End-to-end Delay Analysis for Event-driven Wireless Sensor Network Applications

Bo Jiang

Preliminary examination proposal submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

May 4, 2009
Blacksburg, Virginia

Keywords: Sensor Network, End-to-end Delay, Event-driven, Target Prediction, Real-time Capacity

End-to-end Delay Analysis for Event-driven Wireless Sensor Network Applications

Bo Jiang

(ABSTRACT)

The end-to-end delay is one of the most critical and fundamental issues for wireless sensor networks. Many applications of sensor networks require an end-to-end delay guarantee for time sensitive data. However, the end-to-end delay is difficult to bound for event-driven sensor networks, where nodes generate and propagate data only when an event of interest occurs, thereby producing unpredictable traffic load. Meanwhile, the end-to-end delay is tightly banded with many other factors, e.g., energy and network capacity.

In this dissertation proposal, we analyze the quantitative relation among the end-to-end delay, the probability of guaranteeing this delay, and network parameters for event-driven sensor networks. We consider two sub-delays of the end-to-end delay, i.e., detection delay and queuing delay, based on a two-phase model. This two-phase model includes an event observation phase and a data propagation phase. We use target tracking as the example of an event-driven wireless sensor network application.

For the event observation phase, we present a target prediction and sleep scheduling scheme, called TPSS, to reduce the energy consumption, while satisfying the given delay constraint. Wireless sensor nodes are typically in the sleep state most of the time to prolong the network lifetime, but this will increase the detection delay. Proactive wake-up and sleep scheduling are commonly used approaches to solve this problem. The purpose of TPSS is to select the nodes to be sleep-scheduled and reduce their wake-up time as much as possible, so as to enhance the energy efficiency as well as satisfy the given detection delay constraint. First, we design a target prediction method based on kinematics rules and theory of probability. Then based on the prediction results, we design a novel sleep scheduling mechanism that reduces the number of awakened nodes and schedules their sleep patterns in an integrated manner for enhancing energy efficiency. We analyze the detection delay and the detection probability under TPSS, and conduct simulation-based experimental studies. Our simulation results show that TPSS achieves a better tradeoff between energy efficiency and tracking performance than existing works.

For the data propagation phase, we leverage queuing theory, and analyze the delay from the point of view of network capacity. In a many-to-one data gathering network, the throughput of a node can be estimated based on its distance from the sink node. Thus, the expected waiting time of a packet in the queue, i.e., the queuing delay, can be estimated by approximating all the nodes that are h hops away from the sink node as a queue. Based on the actual network requirement on throughputs and results from queuing theory, we then develop a slack time distribution scheme for unbalanced many-to-one traffic patterns. We also introduce the concept of per-hop success probability, which is defined as the probability for a packet to meet its deadline at each hop. Finally, we define and analyze the network-wide real-time capacity, i.e., given a threshold for the per-hop success probability, how much data (in bits per second) can be delivered to the sink node, meeting their deadlines. An important advantage of the per-hop success probability concept is that application designers can configure a packet's deadline based on the required successful delivery probability.

Contents

1 Introduction **1**

1.1 Overview 1

1.2 Summary of Current Research and Contributions 4

1.3 Summary of Proposed Post Preliminary Exam Work 7

1.4 Proposal Outline 7

2 Related Works **9**

2.1 Real-time 9

2.2 Sleep Scheduling and Target Prediction 9

2.3 Network Capacity 11

2.4 Uniqueness 12

3 Preliminaries, Models, Performance Metrics and Notations **13**

3.1 Assumptions 13

3.2 Models 15

3.2.1 Network Model 15

3.2.2 Target Model 16

3.2.3 Event Model 17

3.2.4 Capacity Model 17

3.2.5 Deadline and Delay Model 17

3.3 Performance Metrics 18

3.4 Summary of Notations 19

4 Detection Delay in Event Observation Phase 21

4.1 Introduction 21

4.2 Target Motion Prediction 24

4.2.1 Calculate the Current State 25

4.2.2 Kinematics-based Prediction 25

4.2.3 Probability-based Prediction 26

4.3 Energy Conservation 28

4.3.1 Reducing the Number of Awakened Nodes 28

4.3.2 Sleep Scheduling for Awakened Nodes 32

4.4 Algorithm Descriptions 33

4.5 Analysis 35

4.5.1 Detection Area 36

4.5.2 Single Node Detection Probability 37

4.5.3 Detection Probability 40

4.5.4 Detection Delay 41

4.6 Performance Evaluation 42

4.6.1 Simulation Environment 42

4.6.2 Experimental Results 44

4.7 Conclusion 48

5 Propagation Delay in Data Propagation Phase 49

5.1 Introduction 49

5.2 Traffic Pattern and Node Throughput 51

5.3 Slack Distribution 52

5.4 Real-time Capacity 56

5.4.1 Definition 56

5.4.2 Examples 57

5.5 Conclusions 59

6 Conclusions, Contributions and Proposed Post Preliminary Exam Work 60

6.1 Contributions 61

6.2 Post Preliminary Exam Work 62

List of Figures

1.1 Six Phase Model in [1] 3

1.2 Simplified Two Phase Model 3

3.1 Toggling Period and Duty Cycle 14

3.2 Network Architecture 15

4.1 Foundation, Approaches, and Objective of TPSS 23

4.2 Target Movement States 25

4.3 Prediction based on Kinematics Rules 25

4.4 Example of Gaussian Distribution of S_n 26

4.5 Probability Density Distribution of Δ_n 26

4.6 Probabilistic Model of Moving Direction 27

4.7 The Relationship between S_n and d 32

4.8 Detection Probability and Detection Delay 36

4.9 Detection Area in the Transmission Range 38

4.10 Simplify $E[T_{detection}|\delta = \hat{\delta}]$ 41

4.11 A Typical Target Route 44

4.12 Extra Energy vs. Node Density 45

4.13 Extra Energy vs. Target Speed 45

4.14 Number of Awakened Nodes vs. Node Density 45

4.15 Number of Awakened Nodes vs. Target Speed 45

4.16 Sensing Energy and Communication Energy when Node Density=1.25 $node/100m^2$ and Target Speed=20 m/s 46

4.17 Average Detection Delay vs. Node Density 47
4.18 Average Detection Delay vs. Target Speed 47
4.19 Relative Changes vs. Node Density 48
4.20 Relative Changes vs. Target Speed 48

5.1 Propagation Path and Virtual Queue 51

List of Tables

3.1 Notation of Parameters 20

4.1 Glossary of Terms 23

4.2 Division of Areas 37

4.3 Single Node Detection Probability and Size of Areas 39

4.4 Energy Consumption Rates 43

Chapter 1

Introduction

1.1 Overview

Wireless sensor networks (or WSNs) which are composed of a large number of multi-functional, low cost sensor nodes are increasingly being used for collecting data from a geographical region of interest and reporting them back to sink nodes (or base stations). Nodes in sensor networks are typically capable of sensing, processing, and communicating data. But at the same time, their power supply, and computing and communicating capabilities are strictly constrained. WSNs are typically applied to diverse application domains, including military applications (e.g., battlefield surveillance), environmental applications (e.g., habitat monitoring), health applications (e.g., telemonitoring of human physiological data), and home applications (e.g., home automation) [2].

In terms of the data delivery model, sensor networks and their applications can be classified as continuous, event-driven, observer-initiated, and hybrid [3]. In the continuous model, sensor nodes periodically deliver the observed data at a specified rate, e.g., temperature monitoring. In an event-driven sensor network, nodes collect and deliver data only when an event of interest, e.g. a vehicle intruding into a surveillance field, occurs. For the observer-initiated model, nodes report their observations in response to an explicit request from sink nodes. These three models can coexist in the same sensor network, which results in a hybrid model. Except for the hybrid model, the event-driven model is the most challenging one among the other three for modeling and analysis of traffic pattern and quality of service (or QoS). The reason is that the occurrence of events is completely unpredictable, resulting in arbitrary traffic patterns.

The event-driven model of sensor networks supports many applications, such as flood detection [4], telemonitoring of human health status [5], vehicle anti-theft [6] and target tracking [7]. Among these applications, those that involve mobile events are more challenging than that for static events, since mobile events need to be tracked in a continuous manner.

We use target tracking, one of the most typical applications for mobile event observation, as the example. Target tracking is usually used for monitoring a geographic region where mobile persons or vehicles may intrude.

Sensor network applications have many critical QoS requirements, among which meeting end-to-end delay constraints is an important one. Many WSN applications require an end-to-end delay guarantee for time sensitive data. For example, sensor and actor networks [8] require sensors to collect and propagate information in a timely manner so that actors can take timely actions. A target tracking system [7] may require sensors to collect and deliver target information to sink nodes before the target leaves the surveillance field. However, the end-to-end delay is difficult to bound for event-driven sensor networks due to their unpredictable traffic pattern. At the same time, the end-to-end delay is often tightly banded with many other factors, for example:

1) *Energy.* Energy efficiency is critical in WSNs, because nodes run on batteries and they are generally difficult to be recharged once deployed. There are many approaches for enhancing energy efficiency such as sleep scheduling [9], optimizing the sensing coverage or the network topology [10], and controlling the RF radio [11]. Sleep scheduling is one of the most commonly used mechanisms, by which most sensor nodes are put into a sleep state for most of the time, and are only awakened periodically or on demand. If energy efficiency is enhanced for prolonging the network lifetime, the quality of service including the end-to-end delay will definitely be impaired. For example, forcing nodes to sleep will result in missing events so as to increase the event observation delay.

2) *Capacity.* The queuing delay is one of the major delay sources during data propagation. Other sources for the propagation delay include the transmission delay and the sleep delay [12, 13]. But the transmission delay is usually specific for the actual hardware and the MAC protocol used [14], thus is relatively fixed for a specific deployment. And in a duty cycling WSN, the sleep delay of each hop is equal to the toggling period. Thus we focus on the queuing delay for data propagation. Queuing delay is caused by constrained network capacity, which defines how much data a WSN can collect and report. When the traffic load in a network exceeds the network capacity, a lot of congestion will happen thus cause a long queuing delay, which contributes to increasing the end-to-end delay.

An analytical description of the end-to-end delay involves parameters from multiple dimensions, including descriptions of energy and capacity. Leveraging the idea of divide and conquer [15], event-driven applications can be divided into multiple phases, so that a high-dimensional design space can be transformed to multiple low-dimensional ones. For example, the operation of a sensor and an actor network may be divided into four phases, including sensing, communicating, processing, and acting. More phases may be inserted if an application requires more operations, such as data aggregation. A partition scheme is mainly dependent on the application requirements. In [1], He *et al.* present a six-phase model for target tracking applications, as shown in Figure 1.1.

Based on such a partitioned phase model, the end-to-end delay may be partitioned into

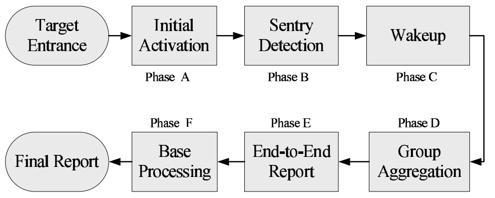

Figure 1.1: Six Phase Model in [1]

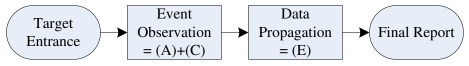

Figure 1.2: Simplified Two Phase Model

multiple sub-delays, one for each phase. Inside each phase, the factors that may impact the delay will be constrained efficiently. For example, network capacity can be considered only for the data propagation phase. If each of these sub-delays can be guaranteed for a given partition scheme, the end-to-end delay will be guaranteed.

For simplicity and feasibility of partitioning the end-to-end deadline, we introduce a two-phase partition scheme for event-driven applications, i.e., event observation and data propagation, as shown in Figure 1.2. Although the six-phase model is practical for actual implementations, it is complicated for analysis, and thereby not feasible for obtaining system-wide, globally optimal solutions. We simplify the partition problem by removing relatively independent and application-specific phases, and establish a simpler but more generic model for event-driven applications. Our modifications include:

1) Removing phase D (group aggregation) and phase F (base processing), because these are relatively independent and can be considered as extended phases based on the fundamental ones;

2) Removing phase B (sentry detection), as it is often specific to applications, hardware and deployment environments; and

3) Combining phase A (initial activation) and phase C (wake-up) into one event observation phase. Unlike in the six-phase model where phase C aims at increasing the confidence in detection and is part of the detection phase, the proactive wake-up mechanism introduced in this proposal aims at preparing neighbors for approaching targets so as to enhance the tracking performance [16, 17]. Tracking is a continuous process consisting of multiple detections, since nodes are required to report the new positions of targets constantly. When nodes work in the default duty cycling mode (usually with a very low duty cycle [18]), the first detection delay may be very long. It is the proactive wake-up mechanism that can significantly reduce the detection delay for the subsequent detections after the first one, by broadcasting alarms to wake up neighbors proactively. In this case, the wake-up delay is not part of the delay in the detection phase. Thus we combine phase C (wake-up) with phase A (initial activation), eventually as the event observation phase.

Thesis Problem Statement. The central question that we are asking in the dissertation proposal is: what is the quantitative relation among the end-to-end delay (denoted as D), the probability of guaranteeing this delay (denoted as P) and network parameters in event-driven sensor networks?

Such a quantitative relation is helpful for system designers to configure a sensor network in order to satisfy certain real-time constraints. Network parameters may include but not limited to the node density, the average target speed, the toggling period, and the duty cycle. Detailed definitions for these parameters will be given in Chapter 3.

If we could establish this quantitative relation, we would be able to answer the following questions based on the two-phase model:

1) Given D and network parameters, what is the maximum P and how to partition D to achieve this maximum P?

2) Given D and P, is this P achievable? If yes, how to partition D and configure the network parameters to achieve it?

3) Given P and network parameters, what is the minimum D to achieve this P? How to partition D to achieve it?

4) Given D and P, what is the minimum possible energy consumption? How to partition D and configure the network parameters to achieve it?

5) Given D and P, what is the minimum required network capacity? How to partition D and configure the network parameters to achieve it?

We analyze this quantitative relation by studying D and P in two phases, each of which corresponds to a phase as part of the end-to-end data collection process. In other words, we consider two sub-delays (i.e., a detection delay and a queuing delay) of the end-to-end delay D, partition the end-to-end probability P of guaranteeing D into a detection probability and a propagation probability, and compute them respectively in two phases.

1.2 Summary of Current Research and Contributions

As previously discussed, we analyze the relation among the end-to-end delay, the probability of guaranteeing this delay, and network parameters with the two-phase model.

For the event observation phase, we establish a quantitative relation between the parameters for sleep scheduling and detection delay/probability. The detection delay is counted starting from the time when a target of interest enters the surveillance field or when it is lost by the nodes, to the time when it is detected and reported. The detection delay is mainly caused by the duty cycling of nodes. For prolonging the network lifetime, nodes should be in their sleep state as long as possible, and only wake up for sensing or communicating periodically or on demand. While sleeping, nodes may miss a moving target and delay the detection. Thus all the parameters for sleep scheduling may influence the detection delay and the detection probability. At the same time, we also consider energy efficiency, so that the energy consumption can be reduced as much as possible while the detection delay is guaranteed probabilistically.

We first develop a *Target Prediction and Sleep Scheduling scheme* (or TPSS) that sleep-schedules nodes for optimizing the tradeoff between detection delay and energy efficiency. The basic idea is that, nodes work in the default duty cycling mode with a very low duty cycle when no event is detected. Once an event is detected, an alarm message is broadcast to the nodes around the target to have their sleep patterns scheduled, e.g., to wake them up for some time. This will improve the tracking performance, including the detection delay and the detection probability. At the same time, nodes are carefully scheduled to reduce the wake-up time as much as possible, so that energy efficiency can be optimized.

The scheduling of nodes depends on the prediction of the target's movement. Once a target's potential movement is predicted, we may take a high probability to awaken nodes on a direction along which the target is highly probable to move, and take a low one to awaken nodes that are not likely to detect the target. Our proposed target prediction is composed of kinematics-based prediction and probability-based prediction. Kinematics-based prediction calculates the expected displacement of the target in a sleep delay, which shows the position and the moving direction that the target is most likely to be in and move along. Based on this expected displacement, probability-based prediction establishes probabilistic models for the scalar displacement and the deviation. These probabilistic models decide which nodes should be awakened for some time, and how long they should wake up.

Based on these probabilistic models, we utilize two approaches to enhance energy efficiency: 1) reduce the number of awakened nodes and 2) schedule the sleep pattern of awakened nodes. On the one hand, those nodes that the target may have already passed during the sleep delay do not need to be awakened. On the other hand, nodes that lie on a direction that the target has a low probability of passing by could be chosen to be awakened with a low probability. Therefore the number of awakened nodes can be reduced significantly. These chosen awakened nodes could be awakened only during the period when the target is highly likely to cross their sensing area. Then their wake-up time can be curtailed as much as possible.

Under TPSS, we then analyze the detection probability and the detection delay. We also conduct simulation-based experimental studies to understand the tradeoff between the tracking performance and energy efficiency. Our simulation results show that, compared with the MCTA algorithm [11] (a past algorithm for reducing the energy consumption for tracking mobile targets), the TPSS scheme exceeds the "net profit" (i.e., subtracting the input, performance loss, from the output, energy efficiency) by $(20\% - 30\%)$.

For the data propagation phase, we leverage results from queuing theory to establish a quantitative relation between network capacity and the queuing delay. The queuing delay is the total time that a packet spends in the queues during propagation. As the network capacity is constrained, packets may have to wait for a while until all the packets with a higher priority than it are transmitted.

The basic idea is that, when approximating all the nodes that are h hops away from the sink node as a queue, the expected waiting time of a packet in the queue can be determined from the queue's outgoing and incoming throughputs. In a many-to-one data gathering network, all the traffic produced by nodes that are h hops or further away from the sink node have to be consumed by nodes that are $h-1$ hops away from the sink node. Based on this observation and a given event distribution model, we can compute the average incoming and outgoing throughputs of a node at hop layer h. Then we assume that all the nodes that are h hops away from the sink node form a queue. Using results from queuing theory, we can calculate the expected queuing delay. Summing up all these expected delays yields an end-to-end expected delay. We introduce a new slack distribution scheme to distribute the end-to-end slack time proportional to the expected delay in the end-to-end expected delay. We prove that with such a distribution scheme, the end-to-end probability of delivering a packet within the given deadline is optimal. In fact, this simultaneously answers the question of real-time capacity, i.e., the network's capability to transmit time sensitive data within deadlines.

The research contributions of this proposal include:

1. We established a quantitative relation among the end-to-end delay, the probability of guaranteeing this delay and network parameters in event-driven sensor networks. Based on such a quantitative relation, system designers can configure the sensor network in order to satisfying certain real-time constraints.

2. We presented a two-phase model for event-driven applications, i.e., event observation and data propagation. This partitioning scheme is simplified, based on a six-phase model, thus feasible for determining the system-wide global optimal solution.

3. We designed a target prediction method based on both kinematics rules and theory of probability. From the prediction result, the system can make a judgment on the probability distribution of the target's position and moving direction.

4. We designed a novel approach that reduces the number of awakened nodes and schedules their sleep patterns in an integrated manner for enhancing energy efficiency.

5. We analyzed the detection delay and the detection probability under the TPSS scheme, and evaluated it by simulation-based experimental studies. The simulation results show that TPSS achieves a better tradeoff between energy efficiency and tracking performance than existing works.

6. We designed a new slack time distribution scheme based on the requirements of realistic networks and results from queuing theory.

7. We introduced the concept of per-hop success probability, in terms of which we show that our slack time distribution scheme is optimal.

1.3 Summary of Proposed Post Preliminary Exam Work

Based on the current research results, the proposed post preliminary exam work includes the following questions:

- *Five questions.* Based on the quantitative relation presented in this proposal, we propose to answer all the five questions listed in Section 1.1 after the preliminary exam.
- *Multiple sink nodes.* In realistic sensor networks, designers usually deploy multiple sinks to reduce the distance from source nodes to the sink, so as to improve the network performance such as the end-to-end delay. The propagation delay and the network capacity of sensor networks with multiple sink nodes may be very different from the one sink case. We propose to extend the analysis result of the data propagation phase to the multiple sink case.
- *Multiple targets.* The current research for the event observation phase is constrained to the single target tracking case. When multiple targets move close to each other, the redundant alarm messages of interfering targets may be leveraged for further enhancing energy efficiency. We propose to answer the question that how much more energy can be saved for multiple target tracking.
- *Dynamic event distribution.* Dynamic event means that the location where events occur is dynamic. Instead of considering static event distribution, it would be more accurate and more adaptive if dynamic event distribution is considered with random processes for the data propagation phase. We propose to extend the analysis result of the data propagation phase to dynamic event distribution.
- *Potential applications.* We discussed several potential applications of the slack time distribution scheme and real-time capacity in this proposal. Many more potential applications can be explored based on the research results of the relation among delay, probability, and network parameters. We propose to provide more scenarios where the results of this proposal can help on the design of a sensor network.

1.4 Proposal Outline

The rest of this dissertation proposal is organized as follows. Past and related works are discussed in Chapter 2. In Chapter 3, we introduce our assumptions, models, performance metrics, and summary of notations. Chapter 4 focuses on the event observation phase, including the target prediction scheme, the energy conservation approaches, distributed algorithm description, analysis and simulation results. In Chapter 5, we discuss the data propagation phase, including the relationship between the event distribution and the expected delay, a new slack distribution scheme, the analysis on per-hop success probability, and the definition of real-time capacity. In Chapter 6, we conclude the proposal, state our contributions, and discuss the proposed post preliminary exam work.

Chapter 2

Related Works

2.1 Real-time

Existing works on the real-time feature of WSNs studied latency and timely delivery from various perspectives, some of which are introduced as follows with the examples. Abdelzaher *et. al.* discussed a WSN's real-time capacity from a macro perspective without making any detailed assumptions in [19]. On the contrary, He *et. al.* presented specific delays for each chain during the end-to-end data propagation of the target tracking application in [1], for which all the calculations were based on the actual implementation of their testbed VigilNet [20]. Some research works utilized real-time performance as a constraint for studying other related problems, e.g., sleep scheduling [21], data aggregation [22], MAC protocol [23], and routing [24]. And some works aimed at achieving an optimized energy-latency tradeoff [25,26].

2.2 Sleep Scheduling and Target Prediction

Past efforts on sleep scheduling can be classified based on several criteria: 1) in terms of collaboration, they may be classified into independent mode [27,28], synchronized mode [29], central controller based mode [17,30], and collaboration mode [21,31–34]; and 2) in terms of target motion engagement, they can be classified as having no engagement [1,21,27,30–34], engaged without prediction [28], and engaged with prediction [11,17,35–37].

In the past, most sleep scheduling works have focused on collaboration mode, and without target motion engagement. Even those which have target motion engaged do not fully leverage the target motion model to improve the tradeoff between energy efficiency and tracking performance. For example, some may schedule the sleep pattern based only on the distance of a node from the target's current position: the further a node is away from the target, the deeper its sleep level would be. In other words, they only consider a target's velocity magnitude. Such a sleep scheduling algorithm is often called the "circle-based scheme" or "legacy scheme" in the literatures [16, 28]. In this legacy circle-based scheme (or Circle), all the nodes in a circle will follow the same sleep pattern. For example, they will wake up at the same time and usually keep active all the time during an expected period, without distinguishing among different possible future directions and speeds of the target.

However, a target in the real world has its specific purpose and will move towards a specific direction in general. Also, it may follow rules of physics instead of moving randomly. Therefore, we can leverage the target's motion model as much as possible: if some rules of classical mechanics are taken into account, not all the nodes in a circle need to be awakened for tracking and not all the awakened nodes need to wake up all the time during the scheduled period, thus the consumed energy could potentially be reduced. The two branches of the classical mechanics are kinematics and dynamics. Kinematics describes the motion of objects without consideration of the circumstances leading to the motion, while dynamics studies the relationship between the motion of objects and its causes [38].

In [11], Jeong *et. al.* present the Minimal Contour Tracking Algorithm (or MCTA) that uses the strategy of reducing the number of awakened nodes. MCTA depends on kinematics to compute the contour of tracking areas. However, they mainly discuss the motion of vehicles and only the vehicular kinematics is used for the computation. Actually a target may also be a person with much more random motion characteristics than a vehicle. Furthermore, MCTA keeps all the nodes in the contour active without any differentiated sleep scheduling.

Xu *et. al.* present a Prediction-based Energy Saving scheme (or PES) in [37]. Similar to ours, PES utilizes the target's predicted motion pattern to reduce the missing rate and save energy. PES also depends on kinematics for the prediction. However, it only uses simple models to predict a possible destination without considering the detailed moving probabilities. Furthermore, PES makes an assumption of a hexagon topology in which all the nodes are evenly dispersed, and neighboring sensor nodes do not have any overlapping sensing area. This is a very strict assumption, as nodes are usually deployed totally randomly. In addition, PES does not consider sleep scheduling, either.

In [39], Taqi *et. al.* discuss a dynamics-based prediction protocol named as A-YAP. They leverage some results of the physics research including the yaw rate and the side force. However, this requires the surveillance system to recognize the target mass, which relies on target classification. In many cases, target classification is difficult especially when the real-time tracking constraint is applied. Moreover, A-YAP also predicts an exact position that the target is probably moving to.

2.3 Network Capacity

The capacity of wireless ad hoc networks has been well studied in the past several years since the work of Gupta and Kumar [40]. In [40], the authors estimate the achievable per node throughput for a static ad hoc network as $\Theta(\frac{1}{\sqrt{n \log n}})$, and further show that the per node throughput cannot exceed $\Theta(\frac{1}{\sqrt{n}})$ even if nodes are optimally deployed and the transmission range is optimally chosen. Their work is based on a one-to-one balanced traffic pattern, where each node generates equal amount of traffic loads and destinations are randomly chosen. Based on [40], many further results were developed. In [41], Grossglauser *et. al.* show that the capacity can be further increased if node mobility is introduced to reduce the path length between the source node and the destination, but with unbounded delay as a cost. In [42], Bansal *et. al.* improve the work in [41] by providing low delay guarantees. Gastpar *et. al.* study a one-to-one traffic pattern in [43], where there is only one source-destination pair and all the other nodes work as relays. Li *et. al.* present the capacity analysis for some typical topologies in [44]. Beyond these typical but simple network architectures, Liu *et. al.* describe the throughput capacity of hybrid wireless networks with sparse base stations connected via a high-bandwidth wired network in [45]. In [46], Jain *et. al.* model the interference using a conflict graph and study the maximum throughput with any given network and workload specified as inputs. Most of these previous works on the capacity of wireless ad hoc networks only consider continuous balanced traffic loads.

Based on the results for wireless ad hoc networks, network capacity has also been studied for wireless sensor networks. One of the major differences of WSNs from ad hoc networks is that WSNs emphasize the many-to-one traffic pattern [47,48], as WSNs are often used to collect and report data to a small number of sink nodes. In addition, due to the resource constraints of WSNs, their network capacity is often discussed together with deployment [49], network architecture [50–52], data aggregation or in-network computation [53,54].

One of the earliest works to study real-time capacity for sensor networks is [19]. Before this effort, there have been some attempts on combining delay and capacity such as [42,55]. However, [42,55] consider delay only as a constraint for the capacity, i.e., these works do not address the question of how much real-time data a network can transmit.

In [19], Abdelzaher *et. al.* present real time capacity analysis for sensor networks with balanced loads and continuous convergecast traffic. They leverage their preceding works on real-time scheduling that specified utilization bounds [56], and assume time independent fixed priority scheduling policy for the analysis. However, [19] does not consider event-driven sensor networks. Moreover, [19] considers a packet's status as unchanged along the whole propagation path. The definition of their synthetic utilization depends only on the packet size and its end-to-end deadline, while these two factors remain unchanged for a packet during its entire lifetime. However, packets often experience variable message velocity (i.e., the speed of data propagation) at multiple hops due to the congestion around sink nodes caused by convergecast traffic. The utilization that packets impose on nodes therefore should be a function of their ever-changing status, e.g., the distance from the sink, the remaining time to the deadline etc.

2.4 Uniqueness

The two-phase model that we use to partition the end-to-end delay is more simple and feasible than existing partition schemes. It excludes the phases that are specific for applications or relatively independent, and only keeps the two core steps for typical applications of event-driven sensor networks. This makes it possible that a high-dimensional design space be partitioned into multiple low-dimensional ones, so that the end-to-end delay is easy to analyze.

Unlike the existing works on sleep scheduling and target prediction, TPSS takes into account both kinematics rules and probability theory to establish a motion model for target prediction, which is suitable for targets that may move highly randomly. Also TPSS schedules the sleep pattern of awakened nodes individually for saving more energy than an effort that solely reduces the number of awakened nodes.

In contrast with past works on network capacity like [19], our work focuses on slack time distribution scheme and real-time capacity of event-driven sensor networks with completely unbalanced traffic pattern, which has not been studied in the past. And we discuss the capacity based on the per-hop throughput and slack time. This makes it possible to distribute the slack time hop by hop, and schedule the packets based on their ever-changing status.

Chapter 3

Preliminaries, Models, Performance Metrics and Notations

Before the detailed discussion, we first introduce our assumptions, models, performance metrics used for evaluation and summary of notations.

3.1 Assumptions

We make the following assumptions:

- *Network.* We assume a flat network architecture consisting of a large number of homogeneous sensor nodes and one sink node. All the sensor nodes are deployed randomly and uniformly in a disk on the plane, and the only sink node is in the center of this disk field. Except for the single sink, there are no other cluster headers or relay nodes that may be used for composing a hierarchical architecture. In this proposal, we use terms "sensor node" and "node" interchangeably for all the sensor nodes except the sink node, and we often refer to the sink node as the "sink" in short.

- *Nodes.* We assume that all the nodes are equipped with omni-directional antennas, and their locations are static and a priori known via GPS [57] or using algorithmic strategies such as [58]. We assume that the transmission power of sensor nodes' communication radio is fixed, thereby the transmission range, denoted as R, is also fixed. Moreover, we assume that the sensing range of nodes, in which events may be observed by nodes, is fixed and denoted as r. Finally, all the sensor nodes are assumed to be well time synchronized using a protocol such as RBS [59].

- *Communication.* We adopt the *protocol model* introduced in [40], where both transmission and interference depend only on the Euclidean distance between nodes. And we assume that there is only one wireless channel. For wireless communication, we assume that nodes communicate with each other through packets, and concurrent transmissions around a receiver may collide. The protocol model does not consider information theoretic techniques such as network coding, interference cancelation, superposition coding, and coherent combining [53]. Therefore, we simplify the problem by avoiding the complexity of information theory and physical model.

- *Sleep pattern.* We assume that a node has two states, *active state* and *sleep state.* When active, a node turns on its processor, sensing devices and RF transceiver for computing, sensing, and communicating. In a sleep state, a node puts most modules/devices into the sleep state except a wake-up timer with extremely low energy consumption [28]. The sleep pattern of a sensor node can be described by an application specific definition

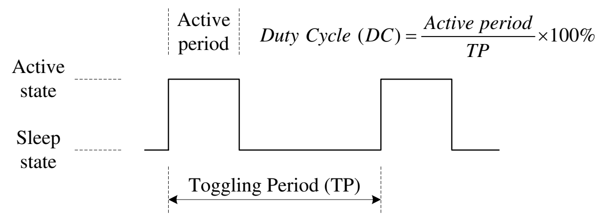
Active period
$Duty\ Cycle\ (DC) = \frac{Active\ period}{TP} \times 100\%$
Active state
Sleep state
Toggling Period (TP)

Figure 3.1: Toggling Period and Duty Cycle

as in [36]. But a more commonly used description is duty cycling with specific toggling period and duty cycle [18]. Figure 3.1 illustrates the concepts of *toggling period* (or TP) and *duty cycle* (or DC). We assume that the default sleep pattern is "random", i.e., before a target is detected thus TPSS is triggered, all the nodes switch between active and sleep states with the same toggling period and the same duty cycle. However, the starting time point of each node's toggling period is random. Within each TP period, a node wakes up and keeps active for $TP * DC$, and then sleeps for $TP * (1 - DC)$ [28]. Also we assume that nodes are able to communicate with each other under such an active/sleep pattern using a MAC protocol such as B-MAC [60].

- *Events.* In an event-driven sensor network, sensor nodes report data towards the sink only when an event of interest occurs. As we use target tracking as our example application, events are the appearance of targets. We assume that targets move in the surveillance field with the ever-changing velocity and direction, and nodes can determine a target's position at a specific time point either by sensing or by calculating—e.g., [16, 61]. Multiple targets are assumed to be distinguished from each other using a target classification algorithm such as [62].

- *MAC protocol.* We assume a MAC protocol with zero overhead and perfect scheduling policy. With a zero-overhead MAC protocol, the traffic transmitted by nodes h hops away from the sink will be able to arrive at nodes $h - 1$ hops away from the sink successfully without being dropped, as long as the total transmission throughput of hop h is less than or equal to the acceptable receiving throughput of hop $h - 1$.

3.2 Models

In this section we introduce the models for the network, the target, events, the capacity, deadline and delay. Although we use target tracking as an example of event-driven applications, we distinguish the event model from the target model. The target model is for target prediction and sleep scheduling, which is concerned with a target's velocity and moving direction etc. But the event model is for data generation and propagation, which is concerned with the distribution of events, and the throughput of nodes etc.

3.2.1 Network Model

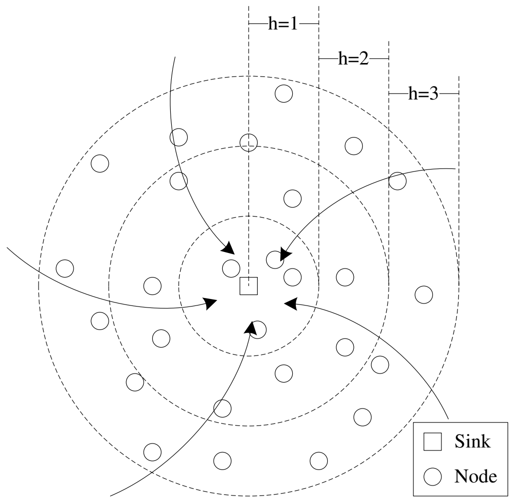

Figure 3.2: Network Architecture

Based on the assumptions that we discussed, Figure 3.2 shows the network architecture in a disk on the plane. The square at the center stands for the sink node, small circles represent nodes, and the large dotted circles show the distance of the nodes from the sink. We count the hop number starting from the sink node. In other words, we say that all the nodes in the ring between $(h-1)R$ and hr belong to *hop layer h*. We denote the maximum hop number as H_{max} and the number of nodes that are located in hop layer h as N_h. In the figure, the curved arrows signify the transmission direction of the convergecast data flows.

With this layered architecture, nodes in hop layer $h > 1$ cannot transmit to the sink directly. Instead, the packets that they generate have to be relayed by nodes in the inner layers. Furthermore, we assume that nodes do not communicate with neighbors that are in the same hop layer; instead they propagate information only to neighbors in the inner layers. With a random and uniform deployment that we assume, such a multi-hop transmission establishes a quantitative relationship among nodes in different layers. Many previous works such as [19, 63] leverage this advantage to simplify the analysis.

We denote the node density of the sensor network as ρ. Now, the number of nodes in hop layer h ($h \geq 1$) can be computed as $N_h = \rho(\pi(hr)^2 - \pi[(h-1)R]^2) = \rho\pi R^2(2h-1)$. In fact, the constant $\rho\pi R^2$ is just the number of nodes in an area within a node's transmission range, i.e., N_1. Thus $N_h = N_1(2h-1)$. When $h = 0$, there is only one sink node, thus $N_0 = 1$.

3.2.2 Target Model

We first define an expression for a vector $\overrightarrow{X}$ in a 2-dimensional plane as $\overrightarrow{X} = (X, \theta)$, where $X = \|\overrightarrow{X}\|$ is its magnitude and $\theta \in (-\pi, \pi]$ is its direction. In an actual field deployed with a WSN, we may simply assign the four directions, south, east, north and west respectively as $-\frac{\pi}{2}$, 0, $\frac{\pi}{2}$ and π.

A target's movement status is a continuous function of time. However, the estimation for a target's movement status is a discrete time process. The surveillance system can only estimate the target states at some time points, and predict the future motion based on the estimation results. Assume that TPSS estimates the target states at time points $\{t_n|n \in N\}$, where $t_i < t_j \ for \ \forall \ i < j \in N$. A state vector, $State(n) = (t_n, x_n, y_n, \overrightarrow{v_n}, \overrightarrow{a_n})$, is defined for each time point t_n to represent the target motion state. In the $State(n)$ vector, (x_n, y_n) is the target position, $\overrightarrow{v_n} = (v_n, \theta_n)$ and $\overrightarrow{a_n}$ are respectively the average velocity vector and the average acceleration vector of the target during (t_{n-1}, t_n), where v_n is the scalar speed and θ_n is the moving direction.

At time point t_n, we may calculate the state vector $State(n)$ based on the discrete time state sequence $\{State(k)|k < n\}$, and then predict $\overrightarrow{S_n}$ (i.e., the displacement vector during $(t_n, t_n + TP)$) according to kinematics rules. These predicted values are denoted as variable names with a prime symbol, such as $\overrightarrow{S_n}' =$ for $\overrightarrow{S_n}$.

Similarly we can also predict the target's displacement within a sleep delay after t_n, denoted as $\overrightarrow{S_n}'$. In the default random sleep pattern, the communication among nodes suffer a sleep delay with a MAC protocol like B-MAC, where a sending node broadcasts the preamble no less than the length of a toggling period to guarantee that each duty cycling receiver can hear it [60]. We suppose the sleep delay exactly as TP for simplification.

However, the predicted displacement $\overrightarrow{S_n}'$ is not enough for describing a target's movement so as to schedule the sleep pattern of nodes. We establish a probabilistic model including two random variables S_n and Δ_n. S_n signifies the magnitude of a target's displacement within a sleep delay, and Δ_n signifies the angle by which the target may deviate from the direction of the predicted displacement $\overrightarrow{S_n}'$. Both S_n and Δ_n are discrete time random processes. Obviously $\mu_{S_n} = \|\overrightarrow{S_n}'\|$ and $\mu_{\Delta_n} = 0$, where μ is a random variable's expected value.

3.2.3 Event Model

For event-driven sensor networks, we establish an event distribution model, which describes how many events may occur and how they occur. Let G_h denote the average traffic (in bits per second) that each node in hop layer h may generate after events are observed. G_h is a function of hop layer h, and represents a distribution of events in the network. For example, a constant G_h (for $\forall$ h) signifies the continuous sensor network model, where each node periodically produces an equal amount of traffic load. For a border surveillance model, where events only occur on the border of the network, G_h is given by:

$$G_h = \begin{cases} constant & , \quad (h = H_{max}) \\ 0 & , \quad (h < H_{max}) \end{cases}$$

In fact, G_h should be a random process $G_h(t)$, as the occurrence of events changes with time. However, we assume that the event distribution is static for simplification. We denote $\hat{G} = \{G_1, G_2, \cdots, G_{H_{max}}\}$ as an arbitrary event distribution.

3.2.4 Capacity Model

Let C_h denote the allowable average transmission throughput (or *outgoing throughput*) of a node in hop layer h. Then the total transmission throughput of nodes in hop layer h will be N_hC_h. Here, the transmission throughput is composed of the traffic that a node produces and that it relays (or *incoming throughput*, denoted as C'_h). It is obvious that $(2i-1)C_i \leq (2j-1)C_j$ if $i > j$, as the total schedulable traffic produced by nodes in the outer layers cannot exceed the traffic that can be consumed by nodes in the inner layers. Many factors may constrain this inequality as a strict one. A typical example is that nodes in hop j may produce traffic themselves. Based on this inequality, we have $C_i \leq \frac{2j-1}{2i-1}C_j < C_j$ for any $i > j$.

Let W denote the fixed transmission capacity of the wireless channel, i.e., each node can transmit or receive data at W *bit/s* at the most. Now, $N_1C_1 \leq W$, i.e., the receiving capacity of the sink cannot exceed the receiving capacity of its wireless channel. Based on the assumption of a zero-overhead MAC protocol, we tighten this inequality as $C_1 = \frac{W}{N_1}$.

3.2.5 Deadline and Delay Model

The slack time of a time-constrained activity denotes the redundant time that is available for the completion of that activity before the expiration of the time constraint. For real-time data delivery in WSNs, the end-to-end *slack time* is the redundant time after subtracting the time necessary, e.g., for propagation, from the deadline. For example, if a packet has a deadline of 2000 ms, but the end-to-end necessary time (e.g., for propagation) is 1500 ms which is impossible to avoid, then the slack time is $2000 - 1500 = 500$ ms. Nodes along the propagation path may safely utilize this 500 ms of slack time for queuing, routing, transmission, and other operations on the packet without jeopardizing the packet deadline.

We focus on the packet waiting time that a packet may incur in the many packet queues as the major consumer for the slack time. This waiting time includes queuing delays and service delays such as time for routing and transmission. Let $D_{e2e,H}$ denote the end-to-end deadline of a packet generated in hop layer H, and τ denote the average per-hop time that is necessary for propagation including the sleep delay TP. Then $L_{e2e,H} = D_{e2e,H} - H\tau$ is the end-to-end slack time, and $L_{h,H}$ is the slack time spared for a node in hop layer h.

3.3 Performance Metrics

In this section we define the following metrics to estimate energy efficiency, tracking performance and the tradeoff between them in the simulation.

Energy efficiency

Since TPSS is a single target tracking scheme and each time the number of nodes involved into tracking is less, the energy consumption on different sensor nodes at different areas within the surveillance field may vary significantly. Given this inequity of the energy consuming load, the network lifetime — time until the first sensor node runs out of power — is a less useful metric for measuring energy efficiency. We define *extra energy* (or EE) as the total extra energy consumed for tracking a single target. The term "extra" here means that we count only the more energy consumed for tracking a target than the energy cost without any target observed during the same tracking period. As we focus on keeping tracking a target instead of data collection, EE does not include the energy consumed for propagating target information towards sink nodes.

Tracking performance

The tracking delay is one of the most important performance metrics for tracking. Usually the detection delay is defined as the delay between when the target enters the surveillance field or gets lost and when it is detected. In [1], the detection delay is further classified as an initial activation delay and a sentry detection delay. However, tracking can be considered as a process of continuously detecting a target, thereby the tracking delay should express the network-wide detection performance. We use *average detection delay* (or AD) for measuring the tracking delay, which is defined as follows. When a target leaves the sensing region of the currently active nodes and no other nodes detect it in time, the target will be lost. A detection delay is defined as the time interval from when a target is lost till it is detected again. AD is defined as the average of the detection delays along the target's traversing route. Before the target is detected for the first time, TPSS scheme is not started and all the nodes work in the random sleep pattern as we assume. Thus the initial detection delay, like the initial activation delay defined in [1], is out of the scope of TPSS's performance. We define a term, *tracking period*, as the time interval from when a surveillance system detects the target for the first time to when the target leaves the surveillance field. Then, AD is estimated only within the tracking period.

Tradeoff

Energy efficiency and tracking performance are two competing factors—e.g., solely optimizing energy efficiency will impair the tracking performance. If we pursue energy efficiency enhancement as outcomes with tracking performance loss as cost, a good tradeoff between them should get more outcomes with less cost. Since energy efficiency and tracking performance have totally different measurements, directly comparing the outcome and the cost with their absolute values is less helpful. Instead, a relative change can show a clearer comparison.

We define the *relative changes* (or RC) for EE and AD as follows respectively. In the equations, the subscript "ref" refers to related works as a reference for comparison.

$$\begin{cases} RC_{EE} & = & (EE_{ref} - EE_{TPSS})/EE_{ref} \\ RC_{AD} & = & (AD_{TPSS} - AD_{ref})/AD_{ref} \end{cases}$$

From the simulation, we will show that TPSS achieves more RC_{EE} than RC_{AD}.

3.4 Summary of Notations

For the convenience of discussion, we first summarize all the notations in Table 3.1.

Table 3.1: Notation of Parameters

Parameter	Definition (Default Value)	Parameter	Definition (Default Value)
Generic Parameters			
r	Sensing range (10 m)	TP	Toggling period (1 s)
R	Transmission range (60 m)	DC	Duty cycle (10%)
$\overrightarrow{X} = (X, \theta)$	A vector with the magnitude X and the direction θ	$\lVert\overrightarrow{X}\rVert$	Magnitude of the vector $\overrightarrow{X}$
ρ	Node density		
Event Observation			
EE	Extra energy consumption	AD	Average detection delay
RC	Relative change	$\{t_n \mid n \in N\}$	Sampling time points
$State(n)$	A target's motion state at t_n	(x_n, y_n)	Target position
$\overrightarrow{v_n} = (v_n, \theta_n)$	Average velocity vector during (t_{n-1}, t_n)	$\overrightarrow{a_n}$	Average acceleration vector during (t_{n-1}, t_n)
v_n	Scalar speed	θ_n	Moving direction
X'	Predicted value of X	$\overrightarrow{S_n}$	A target's displacement vector during $(t_n, t_n + TP))$
S_n	A random variable of the magnitude of a target's displacement in TP	Δ_n	A random variable of the angle by which a target may deviate from the direction of $\overrightarrow{S_n}'$
μ_{S_n}	Expected value of S_n	μ_{Δ_n}	Expected value of Δ_n
σ_{S_n}	Standard deviation of S_n	σ_{Δ_n}	Standard deviation of Δ_n $(\frac{\sqrt{6}}{12}\pi)$
Data Propagation			
H_{max}	Maximum hop number	h	Index of hop layers ($h \in [1, H_{max}]$)
N_h	Number of nodes in hop layer h	W	Fixed transmission capacity of the wireless channel
G_h	Average traffic in bit/s that each node in hop layer h may produce after events are observed	$\hat{G}$	An arbitrary event distribution
C_h	Allowable average outgoing throughput of a node in hop layer h	C'_h	Allowable average incoming throughput of a node in hop layer h
$D_{e2e,H}$	End-to-end deadline for a packet generated in hop layer H	τ	Average per-hop time that is necessary for propagation
$L_{e2e,H}$	End-to-end slack time for a packet generated in hop layer H	$L_{h,H}$	Slack time spared for a node in hop layer h within $L_{e2e,H}$

Chapter 4

Detection Delay in Event Observation Phase

4.1 Introduction

As mentioned in Chapter 1, we use target tracking as the example application. In many surveillance applications, detecting and/or tracking a target (e.g., a human, a vehicle) is one of the main objectives. Tracking usually has more stringent performance requirements than detection: detection succeeds as long as there exists a single sensor node that can detect the intruding target; however, tracking succeeds only if tracking performance constraints are satisfied. The tracking performance constraints are often application-specific. In the existing works, many metrics are used to measure the performance of detection or tracking, e.g., the detection delay [1], the coverage level [28], the probability of false alarm [61], and the tracking error [64]. Among these performance metrics, the detection probability and the average detection delay are two of the most important in many surveillance sensor networks [1, 9]. The most stringent tracking performance criterion is to track with 100% detection probability and zero average detection delay, i.e., at any time there is at least one node that can sense the existence of the target.

Sleep scheduling is one of the most commonly used approaches for reducing the energy consumption [33] for idle listening, which is a major source of energy wastage [21]. For prolonging the network lifetime, most sensor nodes keep in the sleep state with the radio off for most of the time, and only wake up periodically or on demand. However, forcing nodes to sleep will definitely result in target missing thereby impairing the tracking performance of the event observation phase. Fortunately communication between nodes is faster than the movement of real persons or vehicles. Thus on detecting a target, the node (i.e., *alarm node*) may proactively alarm its neighbors to have them prepared for the potentially passing by target, so as to enhance the tracking performance [16, 17]. On receiving an alarm message, each neighbor node (i.e., *candidate node*) may individually make its own decision on whether and when to sleep or wake up, and how long to sleep or wake up. In another word, each candidate node decides whether or not to schedule its sleep pattern and act as a *awakened node*. An extreme case is that each candidate node schedules their sleep pattern to keep active for a long term, which guarantees the tracking performance to be optimized. But sometimes it is unnecessary to force all the candidate nodes to be awakened. As long as the tracking performance constraints (e.g., the detection probability and the detection delay) are satisfied, the energy consumption could be minimized. Notice that the purpose of sleep scheduling here is different from [1] that aims at enhancing the detection and classification confidence.

For reducing the energy consumption, we could reduce the number of awakened nodes as well as curtail their wake-up time. One thing that we can leverage is the prediction for the potential motion of targets [36,37]. In fact, the accuracy of the prediction directly determines the number of awakened nodes: if a system knows the exact route of a target, only those nodes that cover the route need to be awakened; on the contrary if there is completely no prediction, an alarm node will have to wake up all the one-hop neighbors preparing for the approaching target (i.e., candidate nodes are all awakened nodes).

The past efforts about target prediction utilized various predicting approaches: 1) in [37], the authors only predict an exact position that the target is probably moving to; 2) in [11], the algorithm predicts a tracking contour, i.e., a small area that the target may pass with kinematics rules only; and 3) in [39], dynamics is used for prediction, and the system needs to classify what the target is and estimate external causes for the target motion.

Except reducing the number of awakened nodes and shortening their wake-up time, another approach can be used to enhance the energy efficiency for multiple target tracking. That is to leverage the redundant alarm messages of interfering targets. When an alarm node detects a target and plans to broadcast the prediction results for sleep scheduling, its neighbors (i.e. candidate nodes) may have already been scheduled by other alarm nodes for other approaching targets. Then the energy consumed for this alarm broadcast may be saved partially or completely. We will consider multiple targets as independent single ones in this proposal, and leave this problem to the post preliminary exam work.

In this proposal, we discuss the relationship among the detection probability, the detection delay, and the energy consumption. We present a *Target Prediction and Sleep Scheduling scheme* (or TPSS) to reduce the energy consumption as much as possible when simultaneously satisfying the tracking performance constraints. To achieve this objective, we utilize two approaches: 1) reduce the number of awakened nodes; and 2) schedule their sleep pattern to shorten the wake-up time. Both approaches are built upon a target prediction model based on both kinematics rules and probability theory. Figure 4.1 shows the foundation, approaches, the objective of TPSS scheme and the relationship among them.

- The prediction scheme predicts (or describes) the prospective change of a target's

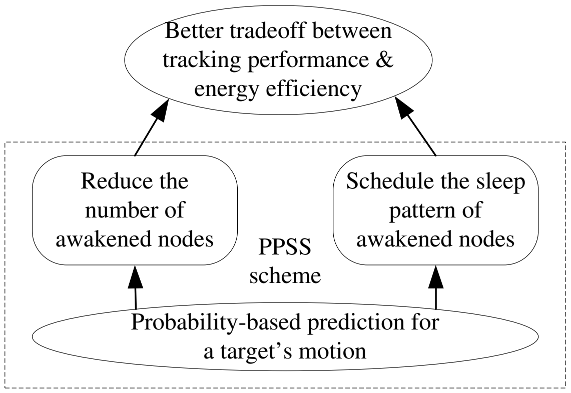
Better tradeoff between tracking performance & energy efficiency
Reduce the number of awakened nodes
Schedule the sleep pattern of awakened nodes
PPSS scheme
Probability-based prediction for a target's motion

Figure 4.1: Foundation, Approaches, and Objective of TPSS

velocity rather than its position. Since velocity is a physical vector, the prediction scheme establishes probabilistic models for both its scalar absolute magnitude (i.e. speed) and its direction. The normal distribution (or Gaussian distribution) is the major probabilistic model that we utilize for prediction.

- For reducing the number of awakened nodes, we introduce a concept of *awake region* and a mechanism for computing the scope of an awake region. An awake region is defined as the region that a target may traverse in a next short term, which should be covered probabilistically by active nodes. As some nodes in an awake region will be proactively awakened and keep active for some time, the number of these awakened nodes in the region should be as less as possible to enhance energy efficiency.

- For scheduling the sleep pattern of awakened nodes, we present a sleep scheduling mechanism to further enhance energy efficiency. This mechanism schedules the sleep patterns of awakened nodes individually according to their distance and direction away from the current motion state of the target. Therefore, the lasting time of awakened nodes in their active state could be reduced to as short as possible.

Finally the glossary of terms for the event observation phase are summarized in Table 4.1.

Table 4.1: Glossary of Terms

Parameter	Definition
Alarm node	The node that detects a target and broadcasts an alarm to wakes up its neighbors
Candidate node	A candidate node that may reschedule its sleep pattern
Awakened node	A candidate node that actually reschedules its sleep pattern
Awake region	An area that awakened nodes may locate in only

The rest of this chapter is organized as follows. Section 4.2 introduces the prediction scheme and the prediction results, i.e., probabilistic models on a target's future motion. Section 4.3 presents the two approaches for minimizing the energy consumption, including the proactive wake-up mechanism based on awake regions, the computation for an awake region's scope, and the selection of awakened nodes. Then we provide distributed algorithm descriptions for TPSS scheme that runs on each individual node in Section 4.4. In Section 4.5 we analyze the detection probability and the detection delay under TPSS scheme. Section 4.6 reports our evaluation results. In Section 4.7, we finally conclude our work for the event observation phase.

4.2 Target Motion Prediction

In the real world, a target's movement is subject to uncertainty, while at the same time it follows certain rules of physics. This apparent contradiction is because: 1) at each instant or during a short time period, there is no significant change on the rules of a target's motion, therefore the target will approximately follow kinematics rules; 2) however, a target's long term behavior is uncertain and hard to predict, e.g., a harsh brake or a sharp turn cannot be predicted completely with kinematics rules. In fact, even for a short term, it is also difficult to accurately predict a target's motion purely with a physics-based model. However, the prediction is absolutely helpful for optimizing the energy efficiency and tracking performance tradeoff. Thus, we consider a probabilistic model to handle as many possibilities of change of the actual target motion as possible.

At each time point t_n, the whole prediction process could be divided into three steps:

1) calculate the current speed v_n, direction θ_n and acceleration $\overrightarrow{a_n}$, based on $State(n-1)$ and the current position (x_n, y_n) that is assumed to be obtained by sensing or by calculating;

2) based on kinematics rules, predict the displacement $\overrightarrow{S_n}'$;

3) establish the probabilistic model including the target's scalar displacement S_n and deviation Δ_n.

Here t_{n+1} is decided by $t_n + TP$, i.e., we predict the potential movement of the target after the sleep delay that an alarm message experiences. Each time when time moves one step forward (e.g. from t_n to t_{n+1}), v'_{n+1} and θ'_{n+1} predicted in step 2 at time point t_n will be dismissed, and the actual values v_{n+1} and θ_{n+1} will be calculated in step 1 at time point t_{n+1}.

Notice that this prediction is based on a previously available observation. If this is the first time that the target is detected and there is no $State(n-1)$, the prediction can be skipped and delayed to the next time point.

4.2.1 Calculate the Current State

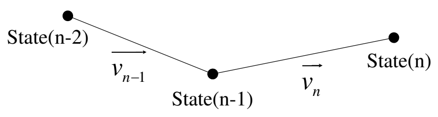

Figure 4.2: Target Movement States

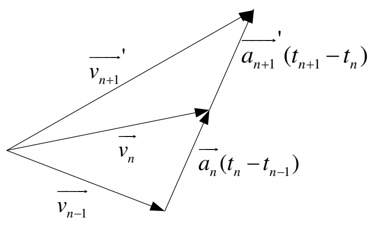
$\overrightarrow{v_{n+1}}'$
$\overrightarrow{a_{n+1}}'(t_{n+1}-t_n)$
$\overrightarrow{v_n}$
$\overrightarrow{a_n}(t_n-t_{n-1})$
$\overrightarrow{v_{n-1}}$

Figure 4.3: Prediction based on Kinematics Rules

First we compute the state vector $State(n)$ based on the previous state $State(n-1)$. Assume that the current time point is t_n, and the state at the previous time point is known as $State(n-1)$. Figure 4.2 shows the target motion at three continuous time points, and Figure 4.3 illustrates the change of the target velocity and its acceleration. As assumed, the target's current location (x_n, y_n) can be determined by sensing or calculating with existing algorithms. Then $\overrightarrow{v_n}$ and $\overrightarrow{a_n}$ can be computed as,

$$\begin{cases} v_n &= \frac{\sqrt{(y_n-y_{n-1})^2+(x_n-x_{n-1})^2}}{t_n-t_{n-1}} \\ \theta_n &= \begin{cases} \arctan\frac{y_n-y_{n-1}}{x_n-x_{n-1}} & , \quad x_n \neq x_{n-1} \\ 0 & , \quad x_n = x_{n-1} \end{cases} \\ \overrightarrow{a_n} &= \frac{\overrightarrow{v_n}-\overrightarrow{v_{n-1}}}{t_n-t_{n-1}} \end{cases} \tag{4.1}$$

Notice that we didn't substitute $t_n - t_{n-1}$ with TP, because the actual time point for sampling t_n depends on whether or not the target is physically detected. TP is only used for prediction.

4.2.2 Kinematics-based Prediction

Then we predict $\overrightarrow{S_n}'$ following kinematics rules. For simplifying the computation, we assume that the acceleration remains unchanged as $\overrightarrow{a_n}$ during (t_n, t_{n+1}) (because otherwise we will have to consider the target's "jerk", i.e. the rate of change of acceleration, which is rarely used). Then

$$\overrightarrow{S_n}' = \overrightarrow{v_n} \cdot TP + \frac{1}{2}\overrightarrow{a_n} \cdot TP^2 \tag{4.2}$$

4.2.3 Probability-based Prediction

Since the alarm message broadcast experiences a sleep delay of TP, the target may have already passed by some candidate nodes before they receive the alarm. For TPSS we setup a probabilistic model for the length of the target's displacement during the sleep delay, which can also show the impact of the target's scalar speed on proactive wake-up and sleep scheduling.

Suppose that the random variable S_n is Gaussian, i.e., $S_n \sim N(\mu_{S_n}, \sigma^2_{S_n})$. The mean is calculated as $\mu_{S_n} = \|\overrightarrow{S_n}'\| = \|\overrightarrow{v_n} \cdot TP + \frac{1}{2}\overrightarrow{a_n} \cdot TP^2\|$ when the acceleration remains unchanged. During the sleep delay TP, the scalar speed is likely to change between $\|\overrightarrow{v_n}\|$ (i.e. the original speed) and $\|\overrightarrow{v_n} + \overrightarrow{a_n} \cdot TP\|$ (i.e. the final speed when the acceleration remains unchanged). Thus S_n is likely to change between $S_A = \|\overrightarrow{v_n} \cdot TP\|$ and $S_B = \|\overrightarrow{v_n} \cdot TP + \overrightarrow{a_n} \cdot TP^2\|$. Obviously μ_{S_n} is in the interval (S_A, S_B) or (S_B, S_A) depending on the included angle between $\overrightarrow{v_n}$ and $\overrightarrow{a_n}$. Let the standard deviation of S_n be $\sigma_{S_n} = |\mu_{S_n} - S_A|$. In general $|\mu_{S_n} - S_B| \neq |\mu_{S_n} - S_A|$, but the difference is very small when the target is not making a sharp turn. Thus the probability of $S_n \in (S_A, S_B)$ or $S_n \in (S_B, S_A)$ is approximately 68% according to the "68-95-99.7 rule" of Gaussian distribution. In summary, $S_n \sim N(\mu_{S_n}, \sigma^2_{S_n})$ where

$$\begin{cases} \mu_{S_n} &= \|\overrightarrow{v_n} \cdot TP + \frac{1}{2}\overrightarrow{a_n} \cdot TP^2\| \\ \sigma^2_{S_n} &= (\|\overrightarrow{v_n} \cdot TP + \frac{1}{2}\overrightarrow{a_n} \cdot TP^2\| - \|\overrightarrow{v_n} \cdot TP\|)^2 \end{cases} \tag{4.3}$$

Figure 4.4 shows an example of $S_n \sim N(20, 25)$.

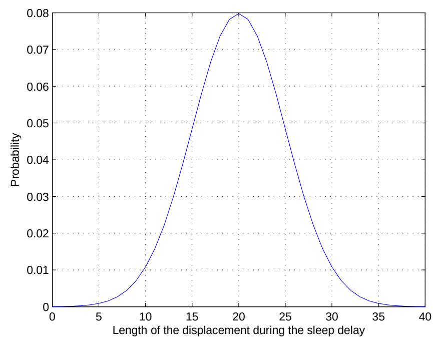

Figure 4.4: Example of Gaussian Distribution of S_n

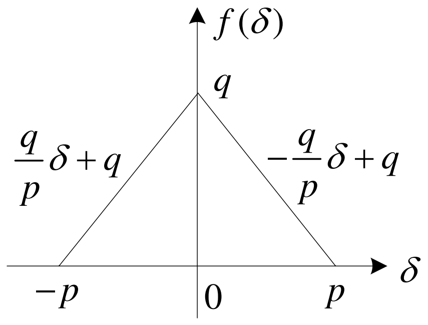

Figure 4.5: Probability Density Distribution of Δ_n

Next, we establish a linear model for Δ_n, i.e., the angle by which the target may deviate from the central direction of $\overrightarrow{S_n}'$. Assume that at time point t_{n+1}, the probabilities that the target turns left or right are completely equal. We configure the probability density function (or PDF) of Δ_n as Equation 4.4, where p, q are coefficients and $p \leq \pi$.

$$f_{\Delta_n}(\delta) = \begin{cases} -\frac{q}{p}\delta + q & , \quad (\delta \geq 0) \\ \frac{q}{p}\delta + q & , \quad (\delta < 0) \end{cases} \tag{4.4}$$

Figure 4.5 shows the probability density distribution of Δ_n. Since the total probability is equal to 1, we have $pq = 1$. Meanwhile the expected value of Δ_n is 0. Thus given a variance $\sigma^2_{\Delta_n}$,

$$\sigma^2_{\Delta_n} = E[\Delta_n^2] - E[\Delta_n]^2 = E[\Delta_n^2] = \int_{-p}^{p} \delta^2 f(\delta) d\delta$$

Then the coefficients p and q are determined by $\sigma^2_{\Delta_n}$ as,

$$\begin{cases} p = \sqrt{6}\sigma_{\Delta_n} \\ q = \frac{\sqrt{6}}{6\sigma_{\Delta_n}} \end{cases} \tag{4.5}$$

The variance $\sigma^2_{\Delta_n}$ can be configured by the application or dynamically computed regarding to the acceleration. Next we present an example of computing $\sigma^2_{\Delta_n}$ from a maximum deviation angle δ_{max}.

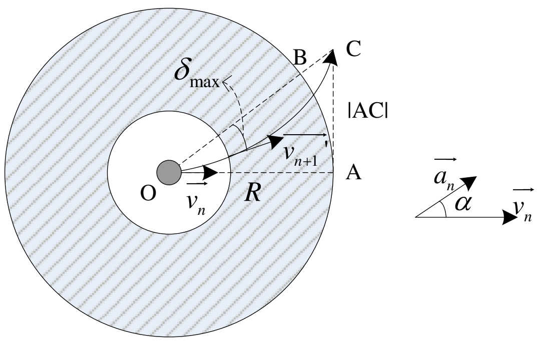

Figure 4.6: Probabilistic Model of Moving Direction

Figure 4.6 shows the computation of the maximum deviation angle, where the small solid circle represents the target's current position, and the large circle is the transmission range of the alarm node. Here we simplify the computation by approximating the target's position with the alarm node's position, as the distance of the target from the alarm node is less than a node's sensing range r, which is neglectable compared with the transmission range R (e.g., the sensing range is configured as 10 m based on empirical data in [1], while the outdoor transmission range provided in the datasheet of Mica2 platform [65] is 500 $ft \approx 152.4$ m).

If the acceleration remains unchanged, the time that a target needs to escape the current alarm node's transmission range R can be computed approximately with the following equation.

$$R = v_n t_{escape} + \frac{1}{2} a_n \cos \alpha t^2_{escape}$$

where α is the included angle between $\overrightarrow{a_n}$ and $\overrightarrow{v_n}$, i.e.,

$$\cos\alpha = \frac{\overrightarrow{a_n} \cdot \overrightarrow{v_n}}{a_n \cdot v_n}$$

Such an approximate computation reduces the calculation complexity significantly by substituting the solving process of a quadratic equation for that of a quartic equation used for precise solution.

Thus

$$t_{escape} = \frac{\sqrt{v_n^2 + 2Ra_n\cos\alpha} - v_n}{a_n\cos\alpha}$$

During this period of time, the target may move along the direction perpendicular to the current velocity for

$$|AC| = \frac{1}{2}a_n \sin\alpha t_{escape}^2 = \frac{\sqrt{1-\cos^2\alpha}}{a_n\cos^2\alpha}\left(v_n^2 + Ra_n\cos\alpha - v_n\sqrt{v_n^2 + 2Ra_n\cos\alpha}\right)$$

Then approximating the length of the arc $\widehat{AB}$ with $|AC|$, we have,

$$\delta_{max} = \frac{|AC|}{R} - \left|\theta'_{n+1} - \theta_n\right|$$

where $\frac{|AC|}{R}$ is the central angle corresponding to the target's deviation.

Let

$$\sigma_{\Delta_n} = \delta_{max} = \frac{\sqrt{1-\cos^2\alpha}}{Ra_n\cos^2\alpha}\left(v_n^2 + Ra_n\cos\alpha - v_n\sqrt{v_n^2 + 2Ra_n\cos\alpha}\right) - \left|\theta'_{n+1} - \theta_n\right| \qquad (4.6)$$

then the probability of $\Delta_n \in (-\delta_{max}, \delta_{max})$ is approximately 68%.

4.3 Energy Conservation

In this section we introduce the approaches for reducing the energy consumption and describe the distributed algorithms.

4.3.1 Reducing the Number of Awakened Nodes

Usually, a sensor node's transmission range R is far longer than its sensing range r. Thus when the nodes are densely deployed to guarantee the sensing coverage, a broadcast alarm message will reach all the neighbors within the transmission range. However, some of these neighbors can only detect the target with a relatively low probability, and some others may even never detect the target. Then the energy consumed for being active on these nodes will be wasted. A more effective approach is to determine a subset among all the neighbor nodes to reduce the number of awakened nodes.

During the sleep delay, the target may move away from the alarm node for a distance. Then it is unnecessary for nodes within this distance to wake up, since the target has already passed by. Meanwhile, all the nodes in an awake region must in the one-hop transmission range of the alarm node. Therefore, an awake region should be in a ring shape, i.e., the part between two concentric circles.

Beyond the effort that limits the awakened nodes within an awake region, the number of awakened nodes can be further reduced by choosing only some nodes in the awake region as awakened nodes. Based on our prediction on the target's moving directions, the probabilities that the target moves along various directions are different. Obviously the number of awakened nodes along a direction with a lower probability could be less than the number along a direction with a higher probability. By choosing an awakened node based on a probability related to the moving directions, awakened nodes can be reduced significantly.

The term awake region, as we use it, is similar to the concept of a cluster used in the network architecture literatures (e.g., [16, 32]) in that it encompasses some of a cluster's functions. However, unlike a cluster's head, neither an alarm node aggregates data from member nodes of the awake region, nor it imposes any control over members. An alarm node's responsibility here is just to broadcast an alarm message on detecting a target. An alarm message carries some target related information so that the neighbors (i.e. candidate nodes) could make decisions on whether or not to get prepared for the approaching target and how to prepare. In fact, an awake region is only a virtual concept. No functions are built upon this concept, except for the selection of awakened nodes. Only nodes within an awake region is likely to detect the target after the sleep delay, and only some of these nodes need to reschedule their sleep patterns preparing for the approaching target.

We first discuss the proactive wake-up mechanism based on awake regions, then introduce the two efforts for reducing the number of awakened nodes.

Proactive Wake-up with Awake Regions

First we present an awake region management mechanism as follows, which is different from cluster management. This mechanism is implemented in the three distributed procedures in the algorithm description.

- *Creation.* On detecting a target, a sensor node will check its own status to determine if it is an awakened node in an existing awake region. If yes, it justifies if the target is leaving the current awake region. If an awake region exists and the target is not going to move out of the current awake region, the node does nothing. Otherwise if no previous awake region exists or if the target is leaving the current awake region, the node will run an alarm node election algorithm, e.g. [66], to decide whether or not to assume an alarm node's responsibility. If this node is elected as the alarm node, it broadcasts an alarm message to all the candidate nodes. On receiving this alarm message, each candidate node individually decides if it is in the scope of this awake region and whether or not to schedule the sleep pattern. Finally, a new awake region comes into being when every awakened node schedules their sleep patterns specifically for the approaching target.

- *Dismissal.* As time progresses, the sleep patterns of awakened nodes will automatically recover back to the default pattern, thus the awake region will be dismissed automatically. There is no explicit dismissal mechanism needed.

The approach for electing an alarm node of [66] is as follows. Upon detection, each node broadcasts a DETECTION message to nodes nearby containing a time stamp recording when the detection is declared. Then it checks all the DETECTION messages received from nodes nearby within an interval, and compares the time stamps of other nodes with its own. Nodes that detect a neighbor node's time stamp is earlier than its own simply keep silent. In fact, a simpler approach also works well. Without any alarm election algorithm used, multiple alarm messages may be broadcast from multiple nodes that detect the same target. Then on receiving the first alarm message, a neighbor node may simply ignore the following ones sent by nodes that are within a $2r$ distance from the first alarm node, as these alarms may be considered as for the same target.

For convenience of discussion, we refer to a sensor node's two working modes as *default mode* and *tracking mode*. When the sleep pattern of a sensor node is not scheduled for a specific target, it works in the default mode and follows the default sleep pattern. When it detects a target or decides to reschedule the sleep pattern on receiving an alarm message, it enters the tracking mode. This rescheduled sleep pattern has an end time point, after which the node will return to the default mode automatically. In other words, the proactive wake-up depends on the alarm broadcasting, and whether to actually wake up proactively is decided by a candidate node itself.

An alarm message contains the following information that is used for a candidate node to make decision and compute the scope of an awake region:

- ID and the position of the alarm node (id_r, x_r, y_r);
- the state vector $State(n)$; and
- some of the prediction results, including $\overrightarrow{S_n}'$, μ_{S_n}, σ_{S_n}, and σ_{Δ_n}.

Constraint on the Awake Region Scope

Denote d the distance of an awakened node from the alarm node. Next we determine an awake region's scope by deciding the value scope of d.

As previously discussed, the target may move by S_n during the sleep delay TP. Here we make the same approximate assumption that the target's position is exactly the alarm node's position to simplify the computation. If we set $d \geq \mu_{S_n} - \sigma_{S_n}$, the probability that awakened nodes cannot cover the target after a sleep delay will be less than 16%. Moreover, it is obvious that $d \leq R$ because nodes outside of the alarm node's transmission range cannot be awakened. Therefore we determine the scope of an awake region as $\max\{\mu_{S_n} - \sigma_{S_n}, 0\} \leq d \leq R$. Thus the number of nodes in an awake region is $\rho\pi\big[R^2 - \max\{\mu_{S_n} - \sigma_{S_n}, 0\}^2\big]$, where ρ is the node density. Figure 4.6 also shows the scope of an awake region, which is filled with diagonals.

Awakened Nodes Selection in an Awake Region

So far the computation of an awake region's scope depends on the target's scalar speed only. Moreover, the decrement percentage of the number of awakened nodes is only $\frac{\max\{\mu_{S_n} - \sigma_{S_n}, 0\}^2}{R^2}$ (e.g., 6.25% when $R = 60$, $\mu_{S_n} = 20$, and $\sigma_{S_n} = 5$), which is not significant enough for enhancing energy efficiency.

As discussed previously, only some of the member nodes in an awake region need to be awakened. By taking into account the prediction results on moving directions, we can further reduce the number of awakened nodes in an awake region so as to save more energy than solely constraining the scope of an awake region.

Since the probability that a target moves along the direction of $\overrightarrow{S_n}'$ (i.e. $E[\Delta_n]$), denoted as θ, is the highest, we force all the nodes on this direction to be awakened. As Δ_n decreases on other directions, the number of awakened nodes on those directions can also be decreased. Define the probability that a candidate node on the direction $(\theta + \delta)$ reschedules its sleep pattern (i.e., becomes an awakened node) as

$$P_{ss}(\delta) = \frac{f_{\Delta_n}(\delta)}{f_{\Delta_n}(0)} = \begin{cases} -\frac{1}{p}\delta + 1 & , \quad (\delta \geq 0) \\ \frac{1}{p}\delta + 1 & , \quad (\delta < 0) \end{cases} \tag{4.7}$$

where "ss" means sleep scheduling.

Then the total number of awakened nodes in an awake region would be

$$
\begin{aligned}
N &= \int_{-\pi}^{\pi} P_{ss}(\delta) \cdot \frac{\rho\pi(R^2 - \max\{\mu_{S_n} - \sigma_{S_n}, 0\}^2)}{2\pi} d\delta \\
&= \rho(R^2 - \max\{\mu_{S_n} - \sigma_{S_n}, 0\}^2) \cdot \int_0^{\pi} \big(-\frac{1}{p}\delta + 1 \big) d\delta \\
&= \frac{\sqrt{6}}{2}\rho\sigma_{\Delta_n}(R^2 - \max\{\mu_{S_n} - \sigma_{S_n}, 0\}^2)
\end{aligned}
$$

As an example, the number of awakened nodes of TPSS is only approximately 19% of that of the Circle scheme when $R = 60$, $\mu_{S_n} = 20$, $\sigma_{S_n} = 5$, and $\sigma_{\Delta_n} = \frac{\pi}{6}$. In another word, the energy consumption of TPSS is only about 19% of that of the Circle scheme.

4.3.2 Sleep Scheduling for Awakened Nodes

After reducing the number of awakened nodes, energy efficiency can be enhanced further by scheduling the sleep patterns of awakened nodes, as not all the awakened nodes need to keep active all the time. We schedule the sleep patterns of awakened nodes by setting a start time and an end time of the active period. Out of this active period, awakened nodes do not have to keep active. Therefore the time that an awakened node has to keep active could be reduced compared with the Circle scheme.

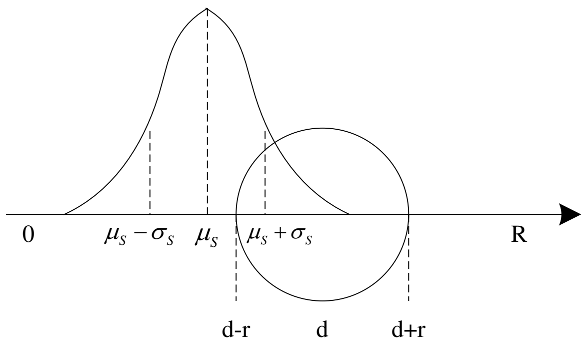

Figure 4.7: The Relationship between S_n and d

As previously stated, the distance of an awakened node to the alarm node is $\max\{\mu_{S_n} - \sigma_{S_n}, 0\} \leq d \leq R$. At the moment that an awakened node receives the alarm message (i.e. after the sleep delay, we denote this time point as $t_{alarmed}$), the relationship between the awakened node's position and the distribution of the target's displacement length during a sleep delay is shown in Figure 4.7. In the figure, 0 means the position of the alarm node and r is the sensing range of nodes. According to the relative positions of the awakened node and the target's expected position after the sleep delay, we make the sleep scheduling decisions as follows.

When $\mu_{S_n} \geq d - r$, the awakened node is required to wake up immediately (i.e. at $t_{alarmed}$) since it is expected that the target has probably entered its sensing range. When $\mu_{S_n} < d-r$, the awakened node is required to wake up at $t_{alarmed} + \frac{d-r-\mu_{S_n}}{TS}$, where we suppose $TS = \frac{\mu_{S_n}}{TP}$ to be the average speed in the awake region. In both cases, the awakened node needs to keep active until $t_{alarmed} + \frac{d+r-(\mu_{S_n}-\sigma_{S_n})}{TS}$, i.e., when the probability that the target has moved out of the sensing range of the awakened node is greater than 84%. For the convenience of discussion, we denote $t_{start} = t_{alarmed} + \frac{max\{d-r-\mu_{S_n},0\}}{TS}$ and $t_{end} = t_{alarmed} + \frac{d+r-(\mu_{S_n}-\sigma_{S_n})}{TS}$. In summary, the rescheduled active period of an awakened node is

$$\left[t_{start} = t_{alarmed} + \frac{max\{d-r-\mu_{S_n},0\}}{TS}, t_{end} = t_{alarmed} + \frac{d+r-(\mu_{S_n}-\sigma_{S_n})}{TS}\right] \tag{4.8}$$

And the time of keeping active is,

$$T_{active} = \frac{min\{2r+\sigma_{S_n}, d+r-(\mu_{S_n}-\sigma_{S_n})\}}{TS}$$

Each awakened node is required to suspend its active/sleep toggling period for keeping active in the tracking mode, and then resume duty cycling after recovering to the default mode. Thus other than the timer for periodic active/sleep toggling (or *default timer*), a new wake-up timer (or *tracking timer*) is needed for the proactive wake-up. Compared with algorithms that have no sleep scheduling, TPSS can save more energy by scheduling the sleep pattern instead of keeping nodes active all the time.

4.4 Algorithm Descriptions

Section 4.4 presented the rules for conserving the energy consumption, including reducing the number of awakened nodes, scheduling their sleep patterns and leveraging the redundant alarm messages of interfering targets. However for the actual implementation, all of these mechanisms have to be distributed on each sensor node. In this section, we present detailed algorithm descriptions for TPSS scheme in three procedures.

Procedure 1 is a handler for the event of detecting a target, which can be triggered by an interrupt that is raised on sensing something.

For the formation frequency of awake regions, the MCTA algorithm [11] uses a "refresh time" concept. Instead, we use the target's motion trend as the criterion: when the target moves close to the edge of the current awake region, a sensor node, who detects the target is leaving and is elected as the alarm node, broadcasts an alarm message to wake up neighbors and form a new awake region.

Procedure 2 describes a sensor node's actions upon receiving an alarm message. This procedure can also be implemented as an interrupt handler.

Algorithm 1: *OnDetectingEvent()* — Triggered when detecting an event

```
1  if (I am in tracking mode) then
2      if (The target is NOT leaving the current awake region) then
3          return;
4  (Optional:) Run an alarm election algorithm;
5  if (I am selected as the alarm node) then
6      Calculate $\overrightarrow{v_n}$ and $\overrightarrow{a_n}$ with Equation 4.1;
7      Predict $\overrightarrow{S_n}'$ with Equation 4.2;
8      Compute $\mu_{S_n}, \sigma_{S_n}, \sigma_{\Delta_n}$ with Equation 4.3 and 4.6;
9      Broadcast $id_r, x_r, y_r, State(n), \overrightarrow{S_n}', \mu_{S_n}, \sigma_{S_n}, \sigma_{\Delta_n}$);
10 return;
```

Algorithm 2: *OnAlarmMsg()*—Triggered when receiving an alarm message

```
1  Compute $d = \sqrt{(x_r - x_o)^2 + (y_r - y_o)^2}$; // $(x_r, y_r)$ is the alarm node's position and $(x_o, y_o)$ is my position
2  if ($d < \mu_{S_n} - \sigma_{S_n}$) then
3      return;                          // constrain the scope of an awake region
4  Compute $\delta$ with $(x_r, y_r)$, $(x_o, y_o)$ and $\overrightarrow{S_n}'$;
5  Generate a random number $random = [0, 1]$;
6  if ($random > P_{ss}(\delta)$) then
7      return;                          // select awakened nodes
8  Compute $t_{start}$ and $t_{end}$ with Equation 4.8;
9  SetTrackingTimer($t_{start}$);        // set the tracking timer for the start of the scheduled sleep pattern
10 return;
```

In step 2 of Procedure 2, the node determines whether or not it is in the scope of the awake region. In step 6, the node decides whether to be an awakened node or not. Finally in step 10, the tracking timer is set so that the node can wake up at the scheduled time point.

Procedure 3 describes the tracking timer processing procedure, which controls the scheduled wake-up/sleep and the mode switch.

Algorithm 3: *OnTrackingTimer()* — Triggered when the tracking timer is out

```
1 if (mode == "default") then
2     mode = "tracking";
3     SuspendDefaultTimer();
4     SetTrackingTimer($t_{end}$);     // set the tracking timer for the end of the scheduled sleep pattern
5 else
6     mode = "default";
7     ResumeDefaultTimer();
8 return;
```

The computation workload of TPSS scheme is mainly located in step 6 and 7 of Procedure 1, i.e., the calculation for $\overrightarrow{v_n}$, $\overrightarrow{a_n}$, and the prediction for $\overrightarrow{S_n}'$, $\mu_{S_n}, \sigma_{S_n}, \mu_{\Delta_n}, \sigma_{\Delta_n}$. Thus the computation workload is aggregated on the alarm node, which is more energy effective than distributed algorithms.

Most of the computations of the three procedures can be completed in the transport layer. The only support that TPSS needs from the link layer (usually a MAC protocol) is an interface for controlling the tracking timer, and suspending/resuming the default active/sleep toggling period.

4.5 Analysis

Based on the target prediction and the two approaches for saving energy for single target tracking, we analyze the detection delay, the tracking probability and the energy consumption of the event observation phase. In [67] and [1], the authors discuss the tracking probability and the detection delay by leveraging two general results from theory of probability. Here we utilize a similar computation process, but add the impact of our target prediction and sleep scheduling scheme. The process of the computation is as follows.

First, we calculate the probability that a single node detects the target, denoted as $P_{single}(Y|\delta = \hat{\delta})$, where Y represents the event of successful detection, and $\hat{\delta}$ is a specific moving direction of the target. This probability is dependent on the relative position of the node to the target's potential route. Thus for all the possibilities where a node may detect the target, this probability needs to be normalized. We integrate it over the area where the node may locate, and compute a normalized probability for single node detection, denoted as $P^1(Y|\delta = \hat{\delta})$. Here the superscript 1 means single node detection.

Then we calculate $P(Y|\delta = \hat{\delta})$ for the moving direction $\hat{\delta}$, i.e., the probability that at least one node around $\hat{\delta}$ detects the target before it moves away for a distance L. And we normalize it as $P(Y)$ over all the possible directions.

Finally we compute $E[T_{detection}|\delta = \hat{\delta}]$ for the moving direction $\hat{\delta}$, i.e., the expected detection delay on $\hat{\delta}$. And we normalize it as $E[T_{detection}]$ (or $D_{detection}$) over all the possible directions.

These calculation operations are based on two general results from theory of probability [67]. The first result is, if the probability of an event A occurring in a single experiment is p, and if the number of experiments conforms to a Poisson Distribution with parameter λ, the probability of event A occurring at least once in the series of experiments is $P = 1 - e^{-p\lambda}$. The second result is, the expected value of a nonnegative random variable is $E[X] = \int_0^\infty [1 - P(X \leq x)]dx$.

In this section, we first introduce the definition of a detection area and its division, then present the computation step by step.

4.5.1 Detection Area

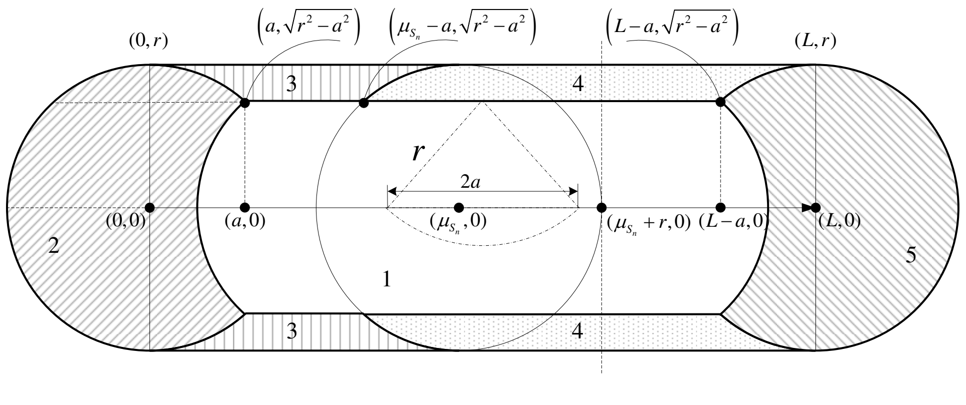
$(0,r)$
$\left(a,\sqrt{r^2-a^2}\right)$
$\left(\mu_{S_n}-a,\sqrt{r^2-a^2}\right)$
$\left(L-a,\sqrt{r^2-a^2}\right)$
(L,r)
3
4
r
$2a$
$(0,0)$
$(a,0)$
$(\mu_{S_n},0)$
$(\mu_{S_n}+r,0)$
$(L-a,0)$
$(L,0)$
2
1
5
3
4

Figure 4.8: Detection Probability and Detection Delay

Suppose that a target is at $(0,0)$ at time point t_n, and moves for a distance L along the direction $\hat{\delta}$ until it arrives at $(L,0)$. As shown in Figure 4.8, all the nodes that have a chance to detect the target locate in the rectangle or two semi-circles. Since tracking is a continuous detecting process, a new detection process may be started after a target is already inside the surveillance field. Therefore nodes in the semi-circle on the left side may also detect the target.

For the simplification of discussion, we let $Area_{total}$ denote this detection area where nodes that may detect the target locate. We divide $Area_{total}$ into five regions, labeled with 1 to 5 and filled with different patterns (except area 1 that is not filled) in the figure. Let $Area_i$ denote the i^{th} area. At the same time, $Area_{total}$ and $Area_i$ are also used to represent the size of these areas.

In average, the point $(\mu_{S_n}, 0)$ is the position of the target after a sleep delay TP. Therefore before the target arrives at $(\mu_{S_n}, 0)$, all the nodes work in the default mode. Only after that, awakened nodes can enter the tracking mode.

Suppose that the target's route overlaps with a node's sensing area by l. Then the probability that this node detects the target is $DC + \frac{l}{TP \cdot TS}$.

We define a parameter a as in

$$DC + \frac{2a}{TP \cdot TS} = 1$$

where $TS = \frac{\mu_{S_n}}{TP}$. Thus

$$a = \frac{1}{2}(1 - DC)\mu_{S_n}$$

Thus when $l \geq 2a$, the node must be able to detect the target, i.e., the detection probability will be 1. Bounded by a, it is obvious that all the nodes in $Area_1$ hold a detection probability of 1. The other four areas are divided based on two criteria: 1) if it is possible that a node enters its tracking mode when the target passes its sensing area; and 2) whether or not the target's route fully overlaps with the sensing area of a node. For all the nodes that may enter the tracking mode when the target passes by, the detection probability should be expressed in two parts, i.e., for the default mode and for the tracking mode. For those nodes that only partially overlap with the target's route, the probability is relatively lower than those nodes that fully overlap. Regarding to these two criteria, the other four areas are divided as in Table 4.2.

Table 4.2: Division of Areas

Areas	2	3	4	5
Criterion 1	No	No	Yes	Yes
Criterion 2	No	Yes	Yes	No

4.5.2 Single Node Detection Probability

For $Area_2$ and $Area_3$, nodes always work in the default mode when they detect the target. Thus the detection probability is $DC + \frac{l}{TP \cdot TS}$. However for nodes in $Area_4$ or $Area_5$, they may work in either the default mode or the tracking mode, according to the scheduling probability $P_{ss}(\delta)$ defined in Equation 4.7. If they work in the default mode, the detection probability is also equal to $DC + \frac{l}{TP \cdot TS}$. If they work in the tracking mode, i.e., selected to be awakened nodes, the detection probability is equal to 1. This is decided by the sleep scheduling scheme discussed in Section 4.3.2, i.e., nodes will wake up when the target passes by. Actually the detection probability may not be strictly 1, as the target may change the movement status abruptly. But this is very rare for the real moving persons or vehicles. We suppose this probability exactly as 1 to simplify the computation. Thus for nodes in $Area_4$ or $Area_5$, the detection probability can be expressed as,

$$(1 - P_{ss}(\delta))\big(DC + \frac{l}{TP \cdot TS}\big) + P_{ss}(\delta) \cdot 1$$

Here another simplification is to utilize $P_{ss}(\delta)$ for all the nodes in $Area_4$ and $Area_5$. We make this simplification approach with two reasons. First, $R >> r$ and L is close to R, thus $L >> r$. In another word, the area shown in Figure 4.8 is a sector-like shape in the alarm node's transmission range, with a very small central angle as shown in Figure 4.9. Therefore the difference between δ of different nodes is small. Second, all the areas are symmetric about the x-axis, and all the probabilities to be integrated are even functions. Thus the difference introduced by $P_{ss}(\delta)$ can be counteracted to the most extent.

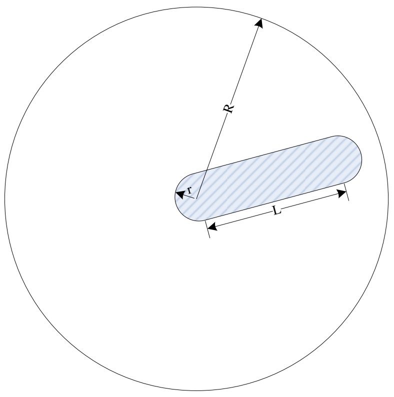

Figure 4.9: Detection Area in the Transmission Range

For a node (x, y) in $Area_3$ or $Area_4$, the target's route fully overlaps with their sensing area. Thus $l = 2\sqrt{r^2 - y^2}$. For a node (x, y) in $Area_2$ or $Area_5$, the target's route partially overlaps with their sensing area. Thus $l = x + \sqrt{r^2 - y^2}$ or $l = L - x + \sqrt{r^2 - y^2}$ respectively.

In summary, the single node detection probability and the area of all the five areas are listed in Table 4.3.

Here $\Phi(r, a) = \pi r^2 - 2r^2 \arccos \frac{a}{r} + 2a\sqrt{r^2 - a^2}$.

Next we integrate and normalize $P_{single}(Y|\delta = \hat{\delta})$ over all the five areas.

Table 4.3: Single Node Detection Probability and Size of Areas

$Area_i$	$P_{single}(Y\|\delta=\hat{\delta})$	Size
1	1	$2r^2 \arccos\frac{a}{r} + 2(L-3a)\sqrt{r^2-a^2}$
2	$DC + \frac{x+\sqrt{r^2-y^2}}{TP\cdot TS}$	$\Phi(r,a)$
3	$DC + \frac{2\sqrt{r^2-y^2}}{TP\cdot TS}$	$2\mu_{S_n}(r-\sqrt{r^2-a^2}) - \Phi(r,a) + 4a\sqrt{r^2-a^2}$
4	$(1-P_{ss}(\delta))\big(DC + \frac{2\sqrt{r^2-y^2}}{TP\cdot TS}\big) + P_{ss}(\delta)$	$2(L-\mu_{S_n})(r-\sqrt{r^2-a^2})$
5	$(1-P_{ss}(\delta))\big(DC + \frac{L-x+\sqrt{r^2-y^2}}{TP\cdot TS}\big) + P_{ss}(\delta)$	$\Phi(r,a)$

$$\begin{aligned}
P^1(Y|\delta=\hat{\delta})\cdot Area_{total} &= \int_{Area_{total}} P_{single}(Y|\delta=\hat{\delta})ds \\
&= \sum_{i=1}^{5}\int_{Area_i} P_{single}(Y|\delta=\hat{\delta})ds \\
&= \int_{Area_1} 1ds + \int_{Area_2}\big(DC + \frac{x+\sqrt{r^2-y^2}}{TP\cdot TS}\big)ds \\
&\quad + \int_{Area_3}\big(DC + \frac{2\sqrt{r^2-y^2}}{TP\cdot TS}\big)ds \\
&\quad + \int_{Area_4}\big[(1-P_{ss}(\delta))\big(DC + \frac{2\sqrt{r^2-y^2}}{TP\cdot TS}\big) + P_{ss}(\delta)\big]ds \\
&\quad + \int_{Area_5}\big[(1-P_{ss}(\delta))\big(DC + \frac{L-x+\sqrt{r^2-y^2}}{TP\cdot TS}\big) + P_{ss}(\delta)\big]ds \\
&= P^1_{random}(Y|\delta=\hat{\delta})\cdot Area_{total} + P^1_{plus}(Y|\delta=\hat{\delta})\cdot Area_{total}
\end{aligned}$$

Here $P^1_{random}(Y|\delta=\hat{\delta})$ is the normalized probability when all the nodes work only in the default mode without sleep scheduling, and $P^1_{plus}(Y|\delta=\hat{\delta})$ represents the enhancement on the probability introduced by TPSS. In fact, the normalized probability when all the nodes work only in the default mode without sleep scheduling $P^1_{random}(Y|\delta=\hat{\delta})$ is independent of δ. Thus in the following discussion we abbreviate it as P^1_{random}. We have,

$$P^1_{random}\cdot Area_{total} = DC\cdot Area_{total} + \frac{L\cdot\Phi(r,a) - 2a\big[\Phi(r,a) - \pi r^2\big]}{TP\cdot TS}$$

and

$$P^1_{plus}(Y|\delta=\hat{\delta})\cdot Area_{total} = \frac{P_{ss}(\delta)}{TP\cdot TS}\{(L-\mu_{S_n})\big[4ar - \Phi(r,a)\big] + 2a\cdot\Phi(r,a) - \frac{8}{3}\Gamma(r,a)\}$$

where $\Gamma(r,a) = r^3 - (\sqrt{r^2-a^2})^3$. Let

$$A = (2-DC)\Phi(r,a) - \frac{8}{3TP\cdot TS}\Gamma(r,a) - 4ar$$

and

$$B = \frac{L}{TP\cdot TS}\big[4ar - \Phi(r,a)\big] + A$$

for concise expression, we have $P^1_{plus}(Y|\delta=\hat{\delta})\cdot Area_{total} = B\cdot P_{ss}(\delta)$.

4.5.3 Detection Probability

As the node density is D_s, we suppose that the number of nodes n in $Area_{total}$ conforms to Poisson distribution with the mean $\lambda = D_s\cdot Area_{total}$. According to the result from theory of probability, we have,

$$\begin{aligned}P(Y|\delta=\hat{\delta}) &= 1 - e^{-\lambda P^1(Y|\delta=\hat{\delta})}\\ &= 1 - e^{-D_s\cdot P^1_{random}\cdot Area_{total}}\cdot e^{-D_s\cdot P^1_{plus}(Y|\delta=\hat{\delta})\cdot Area_{total}}\end{aligned}$$

Integrating and normalizing it over $(-\pi,\pi]$, we can calculate $P(Y)$ as,

$$\begin{aligned}P(Y) &= \int_{-\pi}^{\pi} P(Y|\delta=\hat{\delta})f_T(\hat{\delta})d\hat{\delta}\\ &= \int_{-\pi}^{\pi}\big[1 - e^{-D_s\cdot P^1_{random}\cdot Area_{total}}\cdot e^{-D_s\cdot P^1_{plus}(Y|\delta=\hat{\delta})\cdot Area_{total}}\big]f(\hat{\delta})d\hat{\delta}\\ &= 1 - e^{-D_s\cdot P^1_{random}\cdot Area_{total}}\cdot\int_{-\pi}^{\pi} e^{-D_s\cdot B\cdot P_{ss}(\hat{\delta})}f(\hat{\delta})d\hat{\delta}\end{aligned}$$

Here $f_T(\hat{\delta})$ is the probability density function of the target's actual turning potential. When we also use the prediction model, i.e., Equation 4.4, to describe it, $P(Y)$ can be calculated as,

$$\begin{aligned}P(Y) &= 1 - e^{-D_s\cdot P^1_{random}\cdot Area_{total}}\cdot 2\int_0^{\pi} e^{-D_s\cdot B\cdot(-\frac{1}{p}\hat{\delta}+1)}(-\frac{q}{p}\hat{\delta}+q)d\hat{\delta}\\ &= 1 - \frac{2}{D_sB}\cdot e^{-D_s\cdot(P^1_{random}\cdot Area_{total}+B)}\big[(\frac{1}{D_sB}+1-\pi q)e^{\frac{\pi D_sB}{p}} - (\frac{1}{D_sB}+1)\big]\end{aligned} \tag{4.9}$$

Here p and q are both coefficients for our linear prediction model.

4.5.4 Detection Delay

According to the second result from theory of probability, we have

$$E[T_{detection}|\delta = \hat{\delta}] = \int_0^{\infty} [1 - P(T_{detection} \le t|\delta = \hat{\delta})]dt$$

And letting $L = TS \cdot t$, $P(T_{detection} \le t|\delta = \hat{\delta}) = P(T_{detection} \cdot TS \le L|\delta = \hat{\delta})$ is the probability that a target is detected before it enters for a distance L, i.e., $P(Y|\delta = \hat{\delta})$. Thus

$$\begin{aligned} E[T_{detection}|\delta = \hat{\delta}] &= \int_0^{\infty} e^{-D_s \cdot P^1_{random} \cdot Area_{total}} \cdot e^{-D_s \cdot P^1_{plus}(Y|\delta=\hat{\delta}) \cdot Area_{total}} dt \\ &= exp\big(- D_s\big\{\pi r^2 - (1 - DC)\Phi(r, a) + P_{ss}(\delta) \cdot A\big\}\big) \\ &\quad \cdot \int_0^{\infty} e^{-D_s \cdot TS \cdot \big\{2r \cdot DC + \frac{1}{TP \cdot TS}\big[\Phi(r,a) + P_{ss}(\delta)[4ar - \Phi(r,a)]\big]\big\} t} dt \end{aligned}$$

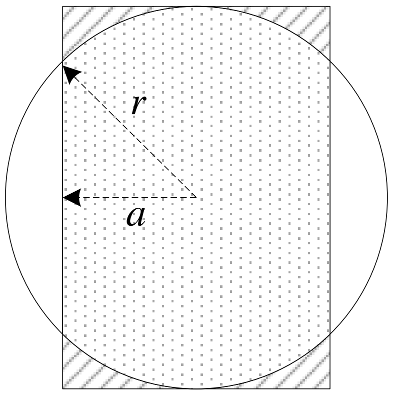

Figure 4.10: Simplify $E[T_{detection}|\delta = \hat{\delta}]$

We simplify this equation by removing $P_{ss}(\delta)$ from the exponential inside the integral. Let us check the meaning of $\Phi(r, a) + P_{ss}(\delta)[4ar - \Phi(r, a)]$. In Figure 4.10, it is easy to prove that $\Phi(r, a)$ is exactly the size of the shape filled with dots. Then $[4ar - \Phi(r, a)]$ is the size of four small areas filled with diagonals. Compared with the dotted shape, the size of small areas filled with diagonals can be ignored, especially when multiplied with a real number $P_{ss}(\delta)$ that is less than 1. Therefore, $\Phi(r, a) + P_{ss}(\delta)[4ar - \Phi(r, a)] \approx \Phi(r, a)$, and

$$E[T_{detection}|\delta = \hat{\delta}] \approx \frac{exp\big(- D_s\big\{\pi r^2 - (1 - DC)\Phi(r, a) + P_{ss}(\delta) \cdot A\big\}\big)}{D_s \cdot TS \cdot \big[2r \cdot DC + \frac{\Phi(r,a)}{TP \cdot TS}\big]}$$

Integrating and normalizing it over $(-\pi, \pi]$, we can calculate $E[T_{detection}]$ as,

$$\begin{aligned} D_{detection} &= E[T_{detection}] = \int_{-\pi}^{\pi} E[T_{detection}|\delta = \hat{\delta}] f_T(\hat{\delta}) d\hat{\delta} \\ &= \frac{e^{-D_s\left[\pi r^2 - (1-DC)\Phi(r,a)\right]}}{D_s \cdot TS \cdot \left[2r \cdot DC + \frac{\Phi(r,a)}{TP \cdot TS}\right]} \cdot \int_{-\pi}^{\pi} e^{-D_s \cdot A \cdot P_{ss}(\hat{\delta})} f_T(\hat{\delta}) d\hat{\delta} \end{aligned}$$

Similarly when we also use the prediction model, i.e., Equation 4.4, to describe it, $D_{detection}$ can be calculated as,

$$\begin{aligned} D_{detection} &= \frac{e^{-D_s\left[\pi r^2 - (1-DC)\Phi(r,a)\right]}}{D_s \cdot TS \cdot \left[2r \cdot DC + \frac{\Phi(r,a)}{TP \cdot TS}\right]} \cdot 2\int_{0}^{\pi} e^{-D_s \cdot A \cdot (-\frac{1}{p}\hat{\delta}+1)} (-\frac{q}{p}\hat{\delta} + q) d\hat{\delta} \\ &= \frac{2e^{-D_s\left[\pi r^2 - (1-DC)\Phi(r,a) + A\right]}}{D_s^2 A \cdot TS \cdot \left[2r \cdot DC + \frac{\Phi(r,a)}{TP \cdot TS}\right]} \cdot \left[(\frac{1}{D_s A} + 1 - \pi q) e^{\frac{\pi D_s A}{p}} - (\frac{1}{D_s A} + 1)\right] \end{aligned} \tag{4.10}$$

Here p and q are both coefficients for the linear prediction model, and

$$A = (2 - DC)\Phi(r,a) - \frac{8}{3TP \cdot TS}\Gamma(r,a) - 4ar$$

4.6 Performance Evaluation

We evaluated TPSS scheme in a simulation environment programmed in C++, and compared it against the Circle scheme and the MCTA algorithm. This section reports our evaluation results measured in terms of the performance metrics that we defined in Section 3.3.

4.6.1 Simulation Environment

Parameters of our simulation environment included the following:

- *Node deployment.* $500 - 2,000$ nodes were randomly deployed in a $200m \times 200m$ area, i.e., the node density is $0.5 - 4.0$ $nodes/100m^2$.
- *Target speed.* Fifteen target speed values in an arithmetic progression were used ($\{2, 4, \ldots, 30\}$ m/s).
- *Target motion.* We used a random curvilinear motion model to describe the actual target motion.
- *Algorithms.* We simulated four schemes including no sleep scheduling (or NOSS), Circle, MCTA, and TPSS.

Table 4.4: Energy Consumption Rates

Status	Energy consumption rate (unit)
Active (P_{active})	9.6 (mJ/s)
Transmit (P_{send})	720 (nJ/bit), 5.76 ($mJ/Byte$)
Receive (P_{rcv})	110 (nJ/bit), 0.88 ($mJ/Byte$)
Sleep (P_{sleep})	0.33 (mJ/s)

We estimated the performance metrics against the node density and the target speed. In the simulation, we configured the node density as 1.25 $node/100m^2$ when retrieving data regarding to various target speeds, and the target speed as 20 m/s when retrieving data regarding to various node densities. For each value of the node density (or the target speed), we simulated 10 deployments (or the route samples), and repeated each deployment (or route sample) 10 times. The system configuration parameters used in the simulation followed the default values given in Table 3.1 and Table 4.4.

The energy consumption data in Table 4.4 come from the actual Mica2 platform [68, 69]. We did not list the energy consumption rate for instruction execution, for it is difficult to measure in the simulation. As a compensation, we increased the energy consumption rate for TPSS's active state by 20%.

The random curvilinear motion model that we used in the experiments was completely different from the model that was established by our prediction in Section 4.2 (because otherwise everything would be perfect). Instead, we leverage some common sense based rules to imitate the actual target motion. Assume that the target moves in a random walk manner starting from a base speed value (e.g. 15 m/s), i.e., at each fixed time interval (e.g. 200 ms), the target speed may increase/decrease by 1 m/s or keep unchanged with a probability of $\frac{1}{3}$ respectively. For the moving direction, we assume that the target may turn with a random angle in $[0, \alpha_{max}]$ at each fixed time interval, where α_{max} decreases as the speed v increases. This matches the actual case because a target, especially a vehicle, may probably turn over if it abruptly changes the direction when moving at a high speed. We describe this relation between v and α_{max} with a linear mapping $\alpha_{max} = av + b$, where a and b are linear mapping constants. Now, two (v, α_{max}) value pairs will fully determine the linear mapping. $\alpha_{max} = 25^o$ when $v = 9$ m/s and $\alpha_{max} = 5^o$ when $v = 30$ m/s were the example value pairs used in our simulation. Figure 4.11 shows an example moving route when a target traverses a tracking field of $200m \times 200m$ with a base speed $v = 20$ m/s.

No sleep scheduling scheme and the Circle scheme are two extreme schemes. When there is completely no sleep scheduling, all the sensor nodes sleep and wake up randomly. The energy consumption is the least, but so is the tracking performance. With the Circle scheme, all the neighbor nodes within one hop range of the alarm node are awakened, and all the awakened nodes keep active all the time until when the target is expected to leave the alarm node's transmission range. Then before awakened nodes recover back to the default sleep

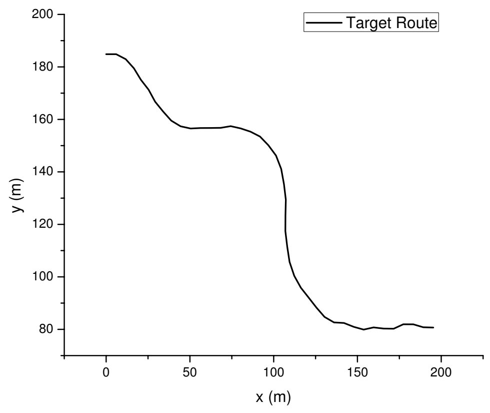

Figure 4.11: A Typical Target Route

pattern or the target leaves the coverage area of awakened nodes, tracking performance is guaranteed, i.e., 100% covered with zero tracking delay. However, energy efficiency of the Circle scheme is the worst, which is same during its active period as the case that keeps all the nodes active all the time. Therefore, NOSS scheme and Circle scheme serve as the upper bound and the lower bound (may be reversed) on performance metrics for other schemes.

4.6.2 Experimental Results

Since our objective is to achieve more RC_{EE} than RC_{AD}, we present the comparison results among the four schemes on EE, AD and relative changes one by one.

Extra energy (EE)

We measure the extra energy consumption EE in the unit J to determine how much more energy is needed for tracking a single target than detecting nothing. Figure 4.12 and Figure 4.13 show the EEs of four schemes at various node densities and various target speeds respectively. In general, the energy consumption for tracking increases as the node density increases, or as the target speed decreases. The former is easy to understand, as denser deployment means more nodes are awakened. The reason for the latter is that a slower target will stay longer in the alarm node's transmission range, thereby require awakened nodes to keep active for a longer time. In both cases, the extra energy for NOSS is the lower bound with the value 0, which is actually the basis of the definition of EE. As the scheme with the simplest sleep scheduling effort, Circle provides an upper bound for MCTA and TPSS in terms of the energy. We observe that TPSS consumes less energy than MCTA.

Something interesting is that, if we check the number of awakened nodes, the curves of MCTA and TPSS show reverse positions. Figure 4.14 and Figure 4.15 illustrate the number

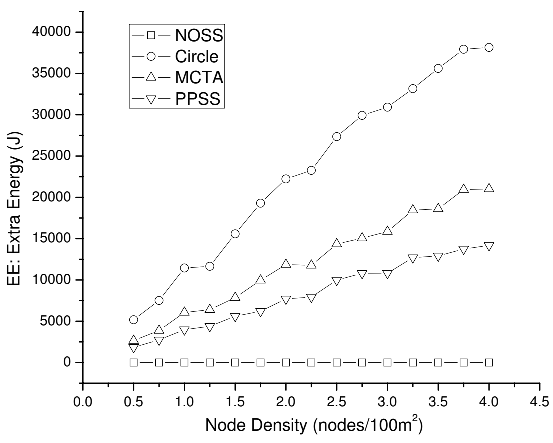

Figure 4.12: Extra Energy vs. Node Density

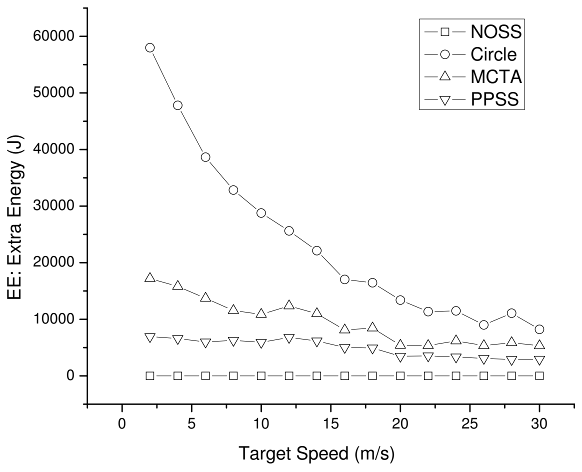

Figure 4.13: Extra Energy vs. Target Speed

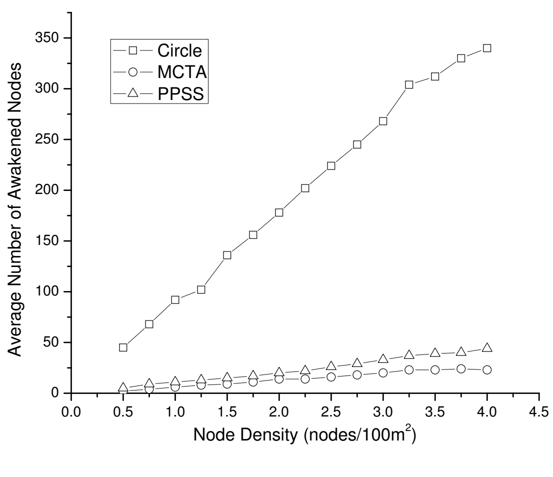

Figure 4.14: Number of Awakened Nodes vs. Node Density

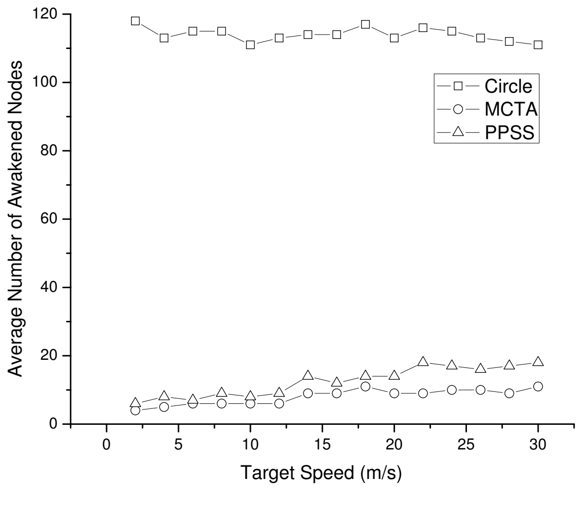

Figure 4.15: Number of Awakened Nodes vs. Target Speed

of awakened nodes (averaged for each proactive wake-up), in which the number of awakened nodes of TPSS is greater than that of MCTA.

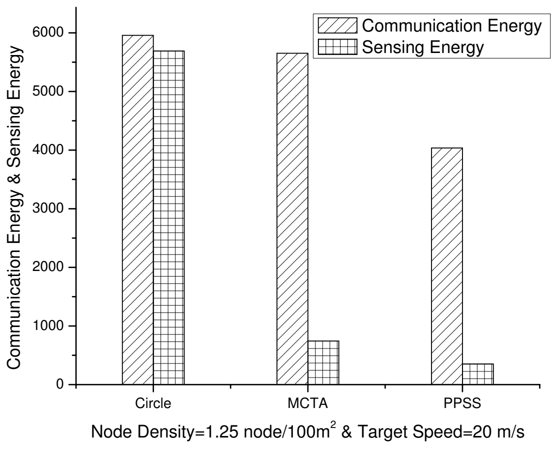

Figure 4.16: Sensing Energy and Communication Energy when Node Density=1.25 $node/100m^2$ and Target Speed=20 m/s

This seeming contradiction can be explained by comparing energy for sensing and energy for communication separately. Note that energy is consumed primarily for sensing and communication. Sensing energy includes P_{active} and P_{sleep}, while communication energy includes P_{send} and P_{rcv}. Figure 4.16 shows the comparison among the three schemes (except Circle) for sensing energy and communication energy separately. We observe that sensing energy and communication energy of TPSS are both less than MCTA. On one hand, TPSS consumes less sensing energy because of the sleep scheduling effort that shortens the active time as much as possible. On the other hand, TPSS consumes less communication energy because the frequency of proactive wake-up of TPSS is lower than that of MCTA. TPSS awakens neighbor nodes up to one transmission range away from the alarm node, and executes the next wake-up only when the target is about to leave the alarm node's transmission range. However, the interval between two adjacent wake-up actions of MCTA depends on the refresh time which is not necessarily as long as the transmission range. Such a short time interval increases the frequency of wake-up actions, thereby increases the energy for sending wake-up messages (or alarm messages). This can be verified by checking the number of proactive wake-up actions: when the node density is 1.25 $node/100m^2$ and the target speed is 20 m/s, the average number of proactive wake-up actions of MCTA and TPSS are respectively 5.48 and 2.91. This accomplishment of TPSS is attributed to the design from its prediction scheme to the sleep scheduling mechanism.

Average detection delay (AD)

Figure 4.17 and Figure 4.18 show the average detection delay on various node densities and various target speeds respectively. The curve for the NOSS scheme is not shown in the figure,

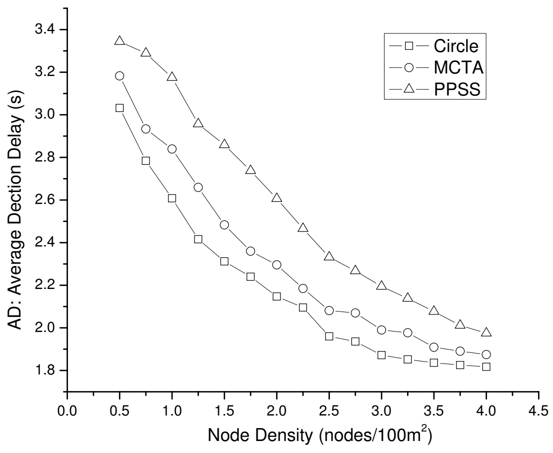

Figure 4.17: Average Detection Delay vs. Node Density

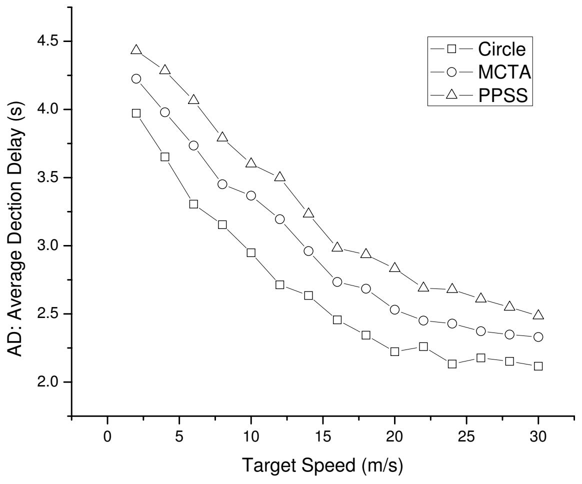

Figure 4.18: Average Detection Delay vs. Target Speed

for otherwise the granularity will be too large to show the difference among Circle, MCTA and TPSS clearly. The NOSS scheme serves as an upper bound for AD, since the detection delay is the longest with no proactive wake-up completely. The values of NOSS's AD spreads around 10 s for the node density case and in $6.5 - 85$ s for the target speed case. We observe that TPSS has longer delay than Circle and MCTA as a cost for enhancing energy efficiency.

Relative changes RC_{EE} and RC_{AD}

From the separate results of absolute values of EE and AD, we cannot determine if the increase in energy efficiency is worthy of the performance loss. According to the definition of RC_{EE} and RC_{AD} in Section 3.3, our goal is to obtain an overall conclusion on the quality of the tradeoff.

Figure 4.19 and Figure 4.20 show the relative changes of EE and AD of TPSS over MCTA (i.e., using MCTA as the reference scheme). Columns in slashes are relatively improved energy efficiency of TPSS over MCTA as the output, and columns in mesh are relatively increased detection delay of TPSS over MCTA as a cost. Then the difference between two styles of columns provide us the "net profits" of TPSS over MCTA.

The height change of columns in Figure 4.19 is a result of multiple competing factors. When the node density is low, both MCTA and TPSS are close to the no sleep scheduling scheme (i.e., the effect of proactive wake-up and sleep scheduling is neglectable). Thus, the difference of TPSS and MCTA on energy and delay is not significant. However, when nodes are highly densely deployed, the number of awakened nodes of TPSS increases faster than MCTA. Thus the improvement on energy efficiency is not as significant as lower densities. Moreover, the coverage of awakened nodes is much better than a sparse deployment, especially for TPSS

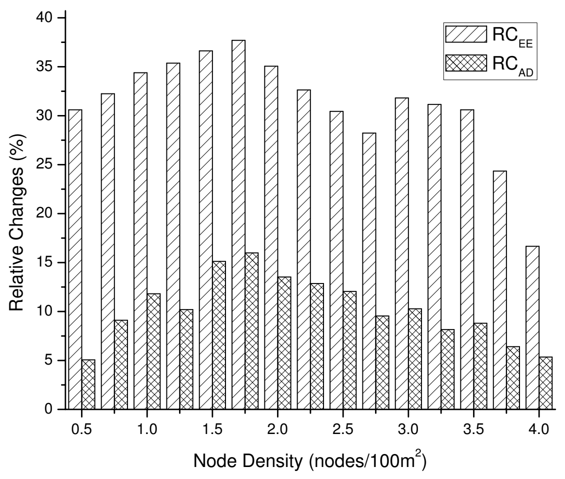

Figure 4.19: Relative Changes vs. Node Density

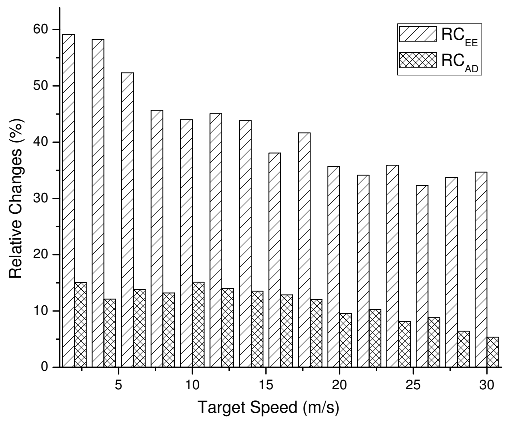

Figure 4.20: Relative Changes vs. Target Speed

which disperses awakened nodes along all the directions. Thus the increase of the detection delay is not as severe as lower node densities.

For most cases, TPSS introduces an improvement of 25% – 45% on energy efficiency based on MCTA, and suffers an increase of only 5% – 15% on detection delay.

4.7 Conclusion

This chapter presented a probability-based prediction and sleep scheduling scheme (TPSS) to achieve better tradeoff between energy efficiency and tracking performance for rare event tracking with sensor networks. We introduce a prediction scheme based on both kinematics rules and probability models, so that the energy consumed by low value-added sensor nodes (i.e., nodes that have a low probability of detecting the target) can be reduced. Based on the prediction results, we discussed two efforts for enhancing energy efficiency, including reducing the number of awakened nodes and scheduling their sleep patterns. We introduced a concept of awake region to reduce the number of awakened nodes, and presented a sleep scheduling mechanism to temporarily adjust their sleep patterns so as to shorten the active time as much as possible. We also experimentally compared TPSS with no sleep scheduling case, the legacy circle-based scheme and the MCTA algorithm. The evaluation results showed that TPSS achieves a better tradeoff between energy efficiency and tracking performance by serving more improvement on energy efficiency than the cost on tracking performance.

Chapter 5

Propagation Delay in Data Propagation Phase

5.1 Introduction

In this chapter, we study the queuing delay, one of the major delay sources during data propagation, together with the real-time capacity. We define real-time capacity as the network's capability to transmit time sensitive data with deadlines. These deadlines may come from the specifics of the applications, e.g., for timely response of actors [8], for timely target tracking [28], etc.

As previously discussed in Chapter 1, constrained network capacity may cause a long queuing delay. In fact, network capacity is important not just for sensor networks. In [44], Li *et. al.* study the capacity of wireless ad hoc networks, showing that the per node throughput actually available is dozens of times smaller than the apparent radio capacity. They also point out that, this is because, the interference among nodes may exceed the range at which they can communicate successfully. Another reason is the overhead of MAC protocols such as headers, RTS/CTS, and ACK packets. The capacity of wireless networks, in general, has been well studied since the work of Gupta and Kumar [40]. Except for the interference and overheads introduced in [44], network size, traffic patterns, and network architecture are the most commonly studied factors that constrain the achievable capacity.

In summary, we present the queuing delay and the per-hop success probability as well as define the real-time capacity for a many-to-one event-driven sensor network with unbalanced traffic loads. The following list shows some major points of design:

1) *Many-to-one.* Unlike ad hoc networks, sensor networks collect data from a large number of sensor nodes and report them to a much smaller number of sink nodes. Usually nodes propagate their measurements to the closest sink node. This data gathering structure leads

to a many-to-one traffic pattern. Without loss of generality, we consider a many-to-one data gathering network with only one sink node.

2) *Per-hop based.* Based on an arbitrary event distribution, we compose the network capacity from the per-hop throughput and slack time. This is because with unbalanced traffic loads, studying network wide conditions from a macro view may not guarantee that each packet can meet its respective deadline.

3) *Slack distribution.* We also present the end-to-end slack time distribution along a packet's propagation path. By subtracting the necessary propagation time from the end-to-end deadline of a packet, we obtain the end-to-end slack time for which the packet can be delayed at the most. Previous works use either a uniform distribution or an exponential distribution for describing the slack time among multiple hops [70]. Uniform distribution allocates the total end-to-end slack time to all the hops equally. Exponential distribution allocates the slack time as $L_h = \frac{L_{e2e}}{2^h}$, where L is the slack time and h is the number of hops from the sink node. With the exponential distribution, nodes close to the sink will therefore obtain more slack time, because of their (expected) greater congestion. However, either uniform distribution or exponential distribution is simple and completely based on mathematical models without considering much of the requirements of realistic networks. Uniform distribution is suitable only for a chain architecture without congestion, and it is a little casual for exponential distribution to claim that hop h deserves a slack time double long as that of hop $h + 1$. We establish a new distribution scheme which is closer to the realistic network and more adaptive by supporting more generic cases.

In details, we first establish a relationship between the event distribution and the expected end-to-end packet delay. Leveraging results from queuing theory, we then design a slack time distribution scheme. For estimating the probability that a packet can reach the sink within its deadline, we introduce the concept of per-hop success probability. This end-to-end probability is computed by multiplying each hop's probability of meeting the deadline of that hop. In terms of the per-hop success probability, we verify that the new slack time distribution scheme is optimal. Finally, we define the real-time capacity of the network, and study two special cases of traffic generation, i.e., a single source case and a continuous case. For the single source case, there is only one source node that may produce traffic in a many-to-one sensor network. On the contrary, each node in the network periodically produces an equal amount of traffic load for the continuous case. We show that our slack distribution scheme yields consistent or similar results for these two special cases as the past works, but is more adaptive by supporting more generic cases.

The rest of this chapter is organized as follows. In Section 5.2, we discuss the relationship between the allowable throughputs and the expected end-to-end delay. In Section 5.3, we present the slack time distribution scheme, and introduce the concept of per-hop success probability. In Section 5.4, we define and analyze real-time capacity, and study two special cases of traffic generation, including the single source case and the continuous case. The study of these special cases serves as a verification for the research result of the data propagation phase. We also discuss the potential applications of the slack time distribution scheme on dynamic priority packet scheduling, and provide its computational complexity. Section 5.5 concludes the chapter.

5.2 Traffic Pattern and Node Throughput

We first establish the relationship between the allowable throughputs and the expected end-to-end delay.

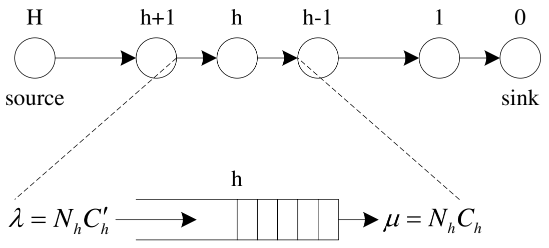

Figure 5.1: Propagation Path and Virtual Queue

Figure 5.1 shows the propagation path of a packet from the source that is H hops away from the sink. Numbers above the node circles represent the hop number. We utilize a queue to describe the relationship between throughputs of adjacent hop layers, so that we could leverage results of queuing theory for our analysis. As shown in Figure 5.1, the mean arrival rate λ of the queue is the total expected incoming throughput $N_h C'_h$, and the mean service rate μ is the total expected outgoing throughput $N_h C_h$.

Lemma 1 (Throughputs) *The outgoing throughput of a node in hop layer h is:*

$$C_h = \frac{W}{N_h}\left(1 - \frac{\sum_{i=1}^{h-1} N_i G_i}{\sum_{i=1}^{H_{max}} N_i G_i}\right) \tag{5.1}$$

The incoming throughput of a node in hop layer h is:

$$C'_h = \frac{N_{h+1}}{N_h} C_{h+1} \tag{5.2}$$

Proof: The total traffic that nodes produce after events are observed, i.e., $\sum_{i=1}^{H_{max}} N_i G_i$, may exceed the receiving capacity of the sink node. Therefore, we need to control the sending rate of source nodes. For the fairness of rate control, a good approach is to cut by their weights. Then the upper bound of the traffic load that a node in hop layer h may generate is:

$$\frac{WG_h}{\sum_{i=1}^{H_{max}} N_i G_i}$$

Since the outgoing throughput of a node is composed of the traffic that it produces and that it relays, we have $C_h = \frac{WG_h}{\sum_{i=1}^{H_{max}} N_i G_i} + C'_h = \frac{WG_h}{\sum_{i=1}^{H_{max}} N_i G_i} + \frac{N_{h+1}}{N_h} C_{h+1}$. Through iteration, we have:

$$C_1 = \frac{1}{N_1} \cdot \frac{W}{\sum_{i=1}^{H_{max}} N_i G_i} \sum_{i=1}^{h-1} N_i G_i + \frac{N_h}{N_1} C_h$$

Thus:

$$C_h = \frac{W}{N_h}\left(1 - \frac{\sum_{i=1}^{h-1} N_i G_i}{\sum_{i=1}^{H_{max}} N_i G_i}\right)$$

With the assumption of zero-overhead MAC protocols, the traffic transmitted by nodes in hop layer $h+1$ can arrive at nodes in hop layer h successfully without being dropped. That is, $N_h \cdot C'_h = N_{h+1} \cdot C_{h+1}$. **Done.**

With Lemma 1, we establish a model for the allowable average incoming and outgoing throughputs of nodes based on the event distribution. Next we introduce our new slack distribution scheme based on this throughput model.

5.3 Slack Distribution

How the end-to-end slack time is distributed to multi-hops embodies how a system considers network congestion and message velocity. The slack distribution scheme may be different in different network environments.

The continuous model and the chain model are two typical special cases of event-driven sensor networks with many-to-one traffic pattern. Recall that each node in a continuous WSN produces an equal amount of original traffic. A continuous sensor network is a special case when traffic loads are extremely average. A many-to-one sensor network degenerates to a chain architecture when only one source node generates traffic and propagates it via a single route. Along the chain path, there is no congestion and the packet velocity does not need to change, since no other traffic will interfere its transmission.

As previously mentioned, most existing works adopt either uniform distribution or exponential distribution for slack time distribution. Uniform distribution allocates the total end-to-end slack time to all the hops from the source to the sink equally, implicitly assuming that a packet suffers the same delay at each intermediate node along the path. In [70], Liu *et. al.* present an exponential distribution scheme for slack time, where the per-hop slack time is computed as $L_h = \frac{L_{e2e}}{2^h}$, where h is the number of hops from the sink node. Obviously, uniform distribution is more suitable for the chain model, while exponential distribution is often utilized for the continuous case. However, both of them are simple and completely based on mathematical models without considering much of the requirements of realistic networks. Moreover, neither of them is suitable for generic event-driven sensor networks with unbalanced traffic patterns.

In a generic sensor network with unbalanced many-to-one traffic pattern, the congestion gets more and more severe as a packet flows from the source towards the sink [71]. Consequently, its message velocity will decrease as a packet approaches the sink. Such a variable message velocity requires variable slack time at each hop, i.e., nodes that are closer to the sink deserve more slack time as packets suffer more delay. At the same time, the congestion degree at each hop depends on the specific traffic status. Therefore, the slack distribution should be based on the event distribution.

For generic many-to-one sensor networks and their random event distribution, we will have to design a more accurate scheme. The new distribution not only needs to consider the specific event distribution, but also has to meet the actual requirements instead of simply establishing a mathematical model.

We discuss the expected delay in two cases, in terms of whether or not nodes produce their own traffic: 1. when nodes in hop layer h produce their own traffic (i.e., $C_h > C'_h$), hop layer h can be approximated with a $M/M/1$ queue; and 2. when nodes do not produce their own traffic (i.e., $C_h = C'_h$), hop layer h can be approximated with a $D/D/1$ queue[1].

Both $M/M/1$ queue and $D/D/1$ queue are simple queue models. Usually $M/M/1$ queue is closer to the actual network environment than the $D/D/1$ queue [73]. However, a constraint of $M/M/1$ queue is that it is stable only if the service rate μ is greater than the arrival rate λ (i.e., $C_h > C'_h$ here). Queuing theory shows that only with this constraint, the queue will not keep growing forever, and the expected waiting time will not approach infinity. For the $C_h = C'_h$ case, the expected waiting time of $M/M/1$ queue will approach infinity and therefore is not useful for delay analysis. Thus, we adopt the $D/D/1$ queue for this case, which is feasible although not as accurate as $M/M/1$ queue.

From queuing theory, the expected waiting time of a packet in a queue of hop layer h is:

$$T_{h,H} = \begin{cases} \frac{1}{\mu-\lambda} & = & \frac{1}{N_hC_h - N_hC'_h} & , & (C_h > C'_h) \\ \frac{1}{\mu} & = & \frac{1}{N_hC_h} & , & (C_h = C'_h) \end{cases} \tag{5.3}$$

where H is the hop layer where the packet is generated.

On average, the tuned traffic load that a node in hop layer h may generate, $\frac{WG_h}{\sum_{i=1}^{H_{max}} N_iG_i}$, is just the difference $C_h - C'_h$ (when $G_h = 0$, $C_h = C'_h$). Substituting this and Equation 5.1

[1]Based on the Kendall notation, $M/M/1$ is a queue with Poisson arrival process, exponential service time, one server, and infinite buffer. $D/D/1$ is a queue with deterministic arrival rate and service time, one server, and infinite buffer [72].

into Equation 5.3, we have:

$$T_{h,H} = \begin{cases} \frac{\sum_{i=1}^{H_{max}} N_i G_i}{W N_h G_h} & , \quad (C_h > C'_h) \\ \frac{\sum_{i=1}^{H_{max}} N_i G_i}{W \sum_{i=h}^{H_{max}} N_i G_i} & , \quad (C_h = C'_h) \end{cases}$$

Now, the expected end-to-end waiting time is:

$$\begin{aligned} T_{e2e,H} &= \sum_{h=1}^{H} T_{h,H} \\ &= \begin{cases} \frac{\sum_{i=1}^{H_{max}} N_i G_i}{W} \sum_{h=1}^{H} \frac{1}{N_h G_h} & , \quad (C_h > C'_h) \\ \frac{\sum_{i=1}^{H_{max}} N_i G_i}{W} \sum_{h=1}^{H} \frac{1}{\sum_{i=h}^{H_{max}} N_i G_i} & , \quad (C_h = C'_h) \end{cases} \end{aligned} \tag{5.4}$$

This establishes a relationship between the event distribution and the expected delay.

We distribute the end-to-end slack time $L_{e2e,H}$ in a proportional manner to the expected delay determined by Equation 5.3. In fact, the distributed per-hop slacks can be considered as per-hop deadlines.

Definition 1 (per-hop Slack) *If the throughputs of all the hops along a packet's routing path are known, the per-hop slack time for hop h is given as:*

$$L_{h,H} = \frac{T_{h,H}}{T_{e2e,H}} \cdot L_{e2e,H}$$

Since the per-hop waiting time of a packet is exponentially distributed [72], the probability that a packet experiences less time than the distributed per-hop slack $L_{h,H}$ is:

$$P[t \le L_{h,H}] = 1 - e^{-\frac{L_{h,H}}{T_{h,H}}} = 1 - e^{-\frac{L_{e2e,H}}{T_{e2e,H}}} \tag{5.5}$$

From Equation 5.5, we observe that by the definition of this slack distribution scheme, the probability that a packet meets the per-hop deadline is independent of the hop number.

Now, we define the *per-hop success probability* as:

$$\begin{aligned} & P_{e2e}(D_{e2e,H}, H) \\ &= P[t_1 \le L_{1,H}, \ \cdots, \ t_h \le L_{h,H}, \ \cdots, \ t_H \le L_{H,H}] \\ &= P[t_1 \le L_{1,H}] \cdots P[t_h \le L_{h,H}] \cdots P[t_H \le L_{H,H}] \\ &= (1 - e^{-\frac{L_{e2e,H}}{T_{e2e,H}}})^H = (1 - e^{-\frac{L_{e2e,H}}{\sum_{h=1}^{H} T_{h,H}}})^H \end{aligned} \tag{5.6}$$

where t_h is the time actually spent on a node in hop layer h. $P_{e2e}(D_{e2e,H}, H)$ is a function of a packet's route length and its deadline in a given network environment with a specific event distribution.

Theorem 1 (Optimal Slack Distribution Theorem) *Based on the relationship between the event distribution and the expected delay, the slack distribution scheme determined in Definition 1 is optimal in terms of achieving the highest per-hop success probability.*

Proof: Without loss of generality, we consider the case of two hops. Thus

$$P[t_1 \leq L_{1,H},\ t_2 \leq L_{2,H}] = (1 - e^{-\frac{L_{e2e,H}}{T_{e2e,H}}})^2$$

Now, assume that a different slack distribution scheme distributes the slack at these two hops as $L_{1,H} + \delta$ and $L_{2,H} - \delta$ ($\forall\ \delta > 0$), instead of $L_{1,H}$ and $L_{2,H}$. Then the probability will become:

$$P[t_1 \leq L_{1,H} + \delta,\ t_2 \leq L_{2,H} - \delta] \tag{5.7}$$
$$= (1 - e^{-\frac{L_{e2e,H}}{T_{e2e,H}}} \cdot e^{-\frac{\delta}{T_{1,H}}}) \cdot (1 - e^{-\frac{L_{e2e,H}}{T_{e2e,H}}} \cdot e^{\frac{\delta}{T_{2,H}}}) \tag{5.8}$$

We substitute the complicated expressions in Equation 5.7 with constants for simplification. Let $Y = e^{-\frac{L_{e2e,H}}{T_{e2e,H}}}$, $A = e^{-\frac{\delta}{T_{1,H}}}$, and $B = e^{\frac{\delta}{T_{2,H}}}$. Then we have $0 < Y < 1$ and $0 < A < 1 < B$. Now, we need to prove that $(1 - AY)(1 - BY) < (1 - Y)^2 \Leftrightarrow (AB - 1)Y < A + B - 2$. This inequality can be proved as,

$$\begin{aligned} Left = (AB - 1)Y &< AB - 1 \\ &= (A - 1)(B - 1) + A + B - 2 \\ &< A + B - 2 = Right \end{aligned}$$

As δ is arbitrary, the slack distribution scheme in Definition 1 is guaranteed to be optimal. **Done.**

The slack distribution scheme presented in Section 5.3 can be used for configuring a packet's per-hop deadline, and accordingly for adjusting the packet priority in the queue. Each time a packet leaves a relay node, its remaining end-to-end slack time $L_{e2e,H}$ can be recalculated, and the allowable slack time for the next hop can be determined accordingly. This builds a foundation for dynamic priority scheduling.

We now determine the computational complexity for determining the slack time. Given an event distribution $\hat{G}$ and a network size H_{max}, the complexity for computing $T_{e2e,H}$ with Equation 5.4 is $O(H_{max}^2)$. Thus, the complexity for computing a new per-hop slack time $L_{h,H}$ with Definition 1, and the complexity for computing the per-hop success probability $P_{e2e}(D_{e2e,H}, H)$ with Equation 5.6 are also $O(H_{max}^2)$. Therefore, the complexity for computing the real-time capacity $RTC(\alpha)$ is $O(H_{max}^3)$.

5.4 Real-time Capacity

We now define and determine the real-time capacity concept. We also study some special traffic generation cases as examples of applying the slack time distribution scheme and the real-time capacity.

5.4.1 Definition

In previous works, the capacity of wireless ad hoc and sensor networks is usually defined as a function of the network size, and upper or lower bounds of the network capacity are often the research focus [40, 47]. Delay has been studied in some works, but usually as a constraint [42]. In [19], the real-time capacity is defined as the upper bound of the sum of all the in-transit packets' ratio of the packet size divided by their respective deadlines, i.e., $C_{RT} = W \sum \frac{C_i}{D_i}$, where C_i and D_i are the transmission time and the deadline of packet i, respectively. This definition works for fixed priority packet scheduling discussed in [19]. However, [19] has used it as a network-wide term without considering per-hop details.

Sometimes network-wide conditions cannot provide sufficient guarantees especially for event-driven sensor networks with unbalanced traffic pattern. For example, two nodes each with 100 *bit/s* capacity cannot simultaneously guarantee the timely delivery of a data flow requiring 20 *bit/s* capacity and another requiring 150 *bit/s* capacity.

We define real-time capacity based on the per-hop success probability defined in Equation 5.6, i.e., the probability that a packet meets the per-hop deadline of all its hops. Let α denote a threshold for the per-hop success probability that must be satisfied. Now, we define real-time capacity as follows:

Definition 2 (Real-time Capacity) *The real-time capacity of an event-driven data-gathering sensor network is defined as the sum of traffic loads, the per-hop success probability of which is higher than a given requirement threshold α i.e.,*

$$RTC(\alpha) = \frac{W}{\sum_{i=1}^{H_{max}} N_i G_i} \sum_{P_{e2e}(D_{e2e,i}, i) \geq \alpha}^{1 \leq i \leq H_{max}} N_i G_i \tag{5.9}$$

As in Lemma 1, $\frac{W}{\sum_{i=1}^{H_{max}} N_i G_i}$ is a factor for rate control.

Thus, this definition of real-time capacity defines how much real-time data in bits can meet their deadlines with a per-hop success probability that is greater than the given threshold α, for an arbitrary event distribution $\hat{G}$.

An important advantage of the per-hop success probability concept lies in its ability for configuring the end-to-end deadline for a packet generated at a node that is H hops away from the sink. In the past, most works do not discuss how the deadline of packets are determined or configured. In contrast, here, we provide an approach for configuring the packet deadline based on the requirement for the delivery probability. For a packet generated by a node in hop layer H, we can compute the lower bound of its deadline when given a requirement threshold of the per-hop success probability α. Formally, this is computed as:

$$\begin{aligned}
&P_{e2e}(D_{e2e,H}, H) = (1 - e^{-\frac{L_{e2e,H}}{T_{e2e,H}}})^H \geq \alpha \\
\iff & e^{-\frac{L_{e2e,H}}{T_{e2e,H}}} \leq 1 - \alpha^{\frac{1}{H}} \\
\iff & L_{e2e,H} = D_{e2e,H} - H\tau \geq -T_{e2e,H} \ln(1 - \alpha^{\frac{1}{H}}) \\
\iff & D_{e2e,H} \geq H\tau - T_{e2e,H} \ln(1 - \alpha^{\frac{1}{H}})
\end{aligned}$$

5.4.2 Examples

We now study some special cases of traffic generation, and show how the real-time capacity and the per-hop success probability can be applied.

Single Source Case

The single source case yields a chain architecture, where there is only one node that produces traffic. Assume that this node is located in hop layer H. For each hop layer $h \in [1, H-1] \cup [H+1, H_{max}]$, nodes do not produce their own traffic, i.e., $G_h = 0$ and $C_h = C'_h$. The only traffic source is shown as $G_H > 0$. Thus, the expected delay for $\forall\ h \in [1, H-1]$ is:

$$T_{h,H} = \frac{\sum_{i=1}^{H_{max}} N_i G_i}{W \sum_{i=h}^{H_{max}} N_i G_i} = \frac{N_H G_H}{W N_H G_H} = \frac{1}{W}$$

For the hop layer H, the source node produces its own traffic. Thus:

$$T_{H,H} = \frac{\sum_{i=1}^{H_{max}} N_i G_i}{W N_H G_H} = \frac{N_H G_H}{W N_H G_H} = \frac{1}{W}$$

Therefore:

$$T_{e2e,H} = \sum_{h=1}^{H} T_{h,H} = \sum_{h=1}^{H} \frac{1}{W} = \frac{H}{W}$$

So the per-hop slack time is $L_{h,H} = \frac{L_{e2e,H}}{H}$, which is independent of the hop number. This is consistent with the uniform distribution [70].

The per-hop success probability is:

$$P_{e2e}(D_{e2e,H}, H) = (1 - e^{-\frac{L_{e2e,H}}{T_{e2e,H}}})^H = (1 - e^{-\frac{WL_{e2e,H}}{H}})^H$$

The real-time capacity of the whole network is either W or 0 depending on whether $P_{e2e}(D_{e2e,H}, H)$ is greater than a given threshold α.

Continuous Case

In a continuous sensor network, all the nodes produce the same traffic load. Therefore, $\forall\ h$, we have $C_h > C'_h$ and $G_h = E$. Assume that packets produced by a node located in hop layer H are configured with an end-to-end deadline $D_{e2e,H}$. Then, the expected delay for $\forall\ h \in [1, H]$ is:

$$T_{h,H} = \frac{\sum_{i=1}^{H_{max}} N_i G_i}{WE} \cdot \frac{1}{N_h}$$

The end-to-end expected delay is:

$$T_{e2e,H} = \sum_{h=1}^{H} T_{h,H} = \frac{\sum_{i=1}^{H_{max}} N_i G_i}{WE} \sum_{h=1}^{H} \frac{1}{N_h}$$

Thus, the per-hop slack time is:

$$L_{h,H} = \frac{\frac{1}{N_h}}{\sum_{h=1}^{H} \frac{1}{N_h}} L_{e2e,H}$$

We may observe that the closer a hop layer is to the sink, the smaller N_h is, and the larger their $L_{h,H}$ will be. This is consistent with the actual requirement that nodes closer to the sink deserve more slack time in many-to-one sensor networks. Compared with the exponential distribution scheme [70], the new slack distribution scheme is optimal in terms of the per-hop success probability, and therefore more accurate for actual network requirements.

Accordingly, we can also compute the per-hop success probability and the real-time capacity based on Equation 5.6 and 5.9. However, the detailed results depend on the configuration of each packet's deadline.

From these two special cases, we can observe that the research results of this chapter (including the slack distribution scheme, the per-hop success probability and real-time capacity) yield consistent or similar results as that of the past works. In addition, our results for event-driven sensor networks are more generic by taking the event distribution into account.

5.5 Conclusions

This chapter investigates the real-time capacity of event-driven data-gathering sensor networks with the unbalanced many-to-one traffic pattern. We establish the relationship between the event distribution and the expected end-to-end delay. We then leverage the $M/M/1$ and $D/D/1$ models of queuing theory to estimate the expected waiting time in the queues, so as to distribute the end-to-end slack time proportionally to this expected delay.

We introduce the concept of per-hop success probability to estimate the probability that a packet can reach the sink meeting its deadline. In terms of the per-hop success probability, we verify that the new slack time distribution scheme is optimal. We then define the real-time capacity of the network. Finally, we study two special cases of traffic generation, including a single source case and a continuous case. Compared with past works, our slack distribution scheme and real-time capacity analysis yield consistent or similar results for these two special cases as that of past works, but is more adaptive by supporting more generic cases. Our work thus lays the foundation for further research on network scheduling for event-driven sensor networks with unbalanced traffic loads.

Chapter 6

Conclusions, Contributions and Proposed Post Preliminary Exam Work

In this dissertation proposal, we analyze the quantitative relation among the end-to-end delay, the probability of guaranteeing this delay and network parameters in event-driven sensor networks. We partition the whole process of event observing and reporting into two phases, i.e., an event observation phase and a data propagation phase. In the end-to-end delay, the detection delay and the queuing delay are analyzed, each for a phase. We use a typical event-driven application, i.e., target tracking, as the example for the convenience of discussion.

For the event observation phase, we present a target prediction and sleep scheduling scheme named as TPSS to reduce the energy consumption when satisfying the given delay constraint. We design a target prediction method based on both kinematics rules and theory of probability. Then based on the prediction results, we enhance energy efficiency by designing a novel sleep scheduling mechanism that reduces the number of awakened nodes and schedules their sleep patterns in an integrated manner. We analyze the detection delay and the detection probability under TPSS scheme, as well as evaluate it with a simulation tool. The simulation results show that TPSS achieves a better tradeoff between energy efficiency and tracking performance than the existing works.

For the data propagation phase, we analyze the delay from the point of view of network capacity. By approximating all the nodes that have the same distance away from the sink node as a queue, the queuing delay can be estimated, which is the major delay source except for the transmission delay and the sleep delay during data propagation. We also develop a new slack time distribution scheme for unbalanced many-to-one traffic patterns. In terms of the per-hop success probability, which is defined as the probability for a packet to meet its deadlines at each hop, we prove that the slack distribution scheme is optimal among all the sub-delay partition schemes. Finally, we define and analyze the network-wide real-time capacity. Real-time capacity is defined as when given a per-hop success probability, how much data (in bit per second) can be delivered to the sink, meeting their deadlines. An important advantage of the per-hop success probability concept is that application designers can configure a packet's deadline based on the required successful delivery probability.

6.1 Contributions

In this proposal, we established a quantitative relation among the end-to-end delay, the probability of guaranteeing this delay and network parameters in event-driven sensor networks. Such a quantitative relation can help system designers to configure a sensor network towards certain real-time constraints. These parameters may include but not limited to node density D_s, average target speed TS, the toggling period TP, the duty cycle DC and σ_{Δ_n} (that may influence the probability with which a node is scheduled).

We presented a two-phase model for event-driven applications, including an event observation phase and a data propagation phase. Compared with the six-phase model used in [1], this partition scheme is simpler but more generic, as it only involves two core steps of event-driven applications. It is more feasible for obtaining system-wide, globally optimal solutions than the six-phase model.

We designed a target prediction method based on both kinematics rules and theory of probability. As far as we know, such an integrated prediction method was not studied in the past. From the prediction result, a target's movement status can be illustrated with probabilistic models, which contribute to scheduling the sleep pattern of nodes in a probabilistic manner. At the same time, the parameters of these probabilistic models can be modified for satisfying given real-time constraints.

For enhancing energy efficiency, we designed a novel approach by reducing the number of awakened nodes and scheduling their sleep patterns in an integrated manner. Compared with past proactive wake-up mechanisms, this integrated effort significantly reduces the energy consumption, without impairing the tracking performance much. Based on our analysis, a network will be able to minimize the energy consumption when satisfying a given delay constraint.

We analyzed the detection probability and the detection delay under TPSS scheme, as well as evaluated it by simulation-based experimental studies. The simulation results show that TPSS achieves a better tradeoff between energy efficiency and tracking performance than the existing works.

For the propagation delay, we leverage the results from queuing theory for estimating the expected queuing delay, which is one of the major delay sources during data propagation. We designed a new slack time distribution scheme based on the requirements of realistic networks and results from queuing theory. Also we introduced the concept of per-hop success probability, in terms of which we proved that the new slack time distribution scheme is optimal. This provides us a delay partition method in the data propagation phase for maximizing the end-to-end probability of meeting the deadline.

6.2 Post Preliminary Exam Work

We propose the following questions for post preliminary exam work:

- *Five questions.* Based on the quantitative relation presented in this proposal, we propose to answer all the five questions listed in Section 1.1 after the preliminary exam. In fact, most of these questions are non-linear programming problems. For example, subject to an upper bounded end-to-end delay, we need to maximize the product of the detection probability and the per-hop success probability corresponding to sub-delays for two phases. Another question that deserves more work is about the difference between the first detection delay and the subsequent detection delays after the first one. When nodes work in the default mode, the active time (i.e., $TP \cdot DC$) is short for prolonging the network lifetime. Thus the initial detection delay will be very long. After the first observation, many awakened nodes know the existence of the target and have already re-scheduled their sleep pattern, which will shorten the subsequent detection delays significantly. However, the delay constraint for detecting and reporting an event may be certain specifically for an application. When partitioning the end-to-end delay, we may need to emphasize the event observation phase for the first detection, and then emphasize the data propagation phase in the following detections. This example also requires a global optimal solution.

- *Multiple sink nodes.* In realistic sensor networks especially those large scale deployments, designers usually deploy multiple sinks to reduce the distance from source nodes to the sink, so as to improve the network performance such as the end-to-end delay. The propagation delay and the network capacity of sensor networks with multiple sink nodes may be very different from the one sink case. Which sink node a source node should report to may be configured by applications or dependent on network optimization. Moreover, the number of hops away from a sink node can be either obtained from the routing protocol or calculated by the distance between nodes by Kleinrock-Silvester formula [74]. We propose to extend the analysis result of the data propagation phase to the multiple sink case.

- *Multiple targets.* The current research for the event observation phase is constrained to the single target tracking case. When multiple targets move close to each other, the redundant alarm messages of interfering targets may be leveraged for further enhancing energy efficiency. At the same time, multiple target tracking may introduce new workload of computation. Therefore, the analysis of the algorithm complexity will also be a new problem. We propose to answer the question that how much more energy can be saved for multiple target tracking, after excluding the extra energy consumed on computation.

- *Dynamic event distribution.* Dynamic event means that the location where events occur is dynamic. Instead of considering static event distribution, it would be more accurate and more adaptive if dynamic event distribution is considered with random processes for the data propagation phase. However, dynamic event distribution will definitely require a dynamic slack distribution scheme and a dynamic delay partition method. Each intermediate node has to check the remaining slack time and adaptively tune its scheduling policy and/or the packet's priority. This may also require the time synchronization. In Chapter 3, we assumed time synchronization mainly for target prediction. However, time synchronization is usually very challenging for realistic applications of sensor networks. We propose to extend the analysis result of the data propagation phase to dynamic event distribution, and consider if it is possible to achieve the target prediction and the propagation delay analysis for dynamic event distribution without time synchronization.

- *Potential applications.* We discussed several potential applications of the slack time distribution scheme and real-time capacity in this proposal. Many more potential applications can be explored based on the research results of the relation among delay, probability, and network parameters. For example, based on real-time capacity, the transmission rate of source nodes can be monitored to ensure the best use of network resources, but avoid any unnecessary congestions and delays at the same time. Another example is for packet scheduling. Packets may be scheduled with dynamic priorities, which are determined by real-time capacity-oriented slack time and delivery probability, among other possible aspects. We propose to provide more scenarios where the results of this proposal can help on the design of a sensor network.

Bibliography

[1] T. He, P. Vicaire, T. Yan, and L. L. et al., "Achieving real-time target tracking using wireless sensor networks," *ACM Transaction on Embedded Computing System (TECS)*, 2007.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks (Amsterdam, Netherlands: 1999)*, vol. 38, no. 4, pp. 393–422, 2002.

[3] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman, "A taxonomy of wireless microsensor network models," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 6, no. 2, pp. 28–36, 2002.

[4] P. Bonnet, J. Gehrke, and P. Seshadri, "Querying the physical world," *Personal Communications, IEEE*, vol. 7, no. 5, pp. 10–15, Oct 2000.

[5] B. G. Celler, W. Earnshaw, E. D. Ilsar, L. Betbeder-Matibet, M. F. Harris, R. Clark, T. Hesketh, and N. H. Lovell, "Remote monitoring of health status of the elderly at home. a multidisciplinary project on aging at the university of new south wales," *International Journal of Bio-Medical Computing*, vol. 40, no. 2, pp. 147 – 155, 1995.

[6] H. Song, S. Zhu, and G. Cao, "Svats: A sensor-network-based vehicle anti-theft system," April 2008, pp. 2128–2136.

[7] S. Oh, L. Schenato, P. Chen, and S. Sastry, "A scalable real-time multiple-target tracking algorithm for sensor networks," *Memorandum*, 2005.

[8] V. C. Gungor, O. B. Akan, and I. F. Akyildiz, "A real-time and reliable transport (rt)2 protocol for wireless sensor and actor networks," in *IEEE/ACM Transactions on Networking*, 2008.

[9] Q. Cao, T. Abdelzaher, T. He, and J. Stankovic, "Towards optimal sleep scheduling in sensor networks for rare event detection," in *Proceedings of the 4th international symposium on Information processing in sensor networks*, no. 4, 2005.

[10] D. Tian and N. Georganas, "A coverage-preserving node scheduling scheme for large wireless sensor networks," 2002.

[11] J. Jeong, T. Hwang, T. He, and D. Du, "Mcta: Target tracking algorithm based on minimal contour in wireless sensor networks," in *INFOCOM*, 2007, pp. 2371–2375.

[12] S. S. Choi, "Analysis of low latency mac protocols for clustered sensor networks," 2008, pp. 1894–1898.

[13] Y. Li, C. S. Chen, Y.-Q. Song, and Z. Wang, "Real-time qos support in wireless sensor networks: a survey," 2007.

[14] X. Xia and Q. Liang, "Latency and energy efficiency evaluation in wireless sensor networks," vol. 3, 2005, pp. 1594–1598.

[15] J. Kleinberg and E. Tardos, *Algorithm Design.* Addison-Wesley Longman Publishing Co., Inc., 2005.

[16] X. Wang, J.-J. Ma, S. Wang, and D.-W. Bi, "Cluster-based dynamic energy management for collaborative target tracking in wireless sensor networks," *Sensors*, vol. 7, pp. 1193–1215, 2007.

[17] J. Fuemmeler and V. Veeravalli, "Smart sleeping policies for energy efficient tracking in sensor networks," *IEEE Transactions on Signal Processing*, 2007.

[18] Y. Gu and T. He, "Data forwarding in extremely low duty-cycle sensor networks with unreliable communication links," in *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, 2007, pp. 321–334.

[19] T. F. Abdelzaher, S. Prabh, and R. Kiran, "On real-time capacity limits of multihop wireless sensor networks," in *RTSS '04: Proceedings of the 25th IEEE International Real-Time Systems Symposium*, 2004, pp. 359–370.

[20] T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic, T. F. Abdelzaher, J. Hui, and B. Krogh, "Vigilnet: An integrated sensor network system for energy-efficient surveillance," *ACM Transactions on Sensor Networks*, vol. 2, no. 1, pp. 1–38, 2006.

[21] G. Lu, N. Sadagopan, B. Krishnamachari, and A. Goel, "Delay efficient sleep scheduling in wireless sensor networks," in *IEEE Infocom*, 2005.

[22] "Adaptive localized qos-constrained data aggregation and processing in distributed sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 9, pp. 923–933, 2006, zhu,, Jin and Papavassiliou,, Symeon and Yang,, Jie.

[23] M. Caccamo, L. Zhang, L. Sha, and G. Buttazzo, "An implicit prioritized access protocol for wireless sensor networks," 2002, pp. 39–48.

[24] O. Chipara, Z. He, G. Xing, Q. Chen, X. Wang, C. Lu, J. Stankovic, and T. Abdelzaher, "Real-time power-aware routing in sensor networks," *Proceedings of the 14th IEEE International Workshop on Quality of Service*, pp. 83–92, 2006.

[25] Y. Yu, B. Krishnamachari, and V. K. Prasanna, "Energy-latency tradeoffs for data gathering in wireless sensor networks," in *INFOCOM*, 2004.

[26] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava, "Optimizing sensor networks in the energy-latency-density design space," *IEEE Transactions on Mobile Computing*, vol. 1, no. 1, pp. 70–80, 2002.

[27] D. Shuman and M. Liu, "Optimal sleep scheduling for a wireless sensor network node," in *Fortieth Asilomar Conference on Signals, Systems and Computers (ACSSC)*, 2006, pp. 1337–1341.

[28] C. Gui and P. Mohapatra, "Power conservation and quality of surveillance in target tracking sensor networks," in *Proceedings of the 10th annual international conference on Mobile computing and networking*, 2004, pp. 129–143.

[29] S. Ren, Q. Li, H. Wang, X. Chen, and X. Zhang, "Design and analysis of sensing scheduling algorithms under partial coverage for object detection in sensor networks," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 3, 2007, pp. 334–350.

[30] S. Liu, K.-W. Fan, and P. Sinha, "Dynamic sleep scheduling using online experimentation for wireless sensor networks," in *SenMetrics*, 2005.

[31] Y. Chen and E. Fleury, "A distributed policy scheduling for wireless sensor networks," in *IEEE Infocom*, 2007.

[32] J. Denga, Y. S. Hanb, W. B. Heinzelmanc, and P. K. Varshney, "Balanced-energy sleep scheduling scheme for high density cluster-based sensor networks," in *Computer Communications: special issue on ASWN04*, vol. 28, 2005, pp. 1631–1642.

[33] Y. Wu, S. Fahmy, and N. B. Shroff, "Energy efficient sleep/wake scheduling for multi-hop sensor networks: Non-convexity and approximation algorithm," in *IEEE Infocom*, 2007.

[34] A. Keshavarzian, H. Lee, and L. Venkatraman, "Wakeup scheduling in wireless sensor networks," in *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2006.

[35] Z. Guo and M. Zhou, "Prediction-based object tracking algorithm with load balance for wireless sensor networks," in *Proceedings of IEEE Networking, Sensing and Control, 2005*, 2005.

[36] X. Wang, J.-J. Ma, S. Wang, and D.-W. Bi, "Prediction-based dynamic energy management in wireless sensor networks," pp. 251–266, 2007.

[37] Y. Xu, J. Winter, and W.-C. Lee, "Prediction-based strategies for energy saving in object tracking sensor networks," in *Proceedings of the 2004 IEEE International Conference on Mobile Data Management (MDM04)*, 2004.

[38] "Kinematics." [Online]. Available: http://en.wikipedia.org/wiki/Kinematics

[39] R. M. Taqi, M. Z. Hameed, A. A. Hammad, Y. S. Wha, and K. K. Hyung, "Adaptive yaw rate aware sensor wakeup schemes protocol (a-yap) for target prediction and tracking in sensor networks," *IEICE - Transactions on Communications*, vol. E91-B, no. 11, pp. 3524–3533, 2008.

[40] P. Gupta and P. Kumar, "The capacity of wireless networks," *Information Theory, IEEE Transactions on*, vol. 46, no. 2, pp. 388–404, 2000.

[41] M. Grossglauser and D. N. C. Tse, "Mobility increases the capacity of ad hoc wireless networks," *IEEE/ACM Trans. Netw.*, vol. 10, no. 4, pp. 477–486, 2002.

[42] N. Bansal and Z. Liu, "Capacity, delay and mobility in wireless ad-hoc networks," *INFOCOM*, 2003.

[43] M. Gastpar and M. Vetterli, "On the capacity of wireless networks: The relay case," *INFOCOM*, 2002.

[44] J. Li, C. Blake, D. S. J. De, C. Hu, I. Lee, and R. Morris, "Capacity of ad hoc wireless networks," in *MobiCom*, 2001, pp. 61–69.

[45] B. Liu, Z. Liu, and D. Towsley, "On the capacity of hybrid wireless networks," *INFOCOM*, vol. 2, pp. 1543–1552, 2003.

[46] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," in *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, 2003, pp. 66–80.

[47] D. Marco, E. Duarte-Melo, M. Liu, and D. Neuhoff, "On the many-to-one transport capacity of a dense wireless sensor network and the compressibility of its data," *Proc. International Workshop on Information Processing in Sensor Networks*, 2003.

[48] H. Gamal, "On the scaling laws of dense wireless sensor networks: the data gathering channel," *Information Theory, IEEE Transactions on*, vol. 51, no. 3, pp. 1229–1234, 2005.

[49] J. L. Bredin, E. D. Demaine, M. Hajiaghayi, and D. Rus, "Deploying sensor networks with guaranteed capacity and fault tolerance," in *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, 2005, pp. 309–319.

[50] E. J. Duarte-melo and M. Liu, "Data-gathering wireless sensor networks: Organization and capacity," *Computer Networks*, vol. 43, pp. 519–537, 2003.

[51] L. Sankaranarayanan, G. Kramer, and N. Mandayam, "Hierarchical sensor networks: Capacity bounds and cooperative strategies using the multiple-access relay channel model," in *First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, 2004, pp. 191–199.

[52] G. Barrenechea, B. Beferull-Lozano, and M. Vetterli, "Lattice sensor networks: capacity limits, optimal routing and robustness to failures," in *IPSN '04: Proceedings of the 3rd international symposium on Information processing in sensor networks*, 2004, pp. 186–195.

[53] A. Giridhar and P. Kumar, "Computing and communicating functions over sensor networks," *Selected Areas in Communications, IEEE Journal on*, vol. 23, no. 4, pp. 755–764, 2005.

[54] A. Giridhar and P. R. Kumar, "Toward a theory of in-network computation in wireless sensor networks," *Communications Magazine, IEEE*, vol. 44, pp. 98–107, 2006.

[55] C. Comaniciu and H. Poor, "On the capacity of mobile ad hoc networks with delay constraints," *Wireless Communications, IEEE Transactions on*, vol. 5, no. 8, pp. 2061–2071, 2006.

[56] T. Abdelzaher, G. Thaker, and P. Lardieri, "A feasible region for meeting aperiodic end-to-end deadlines in resource pipelines," 2004, pp. 436–445.

[57] J. Hightower and G. Borriello, "Location systems for ubiquitous computing," *IEEE Computer*, vol. 34, no. 8, pp. 57–66, August 2001.

[58] R. Stoleru, J. A. Stankovic, and S. Son, "Robust node localization for wireless sensor networks," in *EmNets*, 2007.

[59] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *SIGOPS Oper. Syst. Rev.*, pp. 147–163, 2002.

[60] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, New York, NY, USA, 2004, pp. 95–107.

[61] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, , and H. Z. et al., "A line in the sand: A wireless sensor network for target detection, classification,and tracking," *Computer Networks (Elsevier)*, vol. 46, no. 5, pp. 605–634, 2004.

[62] J. Liu, M. Chu, and J. Reich, "Multitarget tracking in distributed sensor networks," *Signal Processing Magazine, IEEE*, vol. 24, no. 3, pp. 36–46, May 2007.

[63] H. Kang, H. Hong, S. Sung, and K. Kim, "Interference and sink capacity of wireless cdma sensor networks with layer architecture," in *ETRI Journal*, vol. 30, no. 1, 2008, pp. 13–20.

[64] S. Pattem, S. Poduri, and B. Krishnamachari, "Energy-quality tradeoffs for target tracking in wireless sensor networks," in *In Proceedings of IPSN03*, 2003, pp. 32–46.

[65] CrossBow, "Mica data sheet," http://www.xbow.com.

[66] J. Liu, J. Liu, J. Reich, P. Cheung, and F. Zhao, "Distributed group management in sensor networks: Algorithms and applications to localization and tracking," in *Telecommunication Systems*, vol. 26, no. 2-4, 2004, pp. 235–251.

[67] Q. Cao, T. Yan, J. Stankovic, and T. Abdelzaher, "Analysis of target detection performance for wireless sensor networks," in *Intl Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2005, pp. 276–292.

[68] M. Athanassoulis, I. Alagiannis, and S. Hadjiefthymiades, "Energy efficiency in wireless sensor networks: A utility-based architecture," in *European Wireless 2007*, 2007.

[69] J. L. Hill and D. E. Culler, "Mica: a wireless platform for deeply embedded networks," *IEEE Micro*, vol. 22, 2002.

[70] K. Liu, N. Abu-Ghazaleh, and K.-D. Kang, "Jits: just-in-time scheduling for real-time sensor data dissemination," *Fourth Annual IEEE International Conference on Pervasive Computing and Communications*, pp. 5–46, 2006.

[71] M. Ahmad and D. Turgut, "Congestion avoidance and fairness in wireless sensor networks," *IEEE Global Telecommunications Conference (GLOBECOM)*, pp. 1–6, 2008.

[72] M. Zukerman, *Introduction to Queueing Theory and Stochastic Teletraffic Models*, 2008. [Online]. Available: http://www.ee.unimelb.edu.au/staff/mzu/classnotes.pdf

[73] M. Harchol-Balter and D. Wolfe, "In network of queues, m/m/1 can outperform m/d/1," Tech. Rep., 1994.

[74] L. Kleinrock and J. Slivester, "Optimum transmission radii for packet radio networks or why six is a magic number," in *National Telecomm Conference*, 1978.