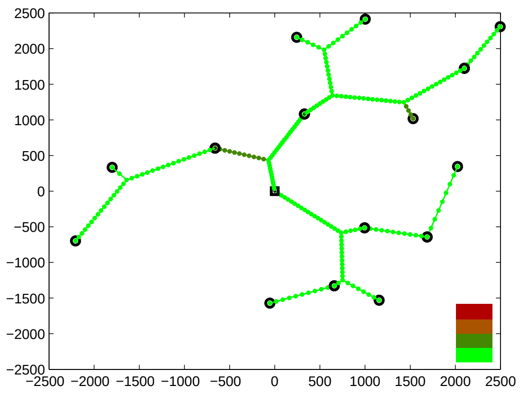
(a) Direct-Connection



(b) Connectivity-Only



(c) Half-Traffic-Aware



(d) Full-Traffic-Aware

Figure 2.10: The different deployment approaches and their residual energy distributions at the end of the network lifetime of each approach. The sink is denoted by the small square at the center. S-nodes are denoted by small circles. R-nodes are denoted by small diamond dots. Each approach uses the same number of R-nodes (230). Residual energy is demonstrated in a different color scale, where redder and darker color denotes higher residual energy while greener and lighter color denotes lower residual energy. A color scale reference is shown at the bottom right corner of each deployment.

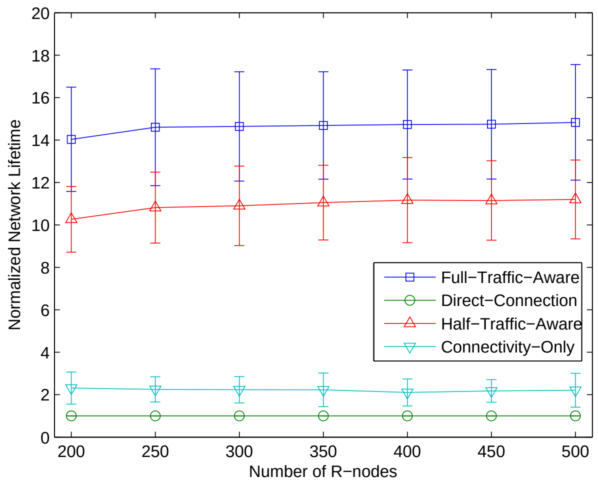


Figure 2.11: Normalized network lifetime with different numbers of R-nodes by numerical analysis.

network lifetime. The second metric is *residual energy*, defined to be the residual energy of all R-nodes at the end of the network lifetime. Since all batteries are expected to be replaced at the same time, lower residual energy indicates less energy wasted by the removed batteries. The third metric is *energy efficiency*, defined to be the amount of traffic relayed to the sink per unit energy cost. We consider this metric as we want to evaluate whether our solution extends the network lifetime at the expense of energy inefficiencies, as discussed by Olariu and Stojmenovic [53].

We set $\alpha = 4$ [53] and $R_{max} = 500m$. The initial energy for each node is set to $\mathbb{E} = T_{min} \cdot 10^8$. In this equation, T_{min} is the minimum network lifetime specified by the application and is set to 1000. For ease of comparison, all results are normalized by the base-line scheme Direct-Connection.

2.4.1 Results by Numerical Analysis

Given a practical solution with the graph topology and the number of R-nodes on each edge, the network lifetime is estimated to be[2]

$$T = \min_{e_i \in E} \frac{\mathbb{E}}{\lambda_{e_i} \cdot (\frac{L_{e_i}}{n_{e_i}})^{\alpha}} .$$

[2]Following our analysis, we omit the small constant c here. In our ns-2 simulation, all the practical factors (e.g., c) are included.

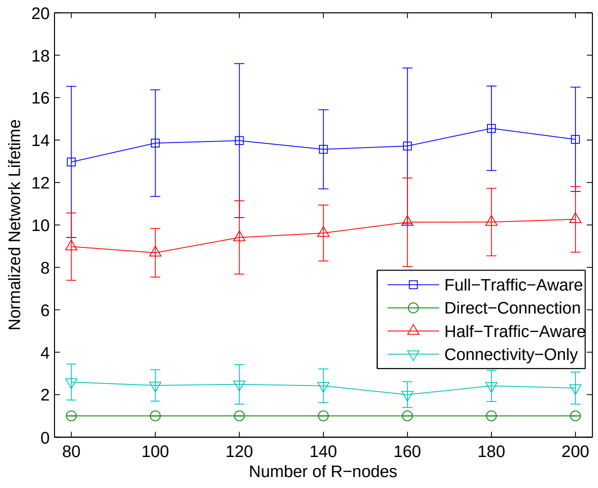


Figure 2.12: Normalized network lifetime with the number of R-nodes ≤ 200 by numerical analysis (the communication range constraint is temporarily relaxed).

The total residual energy is

$$\mathbb{E}_{residual} = \sum_{e_i \in E} \big(\mathbb{E} - \lambda_{e_i} \cdot (\frac{L_{e_i}}{n_{e_i}})^{\alpha} \cdot T\big) \cdot n_{e_i}$$

and the energy efficiency is

$$\frac{T \cdot \sum_{i=1}^{M} \gamma_i}{N \cdot \mathbb{E} - \mathbb{E}_{residual}} = \frac{\sum_{i=1}^{M} \gamma_i}{\sum_{e_i \in E} \lambda_{e_i} \cdot (\frac{L_{e_i}}{n_{e_i}})^{\alpha} \cdot n_{e_i}} .$$

In the remainder of this subsection, we will investigate the impacts of different numbers of R-nodes and S-nodes, respectively.

2.4.1.1 Impact of R-node Number Selection

We first set the number of S-nodes to 25 and compute the numerical results to analyze how the performance of different solutions changes with the number of R-nodes. Given that the field is $5000m \times 5000m$ with $R_{max} = 500m$, for the Direct-Connection scheme the average of the minimum integer number of R-nodes required to work properly (i.e., at least the basic connectivity is

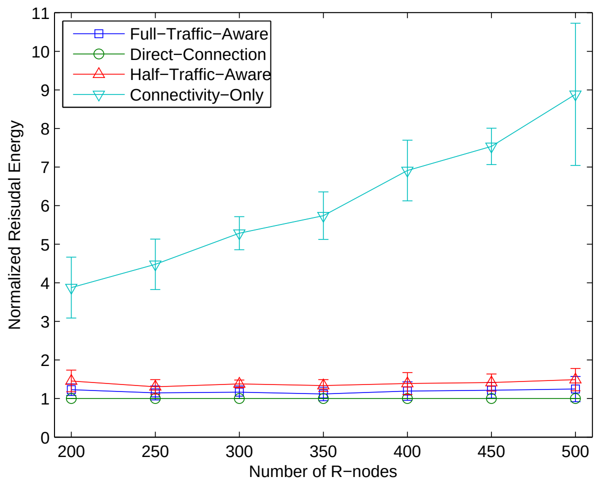


Figure 2.13: Normalized residual energy with different numbers of R-nodes by numerical analysis.

guaranteed) is 100 with the upper bound of the minimum integer number as high as 200 (i.e., on average 4 R-nodes and at most 8 R-nodes for one S-node). As will be discussed later Direct-Connection always needs more R-nodes than other schemes, the numerical analysis starts with 200 R-nodes and ranges up to 500.

Fig. 2.11 shows the results of the network lifetime with different numbers of R-nodes. Interestingly, when the number of R-nodes is greater than or equal to 250, the trends of all solutions stay steadily and are not very sensitive to the changes of the R-node number. Note that the results are normalized by the Direct-Connection scheme which flattens the slope of each solution. On the other hand, when the number of R-nodes is less than 250, the performance of Full-Traffic-Aware and Half-Traffic-Aware seem to deteriorate a little. To investigate how these two schemes perform when the number of R-nodes is closer to the number of S-nodes, we temporarily relax the communication range constraint for Direct-Connection and further conduct numerical analysis with the number of R-nodes ranging from 80 to 200. The results are shown in Fig. 2.12. It can be seen that there are some small fluctuations in the figure. This is because as the number of R-nodes decreases, the marginal effects of the random topology variations and that only an integer number of R-nodes can be used on each edge of a graph topology may become more observable. Even though, the trends of Full-Traffic-Aware and Half-Traffic-Aware are still relatively stable with the performance only slightly decreased. Also, in Figs. 2.11 and 2.12, the network lifetime using Half-Traffic-Aware is roughly up to 10 times longer than when using Direct-Connection and 4 times longer than with

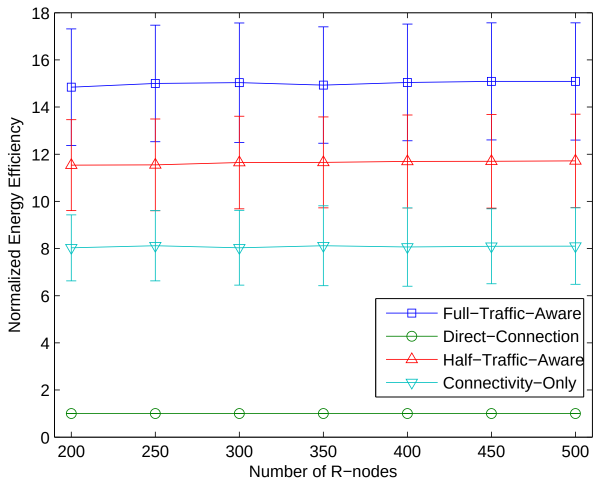


Figure 2.14: Normalized energy efficiency with different numbers of R-nodes by numerical analysis.

Connectivity-Only. Full-Traffic-Aware further raises the network lifetime gain up to 14 times and 6 times, respectively, which is 40% higher than Half-Traffic-Aware. This demonstrates the importance of considering the traffic patterns during both graph topology selection (finding merge vertices) and node deployment stages (discrete R-node assignment and merge vertex adjustments).

Fig. 2.13 shows the total residual energy with different numbers of R-nodes. Note that the value is a lower better. It is not surprising that Direct-Connection, Half-Traffic-Aware and our Full-Traffic-Aware solution have much less total residual energy than Connectivity-Only, since the energy consumption of the first three schemes is balanced by assigning more R-nodes to edges with higher traffic volumes. This also shows that the residual energy of Connectivity-Only increases much faster than the other three solutions. In addition, as Half-Traffic-Aware uses traffic-blind graph topologies as Connectivity-Only does, it runs the second higher. This also matches the residual energy distributions illustrated in Fig. 2.10 where Direct-Connection and our solution have more balanced distributions than Half-Traffic-Aware does.

Fig. 2.14 shows the energy efficiencies of different deployment strategies with different numbers of R-nodes. They are similar to the network lifetime results except that Connectivity-Only has much better energy efficiency than Direct-Connection does. This is because for the Connectivity-Only, most of R-nodes have not yet spent much energy when the first R-node dies. Nevertheless, our Full-Traffic-Aware solution delivers about 15 times as much traffic as Direct-Connection with the same amount of energy consumed. This shows that the extension of the network lifetime by our solution does not come at the expense of energy inefficiencies.

It is worth noting that although selecting different R-node numbers has only marginal impact on performance, there is a minimum number of R-nodes required to guarantee that the WSN system works well. This required minimum varies with the given S-node number, the locations of the S-nodes and their data rates, and the deployment strategy used. During our numerical analysis, we find that the Direct-Connection scheme always needs a higher minimum R-node number than the other three schemes. We thus derive an upper bound on the minimum R-node number required by Direct-Connection for a given S-node number and their locations and data rates. In particular, with the given parameter setting (the initial energy budget $\mathbb{E}$, required minimum network lifetime T_{min}, maximum communication range R_{max}, the number of S-nodes M and their locations s_i and data rates γ_i), an upper bound on the minimum R-node number N required for Direct-Connection can be derived by

$$\begin{cases} \left(\sum_{i=1}^{M} \sqrt[\alpha]{\gamma_i} \cdot \overline{s_i s_0} \right) / \overline{N} \leq \sqrt[\alpha]{\dfrac{\mathbb{E}}{T_{min}}} \\ n_i = \left\lceil \dfrac{\overline{N} \cdot \sqrt[\alpha]{\gamma_i} \cdot \overline{s_i s_0}}{\sum_{j=1}^{M} \sqrt[\alpha]{\gamma_j} \cdot \overline{s_i s_0}} \right\rceil \\ \max\limits_{i=1 \ldots M} \left(\dfrac{\overline{s_i s_0}}{n_i} \right) \leq R_{max} \\ N = \sum\limits_{i=1}^{M} n_i \end{cases}$$

We compute this bound for each test case and use the results as the default R-node number in the remainder of this section.

2.4.1.2 Scalability with S-node Number

We next investigate how our solution performs with different numbers of S-nodes. We also compute the results of the theoretical solution by Eq. 2.1, which serves as a bound to evaluate our practical solution on discrete R-node deployment. For all test cases ($M \leq 15$) that can be verified within acceptable timeframes, our hybrid algorithm successfully returns optimal graph topologies. In these cases the theoretical solution actually serves as an upper bound on the *optimal solution*.

Fig. 2.15 shows the results of the network lifetime with different numbers of S-nodes. Our Full-Traffic-Aware scheme is very close to the theoretical solution (within 13.5%) and it performs much better than the other solutions. As the number of S-nodes increases, the lifetimes of both Half- and Full-Traffic-Aware increase faster and are much higher than those of Direct-Connection

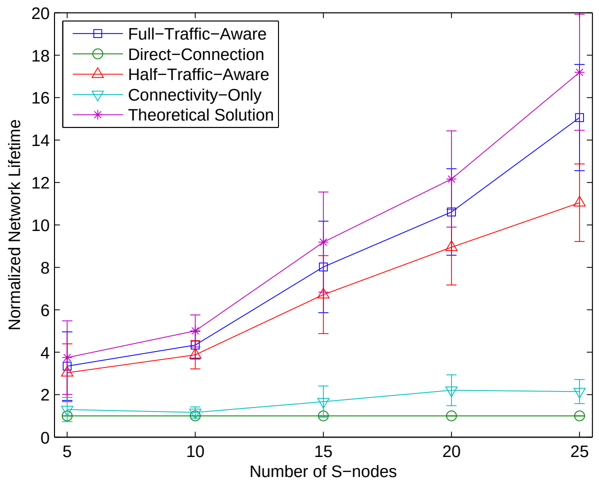


Figure 2.15: Normalized network lifetime with different numbers of S-nodes by numerical analysis.

and Connectivity-Only. One interesting observation is that the lifetime of Connectivity-Only first rises and then drops slightly as the number of S-nodes increases. A closer investigation reveals the reason behind this is that the energy hole phenomenon (see [53]) becomes more significant as the number of S-nodes increases. Fig. 2.10 shows the residual energy distributions of four deployment strategies on a test case of 15 S-nodes used in our evaluation. The energy hole problem can be clearly seen in Fig. 2.10b, where R-nodes close to the sink are depleted while most of other R-nodes still have more than 75% of their initial energy. As the number of S-nodes increases, more traffic will accumulate close to the sink. This dramatically reduces the lifetime if the deployment is not aware of such traffic accumulations as in the Connectivity-Only scheme. The other two schemes and our solution successfully avoid this problem by using algorithms that result in deploying more R-nodes close to the sink, as illustrated in Fig. 2.10a, Fig. 2.10c and Fig. 2.10d. In addition, there are still several edges with residual energy more than 50% of the initial energy in Fig. 2.10c. This is because Half-Traffic-Aware uses the same graph topology as Connectivity-Only which is computed without traffic-awareness.

Fig. 2.16 shows the total residual energy with different numbers of S-nodes. As the number of S-nodes increases, all five schemes behave similarly and are not very sensitive to change in the S-node number. As expected, the theoretical solution has the lowest residual energy, which is followed by Direct-Connection and our Full-Traffic-Aware solution. Half-Traffic-Aware runs forth due to its traffic-blind graph topology selection and Connectivity-Only performs even worse due to the lack

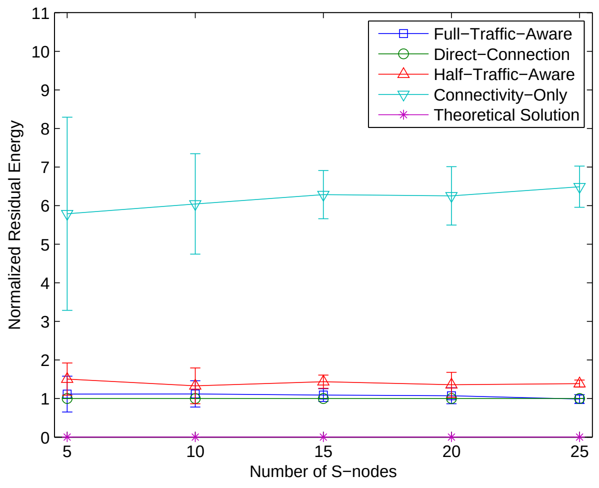


Figure 2.16: Normalized residual energy with different numbers of S-nodes by numerical analysis.

of traffic-awareness in both graph topology selection and node deployment stages.

Fig. 2.17 shows the energy efficiencies of different deployment strategies with different numbers of S-nodes. As in the analysis on the impact of different R-node numbers, energy efficiency also behaves similarly to network lifetime except that Connectivity-Only has much better energy efficiency than Direct-Connection. Our Full-Traffic-Aware solution performs nearly as well as the theoretical solution.

2.4.2 Simulation Results on ns-2

To further evaluate our solution, we conducted extensive simulations on ns-2 [3] which consider both sending and receiving energy consumption, wireless communication loss, collisions and other practical issues. A simple protocol is designed for data collection. The sink broadcasts a control message to start data collection. Each S-node then senses the environment at a predefined average rate and transmits the sensed data. Data losses are handled by both end-to-end and per-hop retransmissions. We modified the standard MAC layer to support dynamic adjustment of transmission ranges by using different power levels. For consistency, we use parameters adopted from Olariu and Stojmenovic [53] as in previous sections, i.e., $\alpha = 4$ and $c = 4500$ for both sending and receiving.

Fig. 2.18, Fig. 2.19 and Fig. 2.20 show the results of network lifetime, residual energy and

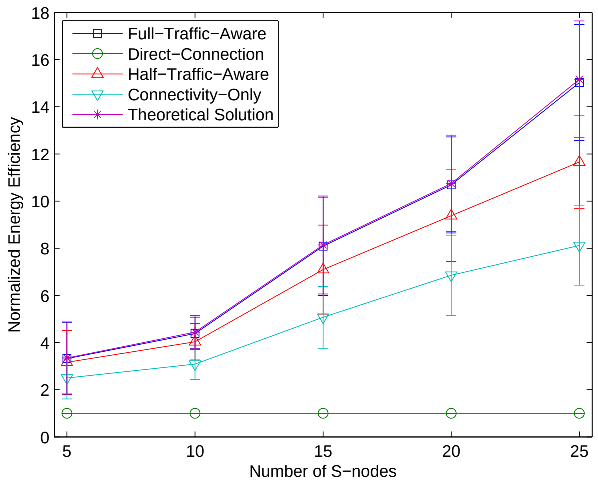


Figure 2.17: Normalized energy efficiency with different numbers of S-nodes by numerical analysis.

energy efficiency with different numbers of S-nodes, respectively. It is easy to see that our Full-Traffic-Aware solution performs best and achieves up to 7 times the lifetime of Connectivity-Only and 15 times that of Direct-Connection when the number of S-nodes increases to 25. On the other hand, the residual energy of our Full-Traffic-Aware solution is almost as low as that of Direct-Connection and is significantly less than that of the other two schemes. Full-Traffic-Aware also has the best energy efficiency. When the number of S-nodes is 25, the average amount of traffic delivered by per unit energy cost through our Full-Traffic-Aware solution is nearly 90% more than with the Connectivity-Only scheme and about 14 times more than the Direct-Connection scheme.

The simulation results match the numerical analysis well which validates the correctness and effectiveness of our approach and analysis. By a careful comparison, we find that the simulation results in general are slightly better than those of the numerical analysis, where the results of the total residual energy is the most obvious. A closer look reveals that due to communication range control, the wireless losses and collisions happen infrequently in all test cases. The only hot-spot identified is the area close to the sink in the Direct-Connection scheme. Recall the example shown in Fig. 2.10a, where the R-nodes in this area are very close to each other and easy to cause collisions even under communication range control. This slightly degrades the performance of Direct-Connection and also makes other schemes better after the normalization.

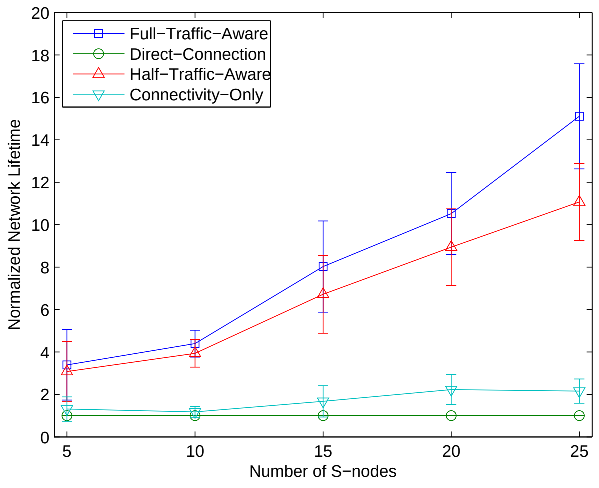


Figure 2.18: Normalized network lifetime with different numbers of S-nodes by ns-2 simulations.

2.5 Summary

In this chapter, we presented an in-depth study of the traffic-aware relay node deployment problem. We developed optimal solutions to maximize network lifetime for the case of one source node with either single or multiple traffic flows. We showed however that the general problem is difficult and that existing connectivity-guaranteed only solutions cannot be directly applied to it. We then transformed our problem into a generalized version of the Euclidean Steiner Minimum Tree problem (ESMT) and proposed a hybrid algorithm. To further improve performance, we also developed algorithms for discrete relay node assignment and further adjustments. We evaluated our solution by both numerical results and ns-2 simulations and observed that for all test cases where the continuous space optimal solution can be computed within acceptable timeframes, our solution is very close to the upper bound of the optimal solution. Our solution yields up to 14 times longer network lifetime than the Direct-Connection scheme and up to 6 times longer lifetime than a state-of-the-art Connectivity-Only algorithm.

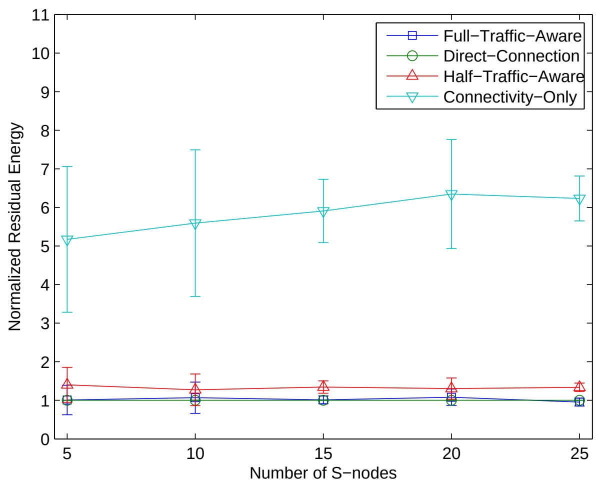


Figure 2.19: Normalized residual energy with different numbers of S-nodes by ns-2 simulations.

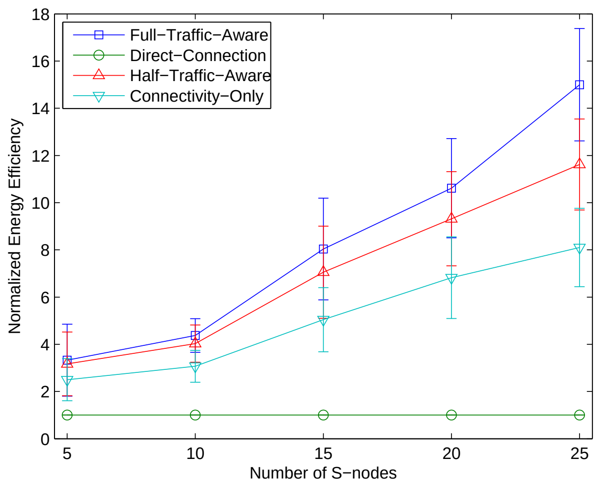


Figure 2.20: Normalized energy efficiency with different numbers of S-nodes by ns-2 simulations.

Chapter 3

Message Dissemination with Low Duty-Cycle

Message dissemination addresses how to disseminate network setup/management or collection command messages to all the sensor nodes efficiently with small transmission costs and low latencies. Previous message dissemination approaches generally assume that all network nodes are active during the dissemination process (referred to as *all-node-active assumption*). This assumption is valid for wired networks and for many conventional multi-hop wireless networks. However, this fails to capture the situation of energy-constrained wireless sensor networks. The sensor nodes often alternate between dormant and active states [23, 45, 80, 87]. In dormant state they go to sleep and consume little energy, while in active state they actively perform sensing and communication tasks, consuming significantly more energy (e.g., typical value of an active MicaZ mote is $56\ mW$ for IEEE802.15.4 radio plus 6 to $15\ mW$ for Atmel ATmega 128L micro-controller and possible energy cost on sensing devices). A low duty-cycle WSN clearly has a much longer lifetime for operation, but breaks the all-node-active assumption. More importantly, duty-cycles are often optimized for a given application or deployment and a message dissemination approach accommodating the schedules is thus expected for cross-layer optimization of the overall system.

In this chapter, we re-formulate the broadcast problem in low duty-cycle wireless sensor networks. To reflect the operational nature of real sensor products [1, 4] and also to simplify exposition, we divide time into equal-length *slots*. The active and dormant periods are both integer multiples of time slots and an active node can either receive or forward at most one message in each slot.

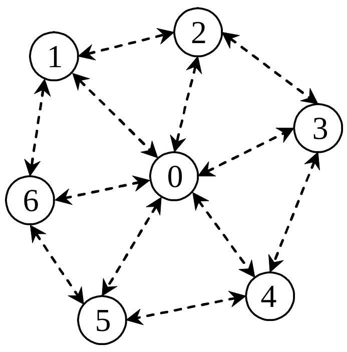


Figure 3.1: A duty-cycle-aware broadcast. We use dashed lines for communications links, reflecting that they are not always available in the presence of duty-cycles.

We do not assume any specific active/dormant schedule in our model. This brings two advantages: first, our solution is generally applicable to diverse schedules and second, our solution provides a generic tool for cross-layer optimization, i.e., for collaborative optimization between active/dormant schedule and broadcast service.

3.1 Motivation

We first consider a small motivational example shown in Fig. 3.1 in which sensor node 0 needs to broadcast a message to all other nodes (1 through 6). Assuming that there is no loss[1], it is easy to find a simple schedule: Node 0 waits until all neighbors wake up and then forwards the message. This strategy has minimum message cost, i.e., only one message is forwarded. However, since the nodes' active/dormant patterns may be noticeably different from each other, it may take a very long time for all of them to become active. At worst, if there is no overlap among their active periods, the time is infinite and the strategy does not work.

An alternative is that node 0 forwards the message as soon as one neighbor wakes up. The latency to accomplish broadcast is thus bounded by the time that the last neighbor is active together with node 0. Node 0 however has to forward the same message 6 times in the worst case.

The problem is further complicated if the broadcast is more than one hop; for example, if node 1 needs to broadcast a message to all others. In this case, for node 4 to receive the message, the

[1]For ease of exposition, we do not consider wireless communication losses at this stage. Loss-tolerant mechanisms will be discussed at the end of this section.

shortest path is through node 0 (a 2-hop path) if all nodes are active. In low duty-cycle networks, however, node 0 may wake up very late, and a 3-hop path through nodes 2 and 3 or through 6 and 5 might be faster. When the network size increases, the difference can be more noticeable, and it is more likely that an all-node-active based solution will fail to cover the whole network.

3.2 Problem Formulation

We now give a formal description of the duty-cycle-aware broadcast problem in wireless sensor networks. We will focus on the dissemination of a single message with a unique identifier (ID) from one source to all other nodes. By assigning different identifiers, our solution can be easily extended to broadcast a series of messages or to broadcast messages from multiple sources. We assume that there are n nodes in the network, indexed from 1 to n. For node i, $X_i(t)$ denotes its active/dormant state at time t, where $X_i(t) = 1$ if it is active and $X_i(t) = 0$ if it is dormant.

We represent the set of 1-hop neighbors of node i by N_i. These nodes can be directly covered by a message forwarded from node i if they are active. Here, we call 1-hop message broadcast transmission from a node to its neighbors a "*forward*", to distinguish this from a network-wide broadcast (or *broadcast* in short).

Without loss of generality, we assume that the message is to be broadcast from node s starting at time t_0. Let (u_i, t_i) denote the i-th forwarding (node u_i forwards the message at time t_i) and let C_i be the set of nodes that receive the message in the i-th forwarding. The problem can be formulated as follows:

Duty-Cycle-Aware Broadcast Problem:
Given node s to broadcast a message starting at time t_0, find a forwarding schedule

$$S = \{(u_1, t_1), (u_2, t_2), \ldots, (u_m, t_m)\} \quad (t_0 \le t_1 \le \ldots \le t_m)$$

that minimizes $f(|S|, t_m - t_0)$, a function of the total message forwarding cost ($|S|$) and the total latency ($t_m - t_0$).

The sequence must satisfy the following constraints:

(1) *Duty-cycle constraint*:

$$X_{u_i}(t_i) = 1, C_0 = \{s\}, C_i = \{j | j \in N_{u_i}, X_j(t_i) = 1\};$$

(2) *Forwarding order constraint*:

$$u_1 = s, \exists j, t_j < t_i, u_i \in C_j, i = 2, 3, ..., m;$$

(3) *Coverage constraint*:

$$\left| \bigcup_{i=0}^{m} C_i \right| = n.$$

The duty-cycle constraint requires that an active node u_i can successfully deliver the message to its neighbor, node j, at time t_i only if j is active at that time. The forwarding order constraint ensures that the message is forwarded hop-by-hop and that only a node that has previously received the message can forward it. Finally, the coverage constraint ensures that the broadcast will cover all the nodes.

The objective function depends on the forwarding cost and the latency and is generally specified by the target application. In this thesis, we will focus on a common linear combination, $f(|S|, t_m - t_0) = \alpha|S| + \beta(t_m - t_0)$. By assigning different weights (α, β), this covers the demands from a broad spectrum of applications. For example, if the broadcast message is about an emergency event and is of small size, a small α with a large β will ensure that the message is quickly delivered to the whole network, though possibly with higher forwarding cost. On the other hand, for large non-urgent messages, such as a code update, a large α with a small β will save forwarding costs and thus energy. It is worth noting that the optimal forwarding sequence, and hence its message and time costs, actually depends on the ratio α/β, and not on their absolute values. We will examine the impact of this ratio and recommend practical settings in Section 3.5.1.

3.3 Centralized Optimal Solution

We first transform the duty-cycle-aware broadcast problem into a shortest path problem in a time-coverage graph. Assuming that there is no wireless loss, given the network topology and the active/dormant patterns, this graph problem is solvable through a centralized dynamic programming algorithm. Its design principle also motivates the distributed implementation to be presented in the next section.

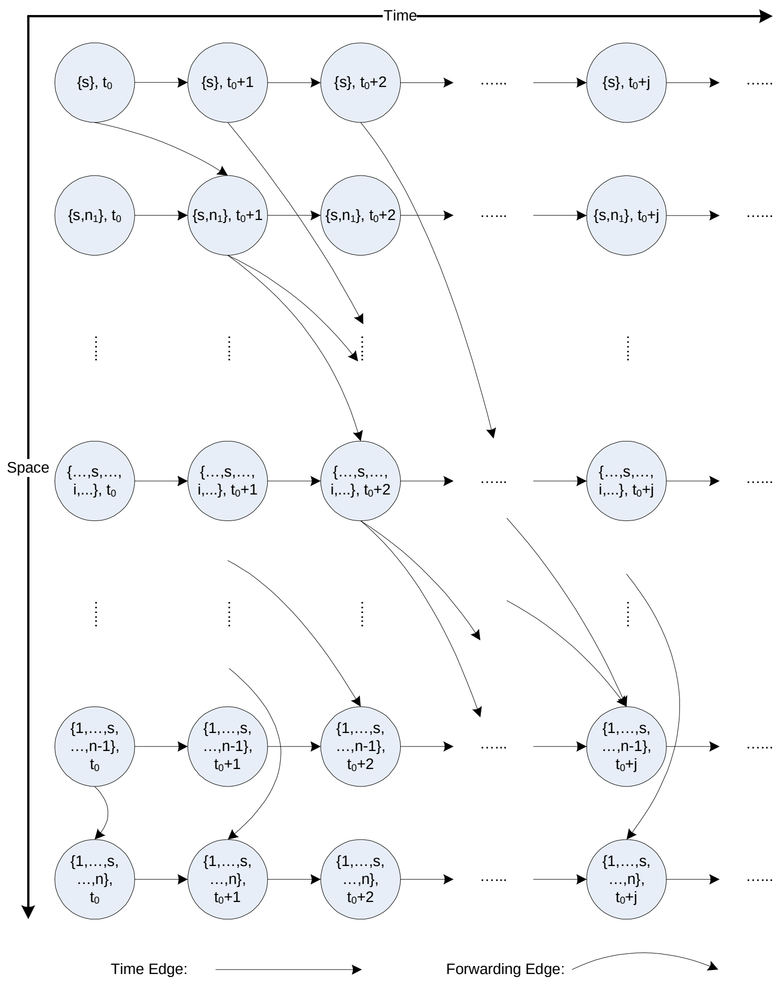


Figure 3.2: A constructed time-coverage graph.

3.3.1 Problem Transformation: Shortest Path to Last-Row

We construct a directed graph $G(V, E)$ as shown in Fig. 3.2 with vertices organized in two dimensions, indexed by time and space coverage, respectively. A vertex $v_{R,t}$ represents that, at time t, the sensor nodes in set R have received the broadcast message, i.e., been *covered*. The index R starts from $\{s\}$, and expands until it becomes $\{1, \ldots, n\}$. Obviously, each index R corresponds to a connected subnetwork in the original wireless sensor network and must include node s. Hence, although there are 2^n subsets of $\{1, \ldots, n\}$, the number of valid R's is much less.

There are two kinds of edges in the graph, referred to as *time edges* and *forwarding edges*. A time edge connects two neighboring vertices $v_{R,t}$ and $v_{R,t+1}$ along a row. This corresponds to the case that no node in R forwards the message at a time t and the state is unchanged in the next time slot. A forwarding edge, on the other hand, corresponds to forwarding events. Specifically, a forwarding edge from $v_{R,t}$ to $v_{R',t'}$ means that, at time t, one or more active nodes in R will forward the message leading to a larger coverage state set R'. Clearly, we have $R \subset R'$ and $R' - R$ is the set of nodes that receive the message in this round of forwarding. We set $t' = t + 1$ as the time index for the destination vertex in the graph. This ensures that a node can only forward a newly received message in a future time slot. The only exception is for vertices in the last row which correspond to the full coverage state and need no more forwarding. We thus set $t' = t$.

This time-coverage graph can be naturally related to the duty-cycle-aware broadcast problem: Each forwarding sequence corresponds to a path from the initial vertex $v_{\{s\},t_0}$ to a vertex in the last row, and vice versa.

For objective function $f(S, t_m - t_0) = \alpha|S| + \beta(t_m - t_0)$, we assign weight β to each time edge since a delay of one time unit is incurred and assign weight $(\alpha p + \beta(t' - t))$ to a forwarding edge from $v_{R,t}$ to $v_{R',t'}$ where p is the number of nodes in R that forward the message at time t. It is clear to see that the duty-cycle-aware broadcast problem is translated into the shortest path problem from $v_{\{s\},t_0}$ to a vertex in the last row in the weighted graph.

3.3.2 Dynamic Programming Algorithm

Let $W(v_{R,t}, v_{R',t'})$ denote the weight of the edge from $v_{R,t}$ to $v_{R',t'}$ and let $W(v_{R,t}, v_{R',t'}) = \infty$ if there is no such edge. Also let $F(v_{R',t'})$ be the total weight of the shortest path from vertex $v_{\{s\},t_0}$ to $v_{R',t'}$. We have the following recurrence relation:

$$F(v_{R',t'}) = \min_{v_{R,t}}(F(v_{R,t}) + W(v_{R,t}, v_{R',t'})),$$

where for $R' = \{1, \ldots, n\}$, we need either $R = R'$ and $t = t' - 1$ or $R \subset R'$ and $t = t'$; Otherwise $R \subseteq R'$ and $t = t' - 1$.

For boundary conditions, we have

$$F(v_{\{s\},t_0}) = 0,$$

and

$$F(v_{R,t_0}) = \infty, \quad \text{for } R \neq \{s\}.$$

Given the relation and the boundary values, we can compute the weight of the shortest path from $v_{\{s\},t_0}$ to each vertex from top to bottom and, for each row, from left to right. The minimum path length among the total weights to the last-row vertices is thus our expected result. The corresponding shortest path (as well as the forwarding schedule that reliably broadcasts the message to all nodes) can be derived by simple backtracking. We refer to this path as the *last row shortest path*.

Since the time dimension of the graph is infinite, there are potentially an infinite number of paths to the last-row. To overcome this problem, we introduce a terminating condition for each row, which guarantees that a shortest path to the last row can be found when the condition is satisfied for each row.

This condition is recursively defined as follows: A row indexed by R is *terminated* at time t if

(1) All the rows that have forwarding edges toward row R have terminated at some time t' before t and

(2) For any edge originated from a vertex of row R after t to a vertex of another row R', there must be at least one edge originated from a vertex of row R of the same or less weight and in time $(t', t]$ to a vertex of row R'.

The first row satisfies the first condition from time t_0. The last row always satisfies the second condition, as there is no forwarding edge from it. We will then have the following theorem.

Theorem 3.3.1 *A shortest path to the last row is found when the last row is terminated.*

Proof: The proof is by induction on the number of the forwarding edges used by a path to the last row. The key idea is that for any path found after a row is terminated we can always construct a path with less or equal total weight where all of its edges pass the row before the row is terminated. The full proof can be found in Appendix A. □

The theorem directly leads to a dynamic programming algorithm as shown in Fig. 3.3. Note that there are two strategies for calculating the recurrence relation: (1) starting from a vertex, find those vertices that have edges to it; and (2) starting from a vertex, follow its edges and find the vertices that these edges lead to. The second strategy is indeed much simpler and more efficient to implement. Specifically, it avoids unnecessary computation of those vertices with ∞ minimum cost because no path from $v_{\{s\},t_0}$ leads to them.

```
Algorithm OptimalForwardingSequence()
1:  $F(v_{\{s\},t_0}) \leftarrow 0$;
2:  $t = t_0$;
3:  while last row is not terminated,
4:    for $\forall$ row $R$ not terminated and $F(v_{R,t})$ exists,
5:      for $\forall$ $v_{R',t'}$ has an edge from $v_{R,t}$,
6:        if $F(v_{R',t'})$ does not exist,
7:          $F(v_{R',t'}) \leftarrow \infty$;
8:        end if
9:        if $F(v_{R,t}) + W(v_{R,t}, v_{R',t'}) < F(v_{R',t'})$,
10:         update $F(v_{R',t'})$;
11:       end if
12:     end for
13:     if row $R$ meets its terminating condition,
14:       terminate row $R$;
15:     end if
16:   end for
17:   $t \leftarrow t + 1$;
18: end while
19: find the minimum in last row;
20: return the last-row shortest path (correspond to the optimal forwarding sequence);
```

Figure 3.3: The dynamic programming algorithm to compute the optimal forwarding sequence.

3.4 Distributed Solution: A Scalable and Robust Implementation

Using the centralized optimal algorithm, it is easy to evaluate the lower bound on the latency or the message forwarding cost to cover a given network, as well as the trade-off between them. This gives a valuable benchmark to assess diverse broadcast strategies for duty-cycle-aware broadcast under ideal situations. It may also be practically useful for small networks with a centralized entity (e.g., the sink or base station) and for infrequent broadcast of large messages, e.g., a code image update. However, for large-scale networks the algorithm will have higher computation cost and, more importantly, it will be more difficult to obtain global connectivity and active/dormant patterns. The error-prone wireless communication raises additional challenges for information collecting and for reliable message forwarding.

In this section, we address these practical issues and present a distributed scalable solution which is resistant to wireless losses.

3.4.1 Generate Scalable Forwarding Sequence

For each sensor node, our distributed solution will focus on finding an optimal forwarding sequence covering nodes within 2 hops, that is, the *1-hop neighbors* of the node and their neighbors, which we call *2-hop neighbors*. The reasons to choose 2 hops are three-fold: First, it minimizes the computation overhead and yet keeps reasonable accuracy. Second, since every node must maintain information about its direct neighbors, the topology and active/dormant information for 1- and 2-hop neighbors can be obtained through a simple beacon protocol without any extra broad-scope protocols for information dissemination. Third, such information is sufficient to avoid most message forwarding contentions which will be further discussed in Section 3.6.

Assume that we are considering forwarding decisions at node w. We define a *Covering set*, or *CovSet*, to be the set of 1- and 2-hop neighbors that are known (by w) covered by at least one forwarding. A CovSet is created when a new broadcast message is received and is updated when node w forwards a broadcast message or a broadcast message is received or overheard. Specifically, when node w forwards a broadcast message, based on the active/dormant patterns of its neighbors, it will learn which neighbors are currently active and thus covered by this message and then add them to its CovSet. Similarly, when a broadcast message is received or overheard, node w will also learn which neighbors of the sender are currently active and add them to its CovSet. For example, in Fig. 3.1, if node 1 is active and nodes 0, 2 and 3 are also active, then after node 0 forwards the broadcast message and node 1 overhears it, nodes 2 and 3 will be included in node 1's CovSet.

The CovSet is node w's view of the broadcast coverage states of its 1- and 2-hop neighbors. Accordingly, we modify the dynamic programming algorithm so that, for node w, it will calculate the forwarding sequence starting from the row equal to its CovSet. The index of the last row will contain only node w and its 1- and 2-hop neighbors.

Another challenge in distributed implementation is that the sequence calculations at different nodes are not necessarily synchronized and are not even consistent. For example, the forwarding sequence calculated by node w might differ from the sequences calculated by it neighbors. To solve this inconsistency, when CovSet is updated, node w will check if this change is consistent with its current forwarding sequence. For example, if CovSet is changed due to an overheard message, node w then checks if this message is forwarded by the sender as indicated in its current forwarding sequence. If not, node w will re-compute the forwarding sequence by incorporating the updated CovSet. Since CovSet expands over time, in each re-computation the first row will become closer to the last row than previously, implying that the computation cost reduces over time.

3.4.2 Accommodate Wireless Losses

Wireless channels are by nature error-prone and thus a neighbor might not successfully receive the message even if it is placed into CovSet. For applications that require stringent coverage, we introduce a *Receiving Set*, or *RcvSet*, for each node w, as the set of 1- and 2-hop neighbors that are known (by w) having already received the message. Specifically, when node w receives a new broadcast message for the first time, it creates an RcvSet for this message and adds the sender of this message into the set. Afterwards, when node w receives or overhears the broadcast message from its neighbor, it will add this neighbor into its RcvSet if this neighbor is not in the set yet. And when the RcvSet includes all its 1-hop neighbors, which guarantees that all its 1-hop neighbors have received the broadcast message (in spite of wireless losses), node w will affirm that no more message forwarding is necessary for itself and thus can safely stop.

To expedite the update of RcvSet, when each node forwards the message it will piggy-back its RcvSet by a bitmap. Its 1-hop neighbors, upon receiving or overhearing the message, will also update their RcvSet according to the piggy-backed RcvSet. Our simulation results suggest that this strategy significantly mitigates the impact of wireless losses and by adding only a few explicit ACKs[2], we can expect ideal (100%) reliability within given delay bounds.

Note that both RcvSet and CovSet are updated from node w's perspective, which might not reflect the real receving/covering status. In particular, RcvSet might be a subset of CovSet only due to wireless losses. We use CovSet in the forwarding sequence calculation, since it is an optimistic estimation and is thus more efficient. However, to prevent CovSet from over-expanding (which would adversely affect the efficiency) we will reset CovSet to RcvSet periodically and also when the node turns active from the dormant state.

3.4.3 Summary

We summarize the core operations of the distributed solution in Fig. 3.4. When a node (e.g. node w) is in active state, it checks whether there is any arrived message (which can be either received or overheard). If so, node w will further process this message based on the message type. If the message is a new broadcast message, node w will create a RcvSet and CovSet for this message, and then add the sender of this message and the neighbors in the RcvSet piggy-backed with this message

[2]In our implementation, to accelerate the convergence of our solution, we also use a few explicit ACKs. For example, when node w receives a broadcast message only sent to itself (which can be determined by checking the piggy-backed RcvSet), it will send out an explicit ACK.

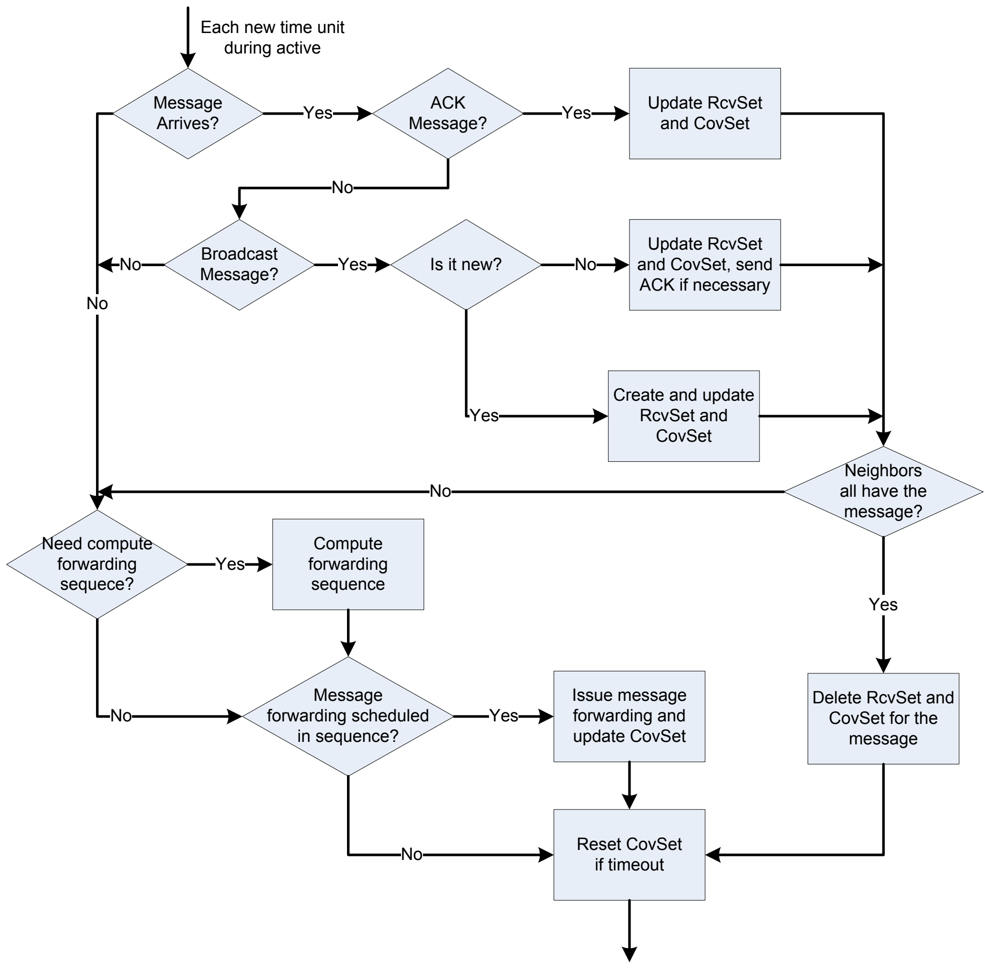

Each new time unit during active
Message Arrives?
Yes
ACK Message?
Yes
Update RcvSet and CovSet
No
Broadcast Message?
No
No
Yes
Is it new?
No
Update RcvSet and CovSet, send ACK if necessary
Yes
Create and update RcvSet and CovSet
No
Neighbors all have the message?
Need compute forwarding sequece?
Yes
Compute forwarding sequence
Yes
No
Message forwarding scheduled in sequence?
Yes
Issue message forwarding and update CovSet
Delete RcvSet and CovSet for the message
No
Reset CovSet if timeout

Figure 3.4: Operation of the distributed solution (for an active node).

into its own RcvSet and CovSet. In addition, node w also adds its neighbors that are currently active and covered by this message into its own CovSet. If the message is a broadcast message that has been received before, node w will then directly update the corresponding RcvSet and CovSet. Node w will also send an ACK if the message is directed only to w. On the other hand, if the arrived message is an ACK node w will update its RcvSet and CovSet by adding the sender of this message.

After processing the newly arrived messages, node w will then check its RcvSet to determine whether all its neighbors have received the broadcast message. If so, node w can safely stop forwarding the broadcast message and release the memory used to store the corresponding RcvSet and CovSet. Otherwise, node w checks whether its CovSet updating is consistent with its current forwarding sequence. If not, node w will further re-compute its forwarding sequence and if there is

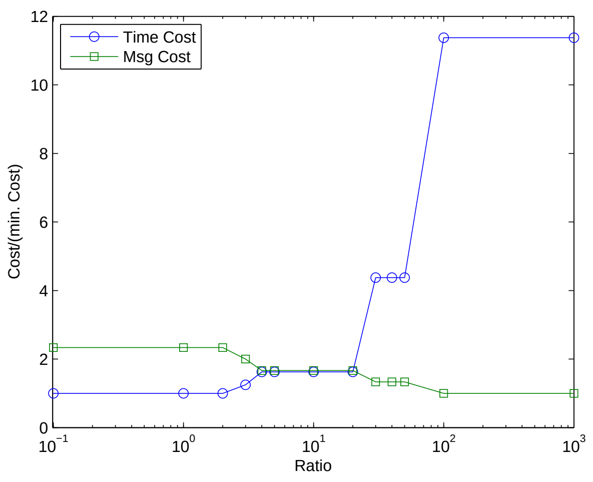


Figure 3.5: The impact of the α/β ratio.

any message scheduled to forward, node w will send out the message. Finally, CovSet will be reset to RcvSet if there is a timeout.

In the next section, we will use simulations to show that our duty-cycle-aware broadcast solution is not only scalable and robust against wireless loss but also retains near-optimal cost.

3.5 Performance Evaluation

In this section, we examine the performance of the proposed solution through extensive simulations. The following two major metrics are used in the evaluation: (1) Message cost, which is the number of forwardings (also reflecting the energy cost), and (2) Time cost, which is the total time slots taken to cover all the sensor nodes. We have examined diverse factors that impact the performance of our solution, including the duty cycle, network size/density and wireless communication losses. In this section, we present the results based on the following typical configurations, which are mainly adopted from Gu and He [22], Guo [25], Kyasanur *et al.* [37] and Stann *et al.* [68]. The sensing field is a square of size 200 m by 200 m and the wireless communication range is set to 10 m. The number of nodes in the network varies from 800 to 2000. For each setting, we randomly generated 10 topologies. Each data point presented in this section is the average of 10 topologies with 10 runs on each topology. The active and dormant patterns are randomly generated following the duty cycle value and exchanged among neighbors during the networking setup stage. We also adopt the

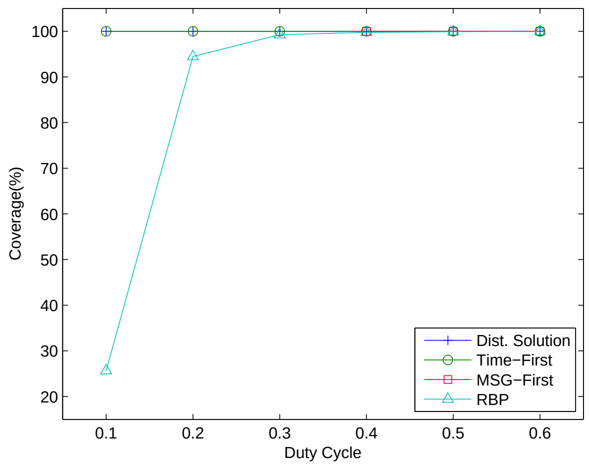


Figure 3.6: Reliability under different duty cycles.

wireless loss model used by Kyasanur *et al.* [37] where packets are randomly dropped based on a predefined packet error probability.

We do not assume any specific active/dormant schedule in our protocol design. We use various randomly generated schedules for performance evaluation and comparison.

3.5.1 Impact of the α/β Ratio

As mentioned earlier, the optimal forwarding sequence depends on the ratio between α and β but not on their exact values. We therefore first investigate the impact of this ratio. To minimize the influences from other uncertainties, we compute the time and message costs of different α/β ratios directly using the centralized solution and assuming that the complete global knowledge is known. The results are shown in Fig 3.5. For ease of comparison, the results are normalized by the respective minimum message and time costs. It is clear to see that as α/β increases, the message cost decreases. The time cost increases dramatically as α/β exceeds 20.

A close look into the forwarding sequences generated by the centralized solution reveals that most messages are forwarded by a node in one of two situations: (1) when all (or almost all) of its non-covered 1-hop neighbors are active at the same time; or (2) when the node or any of its non-covered 1-hop neighbors will turn dormant soon. This obviously can be locally determined, implying that our distributed solution could achieve good performance without global information. This will be confirmed in the following subsections.

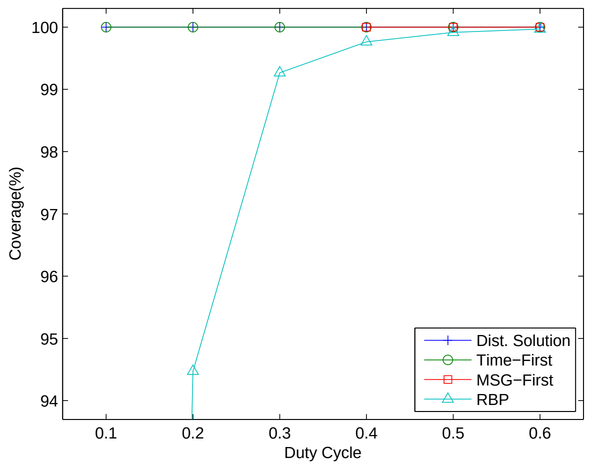


Figure 3.7: Reliability under different duty cycles (amplified).

Through the broadcast process, the proportion of (1) and (2) depends on α/β. An interesting situation occurs when α/β is around 10: Both time and message costs remain stable with relatively low values. In other words, the goals of reducing time and message costs are both achieved in this region, which follows our intuition that, except for extreme cases, fewer message forwardings are accomplished in a shorter time. Hence, for our distributed solution we use $\alpha/\beta = 10$ as the default setting. This enables a good trade-off and results that are consistent with other settings within this region.

To better understand the tradeoff and for comparison, we also checked two extreme cases where α (or β) is set to 0 and the other is greater than 0. These two settings lead to greedy strategies which come close to achieving the lower bounds on the time cost and message cost, respectively, and we thus refer to them as *Time-First* and *Message-First* strategies. In practice, they can be implemented by purely using the aforementioned operation (1) (for Message-First) or (2) (for Time-First) with little modification. We have also embedded the same loss-recovery mechanisms as in our distributed solution.

3.5.2 Adaptability to Duty-Cycle

We first examine the performance of our solution under different duty-cycles, especially under low duty-cycles. The network size is set to 2000 nodes. Based on the values reported by Stann *et al.* [68]

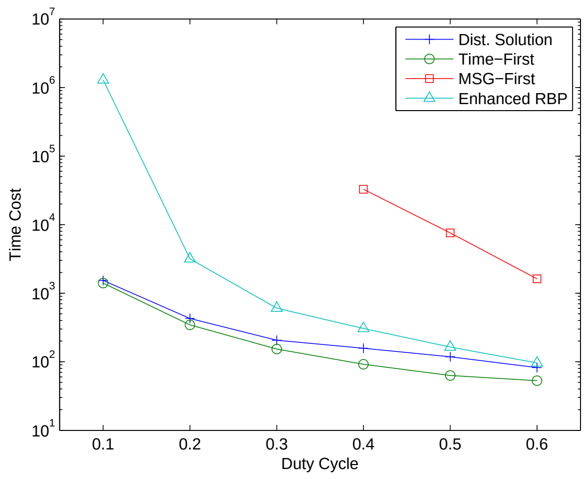


Figure 3.8: Time cost under different duty cycles.

on observed link loss for real-world sensor nodes, the wireless loss rate is set to 0.3. For comparison, we also implemented RBP (Robust Broadcast Propagation) [68], a state-of-the-art broadcast algorithm for WSNs. RBP is flooding-based with local repairs; it targets reliable broadcast but does not explicitly consider duty-cycles. We found that with low duty-cycle, this original RBP performed poorly, even failing to achieve the primary goal of reliability. The results are shown in Fig. 3.6 and detailed in Fig. 3.7 for duty-cycles from 0.2 to 0.6. In contrast to the 100% reliability achieved by our distributed solution and the Time-/Message-First strategies, the reliability of RBP is unacceptable when the duty-cycle is below 0.5.

It is also worth noting that when the duty cycle is lower than 0.4 Message-First fails to terminate in finite time because the probability for all non-covered neighbors being active simultaneously is extremely low. As such, its results for these extremely low duty-cycles are not shown in the figures. In fact, the strategy used in Message-First, i.e., to wait until all non-covered neighbors become active, is our initial attempt to modify RBP to accommodate duty-cycles. Unfortunately, our results suggest this intuitive approach does not work in this new network context.

To achieve a fair comparison, we further enhance RBP by reissuing a broadcast immediately after the previous one until the required coverage reliability is achieved. This is motivated by the reliability compensation technique proposed by Stann *et al.* [68]. The results of the comparison with this Enhanced RBP are shown in Fig. 3.8 and Fig. 3.9 for time and message costs, respectively. We can see that for both time and message costs, our distributed solution outperforms the Enhanced

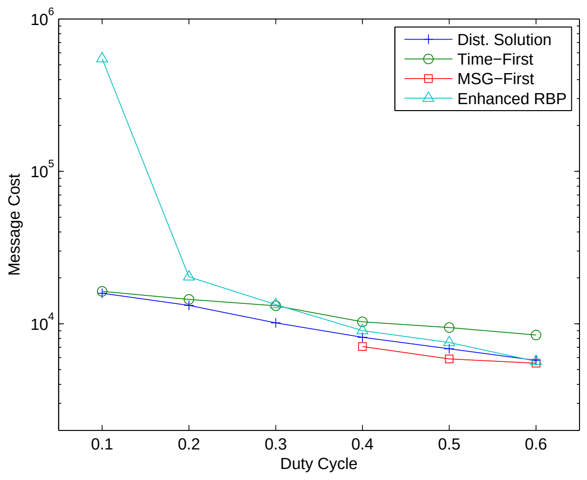


Figure 3.9: Message cost under different duty cycles.

RBP and is close to the practical lower bounds on time and message costs given by the Time- and Message-First strategies, respectively. Moreover, although the Enhanced RBP performs similarly to our distributed solution under moderate duty-cycles, say 0.6, its performance degrades dramatically as the duty-cycle becomes extremely low. Note that the y-axis is in log-scale and the x-axis is from low duty-cycle to high. This further demonstrates the challenges caused by the loss of the all-node-active assumption and the necessity of designing a new broadcast service for low duty-cycle WSNs.

Compared to the Enhanced RBP and the Time-/Message-First strategies, our distributed solution is actually self-adaptive to different duty-cycles. When the duty-cycle increases, i.e., more opportunities to cover multiple neighbors with one forwarding, our solution successfully captures these opportunities and behaves like Message-First with very low message cost. On the other hand, when the duty-cycle becomes extremely low, our solution does not waste time to blindly wait for such opportunities and performs more like Time-First to achieve low time cost.

In short, dropping the all-node-active assumption renders the existing approaches (e.g., the original RBP) to be suboptimal or even failed under low duty-cycles. Such extensions as the Enhanced RBP remain ineffective. On the other hand, our distributed solution adapts well to different duty-cycles, with costs being close to the lower bounds on both time and message forwarding.

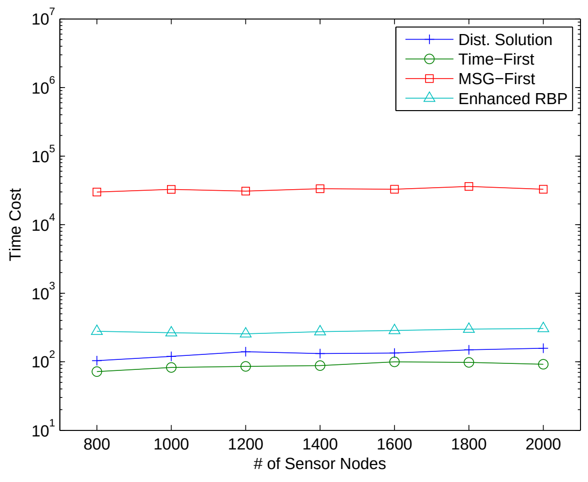


Figure 3.10: Time cost with different numbers of sensor nodes.

3.5.3 Scalability with Network Size

We next evaluate how performance changes with different numbers of sensor nodes. We vary this number from 800 to 2000 and the size of the sensing field is changed accordingly to keep the density constant. The impact of the different node densities will be investigated in the next subsection. Fig. 3.10 and Fig. 3.11 show the results for a default wireless loss rate of 0.3 and a duty-cycle of 0.4. It is clear to see that the time cost of our solution is close to Time-First and much less than those of Enhanced RBP and Message-First. Meanwhile, the message cost of our solution is much less than those of Enhanced RBP and Time-First and is close to that of Message-First. This in turn justifies the existence of a stable region for α/β ratio selection, where both the time and message costs stay relatively low in spite of the network size change.

As the number of nodes increases, the time consumed by our distributed solution, Time-First, and Enhanced RBP all slowly increase (see Fig. 3.10). This is because the time cost increment introduced by the increasing network diameter is insignificant compared to the time cost caused by waiting for nodes to turn active. This is more noticeable for the Message-First approach. In that case, the time cost spent by a node to wait for all its non-covered neighbors to become active together dominates the total time cost. This leads to an almost constant cost with only marginal variations.

The message costs of all four approaches increase almost linearly with the number of nodes.

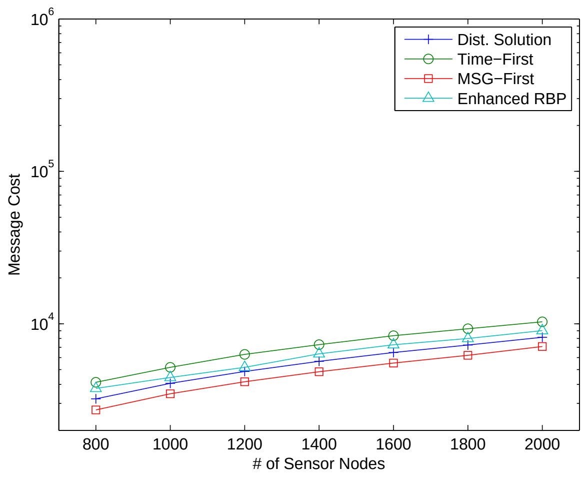


Figure 3.11: Message cost with different numbers of sensor nodes.

Note that it looks sub-linear in Fig. 3.11 due to the log-scale of y-axis. However, our distributed solution grows relatively slower than either Time-First or Enhanced RBP, and is close to Message-First. This implies that our distributed solution is scalable with the network size.

3.5.4 Effect of Network Density

We now examine the effect of different node densities in this subsection. To this end, we set the number of nodes to 2000 and vary the size of the sensing field to change the node density from 3 nodes per $100\ m^2$ to 8 nodes per $100\ m^2$. Fig. 3.12 and Fig. 3.13 give the results for a default wireless loss rate of 0.3 and a duty-cycle of 0.4. It is easy to see that the time cost of our solution is very close to that of Time-First and much lower than those of Message-First and Enhanced RBP in spite of the network density changes. Also, the message cost of our solution stays close to that of Message-First in most cases and is always less than those of the Time-First and Enhanced RBP strategies.

An interesting observation is that as node density increases, the time costs of the four approaches demonstrate different changing patterns. Specifically, our solution, Time-First and Enhanced RBP have lower time costs for higher densities while Message-First suffers higher time cost. The reason for this is that when the node density increases, a node will have more neighbors, which makes that node more likely to be covered by a message forwarding from its neighbors and thus reduces the

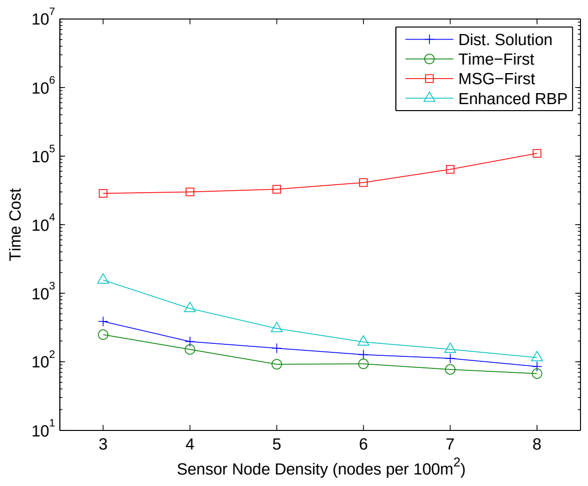


Figure 3.12: Time cost with different sensor node densities.

time cost. On the other hand, in the Message-First strategy, having more neighbors implies that a node has to wait longer for its non-covered neighbors to be active at the same time and, as a result, the total time cost is increased. Another observation is that Enhanced RBP behaves differently for low and high node densities. This can be more clearly seen in Fig. 3.13, as its message cost below the density of 5 nodes per 100 m^2 drops very quickly and then keeps almost constant afterwards. This is due to the local topology adaptation mechanism in RBP, which conducts local repairs by more retransmissions in low node density areas and fewer retransmissions in high node density areas. Since the nodes are randomly deployed in our evaluation, as the average node density increase, high node density areas become more prevalent and then dominate for average density of more than 5 nodes per 100 m^2.

3.5.5 Robustness against Wireless Loss

We also investigate the impact of the wireless communication loss rate. Fig. 3.14 and Fig. 3.15 show the results for a network size of 2000 nodes and duty-cycle of 0.4. Again, for all different loss rates, our distributed solution outperforms both Message-First and Enhanced RBP and is close to Time-First in time cost. In message forwardings, our distributed solution is close to Message-First and is much lower than both Time-First and Enhanced RBP.

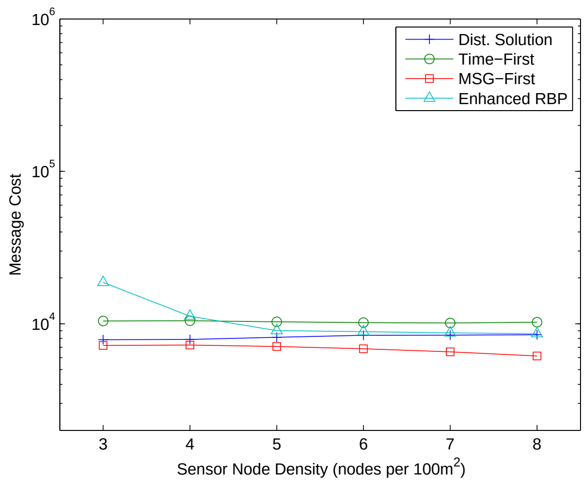


Figure 3.13: Message cost with different sensor node densities.

Furthermore, Fig. 3.14 shows that the time cost of Message-First increases faster (note that y-axis is in log-scale) when the wireless loss rate is greater than 0.1. This is because for the Message-First strategy, if a forwarded message gets lost it takes a long time for a node to wait for all of its non-covered neighbors to become active again to allow another forwarding. With the wireless loss increased, both the probability of such situations occurring and the number of tries raise sharply. Similar situations also happen in Enhanced RBP where when a node misses a broadcast due to wireless loss, it may take quite a few time for next reissued broadcast to arrive at this node. On the other hand, the time costs of Time-First and our distributed solution increase very slowly when the wireless loss rate is less than 0.4. This is because in a low duty-cycle environment, the broadcast messages often have to be forwarded multiple times to cover all the nodes in an area. Due to the broadcast nature of wireless communications, a node may receive or overhear more than one broadcast message forwarding during its active periods. Thus, when the wireless loss rate is low, even if a node fails to receive a forwarded message due to wireless losses, it still has a good chance to receive the message in time by the next few forwardings (though these forwardings may be scheduled to cover other nodes in the area and not intended specifically for this node). However, to cover all the nodes in an area, Time-First requires more messages while our distributed solution successfully balances the time and message costs. Our solution reacts to a loss increase with only a small number of extra message forwardings and achieves a much lower time cost.

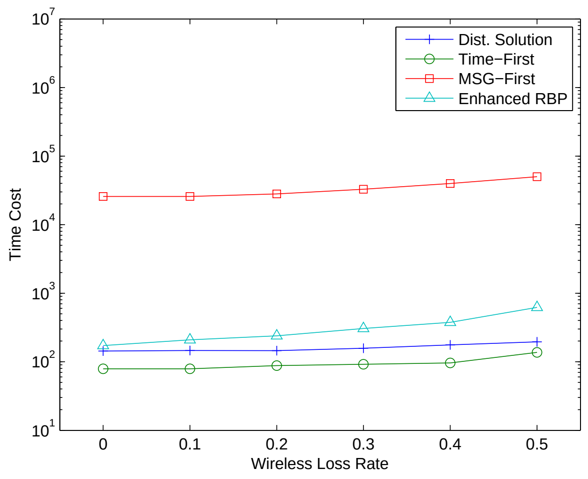


Figure 3.14: Time cost under different wireless loss rates.

3.6 Further Discussion

In summary, our distributed solution achieves near optimal performance: Our time cost is very close to that of the Time-First strategy which is a lower bound on the time cost; and our message cost is very close to that of the Message-First strategy which is a lower bound on the message cost. Yet, given that it involves local operations only, its computation and control overheads both scale well with the network size and density. It is also robust against wireless losses and copes well with different duty-cycles.

It is worth noting that the broadcast module is intended to serve diverse applications. Hence, its control information could be piggy-backed or be overheard during the data transmissions of the served applications. This cross-layer optimization will further reduce the cost and enhance the responsiveness of our solution. Next, we briefly discuss two additional practical concerns for deploying our solution.

Collision reduction: Collision frequently occurs in multi-hop wireless networks, particularly during data bursts such as broadcast. For example, in Fig. 3.1, if node 1 and 4 forward messages to their neighbors simultaneously then node 0 may get nothing because the two forwardings collide with each other. Existing broadcast algorithms rely passively on the MAC layer to avoid or resolve

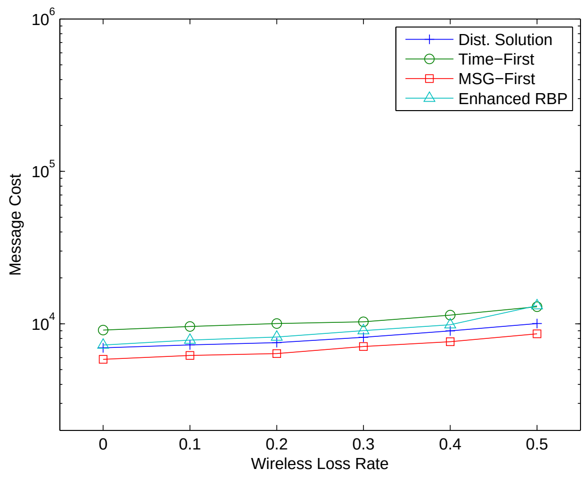


Figure 3.15: Message cost under different wireless loss rates.

collisions [58, 60, 61]. This is indirect and thus can be inefficient. Our solution can pro-actively detect forwarding edges that may lead to collisions and remove them[3]. Furthermore, the asynchronous active/dormant patterns inherently reduce the chances of collision in low duty-cycle networks. We have verified this in our experiments and have found that the collision probability becomes orders of magnitude lower than that in all-node-active networks. This is also observed by Gu and He [22]. It thus becomes a less significant problem.

Flexible reliability: So far we have tried to achieve perfect reliability for broadcast. This is not mandatory in many applications which seek better trade-offs among reliability, efficiency, and delay. Our solution can easily be modified to explore such flexibility. Specifically, it can terminate after all the neighbors not in RcvSet have been covered at least k times (where k is an application-controlled parameter) or when $x\%$ of the neighbors of a node are in its RcvSet (where x can be determined based on local topology information, e.g., using approaches proposed by Stann *et al.* [68]). Our experience has shown that by setting the coverage to 99% of the nodes the time cost can be reduced by 50% to 75% in many low duty-cycle networks. This is because, in such networks, a small portion of nodes with low degree would have low probabilities to activate at the same time as their neighbors

[3]For the collisions of ACKs, since only a few explicit ACKs are used in very specific cases, we assume that the collision probability would be marginal. Even when a collision happens for an ACK sent by a node, our solution can still work well: After the collision, another broadcast message only directed to that node will be forwarded later. This will trigger a second ACK to be sent out.

and consequently a huge time margin has to be spent to cover these nodes.

3.7 Conclusion

In this chapter, we revisited the message dissemination problem in wireless sensor networks with active/dormant cycles. We demonstrated that under low duty-cycles conventional broadcast strategies assuming all-node-active would either suffer from poor performance or simply fail to cover the network. We took the initiative to remodel the broadcast problem in this new context. We showed that it is equivalent to a shortest path problem in a time-coverage graph and presented an optimal centralized solution. This solution has also motivated a distributed implementation that relies only on local information and operations. We examined the performance of our solution under diverse network configurations and compared it with state-of-the-art solutions.

Chapter 4

Elevator-Assisted Data Delivery

Data delivery is the most important stage of data collection applications as it addresses the main issue of how to deliver sensing data from each sensor node to the base station. Since data are required to be delivered to the base station without any information loss, data aggregation/fusion operations [59] are hard to apply. Thus when being relayed hop by hop towards the base station, sensing data from different nodes will merge causing heavier traffic on links close to the base station. This will quickly consume the energy of those sensor nodes that relay the data. Such situations may also occur in other WSN applications where controllable mobile base stations pro-actively move within the sensing field to communicate with sensor nodes for relaying traffic [65]. However, when the environment is harsh and to minimize disturbance, such a solution is often infeasible for data collection.

In one typical data collection application, structural health monitoring (SHM), a high-rise structure is monitored with the sensing data collected from different locations of the structure. Figs. 4.1 and 4.2 show the SHM system deployed on the Guangzhou New TV Tower (GNTVT) in Guangzhou, China, a project in which we participated [2]. Even during the construction phase, the tower was equipped with vibrating wire strain gauge sensors and temperature sensors (Fig. 4.1) to monitor its construction status. After becoming fully operational in November 2010, more advanced sensing devices such as accelerometers and corrosion sensors were deployed on the tower (Fig. 4.2) to monitor its operation and service. As the world's tallest TV Tower, the top of the GNTVT is 600 meters above the ground. Although its horizontal dimension is similar to a normal building (varying from $50m \times 80m$ to $20.65m \times 27.5m$ on different floors), its extreme height creates enormous challenges for sensor data collection beyond those addressed in state-of-the-art mote-like systems. One example is the aforementioned traffic accumulation problem. As data aggregation in SHM is

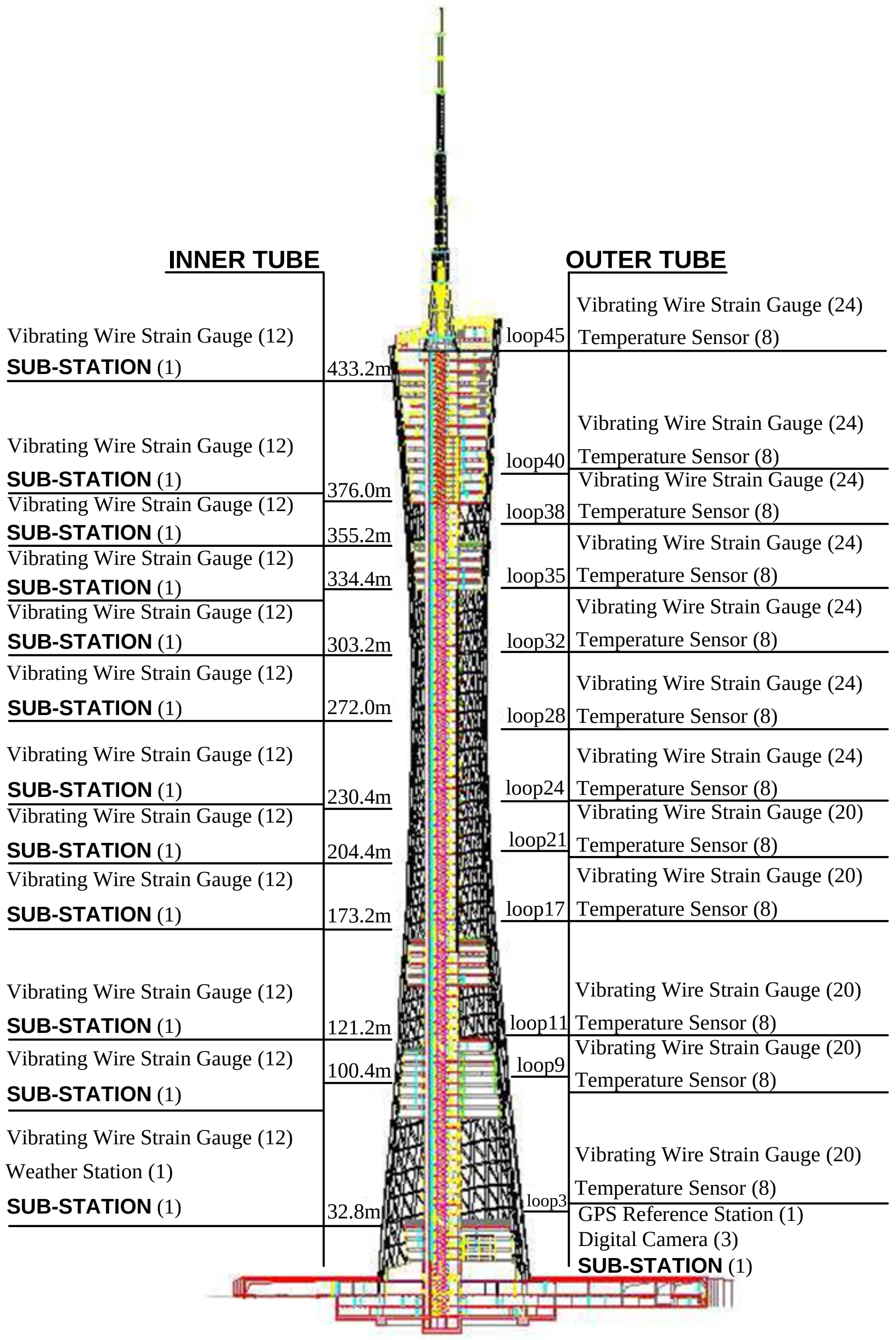

INNER TUBE
OUTER TUBE
Vibrating Wire Strain Gauge (12)
SUB-STATION (1)
433.2m
Vibrating Wire Strain Gauge (12)
SUB-STATION (1)
376.0m
Vibrating Wire Strain Gauge (12)
SUB-STATION (1)
355.2m
Vibrating Wire Strain Gauge (12)
SUB-STATION (1)
334.4m
Vibrating Wire Strain Gauge (12)
SUB-STATION (1)
303.2m
Vibrating Wire Strain Gauge (12)
SUB-STATION (1)
272.0m
Vibrating Wire Strain Gauge (12)
SUB-STATION (1)
230.4m
Vibrating Wire Strain Gauge (12)
SUB-STATION (1)
204.4m
Vibrating Wire Strain Gauge (12)
SUB-STATION (1)
173.2m
Vibrating Wire Strain Gauge (12)
SUB-STATION (1)
121.2m
Vibrating Wire Strain Gauge (12)
SUB-STATION (1)
100.4m
Vibrating Wire Strain Gauge (12)
Weather Station (1)
SUB-STATION (1)
32.8m
loop45 Vibrating Wire Strain Gauge (24) Temperature Sensor (8)
loop40 Vibrating Wire Strain Gauge (24) Temperature Sensor (8)
loop38 Vibrating Wire Strain Gauge (24) Temperature Sensor (8)
loop35 Vibrating Wire Strain Gauge (24) Temperature Sensor (8)
loop32 Vibrating Wire Strain Gauge (24) Temperature Sensor (8)
loop28 Vibrating Wire Strain Gauge (24) Temperature Sensor (8)
loop24 Vibrating Wire Strain Gauge (24) Temperature Sensor (8)
loop21 Vibrating Wire Strain Gauge (20) Temperature Sensor (8)
loop17 Vibrating Wire Strain Gauge (20) Temperature Sensor (8)
loop11 Vibrating Wire Strain Gauge (20) Temperature Sensor (8)
loop9 Vibrating Wire Strain Gauge (20) Temperature Sensor (8)
loop3 Vibrating Wire Strain Gauge (20) Temperature Sensor (8)
GPS Reference Station (1)
Digital Camera (3)
SUB-STATION (1)

Figure 4.1: Sensor layout on the Guangzhou New TV Tower: In-construction monitoring.

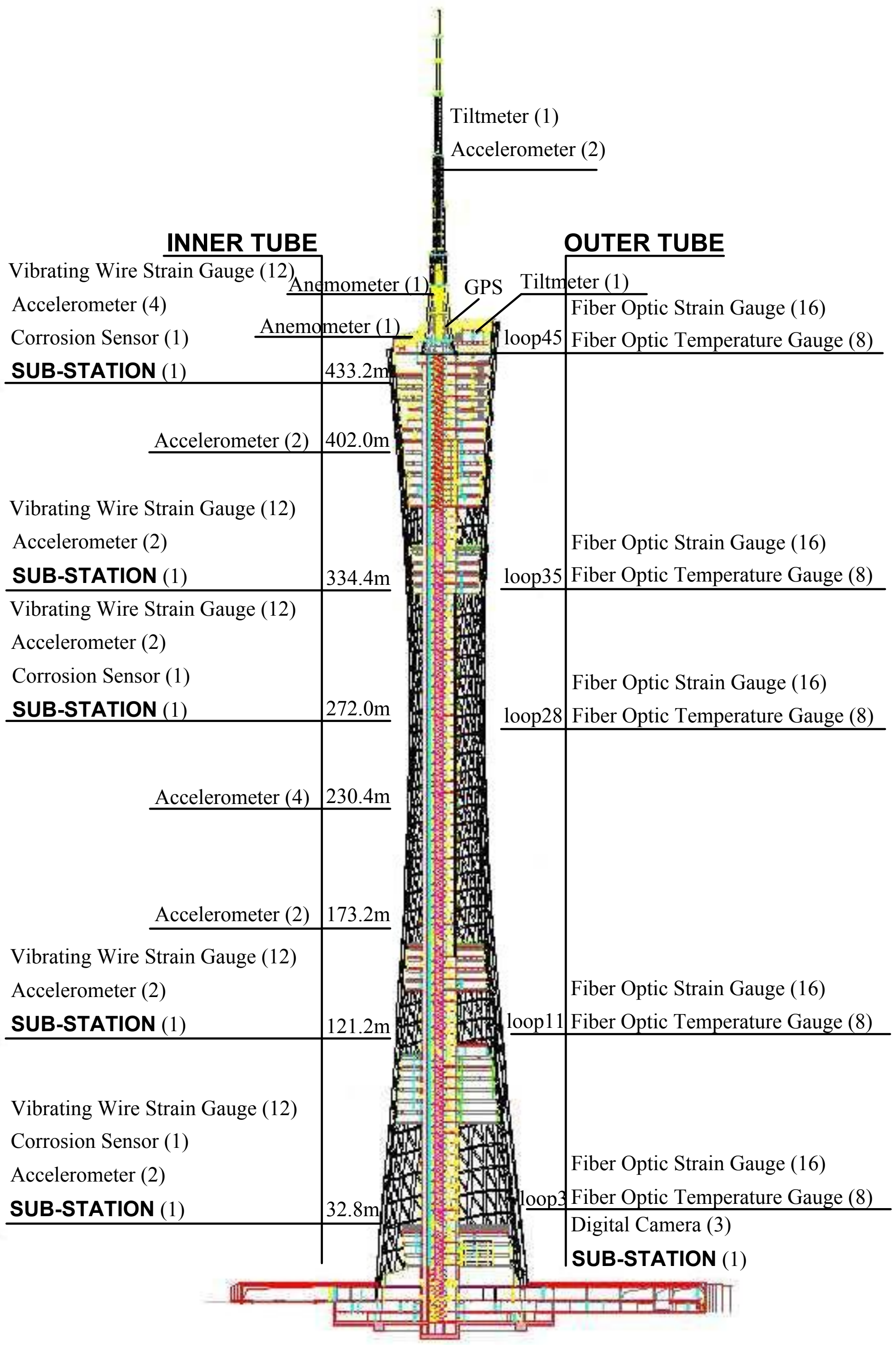


Figure 4.2: Sensor layout on the Guangzhou New TV Tower: In-service monitoring.

not possible currently [39], given the extreme height of the high-rise structures, neither a strategy of long-range one-hop data transmission (partially adopted by the GNTVT in its early stage) nor short-range hop-by-hop communication is cost-efficient. Recent studies combine these two strategies by adjusting wireless communication ranges and/or adding more relay nodes to obtain a more efficient system [28, 83]. However, the intrinsic difficulty remains, i.e., the larger the structure, the longer the distances from the sensors to the base station. This has prevented the GNTVT to benefit from a full-range wireless sensor system.

These high-rise structures, however, require elevators. We thus propose *EleSense*, a novel high-rise structure monitoring framework that uses elevators with minimized disturbance to the sensing environment. In EleSense, a base station is attached to an elevator and can collect data as the elevator moves between floors as it serves passengers. With this scheme, communication distances between sensor nodes and the base station can be greatly reduced and the traffic accumulation problem can be effectively alleviated.

To achieve optimal performance in this unique application, there remain a series of theoretical and practical issues to be addressed. In particular, the motions of a passenger can hardly be predicted in advance. When there is no passenger the elevator may wait at a floor for an unpredictable time. In short, the sensor nodes at different floors may experience different opportunities to transmit data to the base station. Fairness and rate control are difficult to achieve. The problem is further complicated by dynamic wireless interference and collisions due to the moving base station and the limited power of wireless sensor nodes.

In this chapter, we first discuss the background of and challenges to high-rise structure monitoring. We then present a model of the problem as a cross-layer optimization problem. We propose a centralized optimal solution and a practical distributed implementation to solve this problem both theoretically and practically.

4.1 Background and Challenges

In SHM applications, sensors are deployed at critical locations to sample data periodically. For high-rise structures, a commonly adopted data collection strategy is to assign a representative node (e.g., a sub-station in Figs. 4.1 and 4.2) on each floor to collect all of the data from the sensors on this floor. These representative nodes transmit the data back to the base station located at the foot of the structure. Conventionally, data transmission is carried out by wires. For a life-long monitoring system, a wire-dominated system may still be a reasonable choice. In this thesis we focus on the short term (weeks or months) evaluation of structural health [39]. In this scenario, sensor networks are quickly deployed to acquire the data necessary for calibrating structural health models for civil engineering. Based on the calibration, the deployment may be adjusted and new data collection issued for further verification and calibration until civil engineering requirements are met. Such an iterative procedure often starts during the construction of a structure, making a wired system very expensive to deploy. Another major problem is in-construction monitoring where wires could easily be damaged by hammers or drills during construction. Wireless systems are thus welcomed for these situations especially considering the recent development of the state-of-the-art sensor systems.

There are two possible data collection strategies for wireless communication systems: Short-range hop-by-hop routing and long-range single-hop transmission. Long-range single-hop transmission (partially adopted by the GNTVT in its early stage) is costly in communication devices and suffers greatly if the energy supply is limited or difficult to obtain. Hop-by-hop routing is costly for nodes close to the base station as they need to relay large amounts of data[1]. This may cause severe interference and numerous collisions at these nodes as well as unbalanced energy consumption that may shorten the network lifetime. Although carefully combining these two strategies may alleviate these problems, the intrinsic difficulty still remains, i.e., the larger the structure, the longer the distances from the sensors to the base station.

To this end, we propose EleSense as a generic framework for high-rise structure monitoring. In EleSense, the base station is installed on an elevator used by the structure. As passengers go to different floors, the base station moves with the elevator and collects data packets from the sensor nodes that it passes by. Nevertheless, there are still many challenges to be addressed. First, we cannot predict the motions of the elevator. Sensor nodes cannot wait indefinitely for the elevator to move within range. Though currently we focus on non-real-time SHM applications where sensing data can be stored at the local external flash memory for later transmissions, collecting data as quickly as possible is still desirable. Thus, the sensor nodes must decide whether to route the packets through neighboring nodes or to wait for the base station to come within range. When sending data packets, a sensor node must carefully schedule its transmissions to avoid wireless interference and collisions. When a packet is relayed by a neighboring node, the queuing buffer size at that node should also be considered. In the next section we will formally model the high-rise structure monitoring problem by taking these issues into account.

[1]To the best of our knowledge, in civil applications, data aggregation in intermediate nodes is not practical at this time. For background on civil data evaluation, one may refer to Li *et al.* [39].

4.2 Problem Formulation

Assume that there are n floors with one sensor node on each floor[2], denoted as $s_1, s_2, \ldots, s_n$. Let s_0 be the base station attached to the elevator. Let $l_{x,y}$ denote the directed link from node x to node y where $x, y \in \{s_0, s_1, \ldots, s_n\}$. There are $n(n+1)$ directed links among these sensor nodes. We assume that nodes on adjacent floors are within each other's transmission ranges and can communicate. However, this may not be true at nodes on more distant floors. Further, the link between a node and the base station may be unavailable when the base station is far away. We thus define $A(l_{x,s_0}, t)$ to describe the availability of the links to the base station at each time t. $A(l_{x,s_0}, t) = 1$ indicates that at time t the link between node x and the base station is available and $A(l_{x,s_0}, t) = 0$ indicates that it is not. Due to interference and collisions, some links cannot transmit data simultaneously. We define an interference matrix, IM, and let $IM(l_{x_1,y_1}, l_{x_2,y_2}) = 1$ if links l_{x_1,y_1} and l_{x_2,y_2} interfere with each other when transmitting simultaneously. Otherwise, $IM(l_{x_1,y_1}, l_{x_2,y_2}) = 0$.

We assume that all data packets have the same length and define a *time unit* to be the minimum time for a link to reliably transmit a data packet. In SHM applications, data is sampled periodically. We call every period a *round*. Let the data at sensor node s_i be generated at a rate of r_i packets per round for $i = 1, 2, \ldots, n$. Assume that each sensor node can buffer at most B extra data packets. Let t_0 denote the start time of a data collection.

The high-rise structure monitoring problem that exploits elevators can thus be formulated as a cross-layer optimization problem to find a *link-activation schedule* $S = \{(l_{x_1,y_1}, t_1), (l_{x_2,y_2}, t_2), \ldots, (l_{x_k,y_k}, t_k)\}$, $t_0 \le t_1 \le \cdots \le t_k$, subject to the following constraints:

(1) *Link Availability*:

$$\forall (l_{x,y}, t) \in S, \text{ if } y = s_0, \text{ then } A(l_{x,y}, t) = 1 \;;$$

(2) *Link Interference*:

$$\forall (l_{x_i,y_i}, t_i), (l_{x_j,y_j}, t_j) \in S, \text{ if } t_i = t_j, \text{ then } IM(l_{x_i,y_i}, l_{x_j,y_j}) = 0 \;;$$

[2]EleSense can easily be extended to a hierarchial architecture (as the example of the GNTVT) where sensor nodes on the same floor form a cluster with sensing data sent to the representative node (sub-station) by local communications.

(3) *Packet Transmission*:

$$\forall t \in [t_0, t_k], i = 1, 2, \ldots, n,\ r_i + \sum_{(l_{x_j, y_j}, t_j) \in S} I_{[y_j = s_i, t_j \leq t]} \geq \sum_{(l_{x_j, y_j}, t_j) \in S} I_{[x_j = s_i, t_j \leq t]}\ ;$$

(4) *Packet Buffering*:

$$\forall t \in [t_0, t_k], i = 1, 2, \ldots, n, \quad \sum_{(l_{x_j, y_j}, t_j) \in S} I_{[y_j = s_i, t_j \leq t]} - \sum_{(l_{x_j, y_j}, t_j) \in S} I_{[x_j = s_i, t_j \leq t]} \leq B\ ;$$

(5) *Traffic Source*:

$$\forall i = 1, 2, \ldots, n,\ r_i + \sum_{(l_{x_j, y_j}, t_j) \in S} I_{[y_j = s_i]} = \sum_{(l_{x_j, y_j}, t_j) \in S} I_{[x_j = s_i]}\ ;$$

(6) *Traffic Destination*:

$$\sum_{(l_{x_j, y_j}, t_j) \in S} I_{[y_j = s_0]} = \sum_{i=1}^{n} r_i\ ;$$

where $I_{[\cdot]}$ is the indicator function. The first two constraints are requirements from the link layer. Since the base station moves with the elevator, (1) ensures that the links to the base station must be available when being activated and (2) guarantees that two simultaneously activated links do not interfere with each other. The next two constraints restrict the packet routing. For any given time, (3) implies that a node cannot deliver more packets than it has and (4) indicates that a node cannot buffer more packets than its capacity allows. The last two constraints focus on end-to-end traffic where (5) requires that at the end of data collection, each node must send out all its data and (6) dictates that the base station must receive all of the data.

The object is to minimize $f(|S|, t_k - t_0)$, a function of the total number of transmissions $(|S|)$ and the total latency $(t_k - t_0)$. Generally this function is specified by the target application. Here we focus on the commonly-used linear combination $f(|S|, t_k - t_0) = p|S| + q(t_k - t_0)$ that can meet various demands from different data collection applications by assigning different values for p and q. For example, if the collected data is about an emergency event, a small p and a large q will ensure that the data packets are delivered to the base station in real-time, though possibly with more hop-by-hop transmissions. On the other hand, if the data is not urgent but the transmission cost (and thus the energy consumption) is a major concern, a large p and a small q will reduce the number of transmissions and save energy.

4.3 Centralized Optimal Solution

We show that the centralized version of our problem can be solved optimally by assuming that the elevator movements are known in advance. We first provide a novel transformation of the cross-layer optimization problem into a shortest path problem in a time-state graph. We then show that this graph problem is solvable through dynamic programming. This provides an understanding of the intrinsic complexity of the problem. Its design principle also motivates a distributed implementation to be presented in the next section.

4.3.1 Time-State Graph

We construct a directed graph $G(V, E)$ which we call a *time-state graph* as shown in Fig. 4.3. In this graph, vertices are organized in two dimensions indexed by time and state, respectively. Let $M = (m_0, m_1, \ldots, m_n)$ be a state in which node s_i has m_i packets for $i = 0, 1, \ldots, n$. A vertex $v_{M,t}$ represents a situation in which at time t the number of data packets at each node is given by state M. Note that the top row in the figure indicates that the base station s_0 has no packet and each s_i has r_i packets to deliver. The bottom row indicates that at a certain time s_0 has received all of the packets.

There are two kinds of edges in the graph: *time edges* and *transmission edges*. A time edge connects two neighboring vertices along a row, from the earlier to the later. This corresponds to the case in which no node transmits any packet at time t and the state is unchanged in the next time slot. A transmission edge, on the other hand, corresponds to link-activation events. Specifically, a transmission edge ($v_{M,t}$, $v_{M',t'}$) indicates that the network state changes from M at time t to M' at time $t' = t + 1$ by some data transmissions (link activations satisfying the constraints in Section 4.2) at time t. There are a series of special transmission edges ($v_{M,t}$, $v_{M',t'}$) directed to the last row (with $M' = (\sum_{i=1}^{n} r_i, 0, 0, \ldots, 0)$). For these edges we set $t' = t$ since the state in the last row means that all data have been collected and no further transmission is necessary.

4.3.2 Equivalent Problem: Last Row Shortest Path

The time-state graph can be naturally correlated to our high-rise structure monitoring problem. Each link-activation schedule corresponds to a path from $v_{(0,r_1,\ldots,r_n),t_0}$ to a vertex in the last row and vice

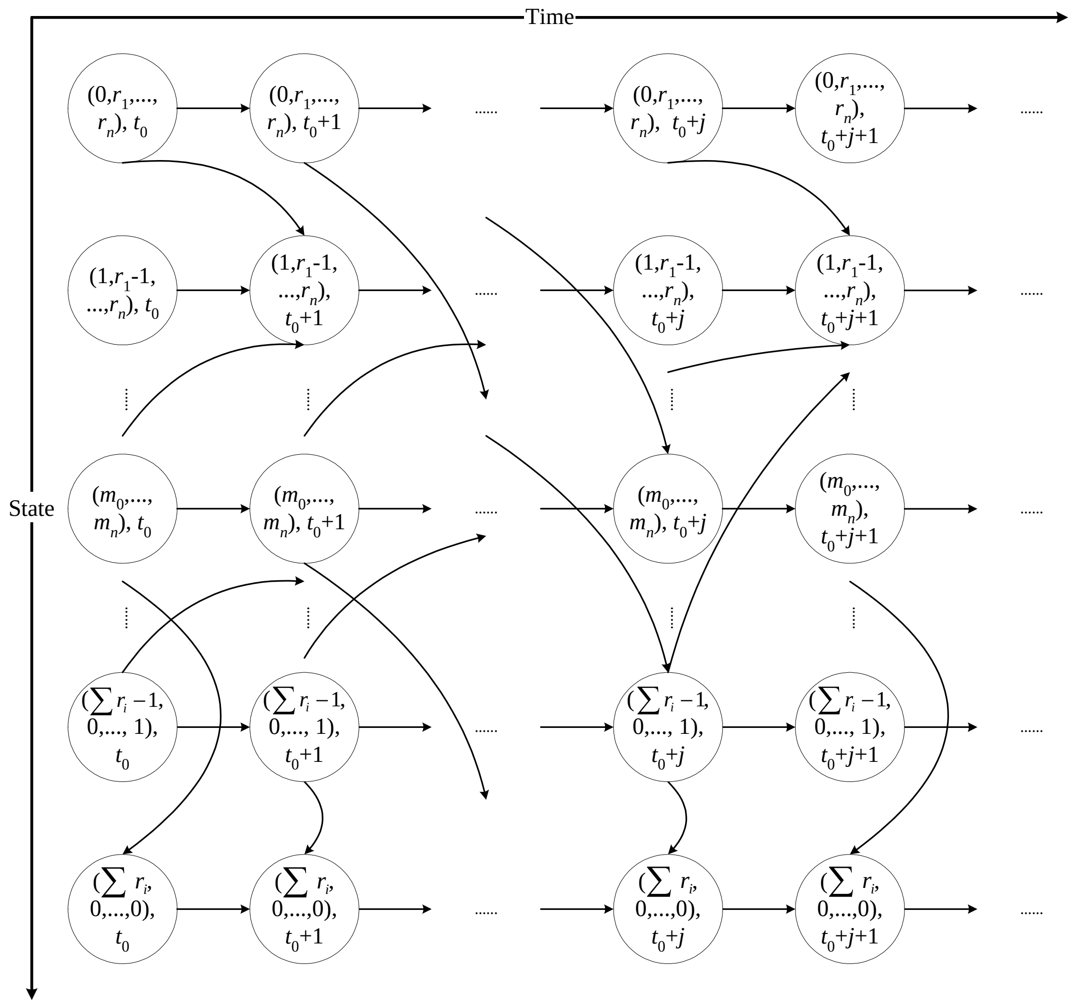

Time
State
(0,r_1,..., r_n), t_0
(0,r_1,..., r_n), t_0+1
(0,r_1,..., r_n), t_0+j
(0,r_1,..., r_n), t_0+j+1
(1,r_1-1, ...,r_n), t_0
(1,r_1-1, ...,r_n), t_0+1
(1,r_1-1, ...,r_n), t_0+j
(1,r_1-1, ...,r_n), t_0+j+1
(m_0,..., m_n), t_0
(m_0,..., m_n), t_0+1
(m_0,..., m_n), t_0+j
(m_0,..., m_n), t_0+j+1
($\sum r_i -1$, 0,..., 1), t_0
($\sum r_i -1$, 0,..., 1), t_0+1
($\sum r_i -1$, 0,..., 1), t_0+j
($\sum r_i -1$, 0,..., 1), t_0+j+1
($\sum r_i$, 0,...,0), t_0
($\sum r_i$, 0,...,0), t_0+1
($\sum r_i$, 0,...,0), t_0+j
($\sum r_i$, 0,...,0), t_0+j+1

Figure 4.3: A constructed time-state graph.

versa. For the objective function $f(|S|, t_k - t_0) = p|S| + q(t_k - t_0)$, we assign weight q to each time edge since a delay of one time unit is incurred. We also assign weight $(p|\mathbb{T}| + q(t' - t))$ to a transmission edge from $v_{M,t}$ to $v_{M',t'}$ where $\mathbb{T}$ is the set of links activated at time t that deliver data packets and change the state from M to M'. Our problem is then translated into the shortest path problem from $v_{(0,r_1,\ldots,r_n),t_0}$ to any vertex in the last row of the weighted graph.

Let $W(v_{M,t}, v_{M',t'})$ denote the weight of the edge from $v_{M,t}$ to $v_{M',t'}$ and let $W(v_{M,t}, v_{M',t'}) = \infty$ if the edge does not exist. Let $F(v_{M',t'})$ be the total weight of the shortest path from vertex

$v_{(0,r_1,\ldots,r_n),t_0}$ to $v_{M',t'}$. We have the following recurrence relation:

$$F(v_{M',t'}) = \min_{v_{M,t}} \big(F(v_{M,t}) + W(v_{M,t}, v_{M',t'})\big),$$

where $t = t'$ for $M \neq M'$ with $M' = (\sum_{i=1}^{n} r_i, 0, \ldots, 0)$; otherwise $t = t' - 1$. For boundary conditions, we have

$$F(v_{(0,r_1,\ldots,r_n),t_0}) = 0$$

and

$$F(v_{M,t_0}) = \infty\,, \quad \text{for } M \neq (0, r_1, \ldots, r_n)\,.$$

Given the recurrence relation and the boundary values, we can compute the weight of a shortest path from $v_{(0,r_1,\ldots,r_n),t_0}$ to each vertex column by column and, in each column, from top to bottom. The minimum weight path to the last-row is our result and the corresponding shortest path can be derived by simple backtracking. We refer to this path as the *last row shortest path*.

There are two strategies to calculate the recurrence relation: (1) starting from a vertex, find those vertices having edges to it; and (2) starting from a vertex, follow its edges to visit the vertices that these edges lead to. Strategy (2) is indeed much simpler and more efficient to implement. Specifically, it avoids unnecessary computation of those vertices with ∞ minimum costs because no path from $v_{(0,r_1,\ldots,r_n),t_0}$ leads to them. We thus implement a dynamic programming algorithm based on Strategy (2) as shown in Fig. 4.4.

It is worth noting that since the time dimension of the graph is unbounded, there are a potentially infinite number of paths to the last row. We thus need a condition to guarantee that a last row shortest path will be found when the condition is met so as to stop the algorithm. Specifically, as shown in line 5-6 of Fig. 4.4, before starting from a vertex $v_{M,t}$ (with $M = (m_0, m_1, \ldots, m_n)$) and following its edges to visit the vertices that these edges lead to, we first set the vertex to be inactive if the following *inactive condition* is fulfilled

$$F(v_{M,t}) + (p+q) \cdot \sum_{i=1}^{n} m_i \geq \min_{t' \leq t} F(v_{(\sum r_i, 0, \ldots, 0), t'})\,.$$

Initially, all vertices are inactive except for the vertex $v_{(0,r_1,\ldots,r_n),t_0}$. A vertex is set to be active if it is visited from a currently active vertex. It is set back to be inactive if it is a vertex in the last row or if all its neighbors have been visited. We then have the following theorem.

Theorem 4.3.1 *A last row shortest path is found when all vertices are inactive.*

Proof: The proof is by contradiction. Assume that the first last row shortest path can only be found after all vertices become inactive. Then this shortest path must contain some vertex $v_{M,t}$ (with $M = (m_0, m_1, \ldots, m_n)$) that is set to be inactive by the inactive condition (otherwise this path would be found by our algorithm before all vertices become inactive). Assume that the last vertex of this shortest path is $v_{(\sum r_i,0,\ldots,0),t'}$. Since the total weight of the path from $v_{M,t}$ to $v_{(\sum r_i,0,\ldots,0),t'}$ is at least $(p+q) \cdot \sum_{i=1}^{n} m_i$, we have

$$F(v_{(\sum r_i,0,\ldots,0),t'}) \geq F(v_{M,t}) + (p+q) \cdot \sum_{i=1}^{n} m_i \geq \min_{t'' \leq t} F(v_{(\sum r_i,0,\ldots,0),t''}) .$$

This means that the total weight of this shortest path is no less than the weight of a shortest path to some last row vertex before $v_{(0,r_1,\ldots,r_n),t'}$. This is a contradiction. □

4.4 Distributed Implementation for Practical Solution

The centralized solution can be computed efficiently on desktop PCs and yield optimal results which provide useful benchmarks and guidelines for system design and performance evaluation. However, to implement and apply it in EleSense, a series of practical issues remain that must be addressed. First, the memory and computation power of a sensor node are very limited compared to a desktop PC. The algorithms designed for the centralized solution can not be directly applied to such highly constrained hardware. Also, the centralized solution needs all elevator movements to be *known a priori* which does not happen in reality. Further, the centralized solution divides time evenly into discrete time units which, in practice, implies that a time synchronization mechanism is necessary to align the clock of different nodes. In this section, we discuss these issues in more detail and provide our distributed implementation which gives a practical solution.

4.4.1 Accommodating Hardware Constraint

As mentioned earlier, a sensor node has limited memory and computation power compared to a desktop PC. For example, the StanfordMote and Imote2 nodes, to be evaluated for the Guangzhou New TV Tower, have $16MHz$ CPU with $256kB$ RAM and $416MHz$ CPU with $32MB$ RAM, respectively. The algorithms designed for the centralized solution take an enormous amount of time

```
Algorithm OptimalLinkActivationSchedule()
1:  $F(v_{(0,r_1,...,r_n),t_0}) \leftarrow 0$; $t \leftarrow t_0$;
2:  set $v_{(0,r_1,...,r_n),t_0}$ active;
3:  while any vertex is active,
4:    for $\forall$ active vertex $v_{M,t}$ and $F(v_{M,t})$ exists,
5:      if $v_{M,t}$ is on last row or $v_{M,t}$ meets inactive condition,
6:        set $v_{M,t}$ inactive;
7:        continue;
8:      end if
9:      for $\forall$ $v_{M',t'}$ has an edge from $v_{M,t}$,
10:       set $v_{M',t'}$ active;
11:       if $F(v_{M',t'})$ does not exist,
12:         $F(v_{M',t'}) \leftarrow \infty$;
13:       end if
14:       if $F(v_{M,t}) + W(v_{M,t}, v_{M',t'}) < F(v_{M',t'})$,
15:         update $F(v_{M',t'})$;
16:       end if
17:     end for
18:     set $v_{M,t}$ inactive;
19:   end for
20:   $t \leftarrow t + 1$;
21: end while
22: find the minimum in last row;
23: return the last-row shortest path (corresponding to the optimal link-activation schedule);
```

Figure 4.4: The algorithm to compute the optimal link-activation schedule.

to finish or fail due to lack of memory when used on the hardware of these sensor nodes. To this end, we design a local search algorithm that can be quickly computed and self-improved during the running time.

The core difference of the local search algorithm is that, instead of exploring the vertices on the time-state graph by columns (as in the dynamic programming algorithm for the centralized optimal solution), the local search algorithm visits vertices in an order based on an evaluation function that estimates the remaining cost to achieve the last row. Fig. 4.5 shows an illustration of the evaluation function design where $E\text{-}MAX$ and $E\text{-}MIN$ are the highest and lowest locations of the elevator moving path computed with the future elevator movements known at a given time instance. Sensor nodes are categorized into three sets, namely, "Direct", "Above" and "Below". The nodes in the Direct set have chances to directly transmit data to the base station. Thus costs are estimated as the costs for direct transmissions. The nodes in the Above set are those physically higher than $E\text{-}MAX$ that need others to help relay packets. Costs are estimated as delivering data to a base station located at $E\text{-}MAX$. Similar estimates are done for the nodes in the Below set, computed for the location at $E\text{-}MIN$. A more detailed description of the evaluation function can be found in Appendix B.

As the vertices on a last row shortest path often have better evaluation function values than other neighboring vertices, visiting vertices in an order based on the evaluation function values allows the local search algorithm to find the last row shortest path more efficiently. In addition, according to the computation power and the affordable memory size of a sensor node, we also limit the number of vertices that the local search algorithm can visit so as to finish the computation within one time unit. If a local search can not find a path to the last row within a time unit (due to a very small search range caused by stringent hardware constraints), it can still choose the path with the best evaluation function value among the explored paths and use that path to generate the link-activation schedule for the next time unit. Moreover, if the best schedule is not found in one time unit, another local search can be issued at the beginning of the next time unit to continue with the previous results and further improve the quality until the best schedule is found.

4.4.2 Scheduling without Apriori Information

The centralized solution requires all elevator movements to be known a priori which is not possible in reality. Since elevator movement is slower than packet transmission, our solution can still work by computing the link-activation schedule for the next time unit given the current position of the elevator.

For better performance, we also note that we can easily know some information about the "near future" when a passenger presses a floor button on the panel of the elevator. That is, a sensor node can calculate precisely where the elevator will move until the elevator reaches its current destination. The information from the pressed floor button can be easily obtained by the base station and then be broadcast to all sensor nodes by inserting a short broadcast interval between neighboring time units. Another way to obtain such information is to let the base station and sensor nodes wire-tap the elevator control panel network, since the information of the pressed floor button is shared within the whole elevator system. The remaining question is how to exploit such information in our distributed implementation. We will investigate this in the remainder of this subsection.

Before a floor button is pressed (triggering a new elevator movement), the link-activation schedule is computed only by the movements known previously. (The elevator is assumed to pause at

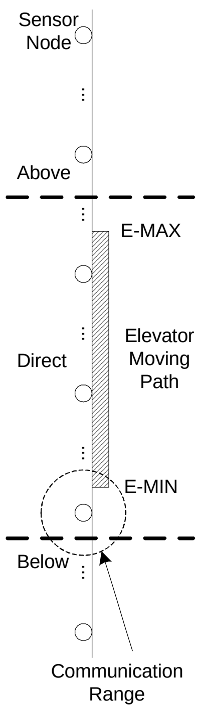


Figure 4.5: Evaluation function design.

its current position after these movements finish.) At each time unit, a sensor node checks for new button presses. If one is detected, it computes a new schedule. Otherwise, it follows the previous schedule.

Therefore, in the distributed implementation, all sensor nodes initially compute the same link-activation schedule using the current elevator position. Then the sensor nodes forward their data packets according to the computed schedule and update their local state and information accordingly. As new elevator movements are detected, the sensor nodes dynamically compute and forward packets by the new link-activation schedule.

4.4.3 Integrating Synchronization

Many synchronization mechanisms have been proposed for wireless sensor networks. Most of them are designed for a flat sensing field with moderate dimensions and the synchronization error is generally linearly proportional to the network diameter [21, 41, 54, 63]. High-rise structures have exaggerated height so that the network diameter could be extremely large. Thus though these previous

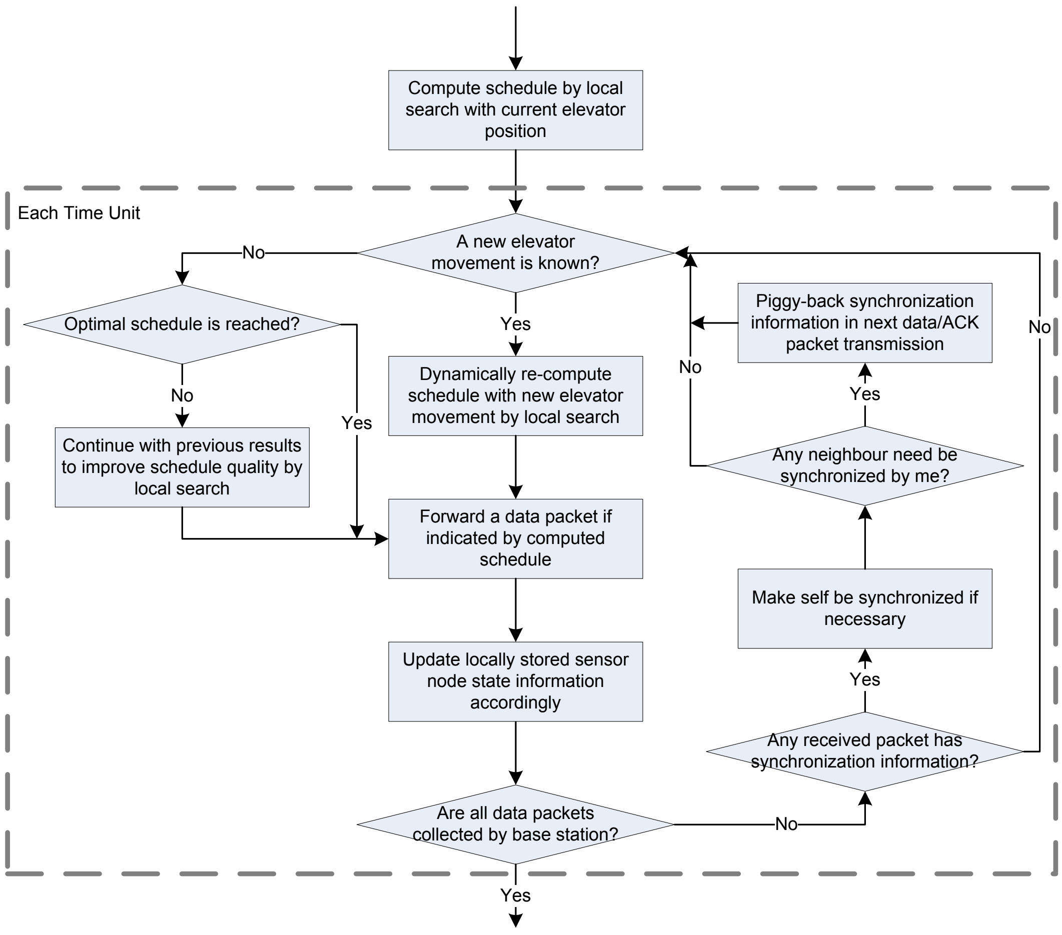

Compute schedule by local search with current elevator position
Each Time Unit
A new elevator movement is known?
No
Optimal schedule is reached?
Yes
No
Continue with previous results to improve schedule quality by local search
Yes
Dynamically re-compute schedule with new elevator movement by local search
Forward a data packet if indicated by computed schedule
Update locally stored sensor node state information accordingly
Are all data packets collected by base station?
Yes
No
Piggy-back synchronization information in next data/ACK packet transmission
No
Yes
Any neighbour need be synchronized by me?
Make self be synchronized if necessary
Yes
Any received packet has synchronization information?
No

Figure 4.6: The operation flow of the practical solution at a sensor node during a data collection. The operations in the dashed rectangle are performed within each time unit.

mechanisms may be applied to EleSense, the increased synchronization error would cause problems if not carefully handled. Since most of these mechanisms need to periodically exchange information through wireless communications, directly applying them would bring unwanted interference and collisions during data collection. This would compromise the computed link-activation schedule and degrade the network performance. In addition, the poor conditions during in-construction monitoring make it difficult to use an out-of-band approach [62]. To this end, we propose a hybrid mechanism that greatly reduces the synchronization error and can be seamlessly integrated with the distributed implementation.

The hybrid mechanism contains an active mode and a passive mode. To alleviate the effects caused by the large height, the active mode exploits elevators. Moving with the elevator, the base station periodically synchronizes with sensor nodes as it passes by. Each sensor node knows its fixed location a priori. When a sensor node is synchronized, it checks whether it is necessary to synchronize with its neighbors. Specifically, if the sensor node is the farthest one (either above or below the base station) that is newly synchronized and some of its neighbors have not been synchronized recently, it synchronizes with those neighbors. To avoid unnecessary interference, the active mode is used only when there is no ongoing data collection. During data collection, we shift to passive mode where the synchronization information is piggy-backed with either a data packet or an ACK packet used in the MAC layer. Since the data packets are collected from all sensor nodes to the base station and the ACKs would go in the opposite direction, these two types of traffic serve perfectly for synchronization with minimal overhead.

As the base station moves with the elevator, the average distance from the sensor nodes to the base station is significantly reduced. Our experience shows that given a typical height of a high-rise structure and the wireless communication range, the synchronization error can be effectively reduced by our hybrid mechanism and is well below the synchronization error required by EleSense.

In summary, Fig. 4.6 gives the main operations conducted by each node during a data collection round in the distributed practical solution. In the following sections, we will show that EleSense can achieve excellent performance with very low cost.

4.5 Performance Evaluation

We evaluate EleSense by both ns-2 simulations and a case study with real configurations from the Guangzhou New TV Tower. We present the simulation results in this section and discuss the results of the case study in the next section.

4.5.1 Methodology

We adopt a typical configuration as follows: The length of a time unit is 1 second. The distance between two neighboring floors is 5 meters. On each floor, the representative node is placed 1 meter to the elevator door with a communication range of 10 meters. We use a random way point model to emulate elevator movements. Specifically, the elevator chooses a destined floor with some probability, moves to it, then pauses for a random period and chooses the next destination. We conducted simulations with different probability distributions, elevator speeds, and other setting values and found that the results were similar in all cases. Due to the space limits we present the

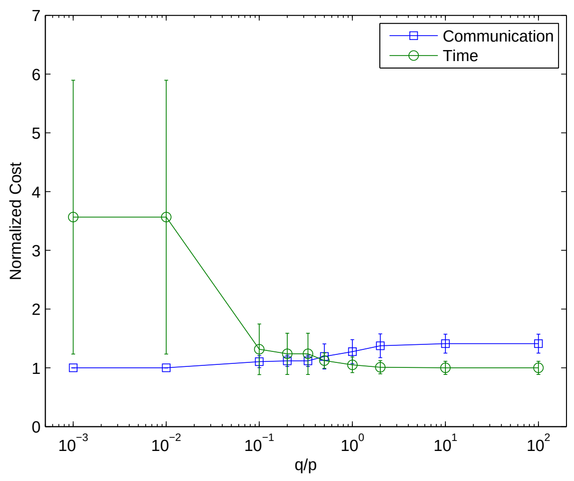


Figure 4.7: Impact of ratio q/p on communication and time costs.

results with the floor destinations chosen with a uniform distribution and an elevator speed of $3m/s$. At the initial stage of each simulation, the communication quality of each possible link is tested and those that can reliably transmit a data packet within one time unit are selected. The IM is then obtained by measuring the simultaneous communication quality through each pair of selected links.

For comparison, we implemented three other approaches: *StaticSense*, *Ele802.11*, and *Static802.11*. StaticSense uses the same cross-layer optimization as EleSense, but the base station is fixed statically at the bottom of the structure. Ele802.11 exploits elevators like EleSense, but using a plain 802.11 MAC layer without the cross-layer optimization. Static802.11 uses a plain 802.11 MAC layer like Ele802.11 and fixes the base station like StaticSense.

In addition to time and communication costs, we are also interested in the following three metrics:

1) *Throughput* is the average number of data packets received by the base station per time unit. High throughput greatly reduces the finishing time of a data collection round and enables real-time monitoring on complicated structure behaviors (such as during a seismic shake or during a typhoon).

2) *Data loss rate* is the number of data packets that fail to arrive at the base station divided by the number of data packets sent from sensor nodes. Since we need raw data for further processing, data loss would greatly affect the usefulness of the collected data. Thus high loss rate means that complicated reliable transmission mechanisms will be needed in practice to compensate for losses.

3) *Fairness* is quantified using the classic Jain's fairness index [42] which is defined as $(\sum x_i)^2/$

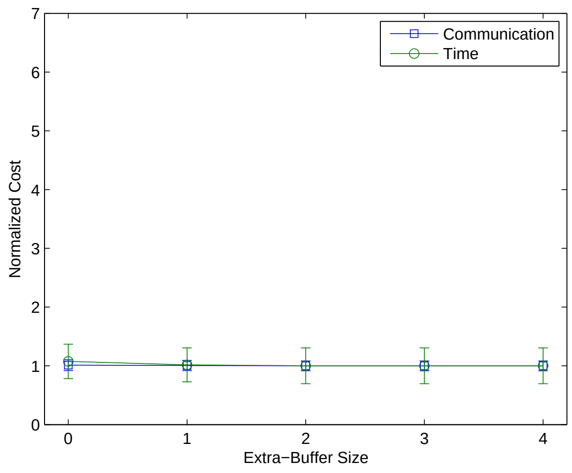


Figure 4.8: Impact of extra-buffer size on communication and time costs.

$(n \cdot \sum x_i^2)$ for demands $x_1, x_2, \ldots, x_n$. Good fairness helps maintain the quality of the collected data, as any sensor node suffering unfairness may lead to inadequate information from the deployment location of that node.

In addition, for the two elevator-assisted schemes (EleSense and Ele802.11) we ran 10 simulations on each setting to alleviate the random effects caused by elevator movements. Each data point in the figures represents the average of 10 runs with an error bar showing the standard deviation.

4.5.2 Impacts of Different Parameter Settings

To mitigate other uncertainties, we first run the centralized optimal solution to investigate the impacts of different parameters. The practical solution will be evaluated in the following subsections.

Fig. 4.7 shows how EleSense performs with different q/p ratios. For ease of comparison, the results are normalized by the corresponding minimum values. When q/p is small ($\leq 10^{-2}$), EleSense yields minimum communication cost but at the expense of excessive latencies. On the other hand, when q/p grows large (≥ 10), the resulting time cost is minimized while introducing the highest communication cost. Moreover, within the region of $[0.2, 0.5]$, both communication and time costs stay low and keep relatively stable. We thus use the middle value $q/p \approx 0.3$ (note that the x-axis is in log scale) as the default for the remaining evaluations.

We next investigate the impacts of different queuing buffer sizes B (defined in Section 4.2). The results are shown in Fig. 4.8. Again we normalize the results by the corresponding minimum

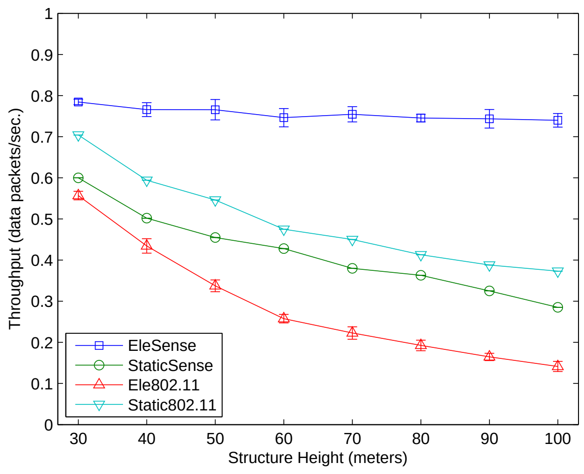


Figure 4.9: Overall throughput as a function of structure height.

values. It is easy to see that as the queuing buffer size increases, both the time and communication costs decrease. And after the queuing buffer size becomes greater than 1, no change on either cost is observed. Therefore, we select 2 as the default size of the queuing buffer.

4.5.3 Scalability with Different Structure Vertical Dimensions

With the default parameter setting, we then conduct simulations on the practical solution to see how EleSense performs as the structure height varies. At this stage, we let each simulation run for 1000 seconds to explore the long-term behavior. In particular, we start a new data collection just after the previous round finishes and stop the simulation at the end of 1000 seconds.

Fig. 4.9 shows the throughput from the four approaches. Recall that a time unit is the minimum time for a link to reliably transmit a data packet. Thus ideally the maximum throughput is at most 1 data packet per second. Intuitively, with the height increasing, the throughput would slightly drop due to the increase in the network diameter and/or the increasing possibility of wireless interference or collisions (as more data packets need to be collected). This is exactly the case for StaticSense, Ele802.11 and Static802.11. EleSense, however, successfully exploits the elevator and greatly reduces the average distances from the sensor nodes to the base station. Its cross-layer optimization also helps resolve possible interference and collisions. This makes the throughput almost invariant with the height and achieves gains of 30.7% to 159.6% over StaticSense and 40.9% to 423.2% over Ele802.11.

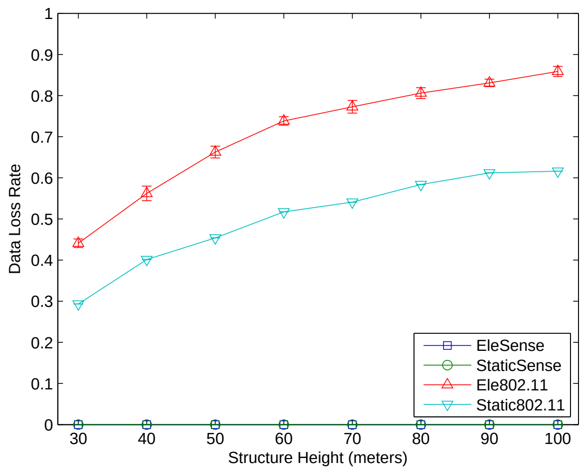


Figure 4.10: Data loss rate as a function of structure height.

One interesting observation is that Ele802.11 performs generally worse than Static802.11. A close look reveals that when the base station moves with the elevator, data traffic may accumulate more quickly by coming from both above and below the base station. This expedites the saturation of the wireless medium near the base station and introduces more significant interference and collisions, resulting in fewer packets being received at the base station and thus lowering the throughput.

Fig. 4.10 presents the results on data loss rate with the four approaches and different structure heights. It is easy to see that with cross-layer optimization, both EleSense and StaticSense achieve zero data loss with the built-in retransmission mechanism of the MAC layer, since most interference and collisions are resolved by the link-activation schedule. On the other hand, the loss rates of Ele802.11 and Static802.11 are generally high and increase with the structure height. This is because larger heights will increase the average distances to the base station and thus raise the possibility of interference and collisions.

Fig. 4.11 gives results on communication cost. It is not surprising that the communication cost rises with the height, due to the increasing average distance to the base station. One exception is StaticSense where the cost first rises and then decreases after the height exceeds 80 meters. This is because as the height increases, more and more data packets will accumulate as relayed to the base station, making it a bottleneck for data collection. At the same time, the cross-layer optimization used in StaticSense successfully suppresses unnecessary transmissions that may cause interference and collisions and thus make the overall communication cost drop a little. On the other hand,

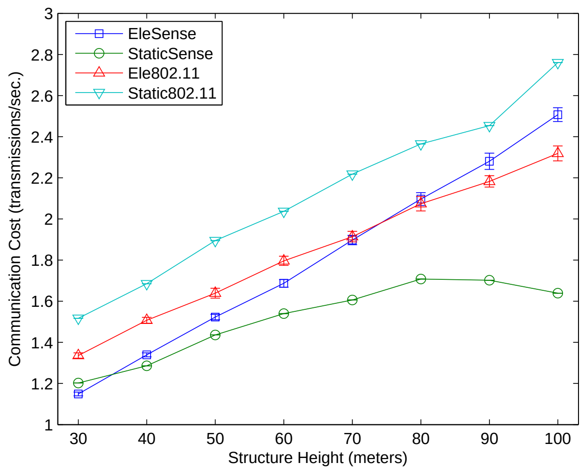


Figure 4.11: Communication cost as a function of structure height.

this bottleneck is well handled in EleSense by attaching the base station to an elevator, allowing more transmissions to be scheduled for more packet deliveries. This also explains why the cost of EleSense becomes even higher than Ele802.11 when the height goes beyond 70 meters. As the communication costs of different approaches are afforded for different throughput, we further normalize the communication cost by the throughput to achieve a more fair comparison. The results are shown in Fig. 4.12. In this figure, although the costs still rise with the height, to successfully deliver a data packet EleSense is less costly (about 58.9% to 73.1% of the runner-up) than the other three approaches.

We also compare the fairness achieved by the four approaches. Fig. 4.13 shows the results with different structure heights. EleSense and StaticSense keep achieving 1.0 in spite of the change of heights. Ele802.11 also has a fairness index above 0.85. But the fairness index of Static802.11 is generally below 0.85 and drops as the height increases. This is because with the presence of wireless interference and collisions (especially when the wireless medium is nearly saturated), the packet delivery ratio of a plain 802.11 decreases as the number of traversed links increases. This favors packets originating from the nodes close to the base station and reduces the fairness.

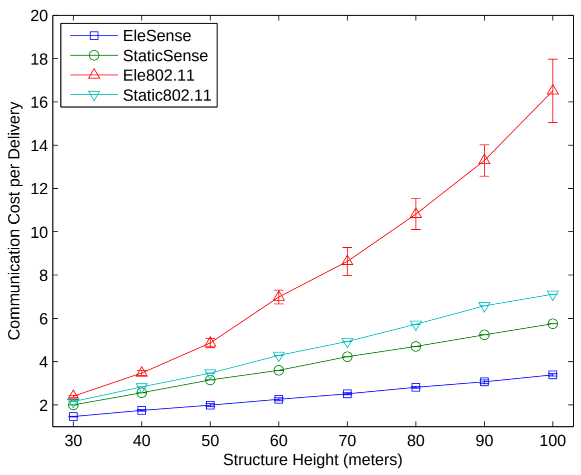


Figure 4.12: Communication cost per delivery as a function of structure height.

4.5.4 Local Search Quality with Different Hardware Constraints

Since our local search algorithm can accommodate different hardware, we are interested in how it performs under various situations, i.e., different search ranges. Fig. 4.14 shows the results of EleSense and StaticSense working with local search on 1000, 2000, 5000 and 10000 vertices per time unit on the time-state graph. The results of EleSense and StaticSense working with the algorithm that searches on the full graph (denoted as full-range) are also shown in this figure. We use the average total weight per data collection round as the metric, since it is the optimization objective. It is worth noting that in the distributed implementation, subsequent elevator movements can not be known until a button on the elevator panel is pressed. Therefore, the schedule computed by the full-range search algorithm may be suboptimal due to the temporary lack of apriori information. This explains why the results of the local search algorithm in EleSense sometimes may be slightly better than those of the full-range search algorithm. On the other hand, since there is no elevator movement in StaticSense, the full-range search algorithm always produces the optimal schedules. In this case, the local search algorithm still works well with small degradation generally below 1%. This demonstrates that the evaluation function used in our local search algorithm can precisely estimate the weight cost to the last row, thus effectively narrowing down the searching branches and yielding high quality results even with a small search range.

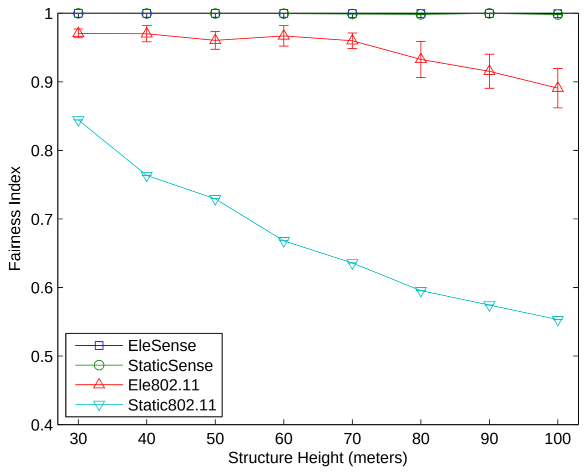


Figure 4.13: Fairness index as a function of structure height.

4.6 A Case Study with the GNTVT

In this section, we conduct a case study with the GNTVT. As shown in Figs. 4.1 and 4.2, the tower has an irregular shape with an exaggerated height. With its publicly accessible floors unevenly spaced, the resulting wireless links as well as their interference and collisions are distributed at great variances. This further varies the capabilities to transmit data to the base station from the nodes on different floors and brings more challenges. To this end, we conduct experiments on the GNTVT to determine the feasibility of EleSense and further evaluate our solution with the collected data sets.

4.6.1 System Deployment and Verification

We adopt the StanfordMote as the data collecting unit and use XStream-PKG $2.4GHz$ RS-$232/485$ RF modem together with a laptop as the base station. We placed sensor nodes on some floors of the tower and the base station in the No. 2 elevator. Our sensor nodes were equipped with $7dBi$ Buffalo high-gain antenna to enhance the signal strength in the poor construction conditions. The speed of the elevator was around $3m/s$. We adopt high precision accelerometers Tokyo Sokushin AS-2000 for acceleration monitoring. To increase the signal strength and the SNR, a signal amplification and conditioning board is developed using TI PGA202. This board works as an amplification middle-ware between the accelerometer and analogue input of our data collection motes. The sampling rate of the 16-bit Analog to Digital Converter (ADC) on the wireless sensing unit is set to $50Hz$, thus

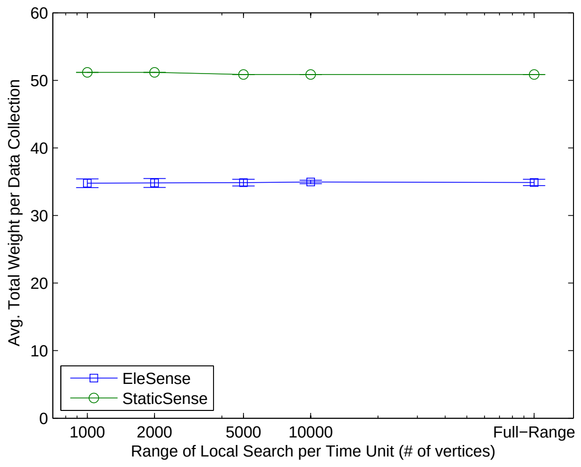


Figure 4.14: Average total weight with different local search ranges.

Elevator Movement	Start Height	End Height	Packets from Each Node	Delivery Ratio
From bottom to top	$0m$	$60m$	20	100%
	$60m$	$120m$	20	100%
	$120m$	$180m$	20	100%
	$180m$	$240m$	20	100%
From top to bottom	$240m$	$180m$	20	100%
	$180m$	$120m$	20	100%
	$120m$	$60m$	20	100%
	$60m$	$0m$	20	100%

Table 4.1: Verification Experiments on the GNTVT.

$50 \times 60 \times 2 = 6000$ bytes are generated every minute. These data are first buffered by the SRAM on the wireless unit. Then we divide each $6kB$ of data (corresponding to one minute's collection) into 20 packets and transmit them back to the moving base station. It is worth noting that during the experiments, the GNTVT was still in construction. To avoid disturbing the construction, we were only allowed to access the section below 240 meters, which was further divided into 4 subsections of 60 meters. To fully understand wireless communication capabilities, we conduct experiments for both upward and downward directions in each subsection as summarized in Tab. 4.1. From the table, it is clear to see that, at each subsection the base station can successfully receive all the collected data while moving with the elevator in both directions. In addition, we also observed

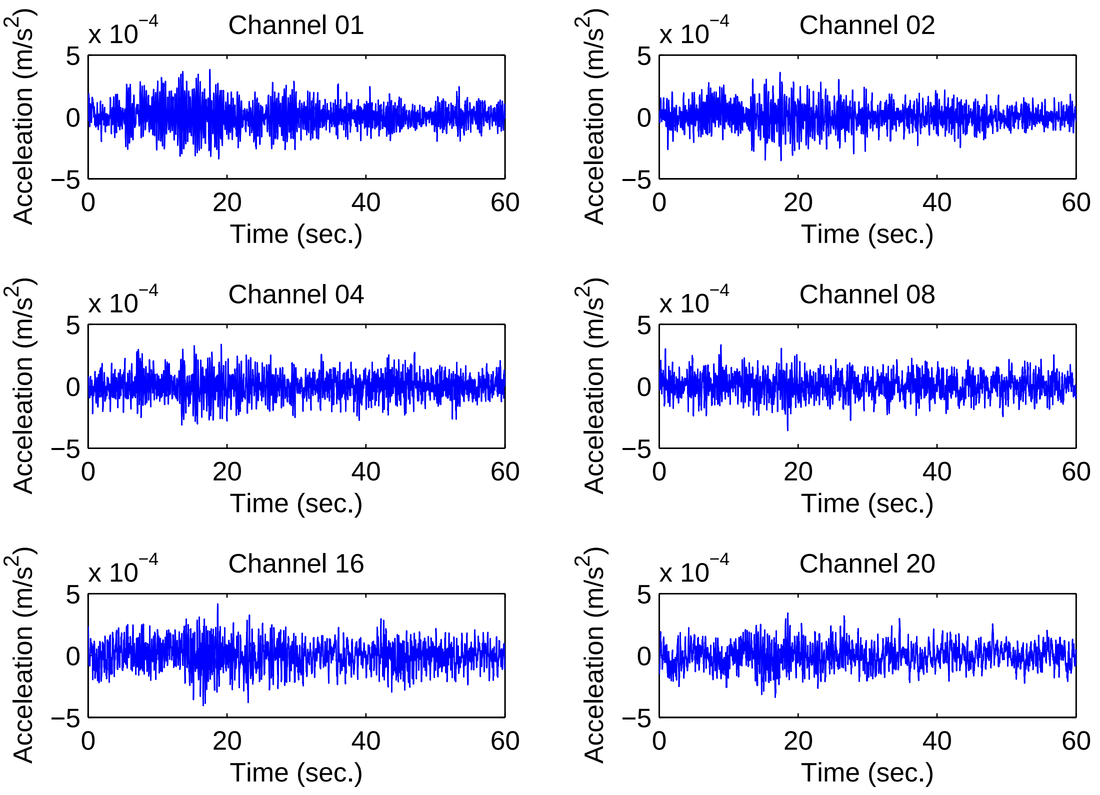


Figure 4.15: Acceleration data collected by experiments on the GNTVT. Each channel corresponds to the readings from one accelerometer sensor.

that the wireless transmission could easily reach $55kbps$. All these validate the feasibility of our EleSense framework.

4.6.2 Further Evaluation

Since the time and region that we can access for conducting experiments were very limited due to the construction of the tower, to further evaluate the performance of our solution on the entire tower, we also conduct emulations with the real data and settings collected on the GNTVT. Our emulations focus on the comparison of EleSense and StaticSense. All the settings are the same as for the experiments on the tower except that the elevator now moves along the entire tower. For current emulations, we only consider the acceleration data collected by experiments on the GNTVT. Fig. 4.15 shows the collected acceleration data where each channel corresponds to the readings from one accelerometer sensor.

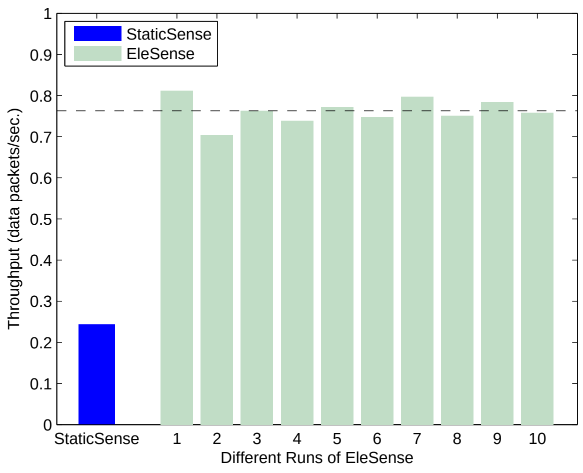


Figure 4.16: Overall throughput with real data sets on the GNTVT.

Similar to the results observed in the simulations, both EleSense and StaticSense schemes achieve zero data loss rate and a fairness index of 1.0. Fig. 4.16 and Fig. 4.17 further show results on throughput and communication costs, respectively, with the dashed line indicating the average of the 10 runs on EleSense. For throughput, EleSense greatly outperforms StaticSense with the average gain as high as 212.7%. Compared to the simulation results, we find that, even under such a stringent scenario the throughput of EleSense still keeps relatively stable. This further verifies its excellent scalability. For communication cost per delivery, EleSense stays much lower than StaticSense with the average reduction of 58.7%. This is similar to in the simulations and demonstrates the superiority of EleSense.

4.7 Conclusion

In this chapter, we presented EleSense, a novel framework for high-rise structure monitoring that uses elevators to reduce the overhead of traffic relaying and to balance loads among sensor nodes. As one of the very first studies in this direction, we focused on the most fundamental practical and theoretical constraints. In particular, we abstracted the high-rise structure monitoring problem from EleSense and formulated it as a cross-layer optimization problem with the considerations of link scheduling, packet routing and end-to-end delivery. We presented a centralized optimal solution through dynamic programming. Our design also motivated a distributed implementation to

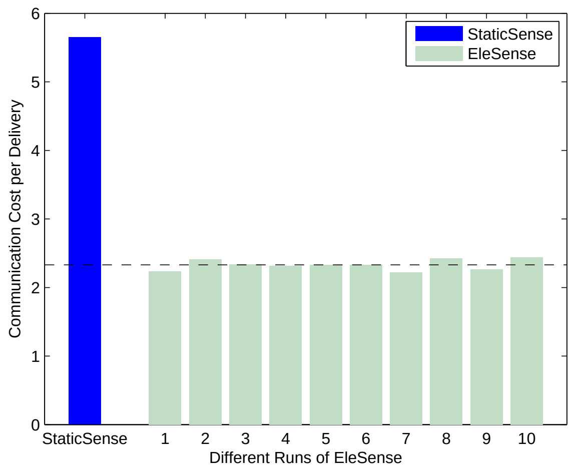


Figure 4.17: Communication cost per delivery with real data sets on the GNTVT.

accommodate the hardware capability of a sensor node as well as address other practical issues. We evaluated EleSense by ns-2 simulations and a case study with real experiments and data sets on the GNTVT. The results showed that the practical solution in EleSense performs only marginally worse (below 1%) than the optimal solution. Moreover, EleSense achieves a throughput gain of 30.7% to 212.7% over the case without elevators and a gain of 40.9% to 423.2% over a straightforward 802.11 MAC scheme without cross-layer optimization. EleSense also greatly reduces the communication cost with excellent fairness and 100% data delivery ratio.

Chapter 5

Future Work

The research field of wireless sensor networks is growing fast. There are diverse new applications with numerous challenges and opportunities. We consider the following areas as our future directions.

5.1 General Data Collection Protocol Design with Cross-Layer Optimization

Previous research has shown that wireless communications constitute the major power consumption of a sensor node. Due to wireless losses and collisions during sensor nodes communicating with each other, retransmissions are needed to ensure that data packets are reliably delivered to the base station. This consumes a significant amount of node power and thus reduces the network lifetime. Many attempts have been made to address this problem. In the network layer, routing topologies may be derived such that data packets can be routed along the links with low loss rates to minimize the number of retransmissions due to wireless losses. In hotspot areas where a great amount of data traffic accumulates and makes the wireless medium nearly saturated, techniques such as hybrid integrating TDMA are used to resolve unnecessary interference and collisions. Either of these solutions, however, would yield local optima due to considering only one source of packet retransmissions. For example, routing packets through the links with low loss rates may make these links become hotspots. Integrating TDMA into the solution without awareness of different link loss rates would result in unexpected packet losses and even compromise the fairness and throughput of the overall data collection. To this end, we plan to explore a cross-layer design approach and model the link scheduling, packet routing, and end-to-end delivery as a single optimization problem with the consideration of wireless losses. As latency is often important in data collection applications, our modeling will also take it into account. Besides modeling and solving the problem itself, we plan to develop practical protocols for reliable and energy-efficient data collection.

5.2 Real-Time Data Collection and Actuation in Cyber-Physical Systems Applications

Another direction that we are interested in is real-time data collection and actuation which is closely aligned with the research frontier of Cyber-Physical Systems (CPS). In recent years, the development of CPS has garnered a significant amount of attention from both government and industrial sectors. At this time, it is essential and critical to further explore and develop systems where computing machines interact with the physical world. In the near future, CPS will be applied in almost every aspect of our daily lives such as transportation, room control, healthcare, and utility supply. Real-time data collection and actuation is closely related to this research frontier, as it enables timely and efficient interactions between the cyber and physical worlds in many CPS applications. Taking structural health monitoring as an example, real-time monitoring can provide fine granularity of monitoring data when there is any emergency such as earthquake, fire, or typhoon nearby, allowing more timely evacuation and greatly reducing losses. Another example is in smart home applications where multiple sensors can be installed on different appliances. These sensors can monitor electricity usage in real-time and adjust the working patterns of appliances to reduce electricity consumption, or even enabling more large-scale optimization over the entire grid by exchanging collected data with the smart energy grid. As CPS applications often put more stress on time, we plan to develop new models that work in real-time and are both reliable and energy-efficient. Also, we consider the application of filtering and coding techniques to smooth data traffic while providing good reliability and satisfactory error rates.

Appendix A

Proof of Theorem 3.3.1

Proof: We first give some definitions that facilitate the proof. Recalling the definition of the terminating condition, we call a row R a *candidate* when all the rows that have forwarding edges toward row R have terminated. If row R is terminated at some time t, then by the terminating condition, for any forwarding edge originating from row R to another row R' at some time greater than t, there must exist at least one edge of the same or less weight originating from row R to row R' between the time that row R becomes a candidate and time t. We call this latter edge a *duplicate* of the former edge.

Since the last edge of a last-row shortest path must be a forwarding edge, we only consider paths whose last edges are forwarding edges. We claim that given a path starting from $v_{\{s\},t_0}$ and with N forwarding edges, we can always construct another path with the same number of forwarding edges, such that in the given path, if the i-th forwarding edge starts from row R and ends at row R', the i-th forwarding edge in the constructed path is also from row R to row R' but with equal or less weight and ending at row R' before it becomes a candidate, for $i = 1, 2, \ldots, N$. We prove this by induction on the number of forwarding edges in the given path.

Base case: We start from the case that the given path uses only one forwarding edge. Clearly this forwarding edge must start from row $\{s\}$ and end at some row R'. If the forwarding edge ends at row R' no later than row R' becomes a candidate, then we take the given path as the constructed new path. Otherwise, the forwarding edge ends at row R' after row R' becomes a candidate. Then by definition, the forwarding edge must leave row $\{s\}$ after row $\{s\}$ is terminated. And by the terminating condition, there must exist a duplicate of the forwarding edge that leaves row $\{s\}$ no later than row $\{s\}$ is terminated (and thus no later than row R' becomes a candidate). Using this duplicate to construct the new path, the new path ends at row R' no later than row R' becomes a candidate.

Advance case: Assuming the claim holds for any given path with k forwarding edges, we now prove the claim also holds for a given path with $(k+1)$ forwarding edges.

We first consider the path, p, consisting of the first k forwarding edges of the $(k+1)$ edge path. By the assumption, we can construct a path p' with k forwarding edges such that in path p, for $1 \leq i \leq k$, if the i-th forwarding edge starts from row R and ends at row R', the i-th forwarding edge in path p' is also from row R to row R' but with equal or less weight and arriving at row R' before it becomes a candidate .

Now consider the $(k+1)$-th forwarding edge. Let row R denote the row where this edge leaves and row R' denote the row where this edge ends. If this edge ends at row R' no later than row R' becomes a candidate, then we just connect this edge with path p' by some necessary time edges and take the resulting new path as the constructed path. Otherwise, if the $(k+1)$-th forwarding edge ends at row R' after row R' becomes a candidate, by definition the $(k+1)$-th forwarding edge leaves row R after row R is terminated. And, by the terminating condition, there must exist a duplicate of this edge no later than when row R is terminated (and thus no later than when row R' becomes a candidate). Note that the duplicate leaves row R after row R becomes a candidate and the last forwarding edge in path p' arrives at row R no later than when row R becomes a candidate. We can then connect this duplicate to path p' by some necessary time edges. Then we obtain a new path with $(k+1)$ forwarding edges such that, for $i = 1, \ldots, k+1$, if the i-th forwarding edge in the given path starts from row R and ends at row R', the i-th forwarding edge in the new path is also from row R to row R' but with equal or less weight and ending at row R' before it becomes a candidate.

Combining the base case and advance case together, the claim holds for any given path, proving the theorem. $\square$

Appendix B

Design of Evaluation Function

In EleSense, we use the following evaluation function for a visited vertex $v_{M,t}$ with $M = (m_0, m_1, \ldots, m_n)$:

$$Eval(v_{M,t}) = \min_{t<t' \leq t_{pause}} \Big(CC(t') + \max(q(t'-t), TC_{Direct}(t') + TC_{Relay}(t')) \Big),$$

where t_{pause} is the finish time of the latest known elevator movement (since then the elevator is assumed to pause), $CC(t')$ is the communication cost, $TC_{Direct}(t')$ is the time cost for the packets that can be directly transmitted from a sensor node to the base station, and $TC_{Relay}(t')$ is the time cost for the packets that have to be relayed to reach the base station.

Since the elevator may move between time t and time t', we define $E\text{-}MAX(t')$ and $E\text{-}MIN(t')$ to be the highest and lowest locations that the elevator reaches during this period. Fig. 4.5 illustrates this. Sensor nodes are then divided into three sets, “Direct”, “Above”, and “Below”. The Direct set contains those sensor nodes with chances to directly send data packets to the base station. The Above set includes those higher than $E\text{-}MAX(t')$ and needing others to help relay the packets. The Below set is similar to the Above set except that its sensor nodes are lower than $E\text{-}MIN(t')$. The communication cost $CC(t')$ is

$$\begin{aligned} CC(t') \quad = \quad & p \cdot \sum_{i=1}^{n} m_i \cdot \Big(I_{[s_i \in Direct]} + HOP(s_i, E\text{-}MAX(t')) \cdot I_{[s_i \in Above]} + \\ & HOP(s_i, E\text{-}MIN(t')) \cdot I_{[s_i \in Below]} \Big), \end{aligned}$$

where $HOP(a, b)$ is the minimum hop count from a to b. $TC_{Direct}(t')$ is computed by

$$TC_{Direct}(t') = q \cdot \sum_{i=1}^{n} m_i \cdot I_{[s_i \in Direct]} \,.$$

We further define four terms that help compute $TC_{Relay}(t')$. We define $TC_{E\text{-}MAX}(t')$ to be the time cost taken by the elevator to reach $E\text{-}MAX(t')$ which is computed as

$$TC_{E\text{-}MAX}(t') = q \cdot t_{E\text{-}MAX(t')} \,,$$

where $t_{E\text{-}MAX(t')}$ is the duration from time t to the time that the elevator achieves $E\text{-}MAX(t')$ for the first time. Similarly, we define $TC_{E\text{-}MIN}(t')$ and compute it as

$$TC_{E\text{-}MIN}(t') = q \cdot t_{E\text{-}MIN(t')} \,,$$

where $t_{E\text{-}MIN(t')}$ is the duration from time t to the time that the elevator achieves $E\text{-}MIN(t')$ for the first time. In addition, we define $TC_{Above}(t')$ to be the time cost for nodes in the Above set to deliver all their data to the base station at $E\text{-}MAX(t')$ and compute $TC_{Above}(t')$ as q times the total duration for packet delivery. If we put the packets in increasing order of their hop distances to the base station, it is easy to determine that when the first packet goes towards the base station, the later packets can just follow it back-to-back like a pipeline. The total duration for packet delivery is thus the time taken for the first packet to reach the base station plus two times the number of remaining packets (since a sensor node can not send and receive data packets within the same time unit). We define $TC_{Below}(t')$ and compute similarly. As the base station may receive packets alternately from both directions (from the Above set as well as from the Below set), we take the maximum (including the time for the elevator to move to the corresponding location) for the final value of $TC_{Relay}(t')$:

$$TC_{Relay}(t') = \max\Big(TC_{Above}(t') + TC_{E\text{-}MAX}(t'), TC_{Below}(t') + TC_{E\text{-}MIN}(t')\Big) \,.$$

Note that since the evaluation function does not include the time cost to hold transmissions to avoid the interference on wireless links, its value is a lower bound on the actual weight cost. Thus to further reduce the number of vertices visited during the local search, we also prune a search branch if the sum of its evaluation function value and its current total weight is greater than the current minimum total weight to the last row.

Bibliography

[1] *MICA2 Datasheet. http://www.xbow.com/products/Product_df_files/Wireless_pdf/MICA2_Datasheet.pdf.*

[2] *Structural Health Monitoring for Guangzhou New TV Tower using Sensor Networks. http://www.cse.polyu.edu.hk/benchmark/.*

[3] *The ns Manual. http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf.*

[4] *TinyOS Tutorial. http://www.tinyos.net/tinyos-1.x/doc/tutorial/.*

[5] G.-S. Ahn, E. Miluzzo, A. T. Campbell, S. G. Hong, and F. Cuomo. Funneling-MAC: A Localized, Sink-Oriented MAC For Boosting Fidelity in Sensor Networks. In *ACM SenSys*, 2006.

[6] N. Aitsaadi, N. Achir, K. Boussetta, and G. Pujolle. Potential Field Approach to Ensure Connectivity and Differentiated Detection in WSN Deployment. In *IEEE International Conference on Communications*, 2009.

[7] N. Aitsaadi, N. Achir, K. Boussetta, and G. Pujolle. Multi-Objectives WSN Deployment: Quality of Monitoring, Connectivity and Lifetime. In *IEEE International Conference on Communications*, 2010.

[8] K. Akkaya and M. Younis. A Survey on Routing Protocols for Wireless Sensor Networks. *Ad Hoc Networks*, 3(3):325–349, May 2005.

[9] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A Survey on Sensor Networks. *IEEE Communications Magazine*, 40(8):102–114, 2002.

[10] X. Bai, S. Kumar, D. Xuan, Z. Yun, and T. H. Lai. Deploying Wireless Sensors to Achieve Both Coverage and Connectivity. In *ACM MobiHoc*, 2006.

[11] X. Bai, D. Xuan, Z. Yun, T. H. Lai, and W. Jia. Complete Optimal Deployment Patterns for Full-Coverage and k-Connectivity ($k \leq 6$) Wireless Sensor Networks. In *ACM MobiHoc*, 2008.

[12] X. Bai, Z. Yun, D. Xuan, T. H. Lai, and W. Jia. Deploying Four-Connectivity and Full-Coverage Wireless Sensor Networks. In *IEEE INFOCOM*, 2008.

[13] G. Barrenetxea, F. Ingelrest, G. Schaefer, and M. Vetterli. SensorScope: Out-of-the-Box Environmental Monitoring. In *ACM/IEEE IPSN*, 2008.

[14] J. L. Bredin, E. D. Demaine, M. Hajiaghayi, and D. Rus. Deploying Sensor Networks with Guaranteed Capacity and Fault Tolerance. In *ACM MobiHoc*, 2005.

[15] N. Burri, P. V. Rickenbach, and R. Wattenhofer. Dozer: Ultra-Low Power Data Gathering in Sensor Networks. In *ACM/IEEE IPSN*, 2007.

[16] Ma. Ceriotti, L. Mottola, G. P. Picco, A. L. Murphy, S. Guna, M. Corrà, M. Pozzi, D. Zonta, and P. Zanon. Monitoring Heritage Buildings with Wireless Sensor Networks: The Torre Aquila Deployment. In *ACM/IEEE IPSN*, 2009.

[17] X. Cheng, D. Z. Du, L. Wang, and B. Xu. Relay Sensor Placement in Wireless Sensor Networks. *Spinger Wireless Networks*, 14(3):347–355, 2008.

[18] S. Deering. *Scalable Multicast Routing Protocol*. PhD thesis, Stanford University, 1989.

[19] C. T. Ee and R. Bajcsy. Congestion Control and Fairness for Many-to-One Routing in Sensor Networks. In *ACM SenSys*, 2004.

[20] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. *IEEE/ACM Transactions on Networking*, 5(6):784–803, 1997.

[21] S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-sync Protocol for Sensor Networks. In *ACM SenSys*, 2003.

[22] Y. Gu and T. He. Data Forwarding in Extremely Low Duty-Cycle Sensor Networks with Unreliable Communication Links. In *ACM SenSys*, 2007.

[23] Y. Gu, J. Hwang, T. He, and D. H.-C. Du. μSense: A Unified Asymmetric Sensing Coverage Architecture for Wireless Sensor Networks. In *IEEE ICDCS*, 2007.

[24] C. Gui and P. Mohapatra. Power Conservation and Quality of Surveillance in Target Tracking Sensor Networks. In *ACM MobiCom*, 2004.

[25] X. Guo. Broadcasting for Network Lifetime Maximization in Wireless Sensor Networks. In *IEEE SECON*, 2004.

[26] C. Hartung, R. Han, C. Seielstad, and S. Holbrook. FireWxNet: A Multi-Tiered Portable Wireless System for Monitoring Weather Conditions in Wildland Fire Environments. In *ACM MobiSys*, 2006.

[27] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan. An Application-Specific Protocol Architecture for Wireless Microsensor Networks. *IEEE Transactions on Wireless Communications*, 1(4):660–670, October 2002.

[28] Y. T. Hou, Y. Shi, H. D. Sherali, and S. F. Midkiff. Prolonging Sensor Network Lifetime with Energy Provisioning and Relay Node Placement. In *IEEE SECON*, 2005.

[29] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *ACM MobiCom*, 2000.

[30] R. Iyengar, K. Kar, and S. Banerjee. Low-coordination Topologies for Redundancy in Sensor Networks. In *ACM MobiHoc*, 2005.

[31] H. Karl and A. Willig. *Protocols and Architectures for Wireless Sensor Networks*. Wiley, 2005.

[32] A. Kashyap, S. Khuller, and M. Shayman. Relay Placement for Higher Order Connectivity in Wireless Sensor Networks. In *IEEE INFOCOM*, 2006.

[33] A. Kashyap, S. Khuller, and M. Shayman. Relay Placement for Higher Order Connectivity in Wireless Sensor Networks. In *IEEE INFOCOM*, 2006.

[34] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon. Health Monitoring of Civil Infrastructures Using Wireless Sensor Networks. In *ACM/IEEE IPSN*, 2007.

[35] Y. Kim, T. Schmid, Z. M. Charbiwala, J. Friedman, and M. B. Srivastava. NAWMS: Nonintrusive Autonomous Water Monitoring System. In *ACM SenSys*, 2008.

[36] J. M. Ko, Y. Q. Ni, H. F. Zhou, J. Y. Wang, and X. T. Zhou. Investigation Concerning Structural Health Monitoring of an Instrumented Cable-Stayed Bridge. *Structure and Infrastructure Engineering*, 2008.

[37] P. Kyasanur, R. R. Choudhury, and I. Gupta. Smart Gossip: An Adaptive Gossip-based Broadcasting Service for Sensor Networks. In *IEEE MASS*, 2006.

[38] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks. In *USENIX NSDI*, 2004.

[39] B. Li, D. Wang, F. Wang, and Y.-Q. Ni. High Quality Sensor Placement for SHM Systems: Refocusing on Application Demands. In *IEEE INFOCOM*, 2010.

[40] M. Li and Y. Liu. Rendered Path: Range-Free Localization in Anisotropic Sensor Networks with Holes. In *ACM MobiCom*, 2007.

[41] Q. Li and D. Rus. Global Clock Synchronization in Sensor Networks. In *IEEE INFOCOM*, 2004.

[42] Y. Li, L. Qiu, Y. Zhang, R. Mahajan, and E. Rozner. Predictable Performance Optimization for Wireless Networks. In *ACM SIGCOMM*, 2008.

[43] G. Lin and G. Xue. Steiner Tree Problem with Minimum Number of Steiner Points and Bounded Edge-Length. *Information Processing Letters*, 69:53–57, 1999.

[44] S. Lindsey, C. Raghavendra, and K. M. Sivalingam. Data Gathering Algorithms in Sensor Networks using Energy Metrics. *IEEE Transactions on Parallel and Distributed Systems*, 13(9):924–935, September 2002.

[45] J. Liu, F. Zhao, P. Cheung, and L. Guibas. Apply Geometric Duality to Energy-efficient Non-local Phenomenon Awareness using Sensor Networks. *IEEE Wireless Communications*, 11(8):62–68, 2004.

[46] E. L. Lloyd and G. Xue. Relay Node Placement in Wireless Sensor Networks. *IEEE Transactions on Computers*, 56(1):134–138, January 2007.

[47] G. Lu, B. Krishnamachari, and C. S. Raghavendra. An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Networks. In *IEEE IPDPS*, 2004.

[48] M. Ma and Y. Yang. SenCar: An Energy-Efficient Data Gathering Mechanism for Large-Scale Multihop Sensor Networks. *IEEE Transactions on Parallel and Distributed Systems*, 18(10):1476–1488, October 2007.

[49] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava. Coverage Problems in Wireless Ad-hoc Sensor Networks. In *IEEE INFOCOM*, 2001.

[50] M. Miller, C. Sengul, and I. Gupta. Exploring the Energy-Latency Tradeoff for Broadcasts in Energy-Saving Sensor Networks. In *IEEE ICDCS*, 2005.

[51] S. Misra, S. D. Hong, G. Xue, and J. Tang. Constrained Relay Node Placement in Wireless Sensor Networks to Meet Connectivity and Survivability Requirements. In *IEEE INFOCOM*, 2008.

[52] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The Broadcast Storm Problem in a Mobile Ad Hoc Network. In *ACM MobiCom*, 1999.

[53] S. Olariu and I. Stojmenovic. Design Guidelines for Maximizing Lifetime and Avoiding Energy Holes in Sensor Networks with Uniform Distribution and Uniform Reporting. In *IEEE INFOCOM*, 2006.

[54] S. PalChaudhuri, A. Kumar Saha, and D. B. Johnson. Adaptive Clock Synchronization in Sensor Networks. In *ACM/IEEE IPSN*, 2004.

[55] N. A. Pantazis and D. D. Vergados. A Survey on Power Control Issues in Wireless Sensor Networks. *IEEE Communications Surveys and Tutorials*, 9(4):86–107, 2007.

[56] L. Paradis and Q. Han. TIGRA: *Ti*mely Sensor Data Collection Using Distributed *Gra*ph Coloring. In *IEEE PerCom*, 2008.

[57] K. S. J. Pister and L. Doherty. TSMP: Time Synchronized Mesh Protocol. In *IASTED International Symposium on Distributed Sensor Networks (DSN)*, 2008.

[58] J. Polastre, J. Hill, and D. Culler. Versatile Low Power Media Access for Wireless Senor Networks. In *ACM SenSys*, 2004.

[59] R. Rajagopalan and P. K. Varshney. Data-Aggregation Techniques in Sensor Networks: A Survey. *IEEE Communications Surveys and Tutorials*, 8(4):48–63, 2006.

[60] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves. Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks. In *ACM SenSys*, 2003.

[61] I. Rhee, A. Warrier, M. Aia, and J. Min. Z-MAC: a Hybrid MAC for Wireless Sensor Networks. In *ACM SenSys*, 2005.

[62] A. Rowe, V. Gupta, and R. (R.) Rajkumar. Low-power Clock Synchronization using Electromagnetic Energy Radiating from AC Power Lines. In *ACM SenSys*, 2009.

[63] T. Schmid, Z. Charbiwala, Z. Anagnostopoulou, M. B. Srivastava, and P. Dutta. A Case Against Routing-Integrated Time Synchronization. In *ACM SenSys*, 2010.

[64] L. Selavo, A. Wood, Q. Cao, T. Sookoor, H. Liu, A. Srinivasan, Y. Wu, W. Kang, J. Stankovic, D. Young, and J. Porter. LUSTER: Wireless Sensor Network for Environmental Research. In *ACM SenSys*, 2007.

[65] R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks. In *IEEE International Workshop on Sensor Network Protocols and Applications*, 2003.

[66] W.-Z. Song, R. Huang, M. Xu, A. Ma, B. Shirazi, and R. LaHusen. Air-dropped Sensor Network for Real-time High-fidelity Volcano Monitoring. In *ACM MobiSys*, 2009.

[67] W.-Z. Song, F. Yuan, and R. LaHusen. Time-Optimum Packet Scheduling for Many-to-One Routing in Wireless Sensor Networks. In *IEEE MASS*, 2006.

[68] F. Stann, J. Heidemann, R. Shroff, and M. Z. Murtaza. RBP: Robust Broadcast Propagation in Wireless Networks. In *ACM SenSys*, 2006.

[69] R. Szewczyk, A. Mainwaring, J. Polastre, and D. Culler. An Analysis of a Large Scale Habitat Monitoring Application. In *ACM SenSys*, 2004.

[70] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong. A Macroscope in the Redwoods. In *ACM SenSys*, 2005.

[71] M. Tubaishat and S. Madria. Sensor Networks: an Overview. *IEEE Potentials*, 22(2):20–23, April/May 2003.

[72] F. Wang and J. Liu. RBS: A Reliable Broadcast Service for Large-Scale Low Duty-Cycled Wireless Sensor Networks. In *IEEE International Conference on Communications*, 2008.

[73] F. Wang and J. Liu. Duty-Cycle-Aware Broadcast in Wireless Sensor Networks. In *IEEE INFOCOM*, 2009.

[74] F. Wang and J. Liu. Networked Wireless Sensor Data Collection: Issues, Challenges, and Approaches. *IEEE Communications Surveys and Tutorials*, 13(4):673–687, 2011.

[75] F. Wang and J. Liu. On Reliable Broadcast in Low Duty-Cycle Wireless Sensor Networks. *IEEE Transactions on Mobile Computing*, 11(5):767–779, May 2012.

[76] F. Wang, D. Wang, and J. Liu. Traffic-Aware Relay Node Deployment for Data Collection in Wireless Sensor Networks. In *IEEE SECON*, 2009.

[77] F. Wang, D. Wang, and J. Liu. Traffic-Aware Relay Node Deployment: Maximizing Lifetime for Data Collection Wireless Sensor Networks. *IEEE Transactions on Parallel and Distributed Systems*, 22(8):1415–1423, August 2011.

[78] F. Wang, D. Wang, and J. Liu. Utilizing Elevator for Wireless Sensor Data Collection in High-Rise Structure Monitoring. In *ACM/IEEE IWQoS*, 2011.

[79] F. Wang, D. Wang, and J. Liu. EleSense: Elevator-Assisted Wireless Sensor Data Collection for High-Rise Structure Monitoring. In *IEEE INFOCOM*, 2012.

[80] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks. In *ACM SenSys*, 2003.

[81] G. WernerAllen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and Yield in a Volcano Monitoring Sensor Network. In *USENIX OSDI*, 2006.

[82] G. Xing, X. Wang, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks. *ACM Transactions on Sensor Networks*, 1(1):36–72, 2005.

[83] K. Xu, H. Hassanein, and G. Takahara. Relay Node Deployment Strategies in Heterogeneous Wireless Sensor Networks: Multiple-Hop Communication Case. In *IEEE SECON*, 2005.

[84] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A Wireless Sensor Network For Structural Monitoring. In *ACM SenSys*, 2004.

[85] G. Xue, T. P. Lillys, and D. E. Dougherty. Computing the Minimum Cost Pipe Network Interconnecting One Sink and Many Sources. *SIAM Journal of Optimization*, 10(1):22–42, October 1999.

[86] G. Xue and Y. Ye. An Efficient Algorithm for Minimizing a Sum of Euclidean Norms with Applications. *SIAM Journal of Optimization*, 7(4):1017–1036, November 1997.

[87] T. Yan, T. He, and J. Stankovic. Differentiated Surveillance Service for Sensor Networks. In *ACM SenSys*, 2003.

[88] H. Zhang and J. Hou. On Deriving the Upper Bound of α-Lifetime for Large Sensor Networks. In *ACM MobiCom*, 2004.

[89] H. Zhang and J. C. Hou. Is Deterministic Deployment Worse than Random Deployment for Wireless Sensor Networks? In *IEEE INFOCOM*, 2005.

[90] Q. Zhang and Y.-Q. Zhang. Cross-Layer Design for QoS Support in Multi-hop Wireless Networks. *Proceedings of the IEEE*, 96(1):64–76, January 2008.

[91] W. Zhang, G. Xue, and S. Misra. Fault-tolrant Relay Node Placement in Wireless Sensor Networks: Problems and Algorithms. In *IEEE INFOCOM*, 2007.