A Holistic Approach to the Exploitation of Energy Harvesting Motivated Networks: Protocols and Countermeasures to DoS Attacks

by

Cong Pu, B.S. and M.S.

A Doctoral Dissertation

In

Computer Science

Submitted to the Graduate Faculty
of Texas Tech University in
Partial Fulfillment of
the Requirements for the Degree of

Doctor of Philosophy

Approved

Dr. Sunho Lim

Chair of Committee

Dr. Yu Zhuang

Dr. Michael (Eonsuk) Shin

Dr. Yong Chen

Dr. Mark A. Sheridan

Dean of the Graduate School

August, 2016

ACKNOWLEDGMENTS

In my pursuit of Ph.D. degree, I have received enormous support and guidance from numerous people. I would like to thank each of them.

First of all, I would like to extend my immense gratitude to Dr. Sunho Lim, my research advisor, teaching mentor, and a friend in life, for his excellent guidance, generous financial support, and patience. My dissertation work would not have been accomplished without his continuous support and professional guidance. Two primary reasons made me join his research group: the areas of research in which he was involved, and what I had heard about his personality. I have not been disappointed in either regard. He has provided guidance in my research work while also allowing me to work independently the majority of the time. The opportunity to work with him is one of the best things that ever happened to me in my three years Ph.D. experience.

Second, I would like to thank other committee members, Dr. Yu Zhuang, Dr. Yong Chen, and Dr. Michael (Eonsuk) Shin, for their valuable comments and suggestions to this dissertation. I feel privileged and honored to have them as members of my dissertation committee. I really appreciate all the committee members for their time, effort and great help in my growth.

Third, I would like to thank the Dept. of Computer Science at Texas Tech University for providing me the teaching opportunity as an instructor during my Ph.D. study. Without this opportunity, I would not have received the 2015 Helen DeVitt Jones Excellence in Graduate Teaching Award.

Most importantly, I would like to give my special thanks to my family. I thank my parents, Guozheng Pu and Zengliang Huang for their constant encouragement, love and enormous support. Whenever I feel tired or depressed, they are always there waiting for me. My lovely wife, Pei Cao, who shared all my experience of excitement and disappointment over the years, has always been the source of support. You are the sunshine in my life!

Finally, I would like to have my best wishes to my soon to be born baby. Welcome to life on Earth, and welcome to your family. May you be surrounded by love, inspired to learn and grow, and always know that you are deeply cherished.

TABLE OF CONTENTS

Acknowledgments ii
Abstract v
List of Tables vii
List of Figures viii
1. Introduction 1
1.1 Challenges and Motivations 1
1.2 Contributions 3
1.3 Organization 5
2. Background 7
2.1 Energy Harvesting Aware Routing Protocols 7
2.2 Countermeasure to Forwarding Misbehaviors 8
3. Light-Weight Forwarding Protocols 12
3.1 Introduction 12
3.2 System Model 13
3.3 Detail Operations 15
3.4 Simulation Testbed 20
3.5 Simulation Results 21
3.6 Summary 26
4. Light-Weight Countermeasure to Forwarding Misbehaviors 28
4.1 Introduction 28
4.2 System and Adversary Models 29
4.3 Single Checkpoint-based Detection 30
4.4 Analysis of the Proposed Countermeasure 33
4.5 Simulation Testbed 39
4.6 Immunity to Other Attacks 44
4.7 Potential Enhancements 45
4.8 Summary 46
5. Camouflage-based Active Detection 47
5.1 Introduction 47
5.2 System and Adversarial Models 48

5.3 Energy Harvesting Motivated Attacks and Implications 50
5.4 The Proposed Detection Scheme 52
5.5 Performance Evaluation 56
5.6 Summary 60
6. Cooperative Detection Scheme 61
6.1 Introduction 61
6.2 System and Adversarial Models 63
6.3 Energy Harvesting Motivated Attack Scenarios and Analysis . 65
6.4 The Proposed Countermeasure 68
6.4.1 SlyDog: Inducement-based Detection 68
6.4.2 LazyDog: Monitor-based Detection 73
6.5 Analysis of The Proposed Countermeasure 74
6.5.1 Detection Rate of SlyDog 74
6.5.2 Detection Rate of LazyDog 78
6.6 Simulation Testbed 81
6.7 Simulation Results 82
6.8 Features and Constraints 87
6.9 Potential Enhancements 88
6.9.1 Dummy Packets 88
6.9.2 Bypass Retransmission 89
6.10 Applicability to Other Attacks 89
6.10.1 Limited Transmission Power Attack 89
6.10.2 Receiver Collisions Attack 89
6.11 Summary 90
7. Conclusion and Future Work 91
Bibliography 94
Biography 100

ABSTRACT

Internet-of-Things (IoT) and its applications are proliferating, in which a myriad of multi-scale sensors and heterogeneous devices (later in short, nodes) are seamlessly blended for a ubiquitous computing and communication infrastructure. In order to overcome limited battery power and extend the network lifetime, energy harvesting from ambient environment has been increasingly popular and playing an important role in realizing a self-sustainable network. Thus, energy harvesting motivated networks (EHNets) are rapidly emerging and becoming a major building block for diverse IoT applications. In EHNets, a link between two nodes may not be stable due to the variable transmission power levels based on non-uniform energy harvesting rates. Each node is also admittedly vulnerable to Denial-of-Service (DoS) attacks because of the lack of centralized coordination, physical protection, and security requirements in the network protocols.

The overall objective of this research is to design, analyze, and evaluate EHNets that can provide reliable, robust, and expected communication performance. We investigate four major research issues. First, light-weight forwarding protocols are proposed to reliably deliver sensory data over time-varying asymmetric links in EHNets. A weighted confirmation scheme, a lazy confirmation scheme, and an asymmetric link aware backoff mechanism are suggested. Second, we propose a light-weight countermeasure to a selective forwarding attack in resource-constrained wireless sensor networks (WSNs), where a randomly selected single checkpoint node is deployed to detect the forwarding misbehaviors. The proposed countermeasure is integrated with timeout and hop-by-hop retransmission techniques to efficiently cover unexpected packet losses due to either forwarding misbehavior or bad channel quality. Third, we propose a new countermeasure, called camouflage-based active detection, to a selective forwarding attack in EHNets. Four adversarial scenarios motivated by energy harvesting and potential forwarding vulnerabilities are also identified and analyzed. Finally, we further extend the camouflage-based active detection to monitor multiple malicious nodes and to detect the forwarding misbehaviors of lurking deep malicious nodes. This countermeasure consists of SlyDog and LazyDog schemes and cooperatively detects the forwarding misbehaviors. The advantages of these techniques are demonstrated through extensive simulation and mathematical analysis.

LIST OF TABLES

4.1 Simulation Parameters of SCAD 39
6.1 Simulation Parameters of EYES 81

LIST OF FIGURES

3.1 Neighbor lists with dual communication ranges. 14
3.2 Acknowledgment packet delivery ratio. 16
3.3 The proposed WCFM scheme. 17
3.4 The pseudo code of the WCFM scheme. 19
3.5 The proposed LCFM scheme. 20
3.6 The pseudo code of the LCFM scheme. 21
3.7 Snapshots of network topology. 22
3.8 Packet delivery ratio as a function of mean periods of energy harvesting and normal states. 23
3.9 Latency as a function of mean periods of energy harvesting and normal states. 24
3.10 Changes of received increment factor. 25
3.11 Changes of the DF ratio and backoff period. 26
4.1 A snapshot of network. 31
4.2 The timeout period increases as the number of hops from the sink increases. 33
4.3 The pseudo code of proposed SCAD detection scheme. 34
4.4 The false detection rate against the number of malicious nodes and channel error rates. 38
4.5 The detection rate against the number of malicious nodes. 40
4.6 The successful drop rate and packet delivery ratios against the number of malicious nodes. 41
4.7 The energy consumption against the number of malicious nodes and packet drop rate. 42
4.8 The number of forwarded and overheard packets against the number of malicious nodes. 42
4.9 The false detection rate against the number of malicious nodes and channel error rate. 43
5.1 The impact of uniform and exponential packet intervals. 49
5.2 A set of adversarial scenarios. 51

5.3 The changes of attack time ratios. 53
5.4 A snapshot of the proposed CAM scheme. 54
5.5 The pseudo code of CAM scheme. 55
5.6 The performance of detection rate and detection latency against energy harvest time. 57
5.7 The performance of PDR and packet buffered ratio against energy harvest time. 58
5.8 The performance of monitor probability and total active and harvest time periods against energy harvest time. 59
6.1 The impact of uniform and exponential *State* packet intervals and number of neighbor nodes. 64
6.2 A set of adversarial scenarios and its vulnerable cases in the presence of malicious nodes in EHNets. 66
6.3 The changes of attack time ratio and number of dropped packets against energy harvest rate and packet injection rate. 67
6.4 The pseudo code of legitimate node's operations. 69
6.5 The proposed *SlyDog* detection strategy. 70
6.6 The pseudo code of *SlyDog*. 72
6.7 The proposed *LazyDog* detection strategy. 73
6.8 The pseudo code of *LazyDog*. 75
6.9 The detection rate against monitor probability and the number of dropped packets. 80
6.10 The performance impact against energy harvest rate, number of malicious nodes, packet injection rate and δ. 83
6.11 The packet delivery ratio against energy harvest rate, packet injection rate and δ. 85
6.12 The energy consumption and monitor probability against energy harvest rate, packet injection rate and δ. 86
6.13 The performance of total active and harvest time periods against energy harvest rate and δ. 87

CHAPTER 1
INTRODUCTION

1.1 Challenges and Motivations

Internet-of-Things (IoT) and its applications are rapidly proliferating, where a myriad of multi-scale sensors and devices (later in short, nodes) are seamlessly blended for a ubiquitous computing and communication infrastructure [1]. Wireless Sensor Networks (WSNs) have been receiving a considerable attention as an alternative solution for scalable monitoring and data collection in a hostile or unattended area. A WSN consists of resource constrained sensor nodes in terms of sensing, computing, or communicating capability. As a part of rapidly emerging Internet of Things (IoT), WSNs will play an important role in building a ubiquitous computing and communication infrastructure. With the prevalence of cloud, social media, and wearable computing as well as the reduced cost of processing power, storage, and bandwidth, it is envisaged that wirelessly connected smart nodes and devices under IoT will enhance flexible information accessibility and availability as well as change our life further.

Nodes are resource constrained in terms of computing and battery-power, but are often required to operate a long-term sensing and communication in a hostile or unattended area, e.g., deployed as a mission-oriented network. As pointed out in [2], a TMote$^{\text{TM}}$ Sky node consumes 64.68 mW in a receive mode. Under the two standard 3,000 mAh AA batteries, if the node is highly utilized, network lifetime is only 5.8 days. Since wireless communication could be responsible for more than half of total energy consumption [3], a significant amount of effort has been devoted to develop energy efficient routing protocols in wireless sensor networks [4]. Due to the limited power in battery-powered WSNs, replacing or replenishing the batteries is ultimately unavoidable and it may be infeasible or even impossible in such a harsh environment. In order to remove batteries or at least reduce the frequency of replacing batteries, energy harvesting from an immediate environment (e.g., kinetic, wireless, solar, etc.) has been increasingly popular for IoT [5, 6, 7] and playing an important role in realizing self-sustainable nodes deployed in a large-scale network. It is also the fact that the U.S. Army will eventually eliminate all the military batteries or at least reduce the frequency of replacing batteries for communication devices [8].

Soldiers will be equipped with batteryless or self-powered communication devices in near future [9]. We envision that energy harvesting will play a pivotal role in making possible self-sustainable wireless devices ranging from nano-scale sensors to handheld mobile devices, and it will serve as a major building block for emerging Internet of Things (IoT) applications. Thus, a newly emerging energy harvesting motivated network (EHNet) foresees diverse applications in civilian and military environments, and will be a part of ubiquitous communication infrastructure [10].

With energy harvesting, sensor devices may contain a different amount of residual energy because of non-uniform energy harvesting rates in WSNs. Depending on the energy availability, nodes can deploy variable transmission power levels and thus, multiple communication ranges commonly exist in the network. For example, variable transmission power levels are easily witnessed in the CISCO Aironet 340 and 350 series and Wi-Fi networks [11] to provide customized services, where computation power, storage limit, and energy consumption are selectively considered. Note that multiple communication ranges can lead to *asymmetric links*. For example, if a node, n_a, can reach nodes, n_b and n_c, but both n_b and n_c or either n_b or n_c may not be able to reach n_a. Since each node can change its transmission power levels based on energy harvesting, a link between two nodes may not be stable. Thus, a route from a data source to a sink also may become unreliable in the presence of time-varying asymmetric links. A bidirectional routing [12] and a tier-based routing framework [13] deployed in asymmetric mobile ad hoc networks (MANETs) cannot directly be applied in resource constrained WSNs. A probabilistic routing [14] and a multiple range convergecast routing [15] have been proposed for heterogeneous WSNs. In these approaches, a small number of nodes is dedicated to communicate with the extended communication range, or each node is able to change its multiple transmission power levels anytime. However, time-varying communication ranges motivated by energy harvesting have not been well considered. To the best of our knowledge, little work has been devoted in a forwarding methodology in the realm of EHNets.

For routing, each node communicates with its neighbor nodes based on a broadcast-based forwarding, and collaboratively routes sensory data through a multi-hop relay. When a node intends to reply a unicast packet, unlike a wired network, all one hop neighbor nodes can still overhear the packet, as if it is a broadcast packet [16]. Since radio link is a shared medium and its radiation pattern is often omni-directional from antenna, it is inherently insecure and thus, adversaries can easily overhear, duplicate, corrupt, or alter data. Nodes deployed in such a hostile or unattended area can also be captured, tampered, or destroyed because they are physically insecure. For example, a malicious node compromised by an adversary can randomly or selectively drop any incoming packet to disrupt network protocols and interfere with on-going communications on purpose or strategically. Note that it is not trivial to differentiate such a misbehavior (or attack) from a temporal node failure or packet loss. Thus, WSNs and EHNets are vulnerable to a well-known denial-of-service (DoS) attack that primarily targets service availability by disrupting network routing protocols or interfering with on-going communications. Diverse countermeasures and their variants [17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30] have been proposed to avoid and/or detect a forwarding misbehavior under an implicit assumption of battery-powered networks, where conventional encryption algorithms and secure routing protocols cannot be directly applied. Unfortunately, forwarding misbehavior and its countermeasure are still under-explored in the realm of EHNets.

1.2 Contributions

To address these limitations and challenges, this dissertation research is to design, analyze, and evaluate EHNets that can provide reliable and robust communication performance. We investigate four major research issues. First, light-weight forwarding protocols are proposed to reliably deliver sensory data over time-varying asymmetric links in EHNets. A weighted confirmation scheme, a lazy confirmation scheme, and an asymmetric link aware backoff mechanism are suggested. Second, we propose a light-weight countermeasure to a selective forwarding attack in resource-constrained wireless sensor networks (WSNs), where a randomly selected single checkpoint node is deployed to detect forwarding misbehaviors. The proposed countermeasure is integrated with timeout and hop-by-hop retransmission techniques to efficiently cover unexpected packet losses due to the forwarding misbehavior or bad channel quality. Third, we propose a new countermeasure, called camouflage-based active detection, to a selective forwarding attack in EHNets. Four adversarial scenarios motivated by energy harvesting and their potential forwarding vulnerabilities are also analyzed.

Finally, we further extend the camouflage-based active detection to monitor multiple malicious nodes and to detect the forwarding misbehaviors of lurking deep malicious nodes. This countermeasure consists of SlyDog and LazyDog schemes and cooperatively detects the forwarding misbehavior. The advantages of these techniques are demonstrated through extensive simulation experiments and mathematical analysis. In summary, we make the following contributions in this dissertation research:

- We propose Weighted Confirmation (WCFM) and Lazy Confirmation (LCFM) schemes to reliably deliver sensory data to the sink. An Asymmetric Link Aware Backoff mechanism is also proposed to avoid packet contentions and collisions by considering the historical statistics of routing and number of neighbor nodes. We evaluate the proposed WCFM and LCFM schemes and their hybrid approach, called Hybrid Confirmation (HCFM), using OMNeT++. We modify a conventional explicit acknowledgment, called Conventional Ack (CAck) scheme, to work in EHNets for performance comparison.

- We propose a single checkpoint based countermeasure, called SCAD, in WSNs. Unlike prior detection schemes, where multiple checkpoint nodes are deployed, the SCAD deploys a single checkpoint-assisted approach and its security resiliency and communication performance are measured. The SCAD is also incorporated with timeout and hop-by-hop retransmission techniques to recover unexpected packet losses due to the forwarding misbehavior or bad channel quality. We propose a simple analytical model of the SCAD and show its numerical result in terms of false detection rate. We also revisit prior checkpoint-based and monitor-based detection approaches and modify them to work in WSNs for performance comparison. We develop a customized discrete-event simulation framework by using the OMNeT++ and evaluate its performance through extensive simulation experiments.

- We investigate four adversarial attack scenarios and analyze their potential forwarding behaviors in EHNets, where each node periodically switches its state between active and harvest. A set of vulnerable cases causing a forwarding misbehavior is identified. We propose a novel camouflage-based active detection scheme and its communication protocol in EHNets, where each node actively disguises itself as an energy harvesting node, monitors its adjacent nodes, and detects a lurking malicious node. We develop a customized simulation framework using OMNeT++, conduct a performance evaluation study in terms of six performance metrics, and show a viable approach to selective forwarding attack in EHNets.

- We investigate a set of adversarial scenarios and analyze its forwarding operations under the *charge-and-spend* harvesting policy in EHNets. Then we identify four vulnerable scenarios and their corresponding potential forwarding misbehaviors. We propose a cooperative countermeasure to efficiently detect the forwarding misbehavior in EHNets, called EYES, and it consists of two mechanisms: SlyDog and LazyDog. In the SlyDog, each node actively disguises itself as an energy harvesting node but in fact monitors its adjacent nodes to detect the forwarding misbehavior of lurking deep malicious nodes. In the LazyDog, however, each node periodically requests its adjacent nodes of a limited history of forwarding operations, and validates any prior uncertain forwarding operation to detect the forwarding misbehavior. We propose an analytical model of the EYES and show its numerical results in terms of detection rate. We also revisit prior detection approaches, Watchdog and HCD, and modify them to work in EHNets. Both single and two malicious nodes cases are applied to HCD and Watchdog, and no malicious node case is also considered as the performance upper bound of packet delivery ratio. In addition, detection strategies of forwarding misbehavior are comprehensively compared in terms of six properties. We conduct extensive simulation experiments using the OMNeT++ for performance comparison and analysis.

1.3 Organization

This dissertation paper is organized as follows. Chapter 1 introduces the challenges and motivations in energy harvesting motivated networks (EHNets) and discussed the overall contributions of the dissertation work. Chapter 2 briefly reviews essential concepts of existing routing protocols and countermeasures to forwarding misbehaviors in EHNets as the background of this study. Chapter 3 describes the proposed lightweight forwarding protocols to reliably deliver sensory data to a sink in the presence of time-varying asymmetric links in EHNets. The design, mathematical analysis, implementation and the experimental results of the proposed light-weight countermeasure to forwarding misbehaviors are presented in chapter 4. The proposed camouflage-based active detection scheme are presented in chapter 5. Chapter 6 describes the design, analytical model, and simulation experiments of the proposed cooperative countermeasure to forwarding misbehaviors. Finally, we conclude this dissertation study in chapter 7.

CHAPTER 2
BACKGROUND

In this chapter, we revisit and analyze prior energy harvesting aware routing protocols and countermeasures to forwarding misbehaviors in various networks.

2.1 Energy Harvesting Aware Routing Protocols

There are two routing paradigms when asymmetric links exist in the network. The first routing paradigm [31] is to avoid using long-range links whenever possible. Since nodes are conventionally powered by batteries, long-range links consuming more energy are not preferred. In this paradigm, the energy consumption can be reduced but the transmission delay can be increased. The second routing paradigm is to maximize the benefit of long-range links in terms of reducing the number of hops and the transmission delay [13]. Prudent power control mechanisms embedded in the link or network layer [32, 33] can reduce the energy consumption by controlling multiple power levels but asymmetric links could be created.

Several routing strategies [13, 12, 14, 15] have been proposed to efficiently deliver sensory data to a sink in the presence of asymmetric links. In [14], the proposed probabilistic routing protocol consists of two steps: searching reverse paths and selecting forwarding nodes. Similar to [12], a reverse path is searched by exchanging control messages. For example, both n_a and n_b exchange *Hello* and *Hello_Ack* messages to create a neighbor list. If n_b receives the *Hello* from n_a but does not receive the *Hello_Ack* for its own *Hello*, an asymmetric link exists between them. Then n_b broadcasts a *Find* message piggybacked with the maximum number of propagation hops and searches a path to n_a. Upon receiving the *Find*, n_a replies a *Path* message containing the reverse path. For routing, a forwarding node is selected based on a delivery probability, which is a historical statistics of routing maintained in each node. A tier-based routing framework [13] is proposed under the consideration of variation in transmission power levels and tries to find a symmetric link to the destination. Each node exchanges *route request (RREQ)* and *route reply (RREP)* messages during a route discovery procedure. A sender repeatedly sends a *RREQ* at different transmission power levels from the lowest to the current power level. Here, the transmission power level is piggybacked in the *RREQ*. Upon receiving the *RREQ*, a receiver replies a *RREP* only if the *RREQ's* transmission power level is less than or equal to its own transmission power level.

In spite of energy efficient routing techniques, the maintenance cost in terms of locating and replacing (or replenishing) batteries becomes non-negligible, and thus, replacing (or replenishing) batteries is ultimately unavoidable in WSNs. The number of studies [34, 35, 36] has been conducted with a set of rechargeable (or renewable) nodes in EHNets, where batteries are replenished by various environmental sources. A solar aware routing [34] is proposed in which sensory data are primarily forwarded to the nodes currently being powered by solar energy. In [35], two geographic routing protocols are proposed in the presence of lossy wireless channel and energy renewable nodes. A forwarding node is selected based on its location, residual energy, and potential energy harvesting rate. Thus, a node located in the shortest path is not simply considered as a forwarding candidate because it may be over utilized and can consume its energy quickly. Similarly a node containing more residual energy is not primarily considered as a forwarding candidate either, because the residual energy may not fully represent its energy availability.

In summary, relatively little effort has been made for developing an acknowledgment technique in EHNets, where time-varying asymmetric links become a major concern.

2.2 Countermeasure to Forwarding Misbehaviors

Both watchdog and pathrater techniques [17, 18] and their variants have been proposed in different networks. In this section, we newly categorize and analyze them in terms of monitor-, acknowledgment-, and inducement-based approaches.

Monitor-based Approach: Each node observes the network condition and communication activities, such as a channel condition, network traffic, or forwarding operation of its adjacent nodes, and checks if there is any abnormality. In the CAD [23], both channel estimation and traffic monitoring are conducted to identify a stand-alone attacker in wireless mesh networks (WMNs). Each node monitors the communication activities of its adjacent nodes by observing downstream and upstream network traffic and estimates packet loss rate. If a node shows higher monitored packet loss rate than the sum of estimated packet loss rate and its corresponding detection threshold, it is suspected as an attacker. In the SCM [24], neighbor nodes located along the routing path between packet sender and receiver become an observer and monitor the forwarding behavior. If one of observing nodes detects a forwarding misbehavior, it generates an *Alarm* packet which is propagated back to the packet source through the observing nodes. The CRS [27] is an extended version of CAD, where each node maintains a reputation table to evaluate the forwarding behavior of its adjacent nodes. This reputation value is calculated based on the deviation of the normal packet loss rate, due to the time- and location-variant channel quality and the link layer collisions, and monitored packet loss rate during a long term. The adjacent node is prosecuted as a malicious node if its reputation value is less than the predefined threshold value. [37] proposes a countermeasure to on-off attacks with selective forwarding, in which a forwarding misbehavior is seldom detected and is confused with a temporary error. Each node monitors the forwarding operation of its adjacent nodes and records good and bad behaviors based on a dynamic sliding window, respectively. This scheme can recognize a pattern of malicious node behavior and help to flexibly design a system that can accept a certain level of security risk based on the accumulated behaviors. In the HCD [38], each node records a limited set of traces about forwarding operations and exchanges it with its adjacent nodes to identify a forwarding misbehavior in EHNets. Each node can gradually reduce the forwarding probability of malicious node in order to exclude the malicious node from participating in the routing operation.

Acknowledgment-based Approach: The key operation is that the intermediate nodes located along the forwarding path between source and destination (e.g., sink) are responsible for monitoring the forwarding operation of its next node and sending an explicit message (i.e., *Ack* packet) to the source. In [20] and its extension CHEMAS [21], a set of checkpoint nodes is randomly selected from a source per packet basis and monitors the forwarding operation by replying an *Ack* packet to the source in WSNs. If an intermediate node located in the forwarding path does not receive the required number of *Ack* packets, it suspects the next located node in the path as a malicious node, generates an *Alarm* packet, and sends it back to the source. However, since multiple number of checkpoint nodes generate *Ack* packets, intermediate nodes may receive and forward the excessive number of packets and consume non-negligible amount of energy. In the [22], the 2ACK is proposed to detect misbehaving links in Mobile Ad-hoc Networks (MANETs), where each intermediate node located along the forwarding path generates an *Ack* packet, and forwards it to two-hop neighbor node in the opposite direction of the data traffic route after receiving the data packet. The destination node of *Ack* packet observes the behavior of link in front of *Ack* packet generator for a period of time. If the link shows a higher *Ack* packet loss ratio than a threshold value, this link is declared misbehaving and added to the blacklist of misbehaving links. In the [25], the traditional end-to-end acknowledgment scheme is conducted to deliver a data packet to the sink in order to reduce network overheard. If the source cannot receive *Ack* packet from the sink within the maximum delay, a secure *Ack* scheme, which is an improved version of [22], is initiated to identify the malicious node located along the forwarding path in MANETs. In the SCAD [30], a light-weight countermeasure is proposed to a selective forwarding attack in resource-constrained WSNs, where a randomly selected single checkpoint node along the forwarding path is deployed to detect forwarding misbehaviors. The proposed countermeasure is integrated with timeout and hop-by-hop retransmission techniques to efficiently cover unexpected packet losses due to the forwarding misbehavior or bad channel quality. The FADE [26] is a variant of the CAD and detects a collaborative selective forwarding attack in WMNs. After each node forwards a packet, it overhears a link-layer acknowledgment and waits for a two-hop acknowledgment from its downstream nodes. Each node also adds its opinion towards the downstream nodes to a separate monitoring packet originally sent from source.

Inducement-based Approach: The basic idea is that a piece of information is hidden or a fake information is utilized to lure the potential malicious nodes to show its possible forwarding misbehavior. The [28] proposes a cooperative bait detection scheme (CBDS) based on the dynamic source routing (DSR) to detect both selective forwarding and blackhole attacks in MANETs. The approach is that a source node selects an adjacent node and uses its address as a bait destination address to entice a malicious node to send back a forged or fake route reply (RREP) packet. Then the malicious node can be identified by using a reverse tracing technique. In the SNBDS [29], each node observes the difference between the sequence numbers of the received RREP packets and that of stored in the routing table based on the ad hoc on-demand distance vector routing (AODV) to detect the next hop located node in MANETs. If the maximum difference is larger than the predefined threshold value, the next node is added to a suspicious node table for malicious node discovery and verification process by using fictitious destination address and destination sequence number. In the CAM [39], each node hides its current operational status and pretends not to overhear the on-going communications, but in fact monitors the forwarding operations of its adjacent nodes to detect a deep lurking malicious node in EHNets. Since malicious nodes are seldom in harvest state and can selectively drop any incoming packet in a short period of time, it is not trivial to detect the forwarding misbehavior.

In summary, most prior countermeasures rely on implicit overhearing that requires nodes to stay in active state for an extended period or depend on explicit acknowledgment that expects the intermediate nodes to generate and forward a large number of packets (e.g., *Ack* packet), resulting in additional energy consumption in battery-supported networks. However, little attention has been paid for self-sustainable nodes in the realm of EHNets, where each node repeatedly changes its state between active and harvest.

CHAPTER 3
LIGHT-WEIGHT FORWARDING PROTOCOLS

In this chapter, we propose two light-weight forwarding protocols, *Weighted Confirmation* (WCFM) and *Lazy Confirmation* (LCFM) schemes, to reliably deliver sensory data to a sink in the presence of time-varying asymmetric links in EHNets.

3.1 Introduction

Wireless Sensor Networks (WSNs) often require long-term sensing/communicating operations on the order of days or even weeks in a hostile and unattended area, e.g., deployed as a mission-oriented network. As pointed out in [2], a TMote$^{\text{TM}}$ Sky node consumes 64.68 mW in a receive mode. Under the two standard 3,000 mAh AA batteries, if the node is highly utilized, network lifetime is only 5.8 days. In battery-powered WSNs, replacing or replenishing the batteries is ultimately unavoidable and it may be infeasible or even impossible in such a harsh environment. Due to the limited battery power, therefore, WSNs powered by diverse environmental sources (i.e., solar, vibration, wind, thermal, etc.) have been widely investigated. This research is also motivated by the fact that the U.S. Army will eventually eliminate all the military batteries or at least reduce the frequency of replacing batteries for communication devices [8]. Soldiers will be equipped with batteryless or self-powered communication devices in near future.

With energy harvesting, sensor devices (later nodes) may contain a different amount of residual energy because of non-uniform energy harvesting rates in WSNs. Depending on the energy availability, nodes can deploy variable transmission power levels and thus, multiple communication ranges commonly exist in the network. For example, variable transmission power levels are easily witnessed in the CISCO Aironet 340 and 350 series and Wi-Fi networks [11] to provide customized services, where computation power, storage limit, and energy consumption are selectively considered. Note that multiple communication ranges can lead to *asymmetric links*. For example, if a node, n_a, can reach nodes, n_b and n_c, but both n_b and n_c or either n_b or n_c may not be able to reach n_a.

Since each node can change its transmission power levels based on energy harvesting, a link between two nodes may not be stable. Thus, a route from a data source to a sink also may become unreliable in the presence of time-varying asymmetric links. To the best of our knowledge, little work has been devoted in a forwarding methodology in the realm of EHNets.

In this research, we propose light-weight forwarding protocols to reliably deliver sensory data to a sink in the presence of time-varying asymmetric links in EHNets. Our contributions are three-fold:

- First, we propose *Weighted Confirmation* (WCFM) and *Lazy Confirmation* (LCFM) schemes to reliably deliver sensory data to the sink. The WCFM scheme differentiates multiple paths between a data source and a sink by assigning multiplicative weights on the paths. In the LCFM scheme, nodes assure a reverse path by waiting for the extended communication range.

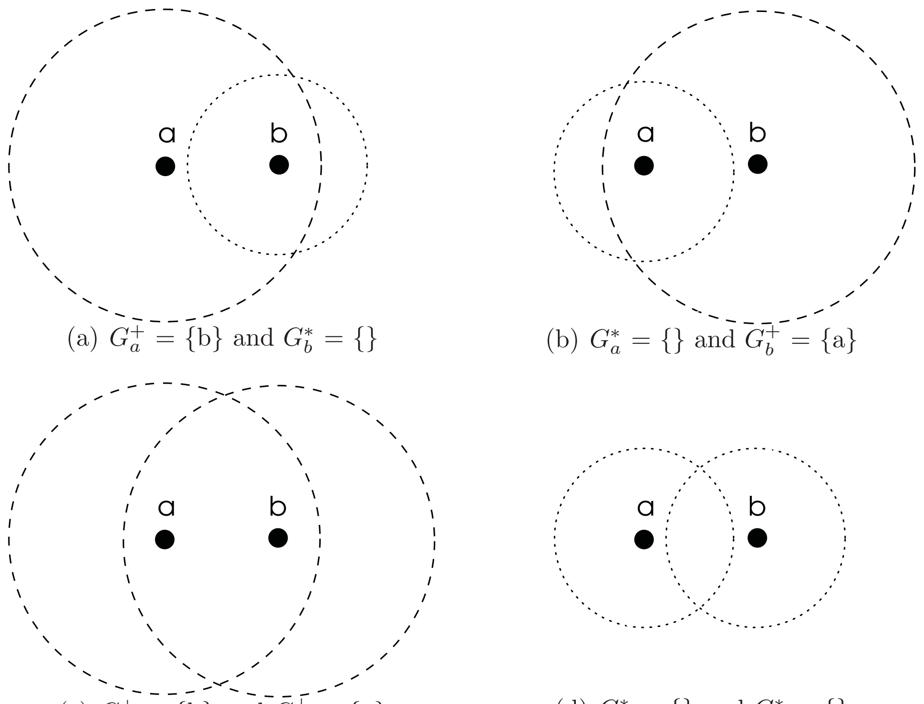
- Second, an *Asymmetric Link Aware Backoff* mechanism is also proposed to avoid packet contentions and collisions by considering the historical statistics of routing and number of neighbor nodes.

- Third, we evaluate the proposed WCFM and LCFM schemes and their hybrid approach, called *Hybrid Confirmation* (HCFM), using OMNeT++. We modify a conventional explicit acknowledgment, called *Conventional Ack* (CAck) scheme, to work in EHNets for performance comparison.

The WCFM, LCFM, and HCFM schemes show higher packet delivery ratio but keep lower latency compared with the CAck scheme. Overall simulation results indicate that the proposed forwarding protocols is a viable approach for reliable asymmetric routing in EHNets.

3.2 System Model

In this research, energy harvesting is modeled by a two-state Markov process with harvest (S_{hv}) and normal (S_{nr}) states. In S_{hv} and S_{nr} states, nodes operate in the extended and normal communication ranges, respectively. A node stays in S_{nr} state for a random amount of time, which is exponentially distributed with a mean λ_{nr}, and changes its state into S_{hv} state. After energy harvesting for some amount of time



(a) $G_a^+ = \{\text{b}\}$ and $G_b^* = \{\}$

(b) $G_a^* = \{\}$ and $G_b^+ = \{\text{a}\}$

(c) $G_a^+ = \{\text{b}\}$ and $G_b^+ = \{\text{a}\}$

(d) $G_a^* = \{\}$ and $G_b^* = \{\}$

Figure 3.1. Neighbor lists with dual communication ranges.

in S_{hv} state, which is also assumed to be exponentially distributed with a mean λ_{hv}, the node changes its state back to S_{nr} state. Both S_{nr} and S_{hv} states are repeated. Upon energy harvesting, each node is able to operate in higher transmission power level to extend its current normal communication range.

Due to multiple communication ranges, each node can have a different set of neighbor nodes. We consider dual communication ranges for the sake of simplicity and categorize node adjacency into four cases as shown in Fig. 3.1. Here, normal and extended communication ranges are marked as dotted and dashed lines, respectively. Each node exchanges an one-hop *Hello* packet, overhears bypassing packets, and maintains a neighbor list, G. The list consists of a set of neighbor nodes reachable with either normal communication range (G^*) or extended communication range (G^+), respectively. For example, in Subfig. 3.1(a), n_a and n_b operate in the extended and normal communication ranges, respectively. n_a can communicate with n_b but n_b cannot. Similarly, in Subfig. 3.1(b), n_b can communicate with n_a but n_a cannot. Both

n_a and n_b can communicate each other with the extended communication range in Subfig. 3.1(c). In Subfig. 3.1(d), both n_a and n_b are not located within their normal communication range.

3.3 Detail Operations

The proposed forwarding protocols consist of three major operations: broadcast-based forwarding, routing history update, and asymmetric link aware backoff. A basic idea is that each node forwards sensory data to the selected node based on its historical statistics of routing. Since a route from a data source to a sink is unreliable in the presence of time-varying asymmetric links, we do not maintain and update a routing table.

First, a simple broadcast-based forwarding is deployed to avoid the exchange of control packets and reliably deliver a data packet through multiple paths. Each node re-broadcasts the received data packet only if it has been forwarded from the node located further from the sink in terms of number of hops. Here, a sink floods an one-time *Hop* packet piggybacked with the number of hops (h, initially set by 0) to the rest of nodes at the initial network setup. When a node receives *Hop* packet, it increments h by one, stores the updated h in a local storage, and rebroadcasts the packet piggybacked with the updated h. When a node receives the *Hop* packet containing higher number of hops, h', it replaces h' with the stored h and rebroadcasts the packet. Thus, each node is aware of how many hops away from the sink.

Second, each node maintains a historical statistics of routing by updating a ratio of the number of delivered packets to the sink (d) to the number of forwarded packets (f), $DF = \frac{d}{f}$. When a node forwards a data packet, it increments the number of forwarded packets by different values (i.e., 1, 0.6, or 0.4). When the sink receives a data packet, it replies a confirmation (*Cfm*) packet, which is relayed back to a source node. When a node receives a *Cfm* packet, it increments the number of delivered packets. If a node has higher DF, it has frequently and successfully delivered data packets to the sink. Unlike prior approach [12, 14], each node does not actively find a reverse path using additional control packets in EHNets. Note that a *Cfm* packet is relayed back to a source node through the intermediate nodes located along the path in best efforts. This is because of time-varying asymmetric links that incur

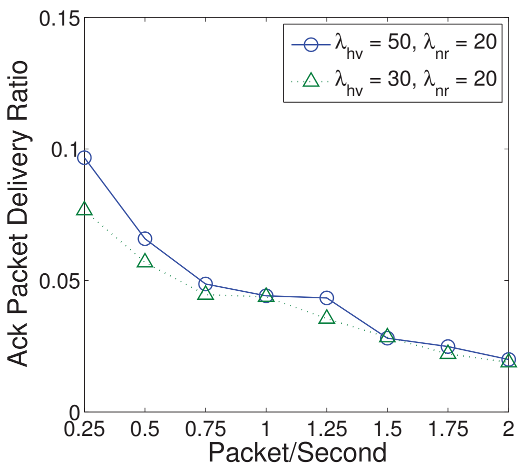


Figure 3.2. Acknowledgment packet delivery ratio.

frequent link disconnections. This approach is different from a conventional explicit acknowledgment scheme for the purpose of reliable routing, where a sink replies an acknowledgment (*Ack*) packet back to a data source if a data packet is successfully received. We observe that replying an *Ack* packet back to a data source is not very efficient in terms of *Ack* packet delivery ratio in EHNets, which supports our approach. As shown in Fig. 3.2, *Ack* packet delivery ratios against different energy harvesting periods, harvest (S_{hv}) and normal (S_{nr}) states, are quite low because of time-varying asymmetric links. Since the data source frequently experiences timeouts and executes retransmissions, a large number of data packets are lost. In this research, we do not consider an implicit acknowledgment scheme based overhearing [16], because the radio should be kept active, resulting in a non-negligible energy consumption.

Due to the multiple paths, the sink may receive the same data packet from a data source multiple times. Upon receiving a data packet, the sink determines whether it already has received the packet routed with the same path. If not, the sink replies a *Cfm* packet. The sink also replies to the later arriving data packets routed with different paths. The sink accepts upto three duplicated data packets routed with different paths. Whenever the sink replies a *Cfm* packet, it piggybacks an increment factor (Δ, initially set by a value 1 (Δ_1)) into the *Cfm* packet. In this research, we propose a *Weighted Confirmation* (WCFM) scheme. Whenever the sink repeatedly replies a *Cfm* packet for the same data packet, it reduces the increment factor, i.e., Δ_1,

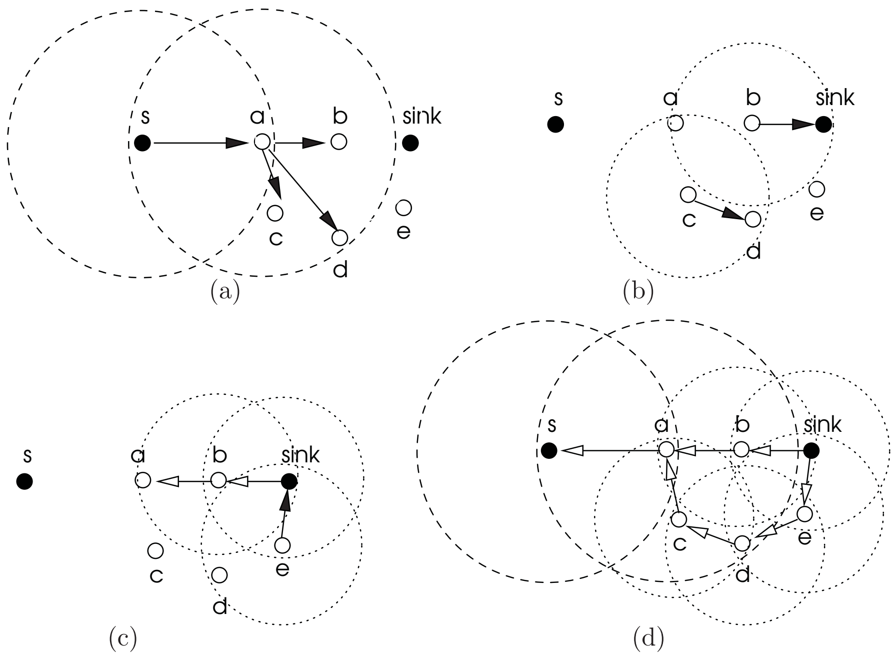


Figure 3.3. The proposed WCFM scheme.

$\Delta_{0.6}$, and $\Delta_{0.4}$. When a node receives a *Cfm* packet, it adds the piggybacked increment factor to the current number of delivered packets (d). Thus, the DF increases with different increment factors. Δ_1 is assigned to the first arriving data packet, because it is expected that the packet has been routed through the shortest path or the path with higher DF. Intermediate nodes can forward the *Cfm* packet at most three times and adjust their number of forwarded packets accordingly. Here, we multiplicatively adjust the increment factor to clearly see the effect of the WCFM scheme on the performance. The rationale behind this approach is to have nodes with higher DF involve in the routing operation frequently and deliver data packets reliably. Due to the communication overhead, this approach limits the number of multiple paths by deploying three increment factors.

For example, n_s initially generates a data packet and sends it toward a sink in Subfig. 3.3(a). Here, both data and *Cfm* packets are forwarded to the directions where black and white arrows indicate, respectively. Both n_s and n_a operate in an extended communication range and the rest of nodes operate in a normal communication range.

n_b, n_c, and n_d receive the forwarded data packet from n_a. Although n_c and n_d receive the packet simultaneously, let say, n_c forwards it. In Subfig. 3.3(b), n_b and n_c forward the packet and the sink receives the packet from n_b. The sink replies a *Cfm* packet piggybacked with the increment factor (i.e., Δ_1) to n_b and receives the same data packet from n_e, as shown in Subfig. 3.3(c). The sink also replies another *Cfm* packet piggybacked with a reduced increment factor (i.e., $\Delta_{0.6}$) for the later arriving packet. Multiple *Cfm* packets are sent back to the data source, n_s, through multi-hop relay as shown in Subfig. 3.3(d). All the intermediate nodes located along to the path between n_s and the sink update their DF based on the increment factors piggybacked in the *Cfm* packets. The pseudo code of major operations in the WCFM scheme is summarized in Fig. 3.4.

If n_a shrinks back to a normal communication range, its reverse link to n_s will be disconnected as shown in Subfig. 3.5(a). To support this, we propose a *Lazy Confirmation* (LCFM) scheme, where n_a does not search a reverse path to n_s but buffers any incoming *Cfm* packets. Then when n_a operates in an extended communication range, it forwards the buffered *Cfm* packets to n_s, as shown in Subfig. 3.5(b). A basic idea of the LCFM scheme is that nodes conservatively forward both data and *Cfm* packets only when their reverse path is available. In contrast to the WCFM scheme, the LCFM scheme does not adjust increment factor but always piggybacks Δ_1 to the *Cfm* packet. The pseudo code of major operations in the LCFM scheme is summarized in Fig. 3.6.

Third, we deploy a simple CSMA/CA MAC protocol for the link layer and propose an *Asymmetric Link Aware Backoff* mechanism. Whenever a node receives a data packet, it executes a backoff procedure before forwarding the packet to avoid possible packet contentions and collisions. A basic idea is that a node containing higher DF has lower backoff period because of its successful history of data deliveries. Also a node operating in an extended communication range has lower backoff period because of its potential to reduce the transmission latency by shortening the number of hops to the sink. To calculate a backoff period, we consider both the DF and the number of neighbors (i.e., $|G^*|$ or $|G^+|$). For example, when a node n_i receives a data packet,

```
Notations:
• $DF_i$, $d_i$, $f_i$: defined before.
• $\Delta_w$, $|\Delta|$: An increment factor ($w$ is 1, 0.6, or 0.4) and its number of increment factors, which is three.
• $pkt[type, src, seq]$: A packet is originally sent from a source node, $n_{src}$, with a sequence number, $seq$. Here, $type$ is either $Data$ or $Cfm$.
• $Q_i[pkt[seq]]$: A queue of received packets in $n_i$.
• $C_i[pkt[seq]]$: A counter of received the same packets in $n_i$.
◇ When a sink, $n_{sink}$, receives a $pkt[Data, s, seq]$,
    if $pkt[seq] \notin Q_{sink}$
        Enqueue the $pkt[seq]$ into $Q_{sink}$, and $C_{sink}[pkt[seq]]$ ++;
        Reply the $pkt[Cmf, sink, seq]$ with $\Delta_{1.0}$;
    else
        $C_{sink}[pkt[seq]]$ ++;
        if $C_{sink}[pkt[seq]] == |\Delta| - 1$
            Reply the $pkt[Cmf, sink, seq]$ with $\Delta_{0.6}$;
        else if $C_{sink}[pkt[seq]] == |\Delta|$
            Reply the $pkt[Cmf, sink, seq]$ with $\Delta_{0.4}$;
        else
            Discard the $pkt$;
◇ When a node, $n_i$, receives a $pkt[type, s, seq]$,
    if $pkt[type] == Data$
        if $pkt[seq] \notin Q_i$
            $C_i[pkt[seq]]$ ++;
            Enqueue the $pkt[seq]$ into $Q_i$;
            $f_i$ ++ and update $DF_i$;
        else
            $C_i[pkt[seq]]$ ++;
            if $C_i[pkt[seq]] == |\Delta| - 1$
                $f_i$ += $\Delta_{0.6}$ and update $DF_i$;
            else if $C_i[pkt[seq]] == |\Delta|$
                $f_i$ += $\Delta_{0.4}$ and update $DF_i$;
            else
                Discard the $pkt$ and return;
        $t_i^{boff}$ = Minimum($\frac{f_i}{|G_i| \cdot d_i} \cdot cw + \delta_i, cw) \cdot t_s$; /* Eq. 3.1 */
        if overhear a $pkt'[Data, k, seq]$ during $t_i^{boff}$, $k \in G_i^*$
            Discard the $pkt$;
        else
            Re-broadcast the $pkt$;
    else /* $pkt[type] == Cfm$ */
        $d_i$ += $\Delta_w$, and update $DF_i$;
        Unicast the $pkt$ after $t_i^{boff}$;
```

Figure 3.4. The pseudo code of the WCFM scheme.

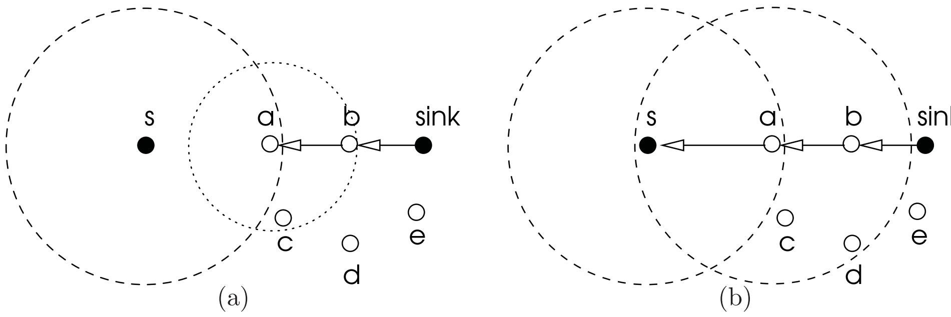


Figure 3.5. The proposed LCFM scheme.

its backoff period is expressed as,

$$t_i^{boff} = \text{Minimum}(\frac{f_i}{|G_i| \cdot d_i} \cdot CW + \delta_i, CW) \cdot t_s, \tag{3.1}$$

where $|G_i|$ becomes either $|G_i^*|$ or $|G_i^+|$ depending on the current transmission power level. Here, $|G_i^+| \geq |G_i^*|$. Also δ_i becomes Uniform$(0, |G_i|)$. In case of $G_i^+ = \{\}$, which means $|G_i^*| = 0$ and $|G_i| = 0$, we replace $|G_i|$ with 1. A small contention window (CW) value (i.e., 32 slots) is used and each slot is 400 μsecs, similar to [40], t_s. If a node overhears a packet being routed during the backoff period, it aborts the backoff procedure and discards the received packet. Upon the backoff expire, if the node does not overhear a packet, it forwards the received packet.

3.4 Simulation Testbed

We develop a customized discrete-event driven simulator using OMNeT++ [41] to conduct our experiments. A 250×250 m^2 rectangular network area is considered, where 140 nodes are randomly distributed in the network. An initial network topology is set in Subfig. 3.7(a), and it changes over simulation time due to the time-varying asymmetric links in Subfig. 3.7(b). The radio model simulates CC2420 with a nominal data rate of 250 Kbps [42]. The radio propagation model is based on the free-space model. A single node generates data traffic with 0.25 to 2 packet injection rates and the data packet size is 1 KByte. The periods of energy harvesting and normal states are assumed to be exponentially distributed with mean λ_{hv} (50 and 30 seconds) and λ_{nr} (20 second), respectively. Depending on the state, normal and extended

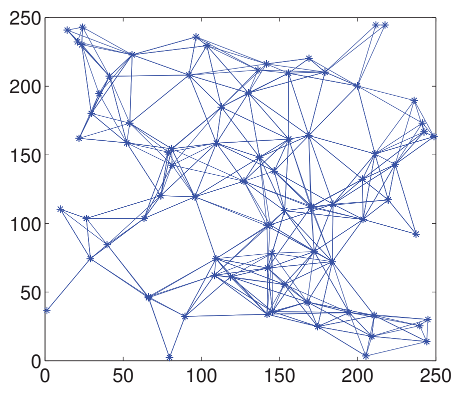
```
Notations:
• $B_i[pkt]$: A buffer of received packets in $n_i$.
◇ When a sink, $n_{sink}$, receives a $pkt[Data, s, seq]$,
    if $pkt[seq] \notin Q_{sink}$
        Enqueue the $pkt[seq]$ into $Q_{sink}$;
        Reply the $pkt[Cmf, sink, seq]$;
    else
        Discard the $pkt$;
◇ When a node, $n_i$, receives $pkt[Data, sink, seq]$,
    if $pkt[seq] \notin Q_i$
        Enqueue the $pkt[seq]$ into $Q_i$;
        $f_i$ ++ and update $DF_i$;
        $t_i^{boff}$ = Minimum($\frac{f_i}{|G_i| \cdot d_i} \cdot cw + \delta_i$, $cw$) $\cdot t_s$;
        if overhear a $pkt'[Data, k, seq]$ during $t_i^{boff}$, $k \in G_i^*$
            Discard the $pkt$;
        else
            Re-broadcast the $pkt$;
    else
        Discard the $pkt$;
◇ When a node, $n_i$, receives $pkt[Cfm, sink, seq]$,
    if $pkt[seq] \notin Q_i$
        Enqueue the $pkt[seq]$ into $Q_i$;
        $d_i$ += $\Delta_1$ and update $DF_i$;
        if a reverse path to $n_j$ is available, $n_j \in G_i^*$
            Unicast the $pkt$ to $n_j$ after $t_i^{boff}$;
        else
            Enqueue the $pkt$ into $B_i$;
            Unicast the $pkt$ to $n_j$ after $t_i^{boff}$, when the reverse path is available;
    else
        Discard the $pkt$;
```

Figure 3.6. The pseudo code of the LCFM scheme.

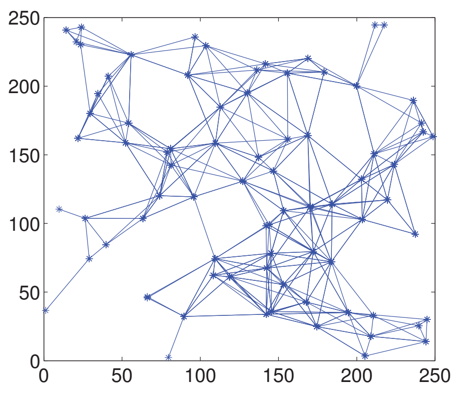
communication ranges are 40.8 m and 52 m. The total simulation time is 1,000 seconds.

3.5 Simulation Results

We vary the key simulation parameters: packet injection rate and period of energy harvesting and normal states. Combinations of the simulation parameters are used to conduct extensive performance evaluation studies. Five performance metrics are measured: packet delivery ratio (PDR), latency, and changes of increment factors, DF, and backoff period. For performance comparison, we modify a conventional



(a) Initial network topology

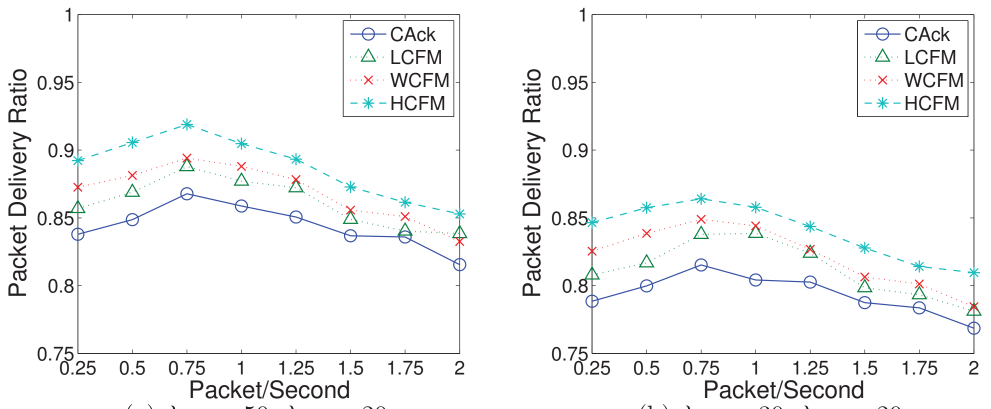


(b) After 500 seconds

Figure 3.7. Snapshots of network topology.

explicit acknowledgment mechanism to work in EHNets, called *Conventional Ack* (CAck) scheme as a base case. In the CAck scheme, a sink replies an *Ack* packet only to the first arriving data packet with the increment factor, Δ_1. The intermediate nodes located along the path relay the *Ack* packet back to a data source after a random backoff. For the sake of simplicity, we do not consider a timeout mechanism for retransmission in the data source in this research. Based on the proposed WCFM and LCFM schemes, we also propose a hybrid approach by combining the weighted factor and reverse path, called *Hybrid Confirmation* (HCFM) scheme. In the HCFM scheme, the sink replies multiple *Cfm* packets piggybacked with multiplicative weights to the later arriving data packets. The intermediate nodes located along the path to the data source relay the *Cfm* packet after the asymmetric link aware backoff. They buffer any incoming *Cfm* packet, if the reverse path is not available.

Packet Delivery Ratio: Fig. 3.8 shows the PDR of four different schemes with varying packet injection rates and periods of energy harvesting and normal states in time-varying network topologies (see Fig. 3.7). Under longer energy harvesting period, as shown in Subfig. 3.8(a), higher PDR is achieved because more nodes operate in an extended communication range. Thus, each node is less likely disconnected with its neighbor nodes. The proposed WCFM, LCFM, and HCFM schemes show higher PDF than that of the CAck scheme. This is because of multiple *Cfm* packets with multiplicative increment factors and buffering any incoming *Cfm* packet, if the

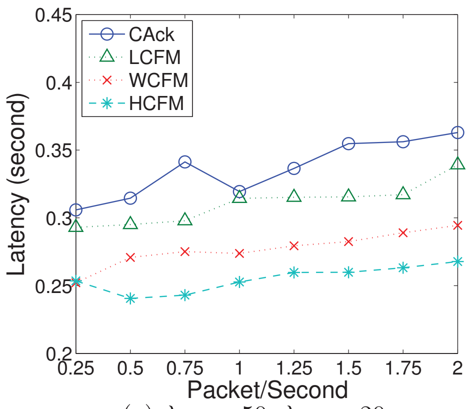


(a) $\lambda_{hv} = 50$, $\lambda_{nr} = 20$ (b) $\lambda_{hv} = 30$, $\lambda_{nr} = 20$

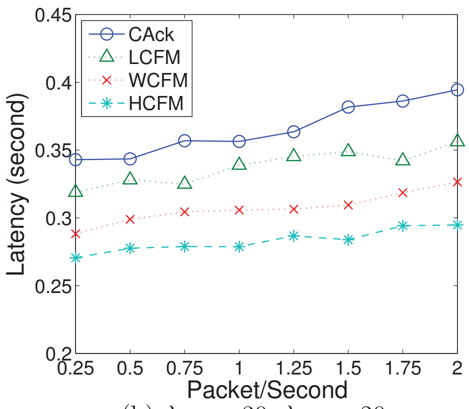
Figure 3.8. Packet delivery ratio as a function of mean periods of energy harvesting and normal states.

reverse path is not available, positively affects the PDR. The HCFM scheme shows the highest PDR for entire packet injection rates because it can identify and deploy multiple reliable paths based on the DF. However, the CAck scheme shows the lowest PDR for entire packet injection rates because data packets are routed through a single path, which is a time-varying asymmetric link and becomes unreliable. In Subfig. 3.8(b), overall PDRs decrease and performance saturation is delayed to 1.0 packet injection rate under shorter energy harvesting period.

Latency: Fig. 3.9 shows the latency of four different schemes. In Subfig. 3.9(a), the HCFM scheme shows the lowest latency for entire packet injection rates because data packets can be delivered reliably through multiple paths based on the DF. Multiple *Cfm* packets with extended communication range can increase the DF, reduce the backoff period, and identify the best path to the sink. Thus, the lowest latency can be achieved. Compared with the LCFM scheme, the WCFM scheme shows shorter latency because multiple *Cfm* packets can provide higher DF value that can lead to shorter backoff period. The CAck scheme shows the highest latency because of a blind random backoff period without considering asymmetric links. In Subfig. 3.9(b), four schemes show a similar pattern of the latency but higher latency is observed compared with the longer harvest period. Due to the short period of extended communication range, more link disconnections and longer waiting time of reverse paths



(a) $\lambda_{hv} = 50$, $\lambda_{nr} = 20$

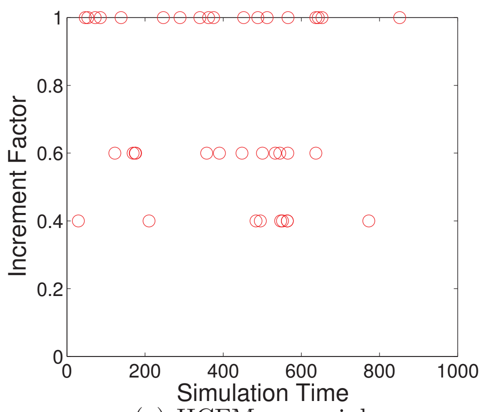


(b) $\lambda_{hv} = 30$, $\lambda_{nr} = 20$

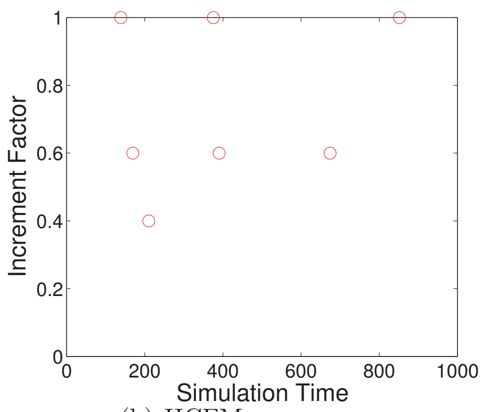
Figure 3.9. Latency as a function of mean periods of energy harvesting and normal states.

are expected.

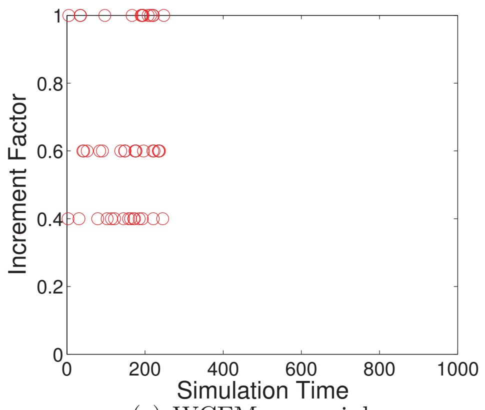
Increment Factor: We randomly select the nodes located near to the sink and data source to trace the values of increment factors, piggybacked in *Cfm* packets, in the HCFM and WCFM schemes. Here, $\lambda_{hv} = 50$, $\lambda_{nr} = 20$, and packet injection rate = 1 packet/second. In Subfig. 3.10(a), the node receives many increment factors in the HCFM scheme. The node can receive multiple *Cfm* packets with different increment factors because the sink replies a *Cfm* packet to the later arriving data packet. More number of Δ_1 than $\Delta_{0.6}$ and $\Delta_{0.4}$ are observed because *Cfm* packets for later arriving data packets are routed through a longer path or less reliable path. Since the node's DF increases, it is more frequently involved in the forwarding and thus, more number of *Cfm* packets are received. In Subfig. 3.10(b), however, the node has not been involved in the forwarding and receives very few *Cfm* packets. Subfig. 3.10(c) shows the effect of time-varying asymmetric links in the WCFM scheme. The node has been actively involved in the forwarding but in later it is removed from the path due to the asymmetric links. Unlike to the LCFM and HCFM schemes, where any incoming *Cfm* packets are buffered, the node can lose *Cfm* packets in the WCFM scheme. Subfig. 3.10(d) shows that the node almost does not receive *Cfm* packets and thus, it is rarely considered as a forwarding node.



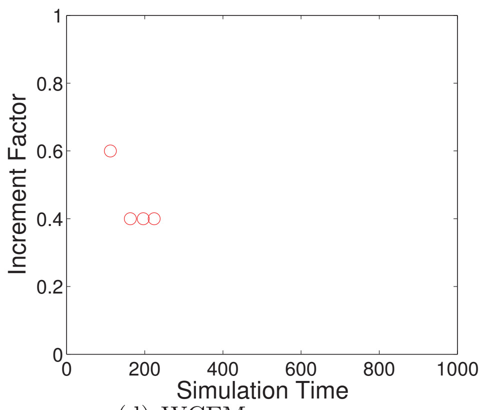
(a) HCFM, near sink



(b) HCFM, near source



(c) WCFM, near sink



(d) WCFM, near source

Figure 3.10. Changes of received increment factor.

DF and Backoff Period: In Subfig. 3.11(a), we observe the changes of the DF in the WCFM, LCFM, and HCFM schemes. Here, $\lambda_{hv} = 50$, $\lambda_{nr} = 20$, and packet injection rate = 1 packet/second. The HCFM scheme shows higher DF than that of other schemes because the sink replies multiple *Cfm* packets, which can also be buffered. Thus, more intermediate nodes can update their DF, i.e., increment the number of forwarded packets. The LCFM scheme shows the lowest DF because the sink replies only to the first arriving data packet with the increment factor of Δ_1. The WCFM scheme shows higher DF than that of the LCFM scheme because of multiple *Cfm* packets with different factors. In Subfigs. 3.11(b), (c), and (d), we compare the backoff periods of three schemes. Since the backoff period is based on the

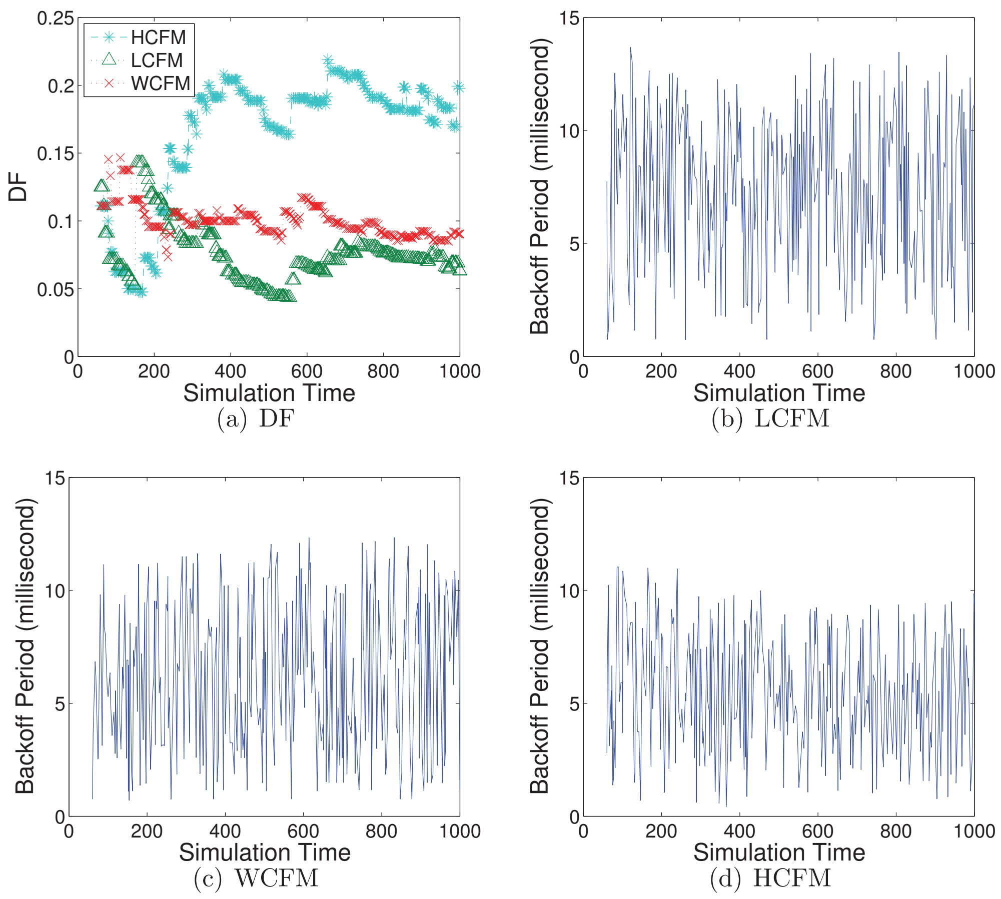


Figure 3.11. Changes of the DF ratio and backoff period.

DF and the number of neighbor nodes, the HCFM scheme shows the lowest backoff period compared with other two schemes. The LCFM scheme shows the highest and highly fluctuated backoff period. Note that the average backoff periods of the LCFM, WCFM, and HCFM schemes are 7.3619 msec, 6.8976 msec, and 5.2905 msec, respectively.

3.6 Summary

In this research, we investigated light-weight forwarding protocols in the presence of time-varying asymmetric links in EHNets. We proposed weighted and lazy route confirmation schemes and an asymmetric link aware backoff mechanism to reliably deliver sensory data. We evaluated their performance through extensive simulation experiments, compared them with an modified conventional explicit acknowledgment scheme, and showed that the proposed forwarding protocols is a viable approach in EHNets.

To see the full potential of the proposed techniques, we relax our assumption on energy harvesting from environmental sources in WSNs. We implicitly assumed that each node uniformly harvests energy and extends its communication range in the network. We are currently investigating a piezoelectric (later piezo) based energy harvesting from ambient vibrations [43] in a mobile tactical network, where only actively moving nodes harvest energy and communicate with an extended communication range. For example, each soldier equipped with a piezo-based energy harvesting kit in his/her shoes moves according to a tactical maneuver within the network and disseminates captured information with other soldiers. We envision that the proposed forwarding protocols can be integrated for reliable data dissemination in a mobile tactical network.

CHAPTER 4
LIGHT-WEIGHT COUNTERMEASURE TO FORWARDING MISBEHAVIORS

In this chapter, we investigate a selective forwarding attack and propose a light-weight detection scheme to forwarding misbehavior in WSNs.

4.1 Introduction

Wireless sensor networks (WSNs) have been receiving a considerable attention as an alternative solution for scalable monitoring and data collection in a hostile or unattended area. A WSN consists of resource constrained sensor nodes (later nodes) in terms of sensing, computing, or communicating capability. As a part of rapidly emerging Internet of Things (IoT), where a myriad of multi-scale nodes and devices are seamlessly blended, WSNs will play an important role in building a ubiquitous computing and communication infrastructure. With the prevalence of cloud, social media, and wearable computing as well as the reduced cost of processing power, storage, and bandwidth, it is envisaged that wirelessly connected smart nodes and devices under IoT will enhance flexible information accessibility and availability as well as change our life further.

Due to the harsh environmental conditions of deployment and the lack of physical protection, however, nodes can be easily captured, tampered, or destroyed by an adversary in WSNs. An open nature of wireless communication can also enable the adversary to overhear, duplicate, corrupt, or alter sensory data. In addition, most conventional network routing protocols are not originally designed to consider the security requirements for malicious attacks. Thus, WSNs are vulnerable to a well-known denial-of-service (DoS) attack that primarily targets service availability by disrupting network routing protocols or interfering with on-going communications.

In this research, we investigate a selective forwarding attack and propose its countermeasure in multi-hop WSNs, where a single or multiple malicious nodes randomly or strategically drop any incoming packet. The selective forwarding attack primarily targets the network routing vulnerabilities of multi-hop WSNs by violating an implicit assumption, i.e., all nodes faithfully and collaboratively route packets to a sink. Unlike a blackhole attack [44], where a malicious node blindly drops any incoming packet, it is a non-trivial problem to detect the forwarding misbehavior from temporal node failures or packet collisions. In light of these, we propose a light-weight countermeasure and its corresponding techniques to energy efficiently detect the selective forwarding attack, and measure its security resiliency and performance tradeoff through an analytical model and extensive simulation experiments. Our major contribution is briefly summarized in two-fold:

- First, we propose a single checkpoint based countermeasure, called *SCAD*, in WSNs. Unlike prior detection schemes [20, 21, 45, 23, 26], where multiple checkpoint nodes are deployed, the SCAD deploys a single checkpoint-assisted approach and its security resiliency and communication performance are measured. The SCAD is also incorporated with timeout and hop-by-hop retransmission techniques to recover unexpected packet losses due to the forwarding misbehavior or bad channel quality.
- Second, we propose a simple analytical model of the SCAD and show its numerical result in terms of false detection rate. We also revisit prior checkpoint-based and monitor-based detection approaches, CHEMAS [21] and CAD [23], and modify them to work in WSNs for performance comparison.

We develop a customized discrete-event simulation framework by using the OMNeT++ [41] and evaluate its performance through extensive simulation experiments in terms of detection rate, successful drop rate, packet delivery ratio, energy consumption, number of forwarded and overheard packets, and false detection rate. The simulation results indicate that the proposed countermeasure is a viable detection approach to a selective forwarding attack.

4.2 System and Adversary Models

When a node detects an event, it becomes a source node, generates a data packet, and forwards the packet towards a sink in WSNs. To deliver the data packet toward the sink, a simple broadcast-based forwarding [46], directed diffusion [47], or geographic-based routing [48] techniques can be deployed. Each node is aware of its one-hop neighbor nodes by exchanging a one-time single-hop *Hello* packet piggybacked with its node *id* [46]. We assume that the network is dense enough to find multiple forwarding candidate nodes. Thus, a single node connecting two sub-networks is not considered because it could be a single point of failure or a malicious node.

A primary goal of the adversary is to attack service availability and degrade the network performance by interfering with on-going communications. An adversary is able to capture and compromise a legitimate node to behave maliciously. A malicious node located along the forwarding path may selectively drop or forward any incoming packet to deafen a sink. The malicious node may also eavesdrop on an on-flying packet and inject false information or modify its packet header to mislead network traffic. However, if a sender can authenticate a packet with a light-weight digital signature [49], a receiver can easily verify the packet and detect any modification. In this research, we primarily focus on the selective forwarding attacks or the adversarial scenarios [20, 21, 45, 23, 26] that cannot be detected by digital signatures and cryptographic primitives.

4.3 Single Checkpoint-based Detection

The SCAD deploys a single checkpoint-assisted approach and consists of three major operations: single checkpoint node selection, timeout, and retransmission. First, when a source node generates a data packet, it randomly selects one of intermediate nodes located along the forwarding path to a sink as a checkpoint node and piggybacks a random number into the packet. Since the source node independently and randomly selects a checkpoint node per-packet basis, it is not trivial for an adversary to predict the checkpoint node for the next data packet. Here, we do not consider dynamically changing routing paths for the same packet during the transmission, because it can exclude the checkpoint node selected by the source node in the path. When a node receives the data packet, it caches the packet in its local storage and checks whether it is selected as a checkpoint node by comparing its one-way hash and map functions [21]. If both functions are equal to one (e.g., selected as a checkpoint node), the node forwards the data packet to the next node and replies an acknowledgment (Ack) packet back to the source node. In Fig. 4.1, a forwarding path from a source node to a sink is depicted with a single checkpoint node. Here, both black and red dots represent a checkpoint node and malicious nodes, respectively. A randomly selected checkpoint node (e.g., n_5) divides the forwarding path into two streams: upstream

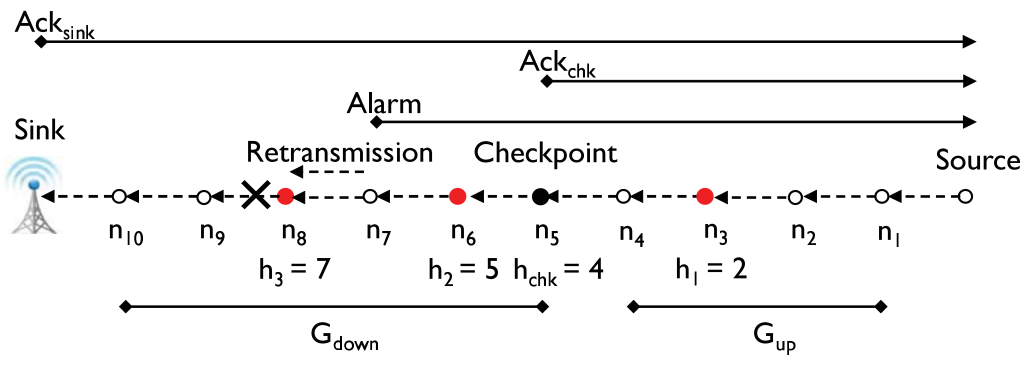


Figure 4.1. A snapshot of network.

(G_{up}: e.g., n_1 to n_4) and downstream (G_{down}: e.g., n_5 to n_{10}). Since both the sink and checkpoint node reply an *Ack* packet, any intermediate node located between the source node and the sink receives one or two *Ack* packets depending on the location of checkpoint node. Note that a malicious node could be selected as a checkpoint node, and thus it could drop a data packet but reply a fake *Ack* packet.

Second, when a node forwards a data packet, it sets a timer for an *Ack* packet originated either from the sink or a checkpoint node, or an *Alarm* packet generated from a downstream node. If the node does not receive the *Ack* or *Alarm* packet before its timer expires, because of possible forwarding misbehavior or bad channel quality, it generates an *Alarm* packet to prosecute the next node for the forwarding misbehavior and forwards the *Alarm* packet back to the source node. The more malicious nodes drop *Ack* or even *Alarm* packets, the more forwarding misbehaviors can be detected because upstream nodes experience more timeouts. The malicious node may fabricate an *Ack* packet but it can be easily detected. This is because each intermediate node can check whether the *Ack* packet was replied from an illegal node by checking its buffered checkpoint seed [21], which is originally generated from the source node and piggybacked to the data packet. A similar light-weight digital signature [49] can also be used to check whether the packet has been modified.

In the SCAD, we propose a timeout technique to reduce unnecessary packet delivery latency, which can be caused by unexpected packet loss due to the forwarding misbehavior or bad channel quality. We define a timeout period as a tuple, $[T^C, T^S]$, where T^C and T^S are timeout periods of an *Ack* packet originated from a checkpoint node (C) and the sink (S), respectively. If a node is located in G_{down}, its T^C is zero. In order to estimate the timeout period, we consider a single-hop based estimated trip time (T_{ETT}) that can be measured from when a node forwards a data packet ($T_{F,data}$) to when it receives an *Ack* packet either from the checkpoint node or the sink ($T_{R,Ack}$). Then T_{ETT} is divided by H_k, which is the number of hops counted from the node to the checkpoint node or the sink when a node forwards a data packet with sequence number $k.T_{ETT}$ is updated by the low pass filter with a filter gain constant α,

$$T^{\ell}_{ETT} = \alpha \cdot T^{\ell}_{ETT} + (1 - \alpha) \cdot T_{ETT,k-1}, \tag{4.1}$$

where $\ell \in \{\text{C, S}\}$. $T_{ETT,k-1}$ is the measurement from the most recently received *Ack* packet and it is expressed as,

$$T_{ETT,k-1} = \frac{T_{R,Ack} - T_{F,data}}{H_{k-1}}. \tag{4.2}$$

Thus, the timeout period is expressed as,

$$T^{\ell} = T^{\ell}_{ETT} \cdot H_k + H_k \cdot \delta, \tag{4.3}$$

where δ is an adjustment factor and $H_k{\cdot}\delta$ is added to consider the packet delivery latency. Fig. 4.2 shows the changes of timeout period against the number of hops from the sink.

Third, we deploy a hop-by-hop retransmission approach to reduce the packet delivery latency and expedite in detecting the forwarding misbehavior in the SCAD. If a node does not receive an *Ack* or *Alarm* packet before its timer expires, it retransmits a cached data packet to the next node after forwarding an *Alarm* packet to the source node. If the node still does not receive an *Ack* or *Alarm* packet again, it forwards another *Alarm* packet again, quits the retransmission, and discards the cached data packet. For example, suppose n_8 drops a data packet forwarded from n_7 in Fig. 4.1. Then n_7 generates an *Alarm* packet and retransmits its cached data packet to n_8. If n_8 drops the retransmitted data packet again, n_7 generates another *Alarm* packet. The more malicious nodes drop retransmitted data packets, the sooner the source

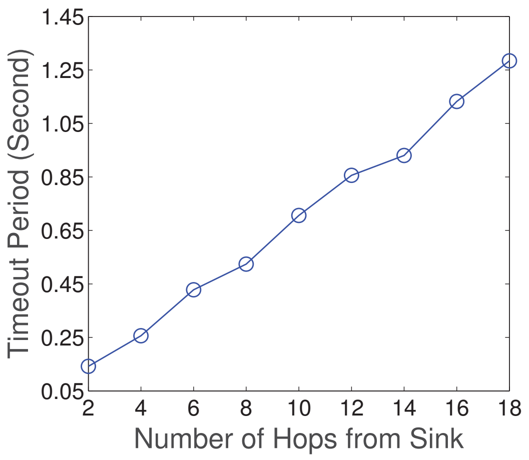


Figure 4.2. The timeout period increases as the number of hops from the sink increases.

node detects their forwarding misbehaviors. Note that the source node may isolate a suspected node from the network after receiving a number of *Alarm* packets by broadcasting a packet piggybacked with the *id* of suspected node, or reducing a forwarding probability of the suspected node [38]. However, this is out of the scope of this research. Major operations of the proposed countermeasure are summarized in Fig. 4.3.

4.4 Analysis of the Proposed Countermeasure

In this part, we analyze the SCAD in terms of average false detection rate. When a packet (e.g., data, *Ack*, or *Alarm*) is lost because of the bad channel quality, however, a node may mistakenly prosecute the next located legitimate node as a malicious node, resulting in the false detection. In Fig. 4.1, for example, n_6 drops a data packet forwarded from n_5. Then n_5 generates an *Alarm* packet to prosecute the forwarding misbehavior of n_6 when its timer expires, and forwards the *Alarm* packet back to the source node. Due to the bad channel quality, the *Alarm* packet can be lost again during the transmission from n_5 to n_4. Then n_4 generates another *Alarm* packet to prosecute the forwarding misbehavior of n_5 when its timer expires, resulting in a false detection. In this analysis, we assume that the bad channel quality in terms of channel error primarily causes packet loss without considering packet drop conducted

```
Notations:
• [$T^C$, $T^S$]: Defined before.
• $pkt[type, seq, chk, x]$: A packet with a sequence number, $seq$, checkpoint node id, $chk$, malicious node id, $x$ and packet type, $type$. Here, $type$ is data, $Ack$ or $Alarm$.
• $Q_i[pkt[seq]]$, $flag_{seq}$, $c^i_{mis}$, $\tau$: A queue of received data packets in $n_i$, a data packet $pkt[seq]$ retransmission flag, the number of detected forwarding misbehaviors of $n_i$ and a forwarding misbehavior threshold.
◇ When a source node, $n_s$, senses an event:
  Send out $pkt[data, seq, chk, none]$;
◇ When the sink, $n_{sink}$, receives an event packet:
  Reply $pkt[Ack, seq, sink, none]$;
◇ When a node, $n_i$, detects a forwarding misbehavior of a malicious node, $n_m$ ($m = i + 1$): $T^C$ or $T^S$ expires;
  if $flag_{seq}$ is false /* Has not retransmitted $pkt[seq]$ */
      Reply $pkt[Alarm, seq, none, m]$;
      Retransmit $pkt[data, seq, chk, none]$;
      $flag_{seq}$ = true;
  else
      Reply $pkt[Alarm, seq, none, m]$;
◇ When a node, $n_i$, receives a $pkt[type, seq, chk, x]$,
  if $pkt[type] == data$
      if $i == chk$
          Enqueue the $pkt[seq]$ into $Q_i$;
          Set up [none, $T^S$]; /* Eq. 1 */
          Forward $pkt[data, seq, chk, none]$;
          Reply $pkt[Ack, seq, chk, none]$;
      else
          if $i < chk$ /* $n_i$ is in the upstream of $chk$ */
              Enqueue the $pkt[seq]$ into $Q_i$;
              Set up [$T^C$, $T^S$];
              Forward $pkt[data, seq, chk, none]$;
          else /* $n_i$ is in the downstream of $chk$ */
              Enqueue the $pkt[seq]$ into $Q_i$;
              Set up [none, $T^S$];
              Forward $pkt[data, seq, chk, none]$;
  end if
  if $pkt[type] == Ack$
      if $sink \in pkt$ /* Ack transmitted from the sink */
          Dequeue the $pkt[seq]$ from $Q_i$;
          Forward $pkt[Ack, seq, sink, none]$;
          Cancel $T^S$;
      else /* Ack is from a checkpoint */
          Forward $pkt[Ack, seq, chk, none]$;
          Cancel $T^C$;
  end if
  if $pkt[type] == Alarm$ /* $x$ is a malicious node */
      if $n_i$ is the source node
          $c^x_{mis} = c^x_{mis} + 1$;
          if $c^x_{mis} \geq \tau$
              Broadcast $isolation$ packet;
      else
          Dequeue the $pkt[seq]$ from $Q_i$;
          Cancel $T^S$ or $T^C$ and $T^S$;
          Forward $pkt[Alarm, seq, none, x]$;
  end if
```

Figure 4.3. The pseudo code of proposed SCAD detection scheme.

by malicious nodes to clearly see the impact on the false detection.

Suppose total N nodes excluding the sink and source node are located in the forwarding path, where m (≥ 1) of them are malicious nodes. φ is a channel error rate, either 10% or 20%. Let P_F be an average false detection rate, which is the sum of average false detection rates of data (P_{FD}), *Ack* (P_{FA}), and *Alarm* (P_{FM}) packet losses. Then P_F is expressed as,

$$P_F = P_{FD} + P_{FA} + P_{FM}. \tag{4.4}$$

First, P_{FD} is expressed as,

$$P_{FD} = \frac{1}{n-m-1}(P_{FD1} + P_{FD2}), \tag{4.5}$$

where,

$$P_{FD1} = \sum_{i=1}^{m} \sum_{j=0}^{h_i - h_{i-1} - 2} (1-\varphi)^{2j+2h_{i-1}} \varphi, \tag{4.6}$$

$$P_{FD2} = \sum_{j=0}^{n-h_m-2} (1-\varphi)^{2j+2h_m} \varphi. \tag{4.7}$$

Here, h_i ($0 \leq i \leq m$, and $h_0 = 0$) is the number of hops from the i^{th} malicious node to the first node (e.g., n_1). P_{FD} is the average false detection rate of data packet loss between the first and the last nodes (e.g., n_1 to n_{10} in Fig. 4.1). In Eq. 4.6, P_{FD1} is the total false detection rates between the first node and the last malicious node (e.g., n_1 to n_8). Note that a data packet loss can lead to both false and correct detection cases. In a false detection case based on Fig. 4.1, if a data packet is lost during the transmission from n_3 to n_4, a malicious node n_3 generates an *Alarm* packet to prosecute the forwarding misbehavior of a normal node n_4. If this *Alarm* packet is forwarded to the source node, a false detection can occur. In case of a correct detection, however, suppose a data packet is lost during the transmission from n_2 to n_3. Then a legitimate node n_2 generates an *Alarm* packet to prosecute a malicious node n_3, which can lead to a correct detection. In Eq. 4.7, P_{FD2} is the total false detection rates between the last malicious node and the last node on the forwarding path (e.g., n_8 to n_{10}). Unlike to P_{FD1}, only a false detection can occur because there is no malicious node between n_8 to n_{10}.

Second, P_{FA} is expressed as,

$$P_{FA} = P_{FA1} + P_{FA2}. \tag{4.8}$$

$$P_{FA1} = \frac{RD_{chk}}{h_{chk} - k}(P_{FA1,1} + P_{FA1,2}), \tag{4.9}$$

where

$$RD_{chk} = (1-\varphi)^{h_{chk}}, \tag{4.10}$$

$$P_{FA1,1} = \sum_{j=0}^{h_{chk}-h_k-1} (1-\varphi)^{h_{chk}-1}\varphi, \tag{4.11}$$

$$P_{FA1,2} = \sum_{i=k}^{1} \sum_{j=0}^{h_i-h_{i-1}-2} (1-\varphi)^{h_{chk}-1}\varphi. \tag{4.12}$$

Also,

$$P_{FA2} = \frac{RD_{sink}}{n-m-1}(P_{FA2,1} + P_{FA2,2}), \tag{4.13}$$

where

$$RD_{sink} = (1-\varphi)^{n-1}, \tag{4.14}$$

$$P_{FA2,1} = \sum_{j=0}^{n-h_m-2} (1-\varphi)^{n-2}\varphi, \tag{4.15}$$

$$P_{FA2,2} = \sum_{i=m}^{1} \sum_{j=0}^{h_i-h_{i-1}-2} (1-\varphi)^{n-2}\varphi. \tag{4.16}$$

Here, h_{chk} is the number of hops from the checkpoint node to the first node (e.g., n_5 to n_1, $h_{chk} = 4$). k is the number of malicious nodes located in G_{up}. $P_{FA1,2}$ becomes zero when $k = 0$. In Eq. 4.8, P_{FA} is an average false detection rate of the first and second *Ack* packet losses from the checkpoint node or the sink to the first node (e.g., n_5 to n_1, or sink to n_1), respectively. RD_{chk} and RD_{sink} are the probabilities that a data packet reaches to the checkpoint node and the sink in Eqs. 4.10 and 4.14, respectively. In Eq. 4.9, P_{FA1} is an average false detection rate of the first *Ack* packet loss during the transmission between the checkpoint node and the first node (e.g., n_5 to n_1).

In Eq. 4.11, $P_{FA1,1}$ is the total false detection rates between checkpoint node and the first malicious node (e.g., n_5 to n_3). Similar to data packet loss, an *Ack* packet loss can lead to both false and correct detections. For example, an *Ack* packet loss during the transmission from n_4 to n_3 can lead to a false detection because a malicious node n_3 generates an *Alarm* packet to prosecute the forwarding misbehavior of a normal node n_4. If an *Ack* packet is lost during the transmission from n_3 to n_2, a correct detection can occur because a normal node n_2 generates an *Alarm* packet to prosecute the malicious node n_3. In Eq. 4.12, $P_{FA1,2}$ is the total false detection rates between the first malicious node and the first node on the forwarding path (e.g., n_3 to n_1). Since no malicious node exists between n_3 and n_1, only a false detection can occur.

In Eq. 4.13, P_{FA2} is an average false detection rate of the second *Ack* packet loss during the transmission between the sink and the first node (e.g., sink to n_1). Similar to the first *Ack* packet loss, both false and correct detections of *Ack* packet loss can occur during the transmission between the sink and the first malicious node. Thus, only a false detection can occur during the transmission between the first malicious node and the first node. In Eqs. 4.15 and 4.16, $P_{FA2,1}$ is the total false detection rates between the sink and the first malicious node (e.g., sink to n_3), while $P_{FA2,2}$ is the total false detection rates between the first malicious node and the first node in the forwarding path (e.g., n_3 to n_1).

Third, P_{FM} is expressed as,

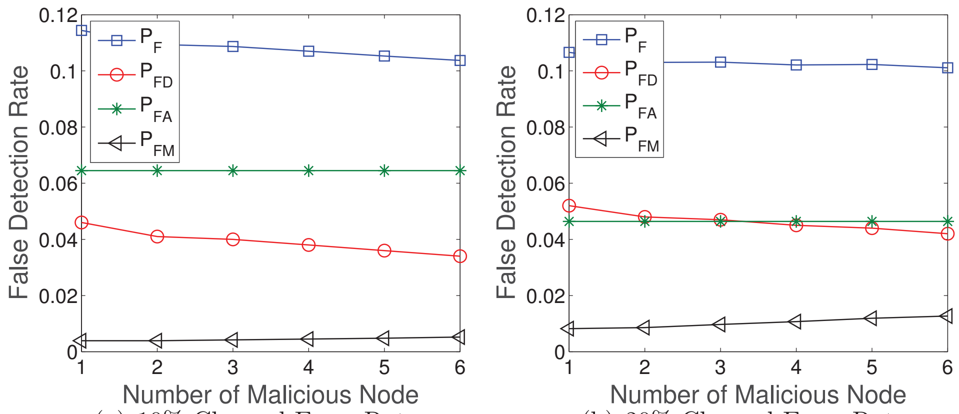
$$P_{FM} = \frac{1}{n-m-1}(P_{FM1} - P_{FM2}), \tag{4.17}$$

where,

$$P_{FM1} = \sum_{i=1}^{n-2}(1-\varphi)^{2i-1}\varphi^2, \tag{4.18}$$

$$P_{FM2} = \sum_{i=1}^{m}(1-\varphi)^{2h_i-1}\varphi^2. \tag{4.19}$$

P_{FM} is an average false detection rate of *Alarm* packet loss between the first and the last nodes. In Eq. 4.18, P_{FM1} includes the probabilities of both false and correct detections for *Alarm* packet loss, respectively. In case of a false detection based on Fig. 4.1, suppose n_6 intentionally drops a data packet and n_5 generates an *Alarm* packet



(a) 10% Channel Error Rate (b) 20% Channel Error Rate

Figure 4.4. The false detection rate against the number of malicious nodes and channel error rates.

to prosecute the forwarding misbehavior of n_6. If the *Alarm* packet is lost during the transmission from n_5 to n_4, n_4 generates another *Alarm* packet to prosecute the forwarding misbehavior of n_5. If this *Alarm* packet is forwarded to the source node, then a false detection can occur. In case of a correct detection, denoted as P_{FM2} in Eq. 4.19, suppose a data packet is lost during the transmission from n_3 to n_4, n_3 generates an *Alarm* packet to prosecute the forwarding misbehavior of n_4, and this *Alarm* packet is lost during the transmission from n_3 to n_2. Then n_2 generates another *Alarm* packet to prosecute the forwarding misbehavior of n_3, leading to a correct detection.

In Fig. 4.4, we show a numerical result of the impact of number of malicious nodes (m) and channel error rate (φ) on the average false detection rate based on the aforementioned analysis. Here, 20 intermediate nodes are located in the forwarding path, where one to six malicious nodes are randomly located. As the m increases, overall P_F decreases with different φ in Subfig. 4.4(a)(b). In particular, higher φ leads to higher P_{FD} in Subfig. 4.4(b). The more data packets are lost, the harder nodes detect whether the packets are lost or dropped. As the m increases, P_{FD} decreases because data packet has higher probability of being dropped by malicious nodes than that of being lost during the transmission. In P_{FA}, malicious nodes are reluctant to drop any *Ack* packet because this forwarding misbehavior may enforce nodes to generate a series of *Alarm* packets. In Subfig. 4.4(b), lower P_{FA} is observed with φ

Table 4.1. Simulation Parameters of SCAD

Parameter	**Value**
Network area	300×300 m^2
Number of nodes	250
Number of malicious nodes	1 to 6
Channel error rate	0 to 10%
Radio data rate	250 Kbps
Packet injection rate	0.5 packet/second
Packet size	1 KByte
Packet drop rate	10% or 20%
Radio range	12.3 m
Radio model	CC2420
Simulation time	1000 seconds

= 20% compared to 10% channel error rate in Subfig. 4.4(a). This is because more data packets are lost during the transmission and thus, the number of *Ack* packets reduces and the m does not affect P_{FA} much. Both m and φ affect P_{FM}. Higher φ leads to higher P_{FM} in Subfig. 4.4(b). As the m increases, P_{FM} slightly increases because $\frac{1}{n-m-1}$ increases in P_{FM}.

4.5 Simulation Testbed

We conduct extensive simulation experiments using the OMNeT++ [41] for performance evaluation and analysis. A 300×300 (m^2) rectangular network area is considered, where 250 nodes are uniformly distributed. The communication range of each node is 12.3 (m). The radio model simulates CC2420 with a normal data rate of 250 Kbps [42]. The channel error rate is randomly changed from 0 to 10% with a step size 2% during the simulation. A packet injection rate is 0.5 packet/second and each packet size is 1 KByte. One to six malicious nodes are randomly located along the forwarding path between a source node and the sink. A set of malicious nodes selectively drops any incoming packet with a packet drop rate, either 10% or 20%. The simulation parameters are summarized in Table 4.1.

In this research, we measure the performance in terms of detection rate, successful drop rate, packet delivery ratio (PDR), energy consumption, number of forwarded and overheard packets, and false detection rate by changing key simulation parameters,

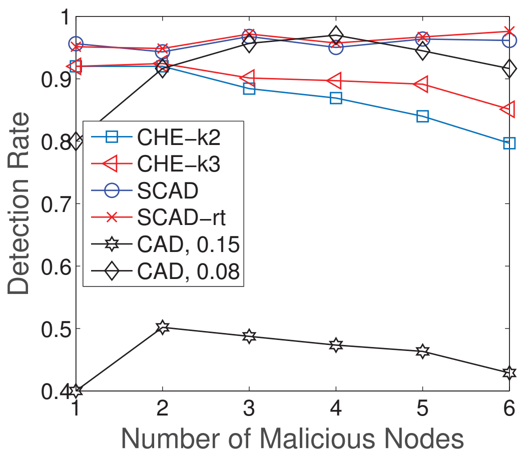
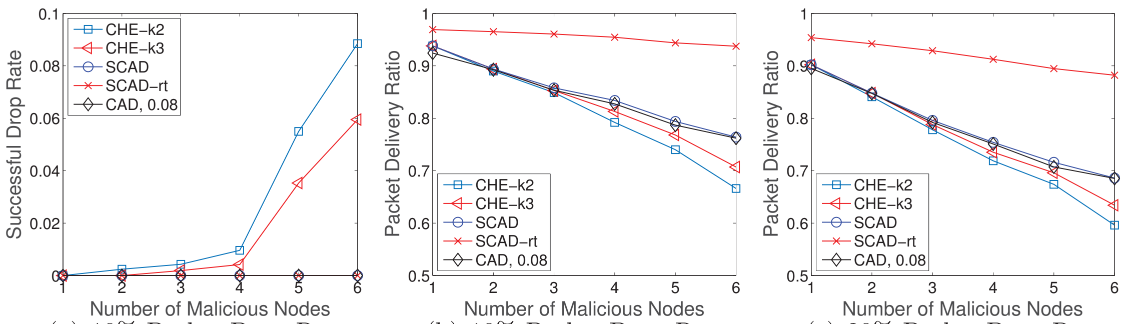


Figure 4.5. The detection rate against the number of malicious nodes.

including number of malicious nodes, packet drop rate, and channel error rate. For performance comparison, we denote the proposed countermeasure without or with retransmission as *SCAD* or *SCAD-rt*, respectively. They are compared with the CHEMAS [21] that is configured with two or three segments (k), denoted as *CHE-k2* or *CHE-k3*, respectively, where an *Ack* packet traverses k segments before being dropped by a checkpoint node. The proposed countermeasure is also compared with the *CAD* [23], where the detection threshold values are set between 0.08 and 0.15.

In Fig. 4.5, as the number of malicious nodes (m) increases, the detection rate decreases in both CHE-k2 and CHE-k3. The probability of multiple malicious nodes being selected as a checkpoint node increases and they may not report the forwarding misbehavior witnessed from adjacent nodes to the source. The lower detection rate is observed with the smaller k. Since *Ack* packet traverses the less number of hops along the forwarding path, each intermediate node receives less number of *Ack* packets forwarded from the downstream. The CAD is sensitive to the detection threshold value and shows about 95% and 50% detection rates in low (0.08) and high (0.15) threshold values, respectively. Due to the temporarily fluctuating channel quality, it becomes an issue to adaptively set the detection threshold value based on the time-varying estimated loss rates. Thus, the detection rate highly depends on the detection threshold value. Unlike to the CAD, both SCAD and SCAD-rt show high and stable detection rates for entire m. Since a single checkpoint node is selected and replies

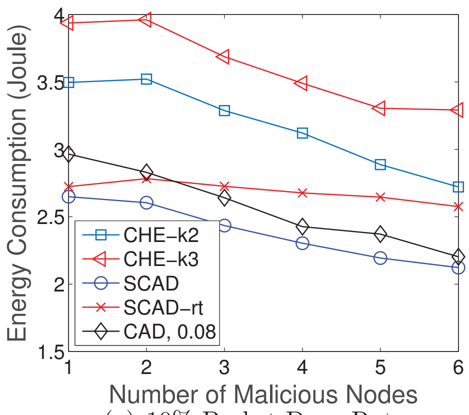


(a) 10% Packet Drop Rate (b) 10% Packet Drop Rate (c) 20% Packet Drop Rate

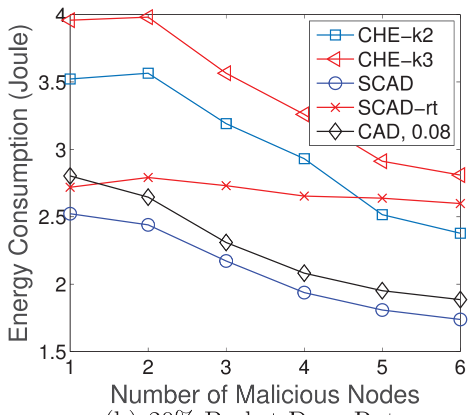
Figure 4.6. The successful drop rate and packet delivery ratios against the number of malicious nodes.

an *Ack* packet, more intermediate nodes are supposed to receive and forward the *Ack* packet to the source. If an upstream legitimate node does not receive an *Ack* packet before its timeout period, it generates an *Alarm* packet to prosecute the next node for forwarding misbehavior.

In Fig. 4.6, both successful drop rate and PDR are measured by varying the m and packet drop rate. In Subfig. 4.6(a), the m significantly affects the successful drop rate in both CHE-k2 and CHE-k3. The CHE-k2 shows higher successful drop rate than that of the CHE-k3. This is because an *Ack* packet travels less number of hops and each intermediate node receives less number of *Ack* packets compared to that of the CHE-k3. Multiple malicious checkpoint nodes can cooperate each other and drop data packets without being detected. Depending on the k, the CHEMAS has a performance tradeoff between security resilience and communication overhead. Note that the SCAD, SCAD-rt, and CAD show zero successful drop rate. In Subfig. 4.6(b), under 10% packet drop rate, PDR quickly decreases as the m increases because more data packets are randomly dropped by malicious nodes. The SCAD, SCAD-rt and CAD show higher PDR than that of the CHE-k2 and CHE-k3 for entire m because the collusion of multiple malicious nodes selected as a checkpoint node does not affect to the SCAD, SCAD-rt and CAD. The SCAD-rt shows the best performance (about 90% or more) because each intermediate node can quickly retransmit its cached data packet to the next node if the data packet is dropped or lost. In Subfig. 4.6(c), overall PDRs decrease with a larger packet drop rate, 20%. However, the SCAD-rt still shows the best performance and the PDR decreases gracefully compared to that of the CHE-k2, CHE-k3 and CAD.

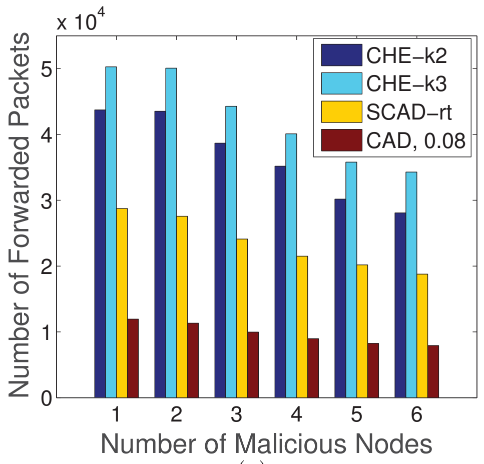


(a) 10% Packet Drop Rate

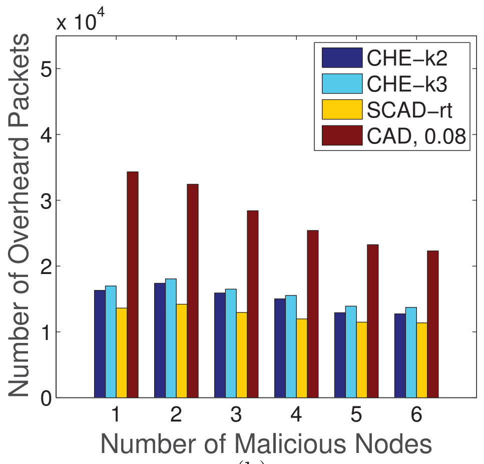


(b) 20% Packet Drop Rate

Figure 4.7. The energy consumption against the number of malicious nodes and packet drop rate.



(a)



(b)

Figure 4.8. The number of forwarded and overheard packets against the number of malicious nodes.

In Fig. 4.7, the energy consumption is measured based on the number of forwarded and overheard packets [50] by varying the m and packet drop rates. In Subfig. 4.7(a), both SCAD and SCAD-rt show lower energy consumption than that of the CHE-k2

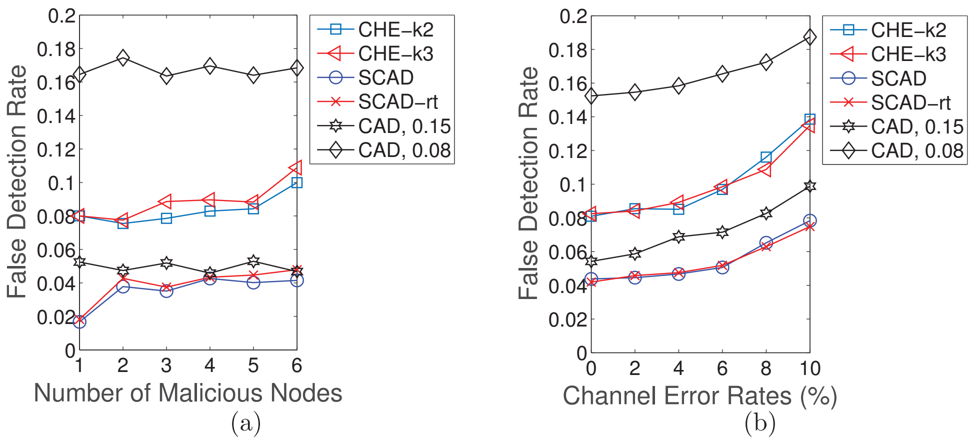


Figure 4.9. The false detection rate against the number of malicious nodes and channel error rate.

and CHE-k3 because of less number of *Ack* packets traversed along the forwarding path. Since an *Ack* packet traverses three and two segments before being dropped by a checkpoint node in the CHE-k3 and CHE-k2, respectively, the CHE-k3 consumes more energy than that of the CHE-k2. The SCAD-rt also consumes more energy than that of the SCAD to retransmit lost or dropped data packets. In Subfig. 4.7(b), overall energy consumptions decrease with higher packet drop rate (20%) because more data packets are dropped by malicious nodes. Note that we measure the number of forwarded and overheard packets in Subfigs 4.8(a) and (b), respectively. The CHE-k2, CHE-k3 and SCAD explicitly send *Ack* packets for detecting forwarding misbehaviors, but the CAD implicitly monitors the network traffic. Thus, intermediate nodes in the CHE-k2, CHE-k3 and SCAD forward more packets but ultimately the CAD overhears more packets, because each node always needs to wake up and observe any on-going packet.

In Fig. 4.9, we measure the false detection rates by varying the m and channel error rates (e). In Subfig. 4.9(a), both SCAD and SCAD-rt show the lowest false detection rate because the number of *Ack* packets generated by a single checkpoint node reduces. Note that this false detection rate is lower than that of the aforementioned analysis (see Fig. 4.4). Since the analysis extensively counts all packet losses due to the bad channel quality, it shows an upper-bound of false detection rate. Multiple checkpoint nodes generate *Ack* packets and each intermediate node frequently forwards them to the source in the CHE-k2 and CHE-k3. Thus, more *Ack* packets can be lost due to the bad channel quality, resulting in higher false detection rate. The CAD with higher detection threshold value (i.e., 0.15) shows the highest false detection rate, because more intermediate nodes mistakenly consider a packet loss as a forwarding misbehavior. In Subfig. 4.9(b), as the e increases, overall false detection rates increase because it becomes harder to detect the forwarding misbehavior of malicious nodes from packet loss due to the bad channel quality.

4.6 Immunity to Other Attacks

We investigate the SCAD whether it can be applied to two well-known attacks: colluding collision attack and power control attack [51].

Colluding Collision Attack: A multiple number of malicious nodes may collude together and create a collision at the next hop on purpose by simultaneously sending packets. The IEEE 802.11 medium access control (MAC) protocol with request-to-send (RTS)/clear-to-send (CTS) exchange can be deployed to reduce packet collisions. However, the 802.11 MAC with RTS/CTS exchange is often disabled in many WSN applications because of its non-negligible energy consumption [51]. Thus, it is not trivial to avoid colluding collision attack, but this attack can be detected by the SCAD. In Fig. 4.1, suppose n_6 sends a data packet to n_7 and its colluding n_8 also simultaneously send any packet to n_7. Then n_7 fails to receive the data packet due to the collision. In the SCAD, since the data packet is lost, the sink will not reply an *Ack* packet back to the source node. Thus, n_5 cannot receive the *Ack* packet from the sink before its timer expires, and it will generate an *Alarm* packet to prosecute the forwarding misbehavior of n_6 and forward the *Alarm* packet back to the source node.

Power Control Attack: A malicious node may control its transmission power and forward a packet to exclude a legitimate node from its communication range. This power control attack is similar to selective forwarding attack and it can be detected by the SCAD. In Fig. 4.1, suppose n_2 forwards a data packet to n_3 and the data packet is relayed to n_4. Then n_2 sets two timers for the *Ack* packets originated from the sink and n_5, respectively. If n_3 reduces its transmission power and forwards the data packet, n_4 fails to receive the data packet. In the SCAD, since n_5 cannot receive the data packet, it will not reply the *Ack* packet back to the source node. Thus, n_2 cannot receive the Ack packet from the checkpoint node before its timer expires, and it will generate an *Alarm* packet to prosecute the forwarding misbehavior of n_3 and forward the *Alarm* packet back to the source node.

4.7 Potential Enhancements

We explore design issues and extensions to see the full potential of our approach for efficiently mitigating the forwarding misbehavior.

Alternative Path for Retransmission: In the SCAD, if a node does not receive an *Ack* or *Alarm* packet before its timer expires, due to the forwarding misbehavior or bad channel quality, it generates an *Alarm* packet to prosecute the next node for its forwarding misbehavior. Then the node retransmits its cached data packet to the same next node based on the proposed hop-by-hop retransmission. If the next node drops the retransmitted data packet again, the source node will choose an alternative forwarding path without including this suspected node. Thus, we plan to deploy a bypass technique [52, 53] in the hop-by-hop retransmission by selecting an alternative forwarding path from the node that prosecutes the next node and generates an *Alarm* packet. This approach can avoid transmitting the cached data packet to the same suspected node over and over until the source node changes the path. For example, when a node detects the forwarding misbehavior of the next node, it selects another one-hop node as a forwarding node and transmits the cached data packet. However, an alternative path may exclude the checkpoint node already selected from the source node during the transmission. Then the node that generates an *Alarm* packet randomly chooses a checkpoint node, piggybacks the id of checkpoint node into the cached data packet, and forwards the data packet towards the sink. Note that when a malicious node selects an alternative path, it may chooses a path which is far longer than the shortest or optimal path to intentionally increase the packet delivery latency, called vampire attack [54].

Active Detection: In the SCAD, a single checkpoint node generates an *Ack* packet and each intermediate node located along the forwarding path *passively* monitors any forwarding behavior of its next node. Similar passive monitoring based approaches are also found in [23, 17, 18, 19, 38]. Since the detection rate highly depends on how frequently malicious nodes conduct the forwarding misbehavior, it can be significantly reduced if multiple malicious nodes collude together. Thus, we consider a camouflage-based detection [39], in which each node pretends not to overhear on-going communication but monitors the forwarding behavior of its adjacent nodes to detect a deep lurking malicious node. We plan to extend the SCAD by deploying an active detection approach, where each intermediate node hides its operational status (i.e., a checkpoint node), counts the number of forwarding misbehaviors, and selects the next forwarding node. A suspected node recorded with a high number of forwarding misbehaviors will not be chosen very often as a forwarding node.

4.8 Summary

In this research, we proposed a light-weight countermeasure, called SCAD, to mitigate the forwarding misbehavior in WSNs. In the SCAD, a single checkpoint-assisted approach incorporated with timeout and retransmission techniques can efficiently improve the detection rate as well as reduce the energy consumption, false detection rate, and successful drop rate. The SCAD can achieve more than 90% PDR with less energy consumption compared to prior CHEMAS and CAD schemes. A simple analytical model of the SCAD and its numerical result in terms of false detection rate are also presented. To see the full potential of our approach, we discuss the design issues and possible extensions of the SCAD. The numerical and simulation results indicate that the proposed countermeasure is a viable approach in WSNs.

CHAPTER 5
CAMOUFLAGE-BASED ACTIVE DETECTION

In this chapter, we investigate one of well-known denial-of-service (DoS) attacks, selective forwarding attack, and proposes a camouflage-based active detection scheme in EHNets.

5.1 Introduction

Energy harvesting from surrounding environmental resources (e.g., vibration, thermal gradient, light, wind, etc.) has been given considerable attention as a way to avoid frequent battery replacements or replenishment. For example, ambient vibration-based energy harvesting has been widely deployed because of the available energy that can be scavenged from an immediate environment, such as a pulse of blood vessel, or a kinetic motion of walking or running [55]. Piezoelectric-based energy harvesting is favored when vibration is the dominant source of environmental energy, and solar light is not always available [43]. Rapidly proliferating wearable devices implanted to anywhere of user (e.g., glasses, clothes, shoes, accessories, or even under skin [56]) are to extend the lifetime of the batteries from an immediate environment, i.e., typical body motions. U.S. Army plans to eliminate all the military batteries or at least reduce the frequency of replacing batteries for communication devices [8]. Soldiers will be equipped with batteryless or self-powered communication devices in near future [9]. We envision that energy harvesting will play a pivotal role in making possible self-sustainable wireless devices ranging from nano-scale sensors to handheld mobile devices, and it will serve as a major building block for emerging Internet of Things (IoT) applications [1]. Thus, a newly emerging energy harvesting motivated network (EHNet) foresees diverse applications in civilian and military environments, and will be a part of ubiquitous communication infrastructure [10].

In this research, we investigate one of well-known denial-of-service (DoS) attacks, selective forwarding attack [44], and its countermeasure in EHNets. In selective forwarding attack, a malicious node randomly or strategically drops any incoming packet in order to disrupt network protocols or interfere with on-going communications on purpose. It is not trivial to identify a malicious forwarding misbehavior from temporal node failures or packet collisions. Note that this is different from a blackhole attack, where a malicious node blindly drops any incoming packet, that can be easily detected. Countering selective forwarding attack and its variants in diverse networks have been actively studied [17, 18, 19, 20, 21, 23]. Unfortunately, selective forwarding attack and its countermeasure are still under-explored in the realm of EHNets.

In light of this, we propose a camouflage-based active countermeasure to selective forwarding attack in EHNets, where each node actively monitors its adjacent nodes and detects forwarding misbehaviors. Our major contribution is summarized in two-fold:

- First, we investigate four adversarial attack scenarios and analyze their potential forwarding behaviors in EHNets, where each node periodically switches its state between active and harvest. A set of vulnerable cases causing a forwarding misbehavior is identified.

- Second, we propose a novel camouflage-based active detection scheme and its communication protocol in EHNets, where each node actively disguises itself as an energy harvesting node, monitors its adjacent nodes, and detects a lurking malicious node.

We develop a customized simulation framework using OMNeT++ [41], conduct a performance evaluation study in terms of six performance metrics, and show a viable approach to selective forwarding attack in EHNets.

5.2 System and Adversarial Models

In this research, each node is assumed to equip a vibration detection card connected with a piezoelectric fiber composite bi-morph (PFCB) W14 and a rechargeable battery [43]. The PFCB-W14 is used as a piezoelectric component to trap immediate environmental vibration energy (e.g., disturbance, walking, or running) and transform it into mechanical vibration energy. Then this mechanical energy can be converted into electrical energy through the direct piezoelectric effect. Energy harvesting is modeled by a two-state Markov process with active (s_a) and harvest (s_h) states. A node stays in active state for an amount of time, which is exponentially distributed with a mean λ_a, and changes to harvest state. After energy harvesting for an amount of time in

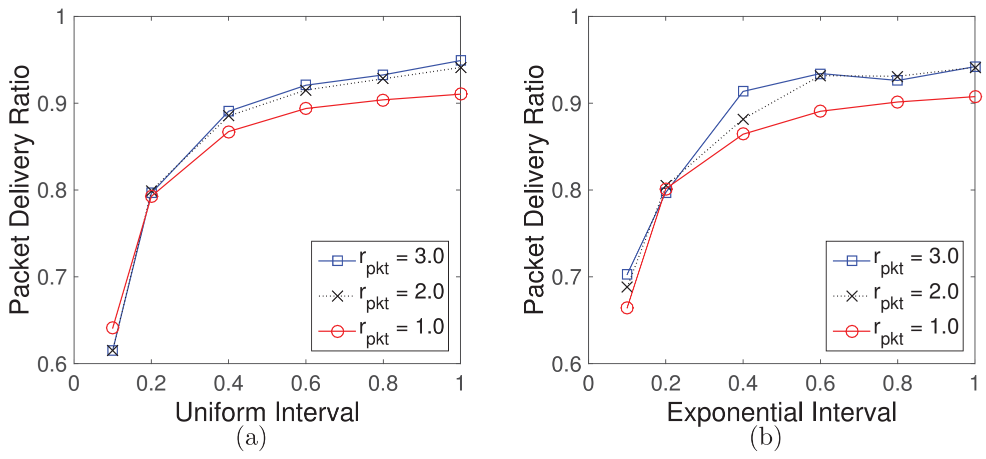


Figure 5.1. The impact of uniform and exponential packet intervals.

harvest state, which is also assumed to be exponentially distributed with a mean λ_h, the node changes back to active state. A node in active state can send/receive and overhear packets. In order to avoid overhead of frequent state changes (i.e., on-off switching cost), a node in harvest state is unable to communicate with other nodes until a certain level of energy is harvested [38]. Each node is aware of its one-hop neighbor nodes by exchanging a one-time single-hop *Hello* packet piggybacked with its node *id* during a network deployment phase [46].

When a node is in harvest state, it periodically broadcasts a one-hop *State* packet to prevent its adjacent neighbor nodes from mistakenly forwarding a packet, resulting in packet loss. In this research, we observe the impact of *State* packet intervals on packet delivery ratio (PDR) in Fig. 5.1, where both uniform and exponential intervals are used by varying packet injection rates (r_{pkt}). Short packet intervals in both uniform and exponential distributions show low PDRs because frequently broadcasted *State* packets can be collided with *Data* packets. As r_{pkt} and interval increase, PDRs increase in both distributions. When the intervals are close to 1.0 (sec), PDRs reach more than 90%. Thus, such a reasonable packet interval is acceptable without significantly affecting the performance in EHNets.

The primary goal of adversary is to attack service availability and degrade the network performance by interrupting on-going communication. The adversary is able to capture and compromise legitimate nodes so that they can behave maliciously.

A malicious node may selectively forward any incoming packet or eavesdrop any on-flying packet and inject false information or modify the packet to mislead the network traffic on purpose. We assume that the malicious node has no energy constraints and it can stay in active state for an extended period. Here, we consider a network where there is at least more than one node to forward a packet to a sink or access point (AP) via multi-hop relay. We do not consider sub-networks connected by a single node because it can be a malicious node or a single-point of failure. If a sender can authenticate a *Data* packet with a light-weight digital signature [49], a receiver can easily verify the packet and detect any modification. In this research, we focus on the adversarial scenarios that cannot be detected by digital signature and cryptographic techniques. We do not consider cryptographic primitives.

5.3 Energy Harvesting Motivated Attacks and Implications

We introduce a set of adversarial scenarios and its vulnerable cases in which a malicious node selectively forwards any incoming packet without being detected in EHNets. An overhearing-based local monitoring is considered to observe the forwarding behavior of adjacent nodes. Although prior local monitoring and acknowledgment-based techniques [17, 18, 19, 20, 21, 23] have been deployed in diverse battery-supported networks, they implicitly assume that nodes stay in active state for an extended period, resulting in non-negligible energy consumption. In this research, each node repeats active and harvest states, and its energy consumption of overhearing can be covered by maximizing the utilization of energy harvesting.

For the sake of simplicity, we use a snapshot of network consisting of four energy harvesting enabled nodes in Fig. 5.2, where a malicious node (n_m) and a node in harvest state are marked as red and shade, respectively. Solid, dotted, and dash-dotted lines represent a forwarding, overhearing, and periodic broadcasting operation, respectively. A packet sender (n_a) forwards a *Data* packet to node (n_c) via one of forwarding candidate nodes (n_b or n_m). Suppose n_m is a malicious node and it can stay in active state for an extended period. When n_a is in active state and has a *Data* packet to send, it selects one of forwarding candidate nodes with equal forwarding probability. If n_a is in harvest state, it holds the packet until it switches back to active state.

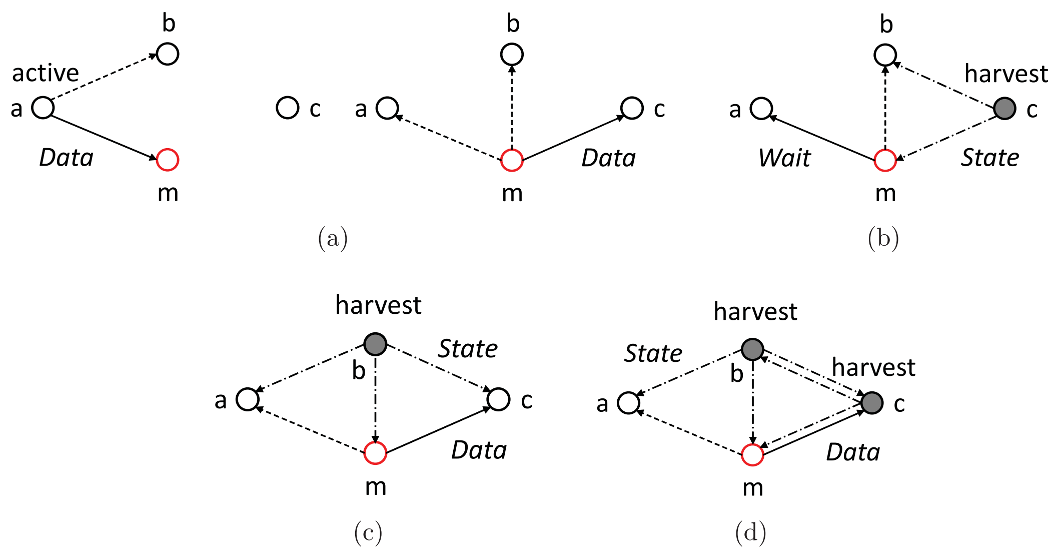


Figure 5.2. A set of adversarial scenarios.

In the first scenario depicted in Subfig. 5.2(a), n_a forwards a received *Data* packet to n_m while n_b can overhear and store the packet in its local cache. If n_m forwards the packet to n_c, both n_a and n_b can overhear the packet and assume that the packet has been successfully forwarded to the next hop, n_c. If n_m drops the packet on purpose, both n_a and n_b cannot overhear it within a timeout period. If n_b does not overhear the packet until the timeout expires, it forwards its cached copy to n_c. When n_a overhears the packet forwarded from n_b, which is different from original forwarder (n_m), n_a can suspect the forwarding misbehavior of n_m. Note that since n_a and n_b are in active state, n_m does not drop the packet because its forwarding misbehavior can be easily detected. Thus, n_m behaves as a legitimate node.

Second, n_c is in harvest state and periodically broadcasts a *State* packet in Subfig. 5.2(b), where both n_b and n_m are aware of the state of n_c. If n_m forwards a *Data* packet to n_c, n_a can overhear it and assume that it has been successfully forwarded to the next hop, n_c. However, n_b can suspect the forwarding behavior of n_m because n_c cannot receive the packet. Thus, n_m does not forward the packet on purpose but holds it until n_c switches back to active state, and replies a *Wait* packet to the packet sender. Then n_a can choose an alternative forwarding node (e.g., n_b).

Third, n_b is in harvest state and periodically broadcasts a *State* packet in Subfig.

5.2(c). If n_m drops a *Data* packet on purpose, n_a can suspect the forwarding behavior of n_m after a timeout period expires. On the other side, if n_m replies a *Wait* packet to the packet sender to delay the packet transmission, n_c can overhear the *Wait* packet and suspect the forwarding misbehavior of n_m. Thus, n_m does not drop the packet but forwards it to the next hop, n_c.

Fourth, both n_b and n_c are in harvest state and periodically broadcast a *State* packet in Subfig. 5.2(d). Since adjacent nodes except the packet sender cannot overhear a packet, n_m can simply forward a packet to the next hop, n_c, resulting in packet loss. n_a can still overhear the packet and thus, the forwarding misbehavior of n_m cannot be detected.

Based on the aforementioned adversarial scenarios, we measure how frequently a malicious node can show its forwarding misbehaviors in terms of attack time ratio (ATR), $\frac{t_{at}}{t_{tot}}$. Here, t_{at} and t_{tot} are total attack time of forwarding misbehaviors and total observation time, respectively. t_{at} is measured by accumulating periods when both adjacent node (n_b) and receiver (n_c) are in harvest state as shown in Subfig. 5.2(d). Average energy harvest time of each node varies between 15 to 40 (sec) and total observation time is 2,000 (sec). In Subfig. 5.3(a), ATR slightly increases (5% to 10%) as energy harvest time increases. As more nodes stay in harvest state, the chance of malicious node to attack without being detected increases. This experiment implies that the malicious node acts as a legitimate node for most of time but attacks during the limited period (10% of t_{tot}) even in high energy harvest time. Since the malicious node can lurk deep but attack only in a vulnerable case, it is not trivial to detect the forwarding behaviors of malicious nodes.

5.4 The Proposed Detection Scheme

We propose a camouflage-based active detection scheme, called CAM, to efficiently detect forwarding misbehaviors of malicious nodes. The basic idea is that each node *actively* disguises itself as an energy harvesting node on purpose and pretends not to overhear, and then monitors any forwarding operation of its adjacent nodes to detect a lurking malicious node. Note that this is different from the prior schemes [17, 18, 19, 23, 38], where each node *passively* monitors any forwarding misbehavior witnessed in a vulnerable case for detection. In this section, we investigate three major

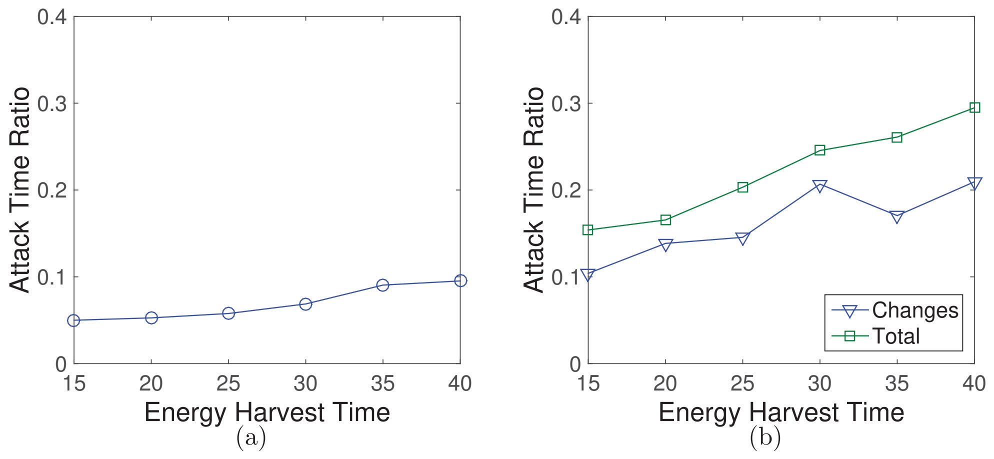


Figure 5.3. The changes of attack time ratios.

issues to implement the CAM scheme: (i) what information should be exchanged and maintained in each node; (ii) how to detect a forwarding misbehavior of lurking malicious node; and (iii) how to adjust actively monitoring a suspected node.

First, when a node receives a *Data* packet, it randomly selects one of active nodes as a forwarding node. If none of forwarding nodes is in active state, the node replies a *Wait* packet to the packet sender and caches the *Data* packet in its local storage. When the node receives a *State* packet from an active forwarding node, it forwards the cached *Data* packet. When a node switches its state, it broadcasts a one-time *State* packet and then periodically broadcasts the *State* packet while it is in harvest state. The node does not periodically broadcast a *State* packet while it is in active state. A *State* packet consists of three components: node id (nid), state ($s \in \{s_{active}, s_{harvest}\}$), and timestamp ($t_{cur}$), where t_{cur} is the current time. When a node receives a *State* packet, it records the packet in a state trace table (ST). For example, when a node n_b receives a *State* packet from n_a, it updates the state of n_a, $ST_b = ST_b \cup [a, s_a, t_{cur}]$. If n_b receives a *State* packet from n_a again but the state of s_a has not been changed, it discards the packet without updating the table.

When a node detects a forwarding misbehavior, it records a number of forwarding misbehaviors of suspected node and updates its monitor probability. In this research, a monitor probability indicates how actively a node monitors the forwarding operation of suspected node, and it is used to decide whether to perform the CAM scheme on

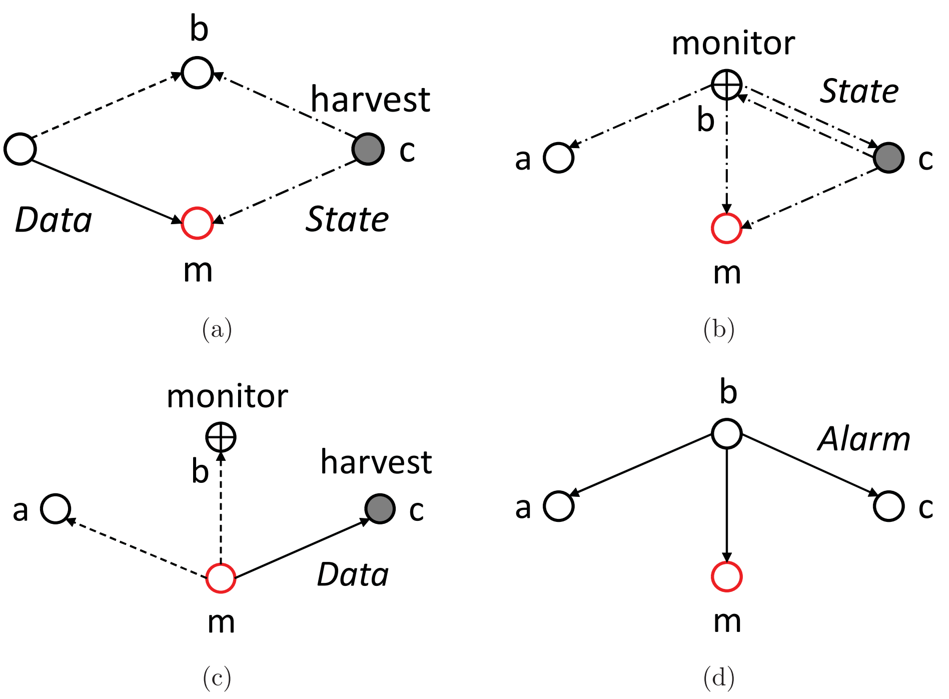


Figure 5.4. A snapshot of the proposed CAM scheme.

suspected node. Initially, each node sets equal monitor probability to all its one-hop neighbor nodes (G^*), $\frac{1}{|G^*|}$. Note that the rationale behind this initialization is to consider a network density. In a dense network, the probability reduces because more number of one-hop neighbor nodes are available to monitor the forwarding operation of suspected node. In a sparse network, however, the probability increases because not many neighbor nodes are available. A set of monitor probabilities is stored and updated in a monitor table (MT). An entry of MT consists of three components: node id (nid), a number of forwarding misbehaviors (c_{mis}), and monitor probability (p).

Second, suppose a node n_b is a legitimate node and overhears a *Data* packet, which is sent from n_a and destined to n_m as shown in Subfig. 5.4(a). Then n_b checks the state of its one-hop neighbor nodes based on the state table, ST_b. If a forwardee node (n_c) is in active state, n_b stays in the current active state without performing the CAM scheme. Since n_b can monitor any forwarding behavior of its one-hop neighbor nodes, n_m will behave as a legitimate node. If the state of n_c is in harvest state as shown

Notations:

• F_i, S_i, $C_{i,j}$, G_i^*: The set of forwardee nodes of n_i, e.g., F_b is $[n_c]$. The set of packet sender of n_i, e.g., S_b is $[n_a]$. The set of common neighbor nodes between n_i and n_j, e.g., $C_{b,m}$ is $[n_a, n_c]$. The set of monitored neighbor nodes of n_i, e.g., G_b^* is $[n_m]$.

• $ST_i[nid, s, t_{cur}]$, $MT[nid, c_{mis}, p]$, nid, s, t_{cur}, c_{mis}, p, τ, δ: Defined before. n_{vim} is the node which is in harvest state and the malicious node forwards packet to. $tget$ is the target node of CAM. src is the source node id of overheard packet. $Fset_g$ is a set of active forwardee node of n_g.

• $pkt[type, fwd, rec, seq]$: A packet is forwarded from n_{fwd} to n_{rec}, with sequence number, seq. Here, $type$ is $data$, $wait$, or $alarm$. If $type$ is $alarm$, rec is considered as malicious node id.

```
◇ $n_g$ overhears the $State$ packet of neighbor node, $n_j$, and then updates $ST_g$.
◇ When $n_g$ receives $pkt[data, s, g, seq]$:
    $Fset_g = \emptyset$;
    for $n_k \in F_g$
        if $ST_g[k].s == ac$
            $Fset_g = Fset_g \cup n_k$;
    if $Fset_g \neq \emptyset$
        Randomly choose a forwarding node ($n_f \in Fset_g$);
        Forward $pkt[data, g, f, seq]$ to $n_f$;
    else
        Cache the packet;
        Forward $pkt[wait, g, s, seq]$ to $n_s$;
◇ When $n_g$ overhears packet $pkt[data, x, y, seq]$:
    if $n_x \in S_g \wedge n_y \in G_g^*$
        for $n_z \in C_{g,y} \wedge n_z \in F_g$
            if $ST_g[z].s == hr$
                $flag_{cam}$ = true; $vim = z$; $tget = y$; $src = x$;
    if $flag_{cam}$ == true $\wedge MT_g[tget].p < rand[0, 1]$
        Broadcast $bogus$ harvest $State$ packet;
        Monitor forwarding behavior of $n_{tget}$;
◇ When $n_g$ overhears packet $pkt[data, tget, vim, seq]$:
    if $ST_g[vim].s == hr \wedge n_{vim} \in C_{g,tget} \wedge flag_{cam}$ == true
        $MT_g[tget].p = MT_g[tget].p + \delta$;
        $MT_g[tget].c_{mis} = MT_g[tget].c_{mis} + 1$;
    if $MT_g[tget].c_{mis} >= \tau$
        Broadcast $pkt[alarm, g, tget, seq]$;
◇ When $n_g$ overhears packet $pkt[wait, tget, src, seq]$:
    if $ST_g[vim].s == hr \wedge n_{vim} \in C_{g,tget} \wedge flag_{cam}$ == true
        $MT_g[tget].p = MT_g[tget].p - \delta$;
        $flag_{cam}$ = false;
```

Figure 5.5. The pseudo code of CAM scheme.

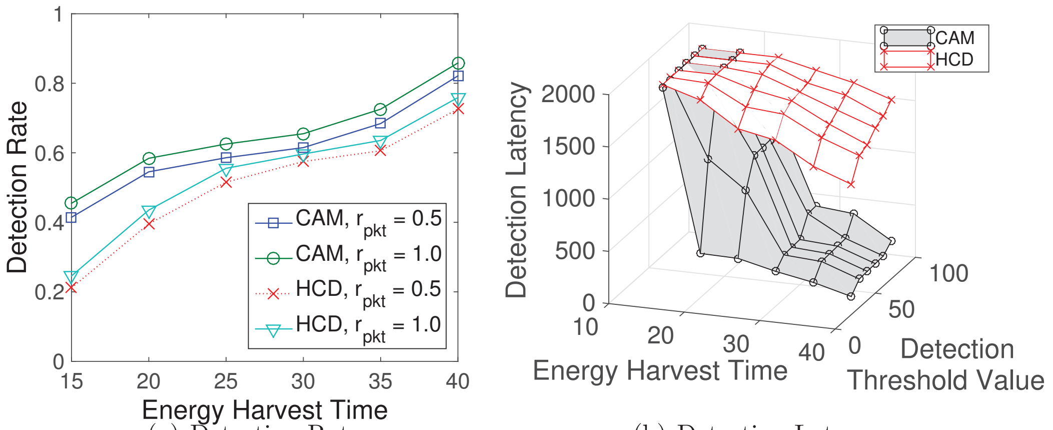
in Subfig. 5.4(a), however, n_b decides whether to perform the CAM scheme based on the monitor probability of n_m, p_m. If a random number (e.g., rand[0, 1]) generated by n_b is less than or equal to p_m, n_b performs the CAM scheme and disguises itself as an energy harvesting node. Then n_b monitors the forwarding operation of n_m while periodically broadcasting a *State* packet piggybacked with harvest state. When n_m overhears a *State* packet, it can be situated in the aforementioned vulnerable case, Subfig. 5.2(d). If n_m simply forwards the *Data* packet to n_c without replying a *Wait* packet back to the packet sender (n_a), this forwarding misbehavior can be detected by n_b as shown in Subfig. 5.4(c). If n_m replies a *Wait* packet, it is considered as a legitimate node. Then n_b broadcasts a *State* packet piggybacked with active state and stops performing the CAM scheme.

Third, when a node detects a forwarding misbehavior, it increments the number of forwarding misbehaviors of suspected node. The node also increases or decreases the monitor probability of suspected node by δ. If the node observes a normal forwarding operation or detects a forwarding misbehavior from suspected node, it decreases or increases the monitor probability by δ, respectively. In addition, when the number of forwarding misbehaviors of suspect node reaches a threshold (τ), the node broadcasts a *Alarm* packet to its one-hop neighbor nodes to prevent the suspected node from involving the forwarding operation as shown in Subfig. 5.4(d). Here, both δ and τ are system parameters and their impacts on the performance are observed in section performance evaluation.

Fourth, we measure the changes of ATR based on the proposed scheme and how additionally a malicious node can reveal its forwarding misbehaviors. In Sugfig. 5.3(b), as average energy harvest time increases, the ATR additionally increases up to 20%. As more nodes can advertise their bogus harvest state, more malicious nodes can frequently be exposed to a vulnerable case. Thus, our approach can increase 15% to 30% of ATR depending on energy harvest time. Major operations of the CAM scheme are summarized in Fig. 5.5.

5.5 Performance Evaluation

We conduct extensive simulation experiments using OMNeT++ [41] to evaluate the performance of proposed scheme. A 150×150m^2 rectangular network area is con-

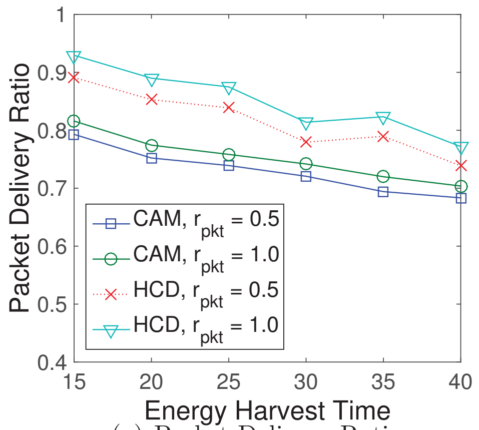


(a) Detection Rate (b) Detection Latency

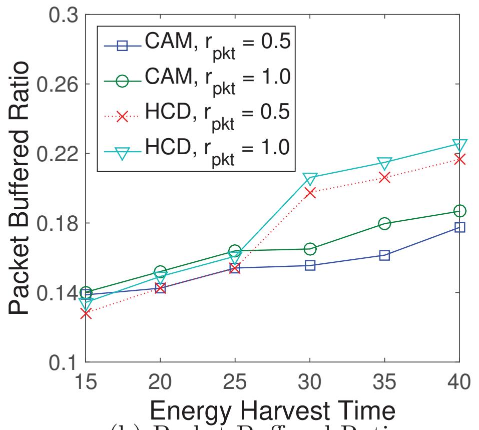
Figure 5.6. The performance of detection rate and detection latency against energy harvest time.

sidered, where 200 nodes are uniformly distributed. The communication range of each node is 12.3 (m). The radio model simulates CC2420 with a normal data rate of 250 Kbps [42]. The radio propagation model is based on the free-space model. A single node generates data traffic with 0.5 and 1 packet injection rates and the data packet size is 1 KByte. The inter-arrival time of traffic is assumed to be exponentially distributed. The periods of active and energy harvest states vary between 50 to 80 seconds and 15 to 40 seconds, respectively. A set of malicious nodes is randomly located along the forwarding path between the packet sender and sink, in which malicious nodes are assumed to monitor network traffic and local network condition, and then perform selective forwarding attacks. In this research, we measure the performance in terms of detection rate, detection latency, packet delivery ratio (PDR), packet buffered ratio, monitor probability, and active and harvest time period by changing key simulation parameters, including packet injection rate (r_{pkt}), energy harvest time (t_h), detection threshold value (τ), and increment weight of monitor probability δ. For performance comparison, we compare our proposed scheme with a hop-by-hop cooperative detection scheme, called HCD [38], which is the first countermeasure to selective forwarding attack in EHNets.

First, we measure detection rate and detection latency by changing r_{pkt}, t_h, and τ in Fig. 5.6. As t_h increases, both detection rates of CAM and HCD schemes increase



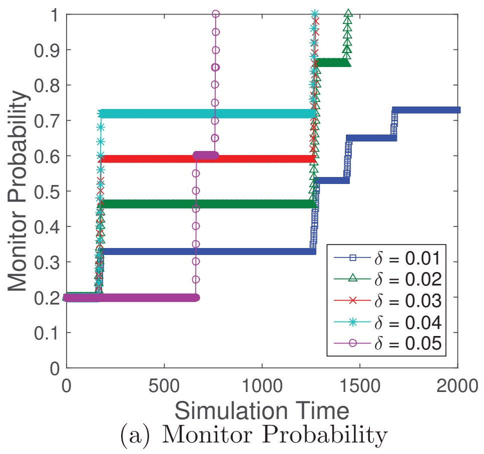
(a) Packet Delivery Ratio



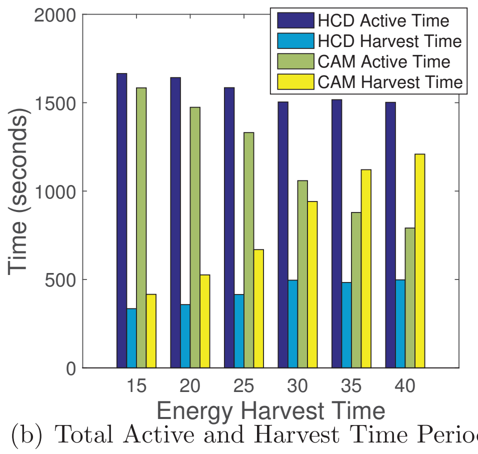
(b) Packet Buffered Ratio

Figure 5.7. The performance of PDR and packet buffered ratio against energy harvest time.

in Subfig. 5.6(a). Since nodes stay in harvest state for a longer period but unable to receive any incoming packet, malicious nodes can have more chances to forward packets to the nodes in harvest state and show frequent forwarding misbehaviors. However, these forwarding misbehaviors can be detected by both CAM and HCD schemes. In particular, the CAM scheme shows higher detection rate than that of the HCD scheme. This is because nodes can actively disguise themselves as energy harvesting nodes, monitor any forwarding operation, and detect more forwarding misbehaviors. Both schemes show the higher detection rate with the larger r_{pkt}. This is because more number of packet is generated at source and more number of packet could be dropped by malicious nodes. Also, more number of forwarding misbehaviors could be detected by both of schemes as well. In Subfig. 5.6(b), the CAM scheme can achieve much more lower detection latency compared to that of HCD. As t_h increases, malicious nodes can frequently have a forwardee node staying in harvest state and show forwarding misbehavior. Thus, adjacent nodes of malicious node can disguise themselves as an energy harvest node and quickly report any forwarding misbehavior to the packet sender. As τ increases, the detection latency increases as well. This is because more number of forwarding misbehavior need to be detected and the elapsed time for reaching τ increases. Unlike our approach, the HCD scheme shows high detection latency for entire t_h and τ. Because a packet sender can detect the forward-



(a) Monitor Probability



(b) Total Active and Harvest Time Periods

Figure 5.8. The performance of monitor probability and total active and harvest time periods against energy harvest time.

ing misbehavior only after receiving a *Mode*[1] packet from its adjacent node. Then the sender can update its mode table of its neighbor nodes, and detect a forwarding behavior by searching the table whether there was any forwarding operation while any forwardee node was in harvest mode.

Second, we measure PDR and packet buffered ratio by varying r_{pkt} and t_h in Fig. 5.7. In Subfig. 5.7(a), PDR decreases as t_h increases because malicious nodes can have higher chances to intentionally forward packets to the nodes staying in harvest state, resulting in more packet losses. The CAM scheme shows lower PDR than that of the HCD scheme because more nodes can temporarily disguise themselves as energy harvesting nodes for detection. This can create more chances for malicious nodes to intentionally forward packets to the nodes staying in harvest state and cause more packets losses. In Subfig. 5.7(b), as t_h increases, a packet sender may not find an active next hop node as a forwardee but buffer a receiving packet in its cache. The CAM scheme shows lower buffered packet ratio than that of the HCD scheme for entire t_h, because more malicious nodes forward packets to the next hop nodes staying in harvest state.

Third, changes of monitor probability with different weights (i.e., δ from 0.01 to

[1]In [38], a node broadcasts a *Mode* packet whenever it changes its state. This is similar to a *State* packet in this research.

0.05) and total active and harvest time periods are observed over simulation time in Fig. 5.8. Whenever a node detects a forwarding misbehavior, it increases the monitor probability of suspected node by δ. Thus, malicious nodes can be monitored more often and most likely be detected for forwarding misbehaviors. In Subfig. 5.8(a), for example, monitor probability of the CAM scheme with $\delta = 0.05$ reaches to 1.0 in about 700 seconds. In Subfig. 5.8(b), total active and harvest time periods of both schemes are measured by t_h. In particular, total active and harvest time periods of the HCD scheme decrease and increase as t_h increases, respectively. This is because nodes of the HCD scheme stay in harvest state for a longer period as t_h increases. However, more total active and harvest time periods of the CAM scheme decrease and increase as t_h increases compared to that of the HCD scheme, respectively. This is because nodes of the CAM scheme can actively disguise themselves as energy harvesting nodes and try to monitor any forwarding operation and detect forwarding misbehaviors.

5.6 Summary

In this research, we proposed a countermeasure to selective forwarding attack in EHNets. Four adversarial scenarios motivated by energy harvesting and their potential vulnerabilities are investigated. Then a camouflage-based active detection scheme is proposed to efficiently detect the forwarding misbehavior. Extensive simulation results indicate that the proposed countermeasure achieves better performance in terms of detection rate and detection latency compared to the existing hop-by-hop cooperative detection scheme, and suggests a new approach to detect lurk deep malicious nodes in EHNets.

CHAPTER 6
COOPERATIVE DETECTION SCHEME

In this chapter, we propose a cooperative countermeasure to efficiently detect the forwarding misbehavior in EHNets.

6.1 Introduction

Internet-of-Things (IoT) and its applications are rapidly proliferating, where a myriad of multi-scale sensors and devices (later in short, nodes) are seamlessly blended for a ubiquitous computing and communication infrastructure [1]. Nodes are resource constrained in terms of computing and battery-power, but are often required to operate a long-term sensing and communication in a hostile or unattended area. Since wireless communication could be responsible for more than half of total energy consumption [3], a significant amount of effort has been devoted to develop energy efficient routing protocols in wireless sensor networks (WSNs) [4]. Due to the limited battery-power, however, it is ultimately unavoidable to replace or replenish batteries. In order to remove batteries or at least reduce the frequency of replacing batteries, energy harvesting from an immediate environment (e.g., kinetic, wireless, solar, etc.) has been increasingly popular for IoT [5, 6, 7] and playing an important role in realizing self-sustainable nodes deployed in a large-scale network. Thus, an energy harvesting motivated network (EHNet) is rapidly emerging and becoming a major building block for IoT applications as well as a part of ubiquitous communication infrastructure.

For routing, each node communicates with its neighbor nodes based on a broadcast-based forwarding, and collaboratively routes sensory data through a multi-hop relay. When a node intends to reply a unicast packet, unlike a wired network, all one hop neighbor nodes can still overhear the packet, as if it is a broadcast packet [16]. Since radio link is a shared medium and its radiation pattern is often omni-directional from antenna, it is inherently insecure and thus, adversaries can easily overhear, duplicate, corrupt, or alter data. Nodes deployed in such a hostile or unattended area can also be captured, tampered, or destroyed because they are physically insecure. For example, a malicious node compromised by an adversary can randomly or selectively drop any incoming packet to disrupt network protocols and interfere with on-going communications on purpose or strategically. Note that it is not trivial to differentiate such a misbehavior (or attack) from a temporal node failure or packet loss. Diverse countermeasures and their variants have been proposed to avoid and/or detect a forwarding misbehavior under an implicit assumption of battery-powered networks, where conventional encryption algorithms and secure routing protocols cannot be directly applied. Unfortunately, forwarding misbehavior and its countermeasure are still under-explored in the realm of EHNets.

The main goal of this research is to relax this implicit assumption in the presence of self-sustainable nodes that periodically harvest the energy and repeat on- and off-periods for communication. More specifically, this research will identify a new type of selective forwarding attacks and investigate a cooperative countermeasure to this forwarding misbehavior in the realm of EHNets, where malicious nodes operate as legitimate nodes most of time and drop any incoming packet during a vulnerable period without being detected. Our major contribution is summarized in three-fold:

- First, we investigate a set of adversarial scenarios and analyze its forwarding operations under the *charge-and-spend* harvesting policy in EHNets. Then we identify four vulnerable scenarios and their corresponding potential forwarding misbehaviors.

- Second, we propose a cooperative countermeasure to efficiently detect the forwarding misbehavior in EHNets, called *EYES*, and it consists of two mechanisms: *SlyDog* and *LazyDog*. In the SlyDog, each node actively disguises itself as an energy harvesting node but in fact monitors its adjacent nodes to detect the forwarding misbehavior of lurking deep malicious nodes. In the LazyDog, however, each node periodically requests its adjacent nodes of a limited history of forwarding operations, and validates any prior uncertain forwarding operation to detect the forwarding misbehavior.

- Third, we propose an analytical model of the EYES and show its numerical results in terms of detection rate. We also revisit prior detection approaches, Watchdog [17] and HCD [38], and modify them to work in EHNets. Both single and two malicious nodes cases are applied to HCD and Watchdog, and no malicious node case is also considered as the performance upper bound of packet delivery ratio. In addition, detection strategies of forwarding misbehavior are comprehensively compared in terms of six properties.

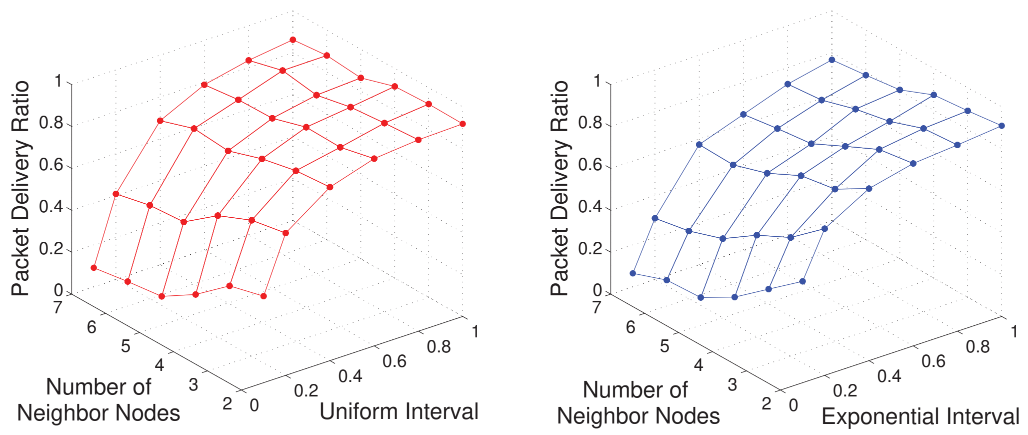
We conduct extensive simulation experiments using the OMNeT++ [41] for performance comparison and analysis. Compared to the Watchdog and HCD, the EYES can not only efficiently detect forwarding misbehavior but also significantly improve the performance in terms of detection rate, detection latency, and packet delivery ratio.

6.2 System and Adversarial Models

In this research, each node is assumed to equip with an energy harvesting device to replenish its rechargeable battery [43]. For example, a piezoelectric fiber composite bimorph (PFCB) W14 (1.3 mW – 47.7 mW) based energy harvesting from an immediate environment (e.g., disturbance, or typical body movements) can generate sufficient power for most small wireless sensors and mobile devices[1] [59, 60, 61]. It is envisaged that multi-scale piezo devices and integrated self-charging power cells (SCPCs) [62] will enhance the efficiency of energy harvesting and achieve a seamless communication. The energy harvesting process is modeled as a two-state Markov process with active (s_a) and harvest (s_h) states. Each node stays in either active or harvest state for a certain period of time, which is exponentially distributed with a mean λ_a or λ_h respectively, and changes to the other state. Note that frequent state changes incur a non-negligible on-off switch cost in terms of energy consumption and operational delay.

In light of this, we adopt the *charge-and-spend* harvesting policy [38, 39, 63, 64], where a node in harvest state is unable to listen and receive any packet until a certain level of energy is harvested. Although the node minimizes its communication activity during harvest state, it periodically broadcasts a one-hop *State* packet to prevent other nodes from mistakenly forwarding a packet to the nodes in harvest state. We observe the impact of *State* packet intervals and number of neighbor nodes on packet delivery

[1]For example, the IEEE 802.15.4-compliant Texas Instrument Chipcon CC2420 radio [57] supports eight different transmission power levels ranging from 3 μW to 1 mW. The IEEE 802.11 a/b/g-compliant Cisco Aironet 340 and 350 series [58] also support four (1, 5, 10, and 30 mW) and six (1, 5, 20, 30, 50, and 100 mW) transmission power levels, respectively.



(a) (b)

Figure 6.1. The impact of uniform and exponential *State* packet intervals and number of neighbor nodes.

ratio (PDR) in Fig. 6.1, where both uniform and exponential packet intervals are used. Short packet intervals show the low PDR because frequently broadcasted *State* packets can be collided with *Data* packets. As the number of neighbor nodes increases, the PDR reduces because more nodes may broadcast *State* packets in harvest state, resulting in more collisions with *Data* packets. When the packet interval is close to 1.0 (sec), the PDR is still above 80% even with the large number of neighbor nodes. This packet interval is acceptable without significantly affecting the performance in EHNets. In addition, each node is aware of its one-hop neighbor nodes by exchanging a one-time single-hop *Hello* packet during a network deployment phase [46].

When a node detects an event, it becomes a source node, generates a data packet, and forwards the packet towards a sink. To deliver the data packet towards the sink, a simple energy-based routing [65] technique can be deployed. An adversary is able to capture and compromise a legitimate node to behave maliciously. The primary goal of adversary is to attack service availability by disrupting network protocols or interfering with on-going communication. A malicious node involved in packet forwarding operation may selectively or strategically drop or forward any packet to deafen a sink. The malicious node may also eavesdrop on an on-flying packet and inject false information or modify its packet header to mislead network traffic. However, if a sender authenticates a packet with a light-weight digital signature [49], a receiver can easily verify the packet and detect any modification. In this research, we consider a dense network, where there is at least more than one neighbor node to forward a packet. Two sub-networks connected with a single node are not considered because it can be a malicious node or a single-point of failure. We assume that a malicious node has no energy constraints and it can stay in active state for an extended period. In this research, we primarily deal with the selective forwarding attack or the energy harvesting motivated adversarial scenarios that cannot be detected by digital signature and cryptographic techniques. We do not consider cryptographic primitives.

6.3 Energy Harvesting Motivated Attack Scenarios and Analysis

In this section, we investigate potential forwarding misbehaviors through a set of adversarial scenarios and observe vulnerable cases in the EHNets, where more than one malicious node are consecutively located along the forwarding path. We consider a snapshot of network consisting of five energy harvesting enabled nodes in Fig. 6.2, where a malicious node and a node in harvest state are marked as red and shade, respectively. Solid, dotted, and dash-dotted lines represent a forwarding, overhearing, and periodic broadcast operation, respectively. Suppose a node (n_a) forwards a *Data* packet to n_c through intermediate nodes, n_b, n_{m_A}, and n_{m_B}, where both n_{m_A} and n_{m_B} are malicious nodes.

Potential Forwarding Behaviors: First, when a packet sender (e.g., n_a, n_{m_A}, or n_{m_B}) forwards a received *Data* packet, its neighbor nodes (e.g., n_a, n_b, or n_{m_A}) can overhear and store it in its local cache as shown in Subfig. 6.2(a). If n_{m_A} drops the packet on purpose, n_b cannot overhear it within a timeout period and forwards its cached copy to n_{m_B}. If n_a overhears the packet forwarded from n_b, which is different from the original forwarder (n_{m_A}), it suspects the forwarding behavior of n_{m_A}. Thus, n_{m_A} does not drop the packet when n_a and n_b are in active state. When n_{m_B} forwards the packet to n_c, both n_b and n_{m_A} can overhear it. n_b assumes that n_{m_B} has successfully forwarded the packet to the next hop, n_c.

Second, if n_{m_A} forwards a received *Data* packet to n_{m_B}, which is in harvest state and periodically broadcasts a *State* packet, n_b can overhear it and suspect the forwarding behavior of n_{m_A} as shown in Subfig. 6.2(b). This is because n_{m_B} is in harvest state and cannot receive any packet. Thus, n_{m_A} does not forward the packet on purpose

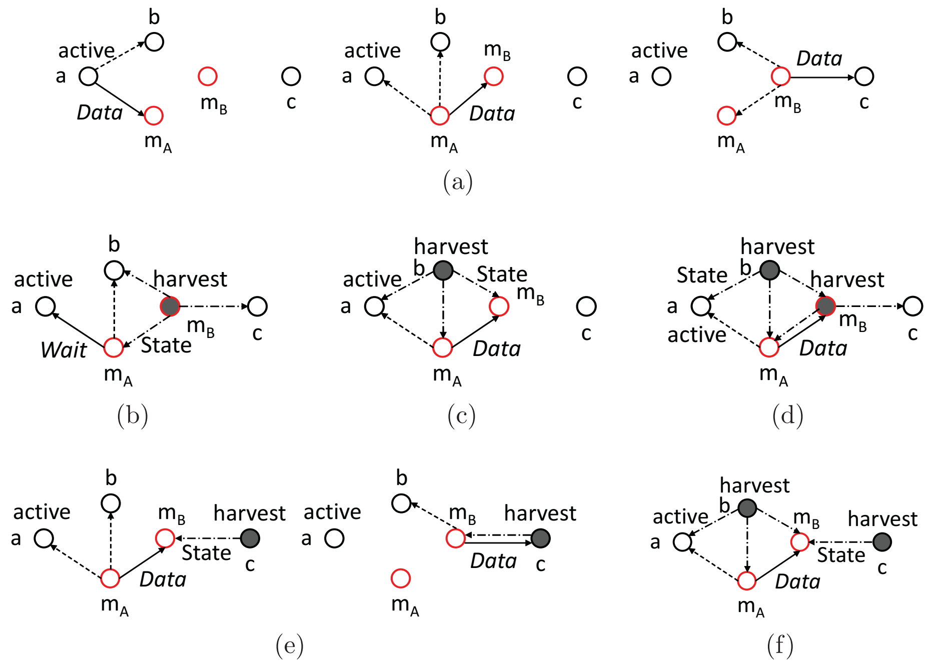


Figure 6.2. A set of adversarial scenarios and its vulnerable cases in the presence of malicious nodes in EHNets.

but holds it until n_{m_B} switches back to active state, and replies a *Wait* packet to the packet sender, n_a, to delay the packet transmission. Upon receiving the *Wait* packet, n_a selects an alternative forwarding node, n_b.

Undetected Vulnerable Cases: Third, suppose n_b is in harvest state and periodically broadcasts a *State* packet as shown in Subfig. 6.2(c). If n_{m_A} drops a received *Data* packet on purpose, n_a can suspect the forwarding misbehavior of n_{m_A} when a timeout period expires. If n_{m_A} simply forwards the packet to n_{m_B}, which will hold it without forwarding to the next hop, the packet is lost without being detected. Since n_b is in harvest state and n_c cannot overhear the packet, this forwarding misbehavior of n_{m_A} and n_{m_B} cannot be detected.

Fourth, both n_b and n_{m_B} are in harvest state and periodically broadcast a *State* packet as shown in Subfig. 6.2(d). Since only n_a can overhear the packet, n_{m_A} simply forwards the packet to n_{m_B}, resulting in packet loss without being detected. Although n_a can overhear the packet, the forwarding misbehavior of n_{m_A} cannot be detected.

Fifth, n_c is in harvest state and periodically broadcasts a *State* packet as shown in Subfig. 6.2(e). Since both n_a and n_b are in active state, n_{m_A} forwards a received *Data*

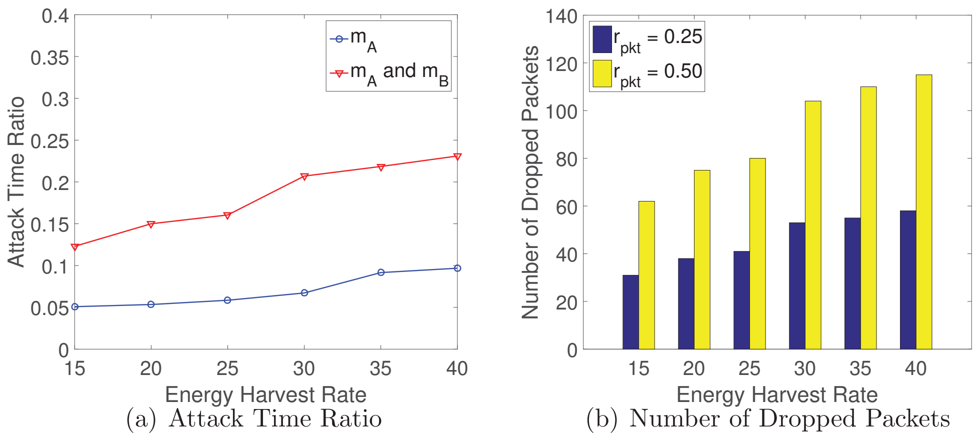


Figure 6.3. The changes of attack time ratio and number of dropped packets against energy harvest rate and packet injection rate.

packet to n_{m_B}. If n_{m_B} holds the packet without forwarding to the next hop, n_b can suspect the forwarding behavior of n_{m_B} when a timeout period expires. In case of when n_c is in harvest state, n_{m_B} simply forwards the packet to n_c, resulting in packet loss without being detected.

Lastly, both n_b and n_c are in harvest state and periodically broadcast a *State* packet as shown in Subfig. 6.2(f). n_{m_B} can either forward the packet to n_c or hold the packet without forwarding to the next hop, resulting in packet loss without being detected. This is because only n_{m_A} can overhear the packet.

Lurking Deep Malicious Nodes: Based on the aforementioned undetected vulnerable cases, we measure the number of dropped packets and how frequently malicious nodes can collude together and conduct undetected forwarding misbehaviors in terms of attack time ratio (ATR), $\frac{t_{at}}{t_{tot}}$, in Fig. 6.3. Here, t_{at} and t_{tot} are total attack time of undetected forwarding misbehaviors and total observation time (e.g., 1,000 (sec)), respectively. t_{at} is measured by accumulating the periods when either adjacent node (n_b) or receiver (n_c), or both of them are in harvest state as shown in Subfigs. 6.2(c), (d), (e), and (f). In Subfig. 6.3(a), the ATR of one malicious node n_{m_A} slightly increases from 5% to 10% as energy harvest rate increases. However, the ATR of two colluding malicious nodes (i.e., n_{m_A} and n_{m_B}) can quickly increase upto 24%. As nodes stay in harvest state for longer period, the chance of malicious nodes to cooperatively conduct forwarding misbehaviors without being detected increases. In Subfig. 6.3(b), the number of dropped packets is measured against energy harvest rate and packet injection rate (r_{pkt}). More number of packets are dropped with higher $r_{pkt} = 0.5$ (pkt/sec). This is because two colluding malicious nodes receive more packets with higher r_{pkt}, more packets are dropped due to undetected forwarding misbehaviors.

6.4 The Proposed Countermeasure

The proposed countermeasure, called *EYES*, consists of two schemes to efficiently detect forwarding misbehaviors of colluding malicious nodes in EHNets: *SlyDog* and *LazyDog*. This is different from the prior approaches, [17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 30], where each node only *passively* monitors any forwarding misbehavior witnessed in a adversarial case for detection.

6.4.1 SlyDog: Inducement-based Detection

The basic idea of SlyDog is that each node *actively* disguises itself as an energy harvesting node on purpose and pretends not to overhear its adjacent nodes. But in fact, each node stealthily monitors any forwarding operation of its adjacent nodes to detect a lurking deep malicious node. Here, the SlyDog is significantly extended from our previous work CAM [39] to detect a collusion of malicious nodes.

Basic Operations: First, when a node receives a *Data* packet, it randomly selects one of adjacent nodes as a forwarding node (or forwardee node). If none of adjacent nodes is in active state, the node replies a *Wait* packet to the prior packet sender and caches the *Data* packet in its local storage. When the node receives a *State* packet from an adjacent node in active state, it forwards the cached *Data* packet. When a node switches its state, it broadcasts a one-time *State* packet. If the node is in harvest state, it periodically broadcasts a *State* packet. Since the node in harvest state is unable to receive any packet based on the charge-and-spend policy, this periodic *State* packet prevents its adjacent nodes from mistakenly forwarding a packet. However, the node in active state does not periodically broadcast a *State* packet. A *State* packet consists of three components: node id (nid), state ($s \in \{s_a, s_h\}$), and timestamp (t_{cur}). When a node receives a *State* packet, it records the packet in a state trace table (ST). For example, when a node n_b receives a *State* packet from n_a, it updates the state of n_a

```
Notations:
• $F_i$: The set of forwardee nodes of $n_i$, e.g., $F_a$ is $[n_b, n_{m_A}]$.
• $FS_i$: The set of active forwardee nodes of $n_i$.
• $ST_i[nid, s, t_{cur}]$, $MT[nid, c_{mis}, p]$, $nid$, s, $t_{cur}$, $c_{mis}$, $p$, $\tau$, $s_a$: Defined before.
• $pkt[type, fwd, rec, seq]$: A packet is forwarded from $n_{fwd}$ to $n_{rec}$, with sequence number, $seq$.
Here, type is $Data$, $Wait$, or $Alarm$. If type is $Alarm$, $rec$ is considered as malicious node id.
Operations:
◇ $n_i$ overhears a $State$ packet of neighbor node, $n_j$:
  Updates $ST_i$ if the state of $n_j$ changes.
◇ $n_i$ receives a $Data$ packet $pkt[Data, s, i, seq]$ from $n_s$:
  $FS_i = \emptyset$;
  for $n_k \in F_i$
    if $ST_i[k].s == s_a$ /* $n_k$ is in active state */
      $FS_i = FS_i \cup n_k$;
  if $FS_i \neq \emptyset$ /* At least one active forwardee node exists */
    Randomly selects a forwardee node $n_f$, $n_f \in FS_i$;
    Forwards the $Data$ packet $pkt[Data, i, f, seq]$ to $n_f$;
  else /* No active forwardee node exists */
    Caches the $Data$ packet;
    Forwards the $Wait$ packet $pkt[Wait, i, s, seq]$ to $n_s$;
◇ $n_i$ isolates the malicious node $n_j$ from network:
  if $MT_i[j].c_{mis} >= \tau$
    Broadcasts the $Alarm$ packet $pkt[Alarm, i, j, seq]$;
```

Figure 6.4. The pseudo code of legitimate node's operations.

(s), $ST_b = ST_b \cup [a, s, t_{cur}]$. If n_b receives a *State* packet from n_a again but the state of n_a has not been changed, it discards the packet without updating the table. The aforementioned operations of a legitimate node is summarized in Fig. 6.4.

Second, each node also maintains an audit table (AT), where each entry consists of five components: one-hop neighbor node's id (sid), two-hop neighbor node's id (rid), and number of overheard packets sent from one-hop neighbor node to two-hop neighbor node (op) during a time interval between t_{begin} and t_{end}. For example, when a node n_a overhears a packet transmission from its one-hop neighbor node (n_b) to two-hop neighbor node (n_c), it sets t_{begin} to the current time and increases $AT_a[b, c].op$ by one. When a node switches to harvest state, it sets t_{end} to the current time.

Third, when a node detects a forwarding misbehavior, it records a number of forwarding misbehaviors of suspected node and updates its monitor probability. In this research, a monitor probability indicates how actively a node monitors the forwarding operation of suspected node, and it is used to decide whether to perform the SlyDog

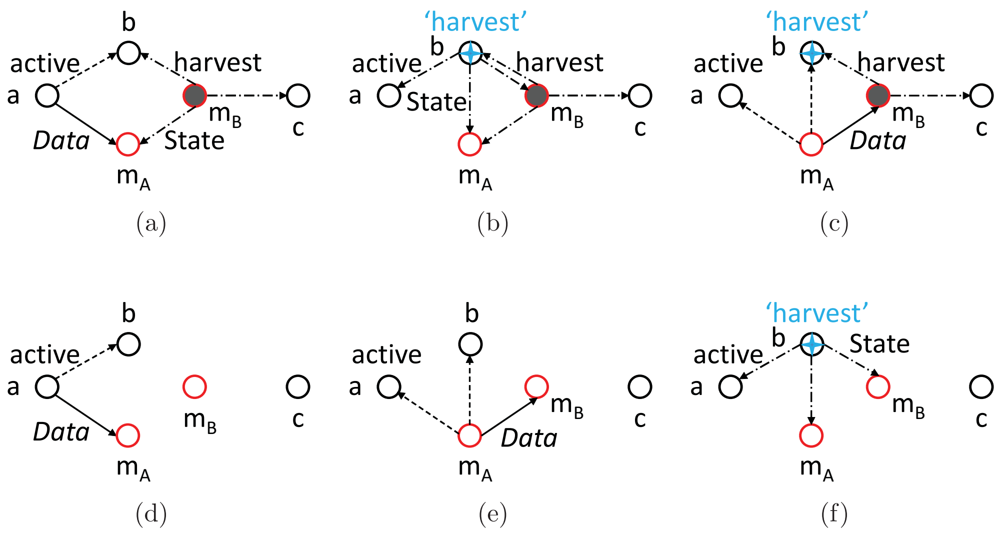


Figure 6.5. The proposed *SlyDog* detection strategy.

on suspected node. Initially, each node sets equal monitor probability to all its one-hop neighbor nodes (G^*), $\frac{1}{|G^*|}$. Note that the rationale behind this initialization is to consider a network density. In a dense network, the probability decreases because more number of one-hop neighbor nodes are available to monitor the forwarding operation of suspected node. In a sparse network, however, the probability increases because a less number of neighbor nodes are available. A set of monitor probabilities is stored and updated in a monitor table (MT). Each entry of MT consists of three components: node id (nid), a number of detected forwarding misbehaviors (c_{mis}), and monitor probability (p).

Fourth, whenever a node detects a forwarding misbehavior, it increments the number of detected forwarding misbehaviors of suspected node by one and increases the monitor probability by δ. Here, δ is a system parameter and its impact on the performance is observed in Section performance evaluations. In addition, when the number of detected forwarding misbehaviors of suspected node reaches a threshold τ, the node broadcasts an *Alarm* packet to its one-hop neighbor nodes to prevent the suspected node from being selected as a forwardee node. The isolation operation of malicious node is summarized in Fig. 6.4.

Detection Operations: First, suppose a node n_b is a legitimate node and overhears a *Data* packet sent from n_a to n_{m_A} as shown in Subfig. 6.5(a). Then n_b checks the state of its one-hop neighbor nodes based on the state table, ST_b. In Subfig. 6.5(b), if a forwardee node (n_{m_B}) is in harvest state, n_b decides whether to perform the SlyDog based on the monitor probability of n_{m_A}, p_{m_A}. n_b generates a random number (e.g., rand[0, 1]) and if it is less than or equal to p_{m_A}, then n_b performs the SlyDog on n_{m_A} and disguises itself as an energy harvesting node. n_b stealthily monitors the forwarding operation of n_{m_A} while periodically broadcasts a *State* packet piggybacked with its harvest state. In Subfig. 6.5(c), when n_{m_A} overhears a harvest *State* packet from n_b, n_{m_A} believes that n_b is in harvest state currently, and this is the aforementioned vulnerable case (see Subfig. 6.2(d)). If n_{m_A} forwards the *Data* packet to n_{m_B} without replying a *Wait* packet back to the packet sender (n_a), this *Data* packet will be lost because n_{m_B} is in harvest state. However, this forwarding misbehavior of n_{m_A} can be detected by n_b. If n_b does not perform the SlyDog on n_{m_A}, it stays in active state and monitors the forwarding behavior of n_{m_A}. If n_{m_A} replies a *Wait* packet, it is considered as a legitimate node. Upon overhearing the *Wait* packet, n_b broadcasts a *State* packet piggybacked with its active state and stops performing the SlyDog on n_{m_A}.

Second, suppose both n_b and n_{m_B} stay in active state as shown in Subfig. 6.5(d). Since n_b is aware of the state of n_{m_B} and monitors the forwarding behavior of its one-hop neighbor nodes, n_{m_A} will behave as a legitimate node and forward a received packet to n_{m_B}. In Subfig. 6.5(e), n_b overhears the packet transmission from n_{m_A} to n_{m_B} and decides whether to perform the SlyDog on n_{m_B} based on the monitor probability of n_{m_B}, p_{m_B}. If n_b decides to perform the SlyDog, it disguises itself as an energy harvesting node and periodically broadcasts a *State* packet piggybacked with its harvest state. In Subfig. 6.5(f), when n_{m_B} overhears a harvest *State* packet from n_b, it is the aforementioned vulnerable case (see Subfigs. 6.2(c) or (f)). If n_{m_B} holds the *Data* packet without forwarding, n_b can detect this forwarding misbehavior since n_b stealthily monitors the forwarding operation of n_{m_B}. If n_{m_B} replies a *Wait* packet back to the packet sender (n_{m_A}), it is considered as a legitimate node by n_b. However, this forwarding behavior can be suspected by n_c because it is in active state and can overhear the *Wait* packet. Major operations of the SlyDog are summarized in Fig. 6.6.

```
Notations:
• $S_i$: The set of packet senders of $n_i$, e.g., $S_b$ is $[n_a]$.
• $F_i$: The set of forwardee nodes of $n_i$, e.g., $F_a$ is $[n_b, n_{m_A}]$.
• $C_{i,j}$: The set of common neighbor nodes between $n_i$ and $n_j$, e.g., $C_{b,m_A}$ is $[n_a, n_{m_B}]$.
• $G_i^*$: The set of monitored neighbor nodes of $n_i$, e.g., $G_b^*$ is $[n_{m_A}, n_{m_B}]$.
• $pkt[type, fwd, rec, seq]$, $\delta$, s, $s_a$, $s_h$, $AT[sid, rid, op, t_{begin}, t_{end}]$, $nid$, $ST[nid, s, t_{cur}]$, $MT[nid, c_{mis}, p]$, $t_{cur}$, $c_{mis}$, $p$, $sid$, $rid$, $op$, $t_{begin}$, $t_{end}$: Defined before.
Operations:
◇ $n_i$ overhears a Data packet $pkt[Data, x, y, seq]$:
  if $n_x \in S_i$ and $n_y \in G_i^*$
      for $n_z \in C_{i,y}$ and $n_z \in F_i$
          if $ST_i[z].s == s_h$ /* Forwardee node in harvest state */
              $flag_{slyA}$ = true; $vim = z$; $src = x$; $tget_{slyA} = y$;
  if $flag_{slyA}$ == true and $MT_i[tget_{slyA}].p <= rand[0, 1]$
      /* $n_i$ performs the SlyDog on $n_{tget_{slyA}}$ */
      Broadcasts bogus harvest State packet;
      Monitors forwarding behavior of $n_{tget_{slyA}}$;
◇ $n_i$ performs the SlyDog on $n_{tget_{slyA}}$: $flag_{slyA}$ = true.
  ▷: $n_i$ overhears a Data packet $pkt[Data, tget_{slyA}, vim, seq]$.
      if $ST_i[vim].s == s_h$ and $n_{vim} \in C_{i,tget_{slyA}}$
          $MT_i[tget_{slyA}].p \mathrel{+}= \delta$; $MT_i[tget_{slyA}].c_{mis} \mathrel{+}= 1$;
  ▷: $n_i$ overhears a Wait packet $pkt[Wait, tget_{slyA}, src, seq]$.
      if $ST_i[vim].s == s_h$ and $n_{vim} \in C_{i,tget_{slyA}}$
          Stops the SlyDog on $n_{tget_{slyA}}$;
      else
          $MT_i[tget_{slyA}].p \mathrel{+}= \delta$; $MT_i[tget_{slyA}].c_{mis} \mathrel{+}= 1$;
  ▷: $n_i$ does not overhear any packet within timeout period.
      $MT_i[tget_{slyA}].p \mathrel{+}= \delta$; $MT_i[tget_{slyA}].c_{mis} \mathrel{+}= 1$;
◇ $n_i$ overhears a Data packet $pkt[Data, tget_{slyA}, rec, seq]$.
  if $rec \in F_i$ and $ST_i[rec].s == s_a$
      if $MT_i[rec].p <= rand[0, 1]$
          /* Performs the SlyDog on $n_{rec}$ */
          Broadcasts bogus harvest State packet;
          Monitors forwarding behavior of $n_{rec}$;
          $flag_{slyB}$ = true; $tget_{slyB} = rec$;
◇ $n_i$ performs the SlyDog on $n_{tget_{slyB}}$: $flag_{slyB}$ = true.
  ▷: $n_i$ overhears a Data packet $pkt[Data, tget_{slyB}, nrec, seq]$.
      $AT_i[tget_{slyB}, nrec].op \mathrel{+}= 1$;
      if $t_{begin} == 0$
          $t_{begin}$ = current time;
  ▷: $n_i$ overhears a Wait packet $pkt[Wait, tget_{slyB}, tget_{slyA}, seq]$.
      Stops the SlyDog on $n_{tget_{slyB}}$;
  ▷: $n_i$ does not overhear any packet within timeout period.
      $MT_i[tget_{slyB}].p \mathrel{+}= \delta$; $MT_i[tget_{slyB}].c_{mis} \mathrel{+}= 1$;
```

Figure 6.6. The pseudo code of *SlyDog*.

6.4.2 LazyDog: Monitor-based Detection

The basic idea of LazyDog is that each node requests its one-hop neighbor node to advertise the number of packets forwarded to its two-hop neighbor nodes during a certain period of time. Since each node can count and record the number of overheard or received packets, this simple information can be used as a clue to detect the forwarding misbehavior. For example, n_b can overhear the packet transmission from n_{m_B} to n_c, but it cannot make sure whether the packet has been successfully received by n_c because n_b is not aware of the state of n_c as shown in Subfig. 6.2(e).

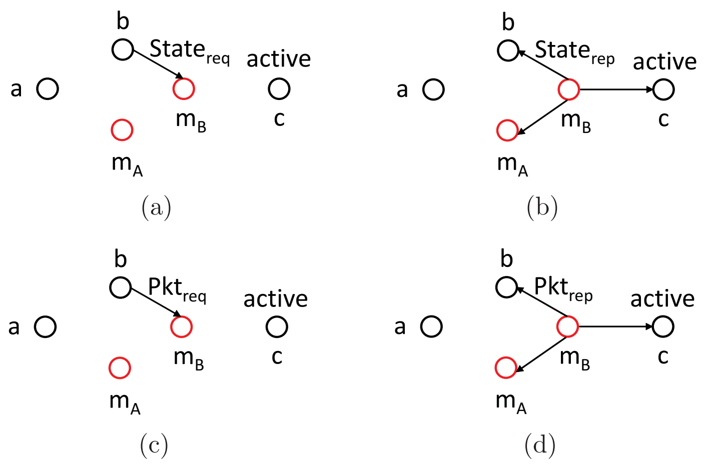


Figure 6.7. The proposed *LazyDog* detection strategy.

Detection Operations: Since n_b is not aware of the state of n_c, it requests n_{m_B} to broadcast the states of one-hop neighbor nodes by sending a $State_{req}$ packet as shown in Subfig. 6.7(a). Then n_b is aware of the active state of n_c after n_{m_B} broadcasts the $State_{rep}$ packet, which contains the states of one-hop neighbor nodes as shown in Subfig. 6.7(b). Then n_b requests n_{m_B} to advertise the number of packets forwarded to n_c during a time period, $AT_b[m_B, c].(t_{begin}, t_{end})$, by sending a Pkt_{req} packet as shown in Subfig. 6.7(c). If n_{m_B} refuses to advertise within a timeout period, n_b suspects the forwarding misbehaviors of n_{m_B} and increments $MT_b[m_B].c_{mis}$ by $AT_b[m_B, c].op$. However, if n_{m_B} advertises, this can be overheard by both n_b and n_c. Then both n_b and n_c compare the received advertisement with their number of packets overheard and number of received packet from n_{m_B} during the time period respectively as shown in Subfig. 6.7(d). If the comparison difference is greater than a predefined threshold value Det_{th} , either n_b or n_c can detect the forwarding misbehavior of n_{m_B}. Major operations of the LazyDog are summarized in Fig. 6.8.

6.5 Analysis of The Proposed Countermeasure

In this section, we analyze the detection rate for a malicious node in the the SlyDog and LazyDog, respectively. In the EYES, the forwarding misbehavior of a malicious node can be detected in the following three cases:

- An adjacent node overhears a *Data* packet which is forwarded to a node in harvest state.
- An adjacent node does not overhear the *Data* packet transmission within the maximum delay.
- An adjacent node receives an inconsistent *Data* packet forwarding summary.

In this analysis, we assume that the packet loss primarily caused by a bad channel quality, and it is independent and is given by C_{ls}. For the sake of simplicity, we conduct our analysis based on the scenarios in Figs. 6.5 and 6.7.

6.5.1 Detection Rate of SlyDog

Considering the scenarios in Subfigs. 6.5(a), (b), and (c), n_a is relaying a *Data* packet $pkt[Data, a, m_A]$ to malicious node n_{m_A}. Then n_{m_A} colludes with another malicious node n_{m_B}, which is in harvest state currently, to drop the packet without being detected. There are six different possibilities for the node n_b to detect any forwarding misbehavior:

1. n_b misses the $pkt[Data, a, m_A]$ and stays in active state;
   n_{m_A} replies the $pkt[Wait, m_A, a]$ to n_a; n_b misses the $pkt[Wait, m_A, a]$ $\Rightarrow$ normal
2. n_b misses the $pkt[Data, a, m_A]$ and stays in active state;
   n_{m_A} replies the $pkt[Wait, m_A, a]$ to n_a; n_b overhears the $pkt[Wait, m_A, a]$ $\Rightarrow$ normal
3. n_b overhears the $pkt[Data, a, m_A]$, performs the SlyDog on n_{m_A}, and changes to harvest state; n_{m_A} forwards the $pkt[Data, m_A, m_B]$ to n_{m_B}; n_b misses the $pkt[Data, m_A, m_B]$ $\Rightarrow$ detected as a packet loss

```
Notations:
• $RP_j$: The set of the number of received Data packets of $n_j$, e.g., $RP_c[m_B]$ is the number
of received Data packets from $n_{m_B}$.
• $FP_i$: The set of the number of forwarded Data packets of $n_i$, e.g., $FP_{m_B}[c]$ is the number
of forwarded Data packet to $n_c$ from $n_{m_B}$.
• $DN_i$: The set of one-hop neighbor nodes of $n_i$.
• $THN_i$: The set of two-hop neighbor nodes of $n_i$.
• $SL_i$: The set of current state of one-hop neighbor nodes of $n_i$.
• $LD_i$: The set of currently active two-hop neighbor nodes of $n_i$.
• $t_{out\_lazy}$: The LazyDog detection window interval.
• $State_{req}$, $State_{rep}$, $Pkt_{req}$, $Pkt_{rep}$ and $Det_{th}$: Defined before.
Operations:
◇ $n_i$ starts the LazyDog detection: $t_{out\_lazy}$ expires.
  Randomly selects $n_t$, $n_t \in DN_i$ and $ST_i[t].s == s_a$;
  Forwards the state request packet $pkt[State_{req}, i, t, seq]$ to $n_t$;
◇ $n_i$ overhears the state reply packet $pkt[State_{rep}, t, SL_t, seq]$.
  for $n_x \in THN_i$
    if $SL_t[x] == s_a$
      $LD_i = LD_i \cup n_x$;
      $flag_{lazy}$ = true; /* Performs the LazyDog on $n_t$ */
  if $flag_{lazy}$ == true
    Forwards packet number request packet $pkt[Pkt_{req}, i, t, seq]$ to $n_t$;
◇ $n_i$ performs the LazyDog on $n_t$: $flag_{lazy}$ == true.
  ▷: $n_i$ doesn't overhear the reply packet $pkt[Pkt_{rep}, t, FP_t, seq]$
    for $n_x \in LD_i$
      $MT_i[t].c_{mis}$ += $AT_i[t, x].op$; $AT_i[t, x].op = 0$;
  ▷: $n_i$ overhears the reply packet $pkt[Pkt_{rep}, t, FP_t, seq]$.
    for $n_x \in LD_i$
      if $(|FP_t[x] - AT_i[t, x].op|) \leq Det_{th}$, $FP_t[x] \in FP_t$
        $AT_i[t, x].op = 0$; /* $n_t$ behaves well on $n_x$ */
      else /* $n_i$ detects the forwarding misbehavior of $n_t$ */
        $MT_i[t].c_{mis}$ += $(|FP_t[x] - AT_i[t, x].op|)$;
◇ $n_x$ overhears the reply packet $pkt[Pkt_{rep}, t, FP_t, seq]$, $n_x \in THN_i$:
  if $(|RP_x[t] - FP_t[x]|) \leq Det_{th}$, $FP_t[x] \in FP_t$
    Discards the packet $pkt[Pkt_{rep}, t, FP_t, seq]$;
  else
    $MT_x[t].c_{mis}$ += $(|RP_x[t] - FP_t[x]|)$;
```

Figure 6.8. The pseudo code of *LazyDog*.

4. n_b overhears the $pkt[Data, a, m_A]$, performs the SlyDog on n_{m_A}, and changes to harvest state; n_{m_A} forwards the $pkt[Data, m_A, m_B]$ to n_{m_B}; n_b overhears the $pkt[Data, m_A, m_B]$ $\Rightarrow$ detected by the SlyDog

5. n_b overhears the $pkt[Data, a, m_A]$ but does not perform the SlyDog on n_{m_A}, stays in active state; n_{m_A} replies the $pkt[Wait, m_A, a]$ to n_a; n_b misses the $pkt[Wait, m_A, a]$ $\Rightarrow$ detected as a packet loss

6. n_b overhears the $pkt[Data, a, m_A]$ but does not perform the SlyDog on n_{m_A}, stays in active state; n_{m_A} replies the $pkt[Wait, m_A, a]$ to n_a; n_b overhears the $pkt[Wait, m_A, a]$ $\Rightarrow$ normal

In the SlyDog, cases 1), 2) and 6) look normal and they represent the legitimate node's operations. However, we classify cases 3), 4) and 5) as abnormal because the malicious node can be detected.

The probability of cases 3), 4) and 5) can be expressed as,

$$P_3 = (1 - C_{ls}) \cdot MT[m_A].p \cdot C_{ls} \tag{6.1}$$

$$P_4 = (1 - C_{ls}) \cdot MT[m_A].p \cdot (1 - C_{ls}) \tag{6.2}$$

$$P_5 = (1 - C_{ls}) \cdot (1 - MT[m_A].p) \cdot C_{ls} \tag{6.3}$$

The probability of existing at least one active adjacent node (i.e., n_b) of n_a, n_{m_A}, and n_{m_B} can be expressed as,

$$P_G = 1 - (1 - P_a)^r \tag{6.4}$$

Here, r is the total number of adjacent nodes of n_a, n_{m_A}, and n_{m_B}. P_a is the probability of node staying in active state. Thus, the detection rate for a malicious node n_{m_A} can be expressed as,

$$\begin{aligned} P_{dt_{sly_A}} &= P_G \cdot (P_3 + P_4 + P_5) \\ &= P_G \cdot (1 - C_{ls}) \cdot (MT[m_A].p \cdot (1 - C_{ls}) + C_{ls}) \end{aligned} \tag{6.5}$$

We also consider the cases shown in Subfigs. 6.5(d), (e), and (f), where n_{m_B} is in active state. There are additional nine different possibilities for n_b to detect any forwarding misbehavior:

7. n_b misses the $pkt[Data, a, m_A]$ and stays in active state; n_{m_A} forwards the $pkt[Data, m_A, m_B]$ to n_{m_B}; n_b misses the $pkt[Data, m_A, m_B]$ and stays in active state; n_{m_B} forwards the $pkt[Data, m_B, c]$ to n_c; n_b misses the $pkt[Data, m_B, c]$ $\Rightarrow$ normal

8. n_b misses the $pkt[Data, a, m_A]$ and stays in active state; n_{m_A} forwards the $pkt[Data, m_A, m_B]$ to n_{m_B}; n_b misses the $pkt[Data, m_A, m_B]$ and stays in active state; n_{m_B} forwards the $pkt[Data, m_B, c]$ to n_c; n_b overhears the $pkt[Data, m_B, c]$ $\Rightarrow$ normal

9. n_b misses the $pkt[Data, a, m_A]$ and stays in active state; n_{m_A} forwards the $pkt[Data, m_A, m_B]$ to n_{m_B}; n_b overhears the $pkt[Data, m_A, m_B]$ and performs the SlyDog on n_{m_B}, changes to harvest state; n_{m_B} holds the packet $\Rightarrow$ detected by the SlyDog

10. n_b misses the $pkt[Data, a, m_A]$ and stays in active state; n_{m_A} forwards the $pkt[Data, m_A, m_B]$ to n_{m_B}; n_b overhears the $pkt[Data, m_A, m_B]$ but does not perform the SlyDog on n_{m_B} and stays in active state;
n_{m_B} forwards the $pkt[Data, m_B, c]$ to n_c; n_b misses the $pkt[Data, m_B, c]$ $\Rightarrow$ detected as a packet loss

11. n_b misses the $pkt[Data, a, m_A]$ and stays in active state; n_{m_A} forwards the $pkt[Data, m_A, m_B]$ to n_{m_B}; n_b overhears the $pkt[Data, m_A, m_B]$ but does not perform the SlyDog on n_{m_B} and stays in active state;
n_{m_B} forwards the $pkt[Data, m_B, c]$ to n_c; n_b overhears the $pkt[Data, m_B, c]$ $\Rightarrow$ normal

12. n_b overhears the $pkt[Data, a, m_A]$ and stays in active state; n_{m_A} forwards the $pkt[Data, m_A, m_B]$ to n_{m_B}; n_b misses the $pkt[Data, m_A, m_B]$ $\Rightarrow$ detected as a packet loss

13. n_b overhears the $pkt[Data, a, m_A]$ and stays in active state; n_{m_A} forwards the $pkt[Data, m_A, m_B]$ to n_{m_B}; n_b overhears the $pkt[Data, m_A, m_B]$, performs the SlyDog on n_{m_B}, and changes to harvest state; n_{m_B} holds the packet $\Rightarrow$ detected by the SlyDog

14. n_b overhears the $pkt[Data, a, m_A]$ and stays in active state; n_{m_A} forwards the $pkt[Data, m_A, m_B]$ to n_{m_B}; n_b overhears the $pkt[Data, m_A, m_B]$ but does not perform the SlyDog on n_{m_B} and stays in active state;
    n_{m_B} forwards the $pkt[Data, m_B, c]$ to n_c; n_b misses the $pkt[Data, m_B, c] \Rightarrow$ detected as a packet loss

15. n_b overhears the $pkt[Data, a, m_A]$ and stays in active state; n_{m_A} forwards the $pkt[Data, m_A, m_B]$ to n_{m_B}; n_b overhears the $pkt[Data, m_A, m_B]$ but does not perform the SlyDog on n_{m_B} and stays in active state;
    n_{m_B} forwards the $pkt[Data, m_B, c]$ to n_c; n_b overhears the $pkt[Data, m_B, c] \Rightarrow$ normal

We classify cases 9), 10), 12), 13), and 14) as abnormal scenarios because the malicious node can be detected. The probability of cases 9), 10), 12), 13), and 14) can be expressed as,

$$P_9 = C_{ls} \cdot (1 - C_{ls}) \cdot MT[m_B].p \tag{6.6}$$

$$P_{10} = C_{ls} \cdot (1 - C_{ls}) \cdot (1 - MT[m_B].p) \cdot C_{ls} \tag{6.7}$$

$$P_{12} = (1 - C_{ls}) \cdot C_{ls} \tag{6.8}$$

$$P_{13} = (1 - C_{ls}) \cdot (1 - C_{ls}) \cdot MT[m_B].p \tag{6.9}$$

$$P_{14} = (1 - C_{ls}) \cdot (1 - C_{ls}) \cdot (1 - MT[m_B].p) \cdot C_{ls} \tag{6.10}$$

Thus, the detection rate for a malicious node n_{m_B} can be expressed as,

$$\begin{aligned} P_{dt_{sly_B}} &= P_G \cdot (P_9 + P_{10} + P_{12} + P_{13} + P_{14}) \\ &= P_G \cdot (1 - C_{ls}) \cdot (2C_{ls} + (1 - C_{ls}) \cdot MT[m_B].p) \end{aligned} \tag{6.11}$$

Finally, the detection rate of the SlyDog can be expressed as,

$$P_{detect\_SlyDog} = P_{dt_{sly_A}} + P_{dt_{sly_B}} \tag{6.12}$$

6.5.2 Detection Rate of LazyDog

In the LazyDog, as shown in Fig. 6.7 for detection, an adjacent node (i.e., n_b or n_c) of malicious node (i.e., n_{m_B}) compares its number of overheard or received packets counted within a time period (t_{begin}, t_{end}) with the number of announcements counted by the malicious node. If the difference is greater than a predefined threshold value Det_{th}, the forwarding misbehaviors of malicious node are detected. Within (t_{begin}, t_{end}), suppose n_{m_B} forwards F_{m_B} number of packets to n_c and F_{m_B} is greater than Det_{th}. Otherwise, the detection rate is zero. We also assume that O_b is the number of packets overheard by n_b sent from n_{m_B} to n_c and R_c is the number of packets received by n_c sent from n_{m_B}, respectively. D_c is the number of dropped packets by forwarding to n_c in harvest state. Thus, the maximum number of packets received by n_c is (F_{m_B}−D_c), which is represented by M_{rec,n_c}.

A malicious node n_{m_B} has one of two options to announce its counter value of forwarded packets to n_c: (i) M_{rec,n_c}, to match with the counter value of the number of received packets at n_c; or (ii) F_{m_B}, to match with the counter value of the number of overheard packets at n_b. For the sake of simplicity, we assume that the malicious node n_{m_B} randomly chooses the option to match with the counter value at n_c or n_b.

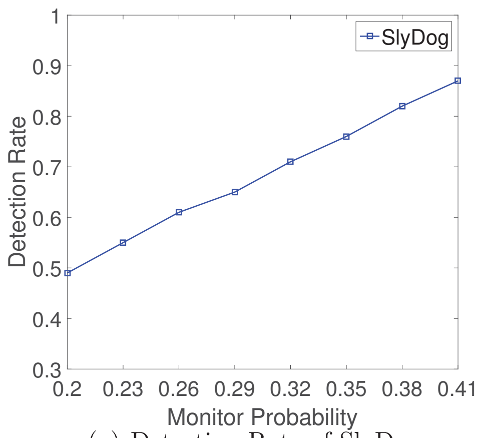
In the view point of n_b, it could be in one of fives states based on the counter value announced by n_{m_B}:

1. n_{m_B} announces F_{m_B}:
   (a) $(F_{m_B} - Det_{th}) \leq O_b \leq F_{m_B} \Rightarrow$ normal
   (b) $0 \leq O_b < (F_{m_B} - Det_{th}) \Rightarrow$ not sure
2. n_{m_B} announces M_{rec,n_c}:
   (a) $M_{rec,n_c} < O_b \leq F_{m_B} \Rightarrow$ detected by the LazyDog
   (b) $(M_{rec,n_c} - Det_{th}) \leq O_b \leq M_{rec,n_c} \Rightarrow$ good
   (c) $0 \leq O_b < (M_{rec,n_c} - Det_{th}) \Rightarrow$ not sure

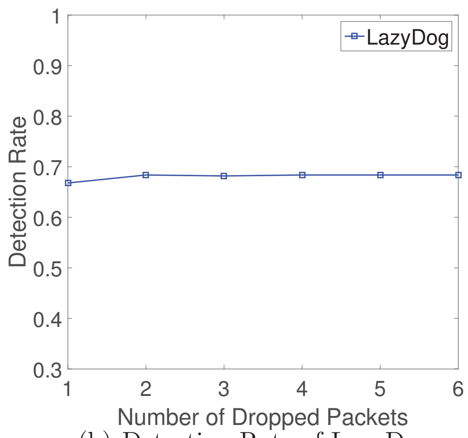
The probability that n_b overhears at least M_{rec,n_c}+1 packets (case *2.a*) can be expressed as,

$$P_{2.a} = \sum_{i=M_{rec,n_c}+1}^{F_{m_B}} \binom{F_{m_B}}{i} (1 - C_{ls})^i \cdot (C_{ls})^{F_{m_B}-i} \tag{6.13}$$

In the view point of n_c, however, it could be in one of four states based on the counter announced by n_{m_B}:



(a) Detection Rate of SlyDog



(b) Detection Rate of LazyDog

Figure 6.9. The detection rate against monitor probability and the number of dropped packets.

3. n_{m_B} announces F_{m_B}:
   (a) $(F_{m_B} - Det_{th}) \leq R_c \leq F_{m_B} \Rightarrow$ normal
   (b) $0 \leq R_c < (F_{m_B} - Det_{th}) \Rightarrow$ detected by the LazyDog
4. n_{m_B} announces M_{rec,n_c}:
   (a) $(M_{rec,n_c} - Det_{th}) \leq R_c \leq M_{rec,n_c} \Rightarrow$ normal
   (b) $0 \leq R_c < (M_{rec,n_c} - Det_{th}) \Rightarrow$ not sure

The probability that n_c receives less than $(F_{m_B} - Det_{th})$ number of packets (case *3.b*) can be expressed as,

$$P_{3.b} = \sum_{i=0}^{\eta=(F_{m_B}-Det_{th}-1)} \binom{\eta}{i} (1 - C_{ls})^i \cdot (C_{ls})^{\eta-i} \tag{6.14}$$

Thus, the detection rate of the LazyDog can be expressed as,

$$P_{detect\_LazyDog} = \frac{P_{2.a} + P_{3.b}}{2} \tag{6.15}$$

In Fig. 6.9, based on the aforementioned analysis, we show the impact of monitor probability and the number of dropped packets on the detection rate of SlyDog and

LazyDog, respectively. Here, we use the following parameters: $r = 3$, $P_a = 67\%$, $C_{ls} = 5\%$, $Det_{th} = 2$ and $F_{m_B} = 10$. In Subfig. 6.9(a), the detection rate of SlyDog increases linearly as the monitor probability increases, because the node has more chances to perform the SlyDog on the suspected node and detects more forwarding misbehaviors with larger monitor probability. In Subfig. 6.9(b), the number of dropped packets does not affect the detection rate of LazyDog much. This is because it is hard to detect whether the packets are dropped by malicious node or lost during the transmission due to a bad channel quality.

Table 6.1. Simulation Parameters of EYES

Parameter	**Value**
Network area	200×200 m^2
Number of nodes	150
Number of malicious nodes along the route	1 or 2
Channel error rate	5%
Radio data rate	250 Kbps
Packet injection rate	0.33 or 0.66 pkt/sec
Packet size	1 KByte
Packet drop rate of HCD and Watchdog	30%
Radio range	12.3 m
Radio model	CC2420
Simulation time	1000 secs
Active time period	50 to 80 secs
Harvest time period	15 to 40 secs

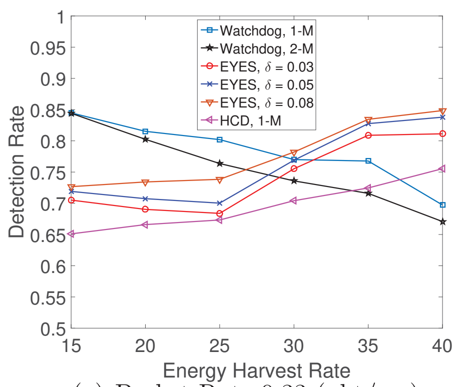
6.6 Simulation Testbed

We conduct extensive simulation experiments using the OMNeT++ [41] to evaluate the performance of proposed approach. A 200×200 (m^2) rectangular network area is considered, where 150 nodes are uniformly distributed. The communication range of each node is 12.3 (m). The radio model simulates CC2420 with a normal data rate of 250 Kbps [42], and the channel error rate is set to 5%. A single node generates data traffic with injection rate 0.33 or 0.66 (pkt/sec) and the data packet size is 1 KByte. The inter-arrival time of traffic is assumed to be exponentially distributed. The total simulation time is 1,000 seconds. The periods of active and harvest states vary between 50 to 80 (secs) and 15 to 40 (secs), respectively. In the proposed approach, two malicious nodes are consecutively located along the forwarding path between packet sender and the sink, in which malicious nodes are assumed to monitor network traffic and local network condition, and then perform selective forwarding attacks cooperatively without being detected.

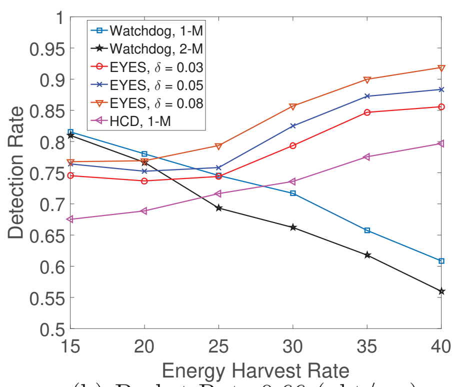
For performance comparison, we compare our proposed schemes *SlyDog* and *LazyDog*, called *EYES*, with a hop-by-hop cooperative detection scheme, called *HCD* [38], which is the first countermeasure to selective forwarding attack in EHNets. The proposed EYES is also compared with the well-known *Watchdog* [17]. We adjust and implement the Watchdog with a single and two consecutively located malicious nodes, denoted as *1-M* and *2-M*, respectively. Here, a malicious node is set to randomly drop received packets with 30% dropping rate in the HCD and Watchdog. The simulation parameters are summarized in Table 6.1.

6.7 Simulation Results

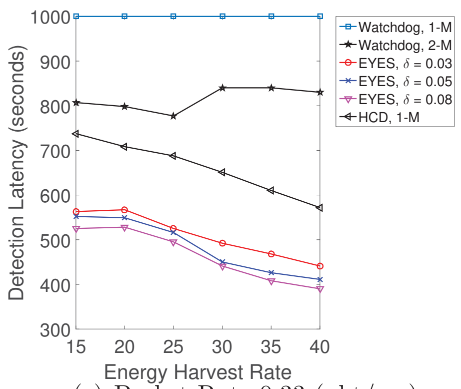
Detection Rate: We first measure the detection rate by changing harvest time (t_h), packet injection rate (r_{pkt}) and δ in Subfigs. 6.10(a) and (b). In Subfig. 6.10(a), under $r_{pkt} = 0.33$ (pkt/sec), as t_h increases, the detection rates of both EYES and HCD increase while that of the Watchdog decreases. In the Watchdog, each node passively changes its state between active and harvest and monitors the forwarding behavior only during active state. As t_h increases, more nodes stay in harvest state for longer time period and the detection rate decreases even though malicious nodes can drop packets with 30% dropping rate. Thus, lower detection rate is observed with two malicious nodes located consecutively in the forwarding path, because more packets are dropped by two malicious nodes and these forwarding misbehaviors cannot be detected. Both EYES and HCD show higher detection rate than that of the Watchdog in high t_h. This is because the SlyDog can actively disguise each node as an energy harvesting node and monitor any forwarding behavior of its adjacent nodes, or exchange the trace information with its adjacent nodes, and detect more forwarding misbehaviors. In particular, the EYES shows higher detection rate than that of the HCD because prior uncertain packet forwarding operations can be verified by the LazyDog, and more forwarding misbehaviors can be detected. In the HCD, the detection rate increases slowly compared to that of the EYES, because the forwarding



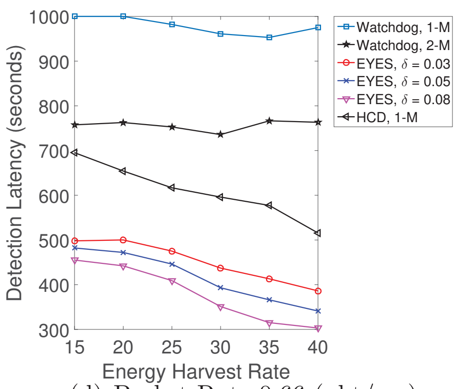
(a) Packet Rate 0.33 (pkt/sec)



(b) Packet Rate 0.66 (pkt/sec)



(c) Packet Rate 0.33 (pkt/sec)



(d) Packet Rate 0.66 (pkt/sec)

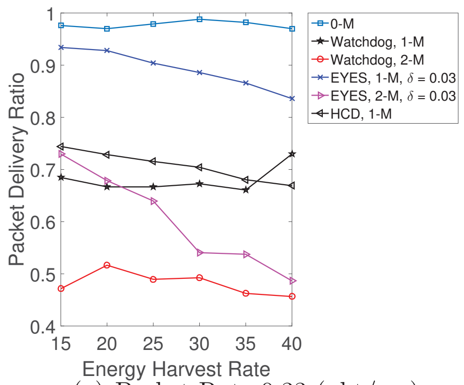
Figure 6.10. The performance impact against energy harvest rate, number of malicious nodes, packet injection rate and δ.

probability of the malicious node is reduced whenever a forwarding misbehavior is detected. Since the malicious node seldom receives the packet, the forwarding misbehaviors of malicious node can be significantly reduced. In the EYES, the detection rate increases as δ increases. This is because monitor probability (p) increases quickly with larger δ and thus, nodes have more chances to disguise themselves as an energy harvesting node and detect more forwarding misbehaviors. In Subfig. 6.10(b), overall detection rates of the EYES and HCD increase with r_{pkt} = 0.66 (pkt/sec), because more packets are forwarded to malicious node with larger r_{pkt}, and then more packets are dropped by malicious node but these forwarding misbehaviors can be detected by the EYES and HCD. The EYES still shows the best performance as t_h increases compared to that of HCD and Watchdog.

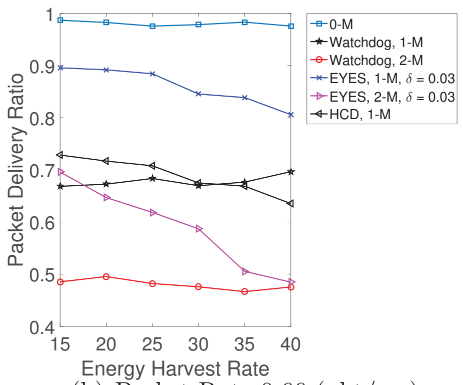
Detection Latency: Second, the detection latency is measured by changing t_h, r_{pkt}, and δ in Subfigs. 6.10(c) and (d). As t_h increases, more nodes are in harvest state and more vulnerable cases are witnessed, i.e., Subfigs. 6.2(c), (d), (e) and (f). In Subfig. 6.10(c), the EYES achieves the lowest detection latency compared to the HCD and Watchdog. This is because adjacent nodes of malicious nodes can disguise themselves as an energy harvesting node, counterfeit vulnerable cases, and finally detect more forwarding misbehaviors. With higher δ, the detection latency decreases because nodes can frequently disguise themselves and monitor any forwarding operation. The LazyDog also helps to reduce the detection latency by detecting the uncertain forwarding behavior of malicious nodes. The EYES can also quickly isolate malicious nodes in the network. Unlike the proposed EYES, the HCD shows higher detection latency for entire t_h. In the HCD, each packet sender can detect the forwarding misbehavior of suspected node only after receiving a *Mode*[2] packet broadcasted from its adjacent nodes. Upon receiving the *Mode* packet, the sender updates the states of its neighbor nodes and searches whether there was any forwarding operation while any forwardee node was in harvest state. The Watchdog shows the highest detection latency because nodes can only detect the forwarding misbehavior in active state. In Subfig. 6.10(d), under higher packet injection rate 0.66 (pkt/sec), overall detection latencies decrease. However, the best performance is still achieved in the EYES and the detection latency decreases quickly compared to that of the HCD and Watchdog.

Packet Delivery Ratio: Third, we measure the packet delivery ratio (PDR) by varying t_h, r_{pkt} and δ in Fig. 6.11. For the purpose of performance comparison, we deploy a no malicious node case under different r_{pkt}, denoted as *0-M*, to see the upper bound of average PDR (about 98% or more). In this case, every node cooperatively forwards the received packet to the sink. The Watchdog is not sensitive to t_h and packet injection rate but to 30% dropping rate, and the PDR is fluctuating about 68% and 47% with a single (*1-M*) and two (*2-M*) malicious nodes, respectively. This is because the malicious node can stay in active state for an extended period but only randomly drops the packet with 30% dropping rate. In Subfig. 6.11(a), the EYES with one (*1-M*) and two (*2-M*) malicious nodes shows higher and lower PDR

[2]In [38], a node broadcasts a *Mode* packet whenever it changes its state. This is similar to a *State* packet in this research.



(a) Packet Rate 0.33 (pkt/sec)

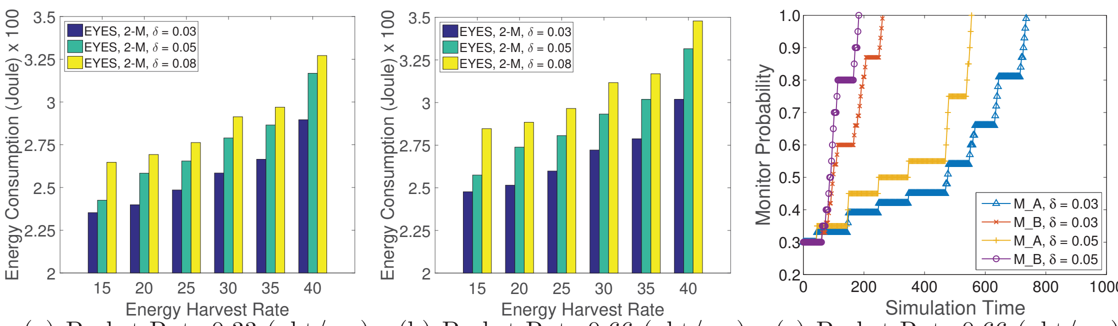


(b) Packet Rate 0.66 (pkt/sec)

Figure 6.11. The packet delivery ratio against energy harvest rate, packet injection rate and δ.

than that of the HCD with a single malicious node, respectively. This is because two malicious nodes located consecutively can collude together and intentionally drop more packets without being detected. Unlike the Watchdog, the HCD can reduce the number of forwarding misbehaviors by decreasing the probability of malicious node being chosen as a forwardee node. Thus, the HCD shows higher PDR than that of the Watchdog with a single malicious node. The HCD also shows lower PDR than that of the EYES with a single malicious node. This is because the malicious node only performs the undetected forwarding misbehavior in the EYES, while the malicious node in the HCD randomly drops the received packet with 30% dropping rate. In Subfig. 6.11(b), overall PDRs decrease with higher packet injection rate 0.66 (pkt/sec) because more packets are dropped by malicious nodes due to more number of generated packets in the network.

Energy Consumption: Fourth, we measure energy consumption in terms of the number of overheard forwarding misbehaviors of malicious nodes [50] in Subfigs. 6.12(a) and (b). An overheard forwarding misbehavior occurs when a malicious node forwards a packet to a legitimate node which is in harvest state, resulting in packet loss. As t_h increases in Subfig. 6.12(a), malicious nodes have more chances to forward packets to the nodes in harvest state and reveal forwarding misbehaviors frequently. However, this forwarding misbehavior can be detected by the SlyDog and then the energy consumption of detection increases. With larger δ, nodes have more chances



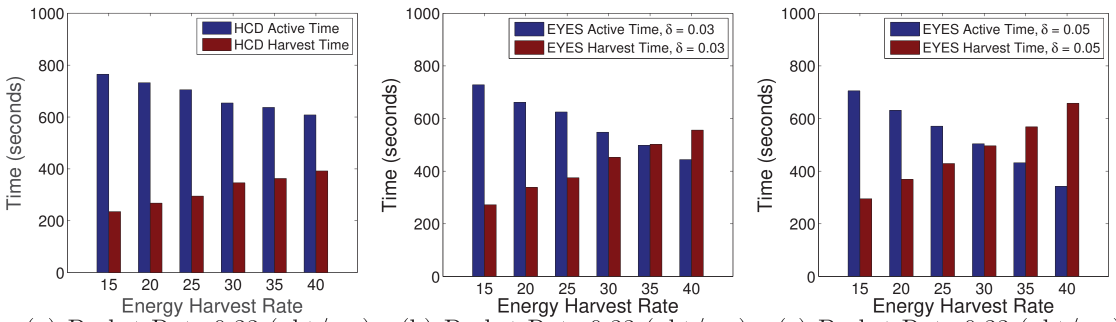
(a) Packet Rate 0.33 (pkt/sec) (b) Packet Rate 0.66 (pkt/sec) (c) Packet Rate 0.66 (pkt/sec)

Figure 6.12. The energy consumption and monitor probability against energy harvest rate, packet injection rate and δ.

to disguise themselves as an energy harvesting node, monitor any forwarding misbehavior, and consume more energy. In Subfig. 6.12(b), more energy consumption is observed with higher packet injection rate, because more packets are dropped by malicious nodes but these forwarding misbehaviors can be detected, which consumes more energy. Thus, the EYES can efficiently utilize the harvested energy to monitor and detect the forwarding misbehaviors of malicious nodes.

Monitor Probability: Fifth, we observe the changes of monitor probability (p) in the presence of two malicious nodes with different weights ($\delta = 0.03$ or 0.05) over simulation period in the EYES as shown in Subfig. 6.12(c). Whenever a node detects a forwarding misbehavior, it increases the monitor probability (p) of suspected node by δ. With larger δ, malicious nodes are monitored more often and thus, their forwarding misbehaviors are detected that leads to quick isolation from the network. For example, the monitor probabilities of malicious nodes n_{m_A} and n_{m_B} (see Subfig. 6.5(a)) reach to 1.0 at 580 and 180 seconds with $\delta = 0.05$, respectively. This indicates that any forwarding operation of two malicious nodes is suspected and monitored. Note that the monitor probability of n_{m_B} reaches to 1.0 earlier than that of n_{m_A} with different δ. Since the prior packet sender of n_{m_B} is n_{m_A}, n_{m_B} tends to perform more forwarding misbehaviors for possible collusion.

Impact of Harvest Time and δ: Finally, we measure the total time periods of nodes staying in active and harvest states in the HCD and EYES by changing t_h and δ over the simulation period as shown in Subfigs. 6.13(a), (b), and (c). In



(a) Packet Rate 0.33 (pkt/sec) (b) Packet Rate 0.33 (pkt/sec) (c) Packet Rate 0.33 (pkt/sec)

Figure 6.13. The performance of total active and harvest time periods against energy harvest rate and δ.

the HCD, since each node does not perform any monitoring operation during the harvest state, total harvest time period increases linearly as t_h increases in Subfig. 6.13(a). In the EYES, however, total harvest time period in Subfig. 6.13(b) increases quickly compared to that of the HCD in Subfig. 6.13(a). Since nodes actively disguise themselves as an energy harvesting node and pretend not to overhear, longer harvest time period is observed in the EYES. With larger $\delta = 0.05$ in Subfig. 6.13(c), more harvest time period is observed than that of $\delta = 0.03$ in Subfig. 6.13(b). This is because larger δ increases the monitor probability quickly, more nodes frequently disguise themselves as an energy harvesting node.

In this section, we discuss the proposed approach in terms of features and constrains and explore a possible extension. We also investigate its immunity to other attacks in EHNets.

6.8 Features and Constraints

The EYES is designed based on three desirable features. First, each node can actively disguise itself as an energy harvesting node to stealthily monitor the forwarding operation of its adjacent nodes. This active detection technique can detect more forwarding misbehaviors within a short time period, then the malicious node can be quickly excluded from participating the forwarding operation in the network. Second, monitor probability indicates how actively a node monitors the forwarding operation of suspected node, and it increases when a forwarding misbehavior is detected. This incremental monitor probability can significantly increase detection rate and reduce detection latency simultaneously, because nodes have more chances to disguise themselves as an energy harvesting node and detect more forwarding misbehaviors. Third, since the node cannot make sure whether the forwarded packet from its adjacent node has been successfully received by its two-hop neighbor node, each node periodically requests its adjacent node to broadcast the number of forwarded packets to its two-hop neighbor node. Thereby, any prior uncertain forwarding operations can be verified.

However, there are a few constraints that need to be further considered. First, the major detection operation of the SlyDog is based on an implicit monitoring technique. In a sparse network, for example in Fig. 6.5, if n_a only has one forwarding node n_{m_A} which colludes with n_{m_B}, it would be hard to detect any forwarding misbehavior of n_{m_A}. This is because there are no other neighbor nodes except for n_a to observe the forwarding misbehavior of n_{m_A}. Second, due to the nature of charge-and-spend energy harvesting policy, malicious nodes still have a chance to perform forwarding misbehaviors without being detected, if their neighbor node has to switch to real harvest state for energy capture.

6.9 Potential Enhancements

In this research, we plan to explore a possible extension to see the full potential of the proposed approach.

6.9.1 Dummy Packets

In the SlyDog, each node actively pretends not to overhear on-going communication of its adjacent nodes, but in fact monitors the forwarding activities to detect a lurking deep malicious node. The monitor probability of the suspected node increases if the forwarding misbehavior is detected, and the suspected node has more chances to be monitored later. Thus, we plan to extend the SlyDog by making the packet sender intentionally distribute dummy packets to the suspected node when there is no on-going packet transmission. Thus, we can lure the suspected node to show its forwarding misbehavior for detection.

6.9.2 Bypass Retransmission

We also plan to deploy a bypass retransmission technique in the SlyDog to quickly recover unexpected packet losses due to the forwarding misbehavior. For example, if a node detects a forwarding misbehavior or packet drop from the suspected node, it retransmits its cached data packet by selecting an alternative forwarding path [52, 53] to avoid the suspected node.

6.10 Applicability to Other Attacks

We further investigate the proposed approach whether it is immune to two other well-known attacks: (i) limited transmission power attack; and (ii) receiver collisions attack [17].

6.10.1 Limited Transmission Power Attack

A malicious node may drop a packet on purpose by transmitting it with reduced transmission power to exclude the legitimate next-hop node from its communication range. This attack is similar to a selective forwarding attack and it can be detected by the EYES. For example, in Subfig. 6.5(e), suppose n_{m_A} forwards a data packet to n_{m_B}. n_b overhears this packet transmission, chooses not to perform the SlyDog on n_{m_B}, and stays in active state. Then n_{m_B} may forward the packet by carefully reducing the communication range that does not reach to n_c but can be overheard by n_b. In the LazyDog, since n_b periodically requests its adjacent node (i.e, n_{m_B}) to advertise the number of packets forwarded to its two-hop neighbor node (i.e., n_c), this forwarding misbehavior can be detected by either n_b or n_c through simple comparison.

6.10.2 Receiver Collisions Attack

A malicious node may create a packet collision at the receiver on purpose by simultaneously sending any packet with the packet sender. It is not trivial to avoid receiver collisions attack but this attack can be detected by the EYES. For example, in Subfig. 6.5(d), suppose n_a sends a data packet to n_{m_A} and n_{m_B} also simultaneously sends any packet to n_{m_A}. Then n_{m_A} fails to receive the packet due to the collision. In the EYES, after n_b overhears the packet transmission from n_a to n_{m_A}, n_b will monitor the following forwarding operation of n_{m_A} no matter whether it performs the SlyDog on n_{m_A}. Since the packet is lost at n_{m_A}, n_b cannot overhear it forwarded from n_{m_A} before its timer expires. Thus, n_b will prosecute the forwarding misbehavior of n_{m_A}.

6.11 Summary

In this research, we investigated the forwarding misbehavior and its countermeasure in the realm of EHNets. Under the charge-and-spend harvesting policy, a set of adversarial scenarios is created and analyzed, and its potential vulnerabilities are also identified. We proposed a countermeasure, called *EYES*, to efficiently detect the forwarding misbehaviors of multiple malicious nodes in the EHNets. The EYES is the combination of inducement- and monitor-based approaches to quickly identify the lurking deep malicious nodes and isolate them from the network. Extensive simulation results indicate that the proposed countermeasure can improve performance in terms of detection rate, detection latency, and PDR compared to prior approaches, the HCD and Watchdog. To see the full potential of the proposed techniques, we plan to investigate a light-weight countermeasure approach [30] to detect the forwarding misbehavior of malicious nodes in the EHNets. Unlike the proposed countermeasure, we try to minimize the number of nodes involved in monitoring the forwarding operations of adjacent nodes. We also plan to design an analytical model and focus on the false detection rate.

CHAPTER 7
CONCLUSION AND FUTURE WORK

Due to the unavoidable battery replacement or replenishment, diverse energy harvesting techniques have been integrated with Wireless Sensor Networks (WSNs) to overcome limited battery power and extend the network lifetime. However, variable transmission power levels based on non-uniform energy harvesting rates can incur asymmetric links. Due to the lack of centralized coordination, physical protection, and security requirements of inherent network protocols, wireless sensor networks (WSNs) are vulnerable to diverse denial-of-service (DoS) attacks that primarily target service availability by disrupting network routing protocols or interfering with on-going communications. This dissertation research proposes the algorithms and communication protocols as a holistic approach to the exploitation of energy harvesting motivated networks. We investigate four major research issues. First, light-weight forwarding protocols are proposed to reliably deliver sensory data over time-varying asymmetric links in EHNets. A weighted confirmation scheme, a lazy confirmation scheme, and an asymmetric link aware backoff mechanism are suggested. We evaluated their performance through extensive simulation experiments, compared them with an modified conventional explicit acknowledgment scheme, and showed that the proposed forwarding protocols is a viable approach in EHNets. Second, we propose a light-weight countermeasure, called SCAD, to a selective forwarding attack in resource-constrained wireless sensor networks (WSNs), where a randomly selected single checkpoint node is deployed to detect forwarding misbehaviors. The proposed countermeasure is integrated with timeout and hop-by-hop retransmission techniques to efficiently cover unexpected packet losses due to the forwarding misbehavior or bad channel quality. In the SCAD, a single checkpoint-assisted approach incorporated with timeout and retransmission techniques can efficiently improve the detection rate as well as reduce the energy consumption, false detection rate, and successful drop rate. The SCAD can achieve more than 90% PDR with less energy consumption compared to prior CHEMAS and CAD schemes. Third, we propose a new countermeasure, called camouflage-based active detection, to a selective forwarding attack in EHNets. Four adversarial scenarios motivated by energy harvesting and their potential forwarding vulnerabilities are also analyzed. Extensive simulation results indicate that the proposed countermeasure achieves better performance in terms of detection rate and detection latency compared to the existing hop-by-hop cooperative detection scheme, and suggests a new approach to detect lurk deep malicious nodes in EHNets. Finally, we further extend the camouflage-based active detection to monitor multiple malicious nodes and to detect the forwarding misbehaviors of lurking deep malicious nodes. This countermeasure consists of SlyDog and LazyDog schemes and cooperatively detects the forwarding misbehavior. The advantages of these techniques are demonstrated through extensive simulation experiments and mathematical analysis. Extensive simulation results indicate that the proposed countermeasure can improve performance in terms of detection rate, detection latency, and PDR compared to prior approaches, the HCD and Watchdog.

As future work, we plan to work on the following topics:

- Although diverse environmental energy harvesting techniques have been well studied in civil and mechanical engineering, the design of energy harvesting sensitive communication algorithms and protocols embedded to the link layer is still in its infancy. In this topic, I plan to investigate the Power Saving Mechanism incorporating with energy harvesting in the IEEE 802.11 Medium Access Control (MAC) layer specification, and propose a novel energy harvesting aware PSM protocol. Note that the 802.11 PSM is originally designed for single-hop wireless Local Area Networks (LANs) without considering rechargeable batteries. The primary goal of this research is to shift the paradigm of energy management from conserving limited battery energy to maximizing the utilization of harvested energy, and ultimately improve the network performance.

- One of the unique characteristics of sensor network is having mobile nodes. A Mobile Ad Hoc Network (MANET) is a collection of mobile nodes that can communicate with each other without the use of a predefined infrastructure or centralized administration. However, MANET is susceptible to selective blackhole attack, which can be easily launched on reactive protocols such as ad hoc on-demand distance vector (AODV) and dynamic source routing (DSR). In this research, I plan to investigate an active detection scheme based on DSR routing protocol, where a source node actively creates a fictitious node and uses its address as a destination address to lure a potential malicious node to send back a fake route reply packet.

- As mobile nodes equipped with more storage and communication capabilities become increasingly popular and prevalent, opportunistic mobile networks are rapidly emerging as an alternative to conventional infrastructure-based communication. Due to the mobility of nodes, it is not trivial to guarantee end-to-end path for communication in a constantly varying network topology. In this research topic, I plan to propose a probabilistic cooperative caching technique to achieve efficient packet delivery in opportunistic mobile networks.

BIBLIOGRAPHY

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions," *Future Generation Computer Systems*, vol. 29, pp. 1645–1660, 2013.

[2] D. Raymond, R. Marchany, M. Brownfield, and S. Midkiff, "Effects of Denial of Sleep Attacks on Wireless Sensor Newtork MAC Protocols," in *Proc. Workshop on Information Assurance*, 2006, pp. 297–304.

[3] R. Kravets and P. Krishnan, "Power Management Techniques for Mobile Communication," in *Proc. ACM MOBICOM*, 1998, pp. 157–168.

[4] N. A. Pantazis, S. A. Nikolidakis, and D. D. Vergados, "Energy-Efficient Routing Protocols in Wireless Sensor Networks: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 551–591, 2013.

[5] M. Gorlatova, J. Sarik, G. Grebla, M. Cong, I. Kymissis, and G. Zussman, "Movers and Shakers: Kinetic Energy Harvesting for the Internet of Things," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 8, pp. 1624–1639, 2015.

[6] P. Kamalinejad, C. Mahapatra, Z. Sheng, S. Mirabbasi, V. Leung, and Y. L. Guan, "Wireless Energy Harvesting for the Internet of Things," *IEEE Communications Magazine*, vol. 53, no. 6, pp. 102–108, 2015.

[7] Y. Wang, Y. Liu, C. Wang, Z. Li, X. Sheng, H. Lee, N. Chang, and H. Yang, "Storage-less and Converter-less Photovoltaic Energy Harvesting with Maximum Power Point Tracking for Internet of Things," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 2, pp. 173–186, 2015.

[8] *A07-034 (Army), Harvesting Energy for Wireless Sensor Networks*, http://www.dodsbir. net/sitis/.

[9] *Killer App: Army Tests Smartphones for Combat*, Wall Street Journal, 6–3–2011.

[10] S. Sudevalayam and P. Kulkarni, "Energy Harvesting Sensor Nodes: Survey and Implications," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 3, pp. 443–461, 2011.

[11] A. Akella, G. Judd, S. Seshan, and P. Steenkiste, "Self-management in Chaotic Wireless Deployments," in *Proc. ACM MOBICOM*, 2005, pp. 185–199.

[12] V. Ramasubramanian and D. Mosse, “BRA: A Bidirectional Routing Abstraction for Asymmetric Mobile Ad Hoc Networks,” *IEEE/ACM Trans. on Networking*, vol. 16, no. 1, pp. 116–129, 2008.

[13] T. Le, P. Sinha, and D. Xuan, “Turning heterogeneity into an advantage in wireless ad-hoc network routing,” *Ad Hoc Networks*, no. 8, pp. 108–118, 2010.

[14] Y. Hu, W. Li, X. Chen, S. L. Xin Chen, and J. Wu, “A Probabilistic Routing Protocol for Heterogeneous Sensor Networks,” in *Proc. IEEE NAS*, 2010, pp. 19–26.

[15] B. Romdhani, D. barthel, and F. Valois, “Routing for Data-Collection in Heterogeneous Wireless Sensor Networks,” in *Proc. IEEE VTC*, 2011, pp. 1–5.

[16] S. Lim, C. Yu, and C. R. Das, “RandomCast: An Energy Efficient Communication Scheme for Mobile Ad Hoc Networks,” *IEEE Trans. on Mobile Computing*, vol. 8, no. 3, pp. 351–369, 2009.

[17] S. Marti, T. J. Giuli, K. Lai, and M. Baker, “Mitigating routing misbehavior in mobile ad hoc networks,” in *Proc. ACM MOBICOM*, 2000, pp. 255–265.

[18] D. R. Raymond and S. F. Midkiff, “Denial-of-Service in Wireless Sensor Networks: Attacks and Defense,” *IEEE Pervasive Computing*, vol. 7, no. 1, pp. 74–81, 2008.

[19] T. H. Hai and E. Huh, “Detecting Selective Forwarding Attacks in Wireless Sensor Networks Using Two-hops Neighbor Knowledge,” in *Proc. IEEE Int’l Symposium on Network Computing and Applications*, 2008, pp. 325–331.

[20] B. Yu and B. Xiao, “Detecting Selective Forwarding Attacks in Wireless Sensor Networks,” in *IEEE IPDPS*, 2006, pp. 1–8.

[21] B. Xiao, B. Yu, and C. Gao, “CHEMAS: Identify Suspect Nodes in Selective Forwarding Attacks,” *Journal of Parallel and Distributed Computing*, vol. 67, no. 11, pp. 1218–1230, 2007.

[22] K. Liu, J. Deng, P. K. Varshney, and K. Balakrishnan, “An acknowledgment-based approach for the detection of routing misbehavior in MANETs,” *IEEE Trans. on Mobile Computing*, vol. 6, no. 5, pp. 536–550, 2007.

[23] D. M. Shila, C. Yu, and T. Anjali, “Mitigating Selective Forwarding Attacks with a Channel-Aware Approach in WMNs,” *IEEE Trans. on Wireless Communications*, vol. 9, no. 5, pp. 1661–1675, 2010.

[24] X. Li, R. Lu, X. Liang, and X. Shen, "Side Channel Monitoring: Packet Drop Attack Detection in Wireless Ad Hoc Networks," in *Proc. IEEE ICC*, 2011, pp. 1–5.

[25] E. M. Shakshuki, N. Kang, and T. R. Sheltami, "EAACK: A Secure Intrusion-Detection System for MANETs," *IEEE Trans. on Industrial Electronicsg*, vol. 60, no. 3, pp. 1089–1098, 2013.

[26] Q. Liu, J. Yin, V. Leung, and Z. Cai, "FADE: Forwarding Assessment Based Detection of Collaborative Grey Hole Attacks in WMNs," *IEEE Trans. on Wireless Communications*, vol. 12, no. 10, pp. 5124–5137, 2013.

[27] J. Ren, Y. Zhang, K. Zhang, and X. S. Shen, "Exploiting channel-aware reputation system against selective forwarding attacks in wsns," in *Proc. IEEE GLOBECOM*, 2014, pp. 330–335.

[28] J.-M. Chang, P.-C. Tsou, I. Woungang, H.-C. Chao, and C.-F. Lai, "Defending against collaborative attacks by malicious nodes in MANETs: A cooperative bait detection approach," *IEEE Systems Journal*, vol. 9, no. 1, pp. 65–75, 2015.

[29] R. H. Jhaveri and N. M. Patel, "A sequence number based bait detection scheme to thwart grayhole attack in mobile ad hoc networks," *Wireless Networks*, vol. 21, no. 8, pp. 2781–2798, 2015.

[30] C. Pu and S. Lim, "A Light-Weight Countermeasure to Forwarding Misbehavior in Wireless Sensor Networks: Design, Analysis, and Evaluation," *IEEE Systems Journal*, 2016.

[31] X. Chen, M. Faloutsos, and S. Krishnamurthy, "Power Adaptive Broadcasting with Local Information in Ad Hoc Networks," in *Proc. of ICNP*, 2003, pp. 168–178.

[32] E.-S. Jung and N. H. Vaidya, "A Power Control MAC Protocol for Ad Hoc Networks," in *Proc. ACM MOBICOM*, 2002, pp. 36–47.

[33] X. Zhang, M. Liu, H. Gong, S. Lu, and J. Wu, "PCAR: A Power Controlled Routing Protocol for Wireless Ad Hoc Networks," in *Proc. of WoWMoM*, 2010, pp. 1–6.

[34] T. Voigt, H. Ritter, and J. Schiller, "Utilizing Solar Power in Wireless Sensor Networks," in *Proc. IEEE LCN*, 2003, pp. 416–422.

[35] K. Zeng, K. Ren, W. Lou, and P. J. Moran, "Energy Aware Efficient Geographic Routing in Lossy Wireless Sensor Networks with Environmental Energy Supply," *Wireless Networks*, vol. 15, no. 1, pp. 39–51, 2009.

[36] Z. A. Eu and H. Tan, "Adaptive Opportunistic Routing Protocol for Energy Harvesting Wireless Sensor Networks," in *Proc. IEEE ICC*, 2012.

[37] Y. Chae, L. C. DiPippo, and Y. L. Sun, "Trust Management for Defending On-off Attacks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 1178–1191, 2015.

[38] S. Lim and H. Lauren, "Hop-by-Hop Cooperative Detection of Selective Forwarding Attacks in Energy Harvesting Wireless Sensor Networks," in *Proc. Int'l Conf. on Computing, Networking and Communications (ICNC)*, 2015, pp. 315–319.

[39] C. Pu and S. Lim, "Spy vs. Spy: Camouflage-based Active Detection in Energy Harvesting Motivated Networks," in *Proc. Military Communications Conference (MILCOM)*, 2015, pp. 903–908.

[40] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," in *Proc. Sensys*, 2004, pp. 95–107.

[41] A. Varga, *OMNeT++*, 2014, http://www.omnetpp.org/.

[42] A. Boulis, *Castalia*, 2014, http://castalia.forge.nicta.com.au.

[43] S. Lim, J. Kimn, and H. Kim, "Analysis of Energy Harvesting for Vibration-Motivated Wireless Sensor Networks," in *Proc. International Conference on Wireless Networks*, 2010, pp. 391–397.

[44] A. D. Wood and J. A. Stankovic, "Denial of Service in Sensor Networks," *IEEE Computer*, vol. 35, no. 10, pp. 54–62, 2002.

[45] Y. Kim, H. Lee, K. Cho, and D. Lee, "CADE: Cumulative Acknowledgment Based Detection of Selective Forwarding Attacks in Wireless Sensor Networks," in *Int'l Conf. on Convergence and Hybrid Information Technology*, 2008, pp. 416–422.

[46] C. Pu, T. Gade, S. Lim, M. Min, and W. Wang, "Lightweight Forwarding Protocols in Energy Harvesting Wireless Sensor Networks," in *Proc. Military Communications Conference (MILCOM)*, 2014, pp. 1053–1059.

[47] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," in *Proc. ACM MOBICOM*, 2000, pp. 264–275.

[48] B. Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," in *Proc. ACM MOBICOM*, 2000, pp. 243–254.

[49] W. Stallings, *Cryptography and Network Security - Principles and Practices, 6th Edition.* Prentice Hall, 2013.

[50] X. Tang and J. Xu, “Extending Network Lifetime for Precision-Constrained Data Aggregation in Wireless Sensor Networks,” in *INFOCOM*, 2006, pp. 1–12.

[51] I. Khalil and S. Bagchi, “Stealthy attacks in wireless ad hoc networks: detection and countermeasure,” *IEEE Trans. on Mobile Computing*, vol. 10, no. 8, pp. 1096–1112, 2011.

[52] J. Deng, R. Han, and S. Mishra, “Insens: Intrusion-tolerant routing for wireless sensor networks,” *Computer Communications*, vol. 29, no. 2, pp. 216–230, 2006.

[53] Q. Fang, J. Gao, and L. J. Guibas, “Locating and bypassing holes in sensor networks,” *Mobile Networks and Applications*, vol. 11, no. 2, pp. 187–200, 2006.

[54] E. Y. Vasserman and N. Hopper, “Vampire attacks: draining life from wireless ad hoc sensor networks,” *IEEE Trans. on Mobile Computing*, vol. 12, no. 2, pp. 318–332, 2013.

[55] M. Gorlatova, J. Sarik, G. Grebla, M. Cong, L. Kymissis, and G. Zussman, “Movers and Shakers: Kinetic Energy Harvesting for the Internet of Things,” in *Proc. ACM SIGMETRICS*, 2014.

[56] *Wearable computing is here already: How hi-tech got under our skin, July 2013*, http://www.independent.co.uk.

[57] *2.4 GHz IEEE 802.15.4/ZigBee-ready RF Transceiver*, http://www.ti.com.cn/cn/lit/ds/symlink/cc2420.pdf.

[58] *Cisco Aironet 802.11a/b/g wireless CardBus adapter*, http://www.cisco.com.

[59] T. Starner, “Human-powered Wearable Computing,” *IBM Systems Journal*, vol. 35, no. 3 & 4, pp. 618–629, 1996.

[60] T. Starner and J. A. Paradiso, “Human Generated Power for Mobile Electronics,” in *CRC Press*, 2004, pp. 1–35.

[61] Z. L. Wang, *Nanogenerators for Self-powered Devices and Systems.* Georgia Institute of Technology, Atlanta, USA, 2011.

[62] X. Xue, S. Wang, W. Guo, Y. Zhang, and Z. L. Wang, “Hybridizing Energy Conversion and Storage in a Mechanical-to-Electrochemical Process for Self-Charging Power Cell,” *Nano Letter*, vol. 12, no. 9, p. 50485054, 2012.

[63] Z. A. Eu, H. Tan, and W. K. G. Seah, “Design and Performance Analysis of MAC Schemes for Wireless Sensor Networks Powered by Ambient Energy Harvesting,” *Ad Hoc Networks*, vol. 9, no. 3, pp. 300–323, 2011.

[64] C. Fujii and W. K. G. Seah, “Multi-tier Probabilistic Polling for Wireless Sensor Networks Powered by Energy Harvesting,” in *IEEE ISSNIP*, 2011, pp. 383–388.

[65] K. Zeng, K. Ren, W. Lou, and P. J. Moran, “Energy aware efficient geographic routing in lossy wireless sensor networks with environmental energy supply,” *Wireless Networks*, vol. 15, no. 1, pp. 39–51, 2009.

BIOGRAPHY

Cong Pu received the B.S. degree in computer science and technology from Zhengzhou University, Zhengzhou, China, in 2009, the M.S. degree in computer science from Texas Tech University, Lubbock, TX, USA, in 2013.

He is currently a Ph.D. candidate and a graduate part-time instructor with the Department of Computer Science, Texas Tech University. His research interests include wireless sensor networks, energy harvesting motivated networks, vehicular ad hoc networks, and denial-of-service attacks.

He was the recipient of the 2015 Texas Tech Helen DeVitt Jones Excellence in Graduate Teaching Award.

- C. Pu and S. Lim, A Light-Weight Countermeasure to Forwarding Misbehavior in Wireless Sensor Networks: Design, Analysis, and Evaluation, IEEE Systems Journal, 2016.
- C. Pu and S. Lim, Spy vs. Spy: Camouflage-based Active Detection in Energy Harvesting Motivated Networks, in Proc. Military Communications Conference (MILCOM), 2015, pp. 903–908.
- C. Pu, T. Gade, S. Lim, M. Min, and W. Wang, Lightweight Forwarding Protocols in Energy Harvesting Wireless Sensor Networks, in Proc. Military Communications Conference (MILCOM), 2014, pp. 1053–1059.