Event Reliability in Wireless Sensor Networks

by

Muhammad Adeel Mahmood

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Doctor of Philosophy
in Engineering.

Victoria University of Wellington
2019

Abstract

Ensuring reliable transport of data in resource-constrained Wireless Sensor Networks (WSNs) is one of the primary concerns to achieve a high degree of efficiency in monitoring and control systems. The two reliability mechanisms typically used in WSNs are *packet* reliability and *event* reliability. *Packet* reliability, which requires all packets from all the sensor nodes to reach the sink, can result in wastage of the sensors' limited energy resources. Event reliability, which only requires that one packet related to each event reaches the sink, exploits the overlap of the sensing regions of densely deployed sensor nodes to eliminate redundant packets from nodes in close proximity that contain duplicate information about an event.

The majority of previous research in this area focuses on *packet* reliability rather than *event* reliability. Moreover, the research that does focus on *event* reliability relies on the sink to impose some form of control over the flow of data in the network. The sinks' centralized control and decision-making increases the transmission of unnecessary packets, which degrades overall network performance in terms of energy, congestion and data flow.

This thesis proposes a distributed approach to the control of the flow of data in which each node makes in-node decisions using data readily available to it. This reduces the transmission of unnecessary packets, which reduces the network cost in terms of energy, congestion, and data flow. The major challenges involved in this research are to: (i) accurately identify that multiple packets are carrying information about the same event, (ii) reliably deliver the packets carrying information about the unique event, (iii) ensure that enough information about the area of interest is reliably delivered to the sink, and (iv) maintain the event coverage throughout the network.

This thesis presents the Event Reliability Protocol (ERP) and its extension, the Enhanced Event Reliability Protocol (EERP). The protocols aim for the reliable transmission of a packet containing information about each unique event to the sink while identifying and minimizing the unnecessary transmission of similar redundant packets from nodes in the region of the event. In this way, the sensor nodes consume less energy and increase the overall network lifetime. EERP uses a multilateration technique to identify multiple packets containing similar event information and thus is able to filter redundant packets of the same event. It also makes use of implicit acknowledgment (iACKs) for reliable delivery of the packets to the sink node. The process is based on the hop-by-hop mechanism where the decisions are made locally by the intermediate nodes.

The thesis reports on simulations in QualNet 5.2 for verifying the accuracy of our event identification and event reliability mechanisms employed in the ERP and EERP. The results show that EERP performs better in terms of minimizing overall packet transmission and hence the energy consumption at the sensor nodes in a WSN. Also, the results for event identification mechanism and reliable event delivery show that EERP considerably improves upon other protocols in terms of unique events delivery.

Dedication

To Abbu...

Acknowledgments

Praise be to Almighty ALLAH alone, who created us empty of any knowledge bestowed upon us hearing, vision, and intellect so that we must be thankful to Him.

Praise be to Him alone, who let wonderful, encouraging and supporting people come into life without whom this journey would have been impossible. My mentors, my guides Dr. Ian Welch and Dr. Peter Andre (Pondy). I am indebted to them for their outstanding supervision, support and mentoring. I feel lucky to have had the opportunity to work with Pondy, whom I greatly admire for his invaluable advices that go well beyond the pursuit of an academic title and the fulfilment of a research project. Prof. Winston Seah for all his valuable advices and suggestions that led me into experiences that changed my life forever.

I can't be thankful enough to Almighty, Who gave me friends like family. My friend, my brother Dr. Mumraiz Khan Kasi. You are forever my rock. I made such amazing friends at Victoria University who never failed to amaze and inspire me. This acknowledgement would be incomplete without mentioning Dr. Jawad Mirza, Dr. Saqib Saleem, Dr. Ibrahim Rehman, Dr. Abdul Wahid, Dr. Muhammad Iqbal, Dr. Mudassar Altaf, Dr. Soha Ahmed, Dr. Harith Al-Sahaf, Dr. Muhammad Nekooei, Dr. Muhammad Alzeer, Dr. Harsh Tataria, Salman Rasheed, Hassan Tariq, Dr. Saud Naqvi, Asim Masood and my dearest Dr. Umar Ahmed. Their companionship has been a source of great relief and entertainment in this intellectually challenging journey. I am forever grateful to my constant support Amin Syed, Afraz Liaquat and Dr. Aymen Sajjad.

My amazingly wonderful colleagues at work. I cannot show enough gratitude to my manager Bruce Parrot for being so understanding, supportive and an awesome boss during this time. I would also like to take this opportunity to thank Prof. Dale Carnegie, Suzan Hall, Patricia Stein, Ambreen Khan-Evans, Dr. Diana Siwiak, Dr. Craig Watterson and all the staff in ECS and University Hall Accommodation for their help and support.

He provided me with love and unconditional support in the form of my family. Their sacrifices can never be forgotten. I remain indebted to my parents for teaching me the value of education and especially my mother and sisters, for all their prayers and sacrifices during this time. I owe my deepest gratitude to my beloved wife, Saniya, for always being there for me during all the highs and lows we faced together. It was not possible to accomplish this milestone without her constant care, support, cooperation and encouragement. Last but not the least I owe to the two angels in my life, Rameen and Maham, my beautiful daughters who will no longer have to wait for their dad being away to write just one more paragraph or run another experiment.

Contents

List of Figures

List of Abbreviations

WSN Wireless Sensor Network

ERP Event Reliability Protocol

EERP Enhanced Event Reliability Protocol

MEMS Micro-Electro-Mechanical Systems

iACK implicit Acknowledgment

eACK explicit Acknowledgment

NACK Negative Acknowledgment

SNR Signal to Noise Ratio

PoI Point of Ineterest

PDA Personal Digital Assistant

RF Radio Frequency

PHY PHYsical layer

LR-WPANs Low-Rate Wireless Personal Area Networks

WFD Water Framework Directive

MAC Medium Access Control

EAST	Efficient Data Collection Aware of Spatio-Temporal correlation
TCT	Temporal Coherency Tolerance
QoM	Quality of Monitoring
STCP	Sensor Transmission Control Protocol
RTMC	Reliable Transport with Memory Consideration
DTSN	Distributed Transport Sensor Networks
AW	Acknowledgement Window
EAR	Explicit Acknowledgment Request
DFRF	Directed Flood-Routing Framework
SWIA	Stop-Wait Implicit Acknowledgement
RBC	Reliable Bursty Convergecast
LPL	Low Power Listening
TRCCIT	Tuneable Reliability with Congestion Control for Information Transport
HACK	Hybrid Acknowledgment
RMST	Reliable Multi-Segment Transport
PFSQ	Pump Slowly Fetch Quickly
ESRT	Event-to-Sink Reliability Protocol
DST	Delay Sensitive Transport
ART	Asymmetric and Reliable Transport
LTRES	Loss-Tolerant Reliable Event Sensing

QERP	Quality-based Event Reliability Protocol
CD	Contribution Degree
PORT	Price-Oriented Reliable Transport Protocol
OREC	Optimum Reed-Solomon Erasure Coding
GAs	Genetic Algorithms
DTSN	Distributed Transport for Sensor Networks
FEC	Forward Error Correction
ULP	Unconditional Loss Probability
RTSN	Reliable Transfer on Sensor Networks
TRSN	Transmission Reliability in Sensor Networks
eNodes	essential Nodes
RDTS	Reliable erasure-coding based Data Transfer Scheme

Chapter 1

Introduction

Wireless Sensor Networks (WSNs) are the preferred choice for the design and deployment of next-generation monitoring and control systems [1,2]. It has now become feasible to deploy large numbers of low-cost sensors to monitor regions over the ground surface, underwater or in the atmosphere [3]. The most significant benefit of wireless sensor networks is that they have the potential to provide a wealth of data about the environment in which they are deployed and deliver this data across the network to the end-users [6,10]. Moreover, they extend computation capability to physical environments that human beings cannot reach [5]. They can operate for prolonged periods in habitats that are hostile, challenging or ecologically too sensitive for human visitation [12].

At the heart of WSNs are the sensor nodes. A sensor node is a device that possesses sensing, computation and communication capabilities. Depending on its sensing components and the application requirements, a sensor node can be used to monitor phenomena such as temperature, light, motion, pressure, and humidity. The processing module of a sensor node performs computation on the sensed data and also on data received from other sensor nodes. The communication module in a sensor node is used to send and receive data packets to and from the neighboring nodes [11]. Since a single sensor provides only limited information, a network of these sensors is used to provide coverage over large environments.

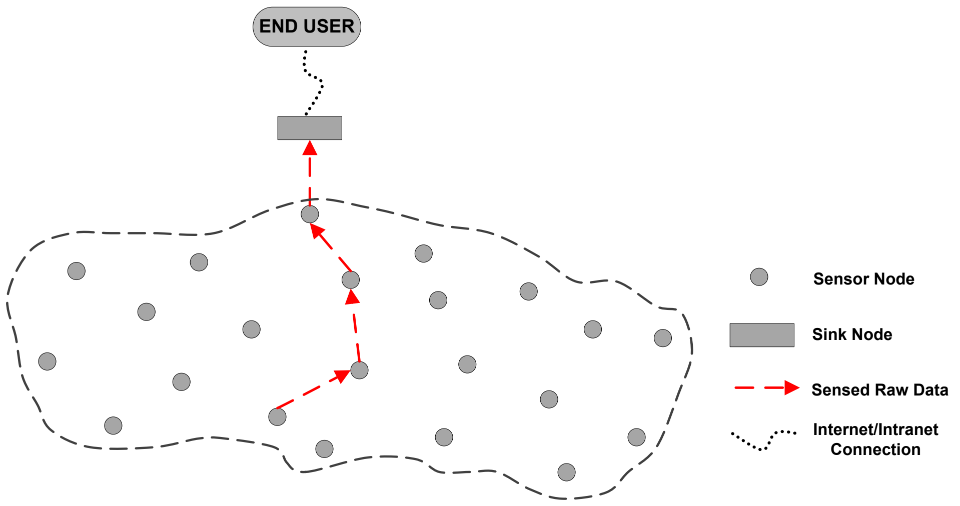


Figure 1.1: Wireless Sensor Network

Figure 1.1 shows a typical Wireless Sensor Network (WSN) comprising several sensor nodes and a sink node. The sensor nodes continually sense data from the environment and send it to the sink node. In most scenarios, tens to as many as thousands of such sensor nodes are distributed throughout a region of interest, where they self-organize into a network through wireless communication and collaborate with one another to accomplish a common task [2]. The sink receives all the sensed data from these sensor nodes, processes it and sends it to the end user. Because of the limited resources in WSNs, especially power limitations, it is always desirable to minimize the network traffic in a WSN and only transmit the packets that are needed for the application.

In a WSN, an event of interest can be described as a meaningful change in the state of the environment that the application users are interested in and require the sensors to observe it. Mostly, WSN applications require a single notification of the occurrence of an event rather than all the individual packets from each node in the network that may have sensed the event. This, in turn, requires a sophisticated mechanism that identifies whether several packets are carrying information about the same event or not. If the nodes are able to perform this event identification, it will be possible to reliably transmit event information from an area of interest to the destination without having to send every sensor report. Reliability in this context aims to ensure that at least one of the many packets from the sensors that detected an event is reliably delivered to the destination, even if some packets are not forwarded to the destination from all the sensor nodes.

1.1 Problem Statement

Most research to date on reliable transmission in WSNs has been done to ensure traditional *packet* reliability, where all the packets carrying sensed data from all the sensor nodes need to be reliably transported to the sink [14]. Since wireless links are error-prone, packet loss may occur. *End-to-end packet* reliability requires every packet to be acknowledged and each lost packet to be retransmitted. These retransmissions and acknowledgments increase the packet transmission overheads and can make the network congested.

Comparatively fewer studies have focused on *event* reliability where packet loss can be tolerated as long as the sink receives at least one packet containing the sensed data of an event [14]; therefore, event reliability does not require the retransmission of every lost packet. This reduces unnecessary transmissions by requiring the network to ideally transmit only a single packet from one node among a number of closely located nodes that have sensed the event. However, a decentralized process of identifying that several packets contain information about the same event is missing in the existing literature. This is one of the most important factors to be considered for event reliability and is one of the major challenges involved in this research.

Once packets containing information about the same event can be identified in a distributed fashion, the next challenge is to reliably transport the event information in resource-constrained WSNs. In WSNs, *hop-by-hop event* reliability as opposed to *end-to-end packet* reliability is sufficient for most event-driven applications [92]. The traditional approach of *end-to-end* reliability is inefficient as it introduces long delays, increased traffic and packet loss, all of which causes energy wastage that has been the primary concern for resource-constrained WSNs. The *hop-by-hop* mechanism that performs the loss recovery at each hop is a more efficient approach. The challenge is to integrate the *hop-by-hop* approach with an event identification method to create a *hop-by-hop* event reliability mechanism that reliably delivers sufficient event information while minimizing the traffic overhead.

1.2 Research Aim & Objectives

This thesis aims to solve the challenge of reliable event delivery with distributed event identification in an energy-efficient manner. It improves the scalability and coverage of event detection throughout the network by minimizing the transmission of data packets coming from multiple nodes in an event's locality while reliably delivering information about the events to the sink under different network conditions, making the network more energy efficient, scalable and less congested. To ensure energy optimization this thesis will adopt a non-sink centric approach, where instead of relying on the sink to perform event identification and detect packets carrying duplicate event information, each node makes an in-node decision more efficiently by using the data readily available to it.

This approach leads to two main objectives:

- Design a methodology that is able to identify whether two or more packets are carrying information about the same events or not, based on spatio-temporal correlation by performing in-node data processing.

- Design a mechanism that ensures the reliable delivery of information about event occurrences from source nodes to the sink node.

The main purpose of performing the event identification process at the node level instead of relying on the sink is to use the data processing capabilities within the node in order to reduce the communication cost. In addition, the non-sink centric event identification would help to reduce overall traffic by avoiding the unnecessary transmission of duplicate packets that are coming from multiple sensor nodes in an event's locality and unnecessary transmissions of control packets from the sink. A consequence is a reduced level of network congestion and packet loss rate that further supports reliable event delivery.

To ensure the reliable delivery of event information, a distributed approach is devised where, instead of relying on the sinks' centralized decision making, the transmission and retransmission decisions are taken at the node level to ensure reliability. These decisions are required when the sender node needs to decide whether it should transmit the unsent packets or retransmit its lost packet or not. This results in the reliable transmission of the packets by ensuring that the sink only gets a sufficient number of packets carrying information about a certain event instead of all sensed packets for that event. Subsequently, different acknowledgment mechanisms are also evaluated with the aim of improving the energy resource utilization and the scalability of event detection.

To further strengthen the event identification process, the nodes are required to identify whether multiple packets with similar spatio-temporal information are actually carrying information about the same event or multiple unique events. This not only strengthens the overall event identification process but also improves the scalability of event detection and reliability in the wireless sensor networks.

1.3 Contributions

This thesis makes four major contributions:

- A novel 3D-Reliability reference model that establishes a common taxonomy to evaluate the research in WSN reliability (see chapter 3). It provides a detailed analysis and comparison of state-of-the-art schemes and protocols that aim to achieve reliable transmission of data in WSNs based on the 3D-Reliability reference model.

  [Published in M. A. Mahmood, W. K. G. Seah and I. Welch, "Reliability in wireless sensor networks: A survey and challenges ahead", Computer Networks, Volume 79, Pages 166-187, 2015 - https://doi.org/10.1016/j.comnet.2014.12.016]

- A selective retransmission-based distributed event reliability mechanism that provides event reliability by minimizing the unnecessary retransmission of data packets coming from multiple nodes in an event's locality. The mechanism exploits the high level of spatial locality in sensor data tends to exhibit and uses a novel region-based selective retransmission mechanism that takes advantage of the broadcast nature of wireless channel by using an implicit acknowledgment mechanism (see Chapters 4-6). The mechanism performs efficient event reliability while reducing overall network cost in terms of energy, congestion and network traffic.

  [Published in M. A. Mahmood and W. K. G. Seah, "Event Reliability in Wireless Sensor Networks", in Proceedings of the Seventh International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), pages 730734, Adelaide, Australia, 6-9 December 2011]

- A multilateration-based distributed event reliability mechanism that provides reliable transmission of event information to the sink. The mechanism uses the spatio-temporal information of the originating nodes and combines a greedy strategy with a multilateration algorithm in a novel way to perform event identification before reliably transmitting the event information towards the sink node (see Chapter 4-6). The analysis takes note of the error rate in estimating the event's location using multilateration with increasing node density (see Chapter 6).

[Published in M. A. Mahmood, P. Andreae and I. Welch, "Enhanced Event Reliability in Wireless Sensor Networks", in Proceedings of the 32nd IEEE International Conference on Advanced Information Networking and Applications (IEEE AINA-2018), Cracow, Poland, 16-18 May 2018

and

M. A. Mahmood, I. Welch and M. K. Kasi, "Multilateration-based Event Identification in a Wireless Sensor Network", in Proceedings of the ACM International Conference on Engineering & MIS (ICEMIS), Istanbul, Turkey, 19-21 Jun 2018]

- An event generation application for IEEE 802.15.4 networks to evaluate the distributed event identification and reliability mechanisms in a realistic environment. Unlike other schemes, which assume that event generation has happened, this application actually generates single/multiple events sensed and transmitted by the in-range nodes towards the destination. This makes the performance evaluation more realistic in realistic scenarios (see chapter 6 and Appendix A).

1.4 Structure of the Thesis

This section provides a brief description of the remainder of the thesis.

Chapter 2 presents the background information necessary for an understanding of the concepts presented in this thesis. It discussed some potential application areas of wireless sensor networks and their requirements. The challenges involved in achieving energy efficient and reliable transmission are also discussed in detail while examining critical requirements for event reliability.

Chapter 3 focuses on research work related to reliability in wireless sensor networks. It examines how others have addressed the issue of energy efficient reliable in WSNs. It classifies the research in the field of reliability in WSNs by presenting a systematic and comprehensive taxonomy of the reliability schemes, which are subsequently discussed in depth. It compares and analyses the existing schemes using different criterion for comparison by presenting a qualitative analysis and identifies the strengths and weaknesses of existing works.

Chapter 4 provides design of the protocols presented in this thesis based on the arguments and requirements presented in the previous chapters. This contributes towards building the basic architecture and design of the mechanisms proposed in this thesis by using step-by-step process flows that in turn provide the grounds for the implementation of the contributions of this thesis as presented in Section 1.3. For a better understanding of the system design and operation, this chapter also presented a scenario-based conceptual model with a simple example scenario and a sequence diagram explaining the details of the proposed mechanisms.

Chapter 5 presents a prototypical implementation of basic and enhanced ERP. It also presents the operations performed on the different level of the node categories based on the conceptual design presented in the previous chapter to achieve the aims and objectives of this thesis. The implementation details presented in this chapter addresses how to perform event reliability in a distributed fashion in combination with an efficient event identification mechanism in a resource-constrained wireless sensor network. The processes and procedures involved in the implementation of the proposed work are explained in detail in the form of pseudo codes according to different roles associated with each node category.

Chapter 6 reports on the evaluation of the performance of the proposed mechanisms in terms of achieving event reliability, event identification, energy efficiency and event coverage throughout the network. The evaluation is based on the simulation results in various scenarios. It presents a performance analysis of the EERP as compared to other related schemes. The analysis is conducted to demonstrate that the proposed mechanisms achieve the overall objectives of this thesis by utilizing sensor's processing capabilities to improve event reliability in a WSN with reduced energy consumption.

Chapter 7 concludes by summarizing the major findings of the thesis, discusses the research objectives achieved and presents future directions that may be helpful for the extension of this work.

Chapter 2

Background

This chapter presents an overview of WSNs that is necessary for a general understanding of the issues discussed in the remainder of this thesis. This background study focuses on the factors that need to be considered for achieving energy efficient reliable transmission in WSNs with distributed event identification.

Section 2.1 and Section 2.2 provide an overview of the WSN and its applications. Section 2.3 discusses the challenges involved for an efficient WSN. Section 2.4 identifies initial research solutions to address the challenges described in the previous section. Section 2.5 provides the functional requirements of energy efficient event reliability in WSNs and Section 2.6 concludes by summarizing this chapter.

2.1 Wireless Sensor Networks (WSNs)

Advances in micro-electro-mechanical systems (MEMS) have allowed the integration of data processing and wireless communication capabilities into small, inexpensive, battery-powered devices called sensor nodes. Typically, a sensor node consists of sensing, computing, and communication components [23]. Depending on their sensing components, sensor nodes can be used to monitor phenomena such as temperature, light, humidity and other environmental features. The communication module in a sensor node is used to send and receive information from neighboring nodes [39]. The processing module of the sensor node performs computation on the sensed values and also on other received values from its neighbors. Since a single sensor node provides only limited information, a network of these sensors is used to manage large environments [38].

In most settings, large numbers of such sensor nodes are distributed throughout a region of interest, where they self-organize into a network through wireless communication and collaborate with each other to accomplish an assigned task [9, 44]. Such a network is called a Wireless Sensor Network where the sensor nodes continually sense data from the environment and send them to the sink node. The sink node receives all the information from these sensor nodes, processes it and sends it to the end-user [5]. Typically the highest energy component of a WSN is the communication of packets through the network.

2.2 WSN Applications & Requirements

The integration of sensing, processing and communication components into small, battery-powered sensor nodes opens the door to a wide range of real-world applications [22]. There is no single set of design requirements that fulfill the entire diversified range of wireless sensor network applications; therefore, it is critical to explore the application-specific characteristics and requirements before designing a WSN application. This section presents an overview of three general application areas and their requirements.

2.2.1 Environmental Monitoring

WSN applications for environmental monitoring and control have numerous potential benefits for scientific communities and society as a whole [24]. Often a large number of sensors are required to gather the data in an environment to monitor and control environmental trends and inter-dependencies within various habitats [12]. These applications can involve both indoor and outdoor environments which may consist of large monitored areas (i.e., hundreds of square meters and kilometers) and may also require extended periods (i.e., months or years) of monitoring to examine long-term and seasonal trends. This may also require significant numbers (i.e., tens to thousands) of sensor nodes to be deployed over the desired area in order to collect detailed and meaningful information about the environment [25].

An example of a WSN environmental monitoring application is GlacsWeb [26], which aims to monitor glaciers for extended periods (i.e., months or years) to understand what is happening under the glaciers and the possible effects on climate change and global warming. Burrel et al. [27] deployed WSN to monitor vineyards to prevent freezing damage to crops. Werner-Allen et al. [28] used seismic and acoustic sensor nodes to monitor active volcanoes in Ecuador. Likewise, SmartCoast [29] was developed to monitor water quality based on the EU Water Framework Directive (WFD).

Such applications often require the system to sense and respond to changes in the environment. Therefore, sensor nodes perform data collection by continuously sensing and reliably transmitting the event information back to the sink node for further analysis. Typically, these applications require low sampling rates, since the most common factors to be monitored, such as temperature, humidity and light, do not change quickly.

However, these applications need to operate for long periods in unattended areas; therefore, the node needs to operate in an energy efficient manner to extend the overall network lifetime. The event-driven nature of these applications requires a sophisticated mechanism to reliably deliver the environmental events from source to destination. Extended network lifetime, event reliability and efficient energy use are thus the most critical requirement of environmental monitoring applications, while data transmission rates can be delayed or reduced in order to improve overall network lifetime.

2.2.2 Surveillance

One of the key advantages of wireless sensor networks is their ability to track and detect patterns in their surroundings, which makes WSNs an integral part of surveillance applications. Military applications are one of the major areas where the fast deployment and self-organizing capabilities of WSNs make them a popular choice to perform battlefield surveillance [30]. In addition to this, WSNs can also be deployed to monitor buildings, airports, shops and homes by promptly sensing and reporting the detected information back to the sink node.

VigilNet [31] is an example of a surveillance application where sensor nodes with magnetic capabilities are deployed over an area of interest to detect magnetic fields generated by the movement of vehicles or other metallic objects. The ”A line in the Sand” [32] project by Ohio State University is another similar surveillance application, which like VigilNet, focuses on the detection and tracking of metallic objects such as armed soldiers and vehicles via a network of nodes deployed over the surveillance area.

Unlike environmental monitoring applications, surveillance applications must immediately report the sensed data back to the sink node in a reliable and timely fashion. It is also crucial for this category of applications to locate and track selected moving objects within the WSN. Reliable object detection within time bounded latency is an integral part of surveillance applications, as they require the event data to be immediately reported to the sink in a timely and reliable fashion. Low latency, packet reliability and energy efficiency are thus the most critical requirements of surveillance applications.

2.2.3 Health Care

Advances in bio-medical, telemonitoring and tracking devices have made it possible to use WSNs in a variety of healthcare applications. The integration of wireless sensors with healthcare applications is highly convenient and beneficial not only for doctors but also for patients and disabled people [33]. These applications are mostly used in indoor conditions so usually have smaller monitoring areas and fewer nodes than environmental monitoring and surveillance applications. In healthcare, WSNs are mostly used to monitor patient's movements with the help of tracking devices while reporting these movements back to the relevant authorities.

The CodeBlue [34] project developed at Harvard University is an excellent example in the area of health-care applications. CodeBlue continuously monitors patients based on data gathered from wearable sensors, monitoring patient's blood pressure, heart rate, muscular activities and physical movement. The data gathered is continuously transmitted to a Personal Digital Assistant (PDA) monitored by medical personnel. Other healthcare applications of WSNs include tracking and monitoring doctor and patient movements inside hospitals [35] and overseeing drug administration [37]. Eldercare is another application [36], where sensors can send an automatic notification to a contact center in the case of any emergency situations.

Healthcare applications require high accuracy to track and monitor patient information, while reliably returning data to medical personnel in a timely fashion. Moreover, the privacy of patient data is of utmost importance in healthcare applications. Thus, special consideration of authentication and authorization is required to prevent unauthorized disclosure of information.

2.2.4 Common Application Requirements

The most common and critical requirements in the three general application areas in WSNs are energy efficiency and reliability. Due to the dynamic nature of WSNs, it can accommodate a huge range of applications having a diverse set of requirements. However, energy efficiency and reliable delivery of event information remain the most common and critical requirements for a WSN application.

2.3 Challenges of Event Reliability in WSNs

This section discusses the challenges that complicate the identification of events in WSNs. WSNs are inherently constrained by the limited resources of the sensor nodes and it is essential to understand the constrained capabilities of sensor nodes in order to address the challenges faced by the WSNs.

2.3.1 Centralized Event Identification Constraints

Most applications of WSNs are event-driven, where the primary purpose is to monitor and control the area of interest for particular event happenings and use a centralized event identification approach. In a WSN, an event of interest can be described as a meaningful change in the state of the environment that the application users are interested in and require the sensors to observe. Sensor nodes detect an event of interest and put the event data in a packet and send it through the network to the sink. In centralized event identification, sensor nodes do not examine the packets and simply forward them to the sink. The sink is responsible for analyzing the information in the packets and identifying the different events that have been reported to it. The sink may regulate the flow of data for a particular area of interest by asking the sensors to either send more packets or reduce their rate to get the information about events happening in that area by sending more packets back through the network. This centralized flow control increases the overall communication overhead of the resource constrained WSNs. Therefore it is desirable to use a distributed event identification approach that uses the processing capabilities of the sensor nodes to reduce the communication overhead.

2.3.2 Event Reliability Constraints

For successful monitoring of an environment, the critical data collected by the sensor nodes need to be reliably delivered to the sink. The critical requirement for the reliability of event information is that packets from the sensors are received by the sink. There are many causes for packets loss - transmission errors, packet collision, interference, node failure (due to energy depletion) or other unforeseeable reasons. One cause of loss is congestion, which is particularly problematic in centralized event identification and reliability. The congestion causes the loss of packets, which may affect the reliability of communicating the event information to the sink; therefore, a centralized approach is inefficient for maintaining event reliability in WSNs.

Furthermore, due to the short range of the sensor nodes, data might have to travel a large number of hops which, in turn, introduces many possibilities of error that can also become the cause of packet loss. Given the error-prone nature of wireless links, ensuring reliable transfer of data from resource constrained sensor nodes to the sink continues to be one of the significant challenges in the field of WSNs. A centralized approach for maintaining event identification and reliability is a particular problem because in centralized event identification and reliability, where all sensor nodes in the network forward their sensed data towards the sink, the area around the sink becomes more congested with all the traffic flowing towards and converging on the sink.

In WSNs, existing methods to ensure reliability belong mainly to one of two techniques: retransmission and redundancy. Each of these techniques ensures reliability by recovering the lost data using a hop-by-hop or an end-to-end method. Another dimension is that they can adopt either packet or event level reliability based on the application requirements. The terminology used for these techniques and methods to ensure reliability in WSNs are defined as follows.

Retransmission vs. Redundancy: Retransmission is the traditional way of ensuring reliability, where the sender node, after transmitting its packet, waits for the acknowledgment of its sent packet from the next hop node on the path to the sink. In the case the sender does not receive any acknowledgments, it deems the sent packet has been lost. Thus, in order to ensure reliability, the lost packet needs to be retransmitted. On the other hand, instead of retransmitting the packet, the redundancy approach aims to allow the receiver to correct damaged packets. The sender, when sending the packet, adds some extra information to the packet that the receiver can use to reconstruct the packet if some bits are lost or corrupted. The redundancy approach is unable to deal with major errors in packets or complete packet loss.

End-to-end ("connection oriented") vs. Hop-by-hop ("link oriented"):

Both retransmission and redundancy-based reliability techniques can be performed using either a hop-by-hop or an end-to-end method. We refer to hop-by-hop as a link-oriented mechanism, as reliability needs to be maintained across every single link. In the hop-by-hop method, every single hop from source to destination is responsible for ensuring the reliable transmission of the sensed data. Similarly, the end-to-end method can be referred to as a connection-oriented mechanism, where only the two endpoints (i.e., the source and destination nodes) are responsible for ensuring reliability, while the intermediate nodes simply relay the packets between the source and the destination.

Packet vs. Event level Reliability: Packet or event level reliability is concerned with how much sensed data needs to be delivered to the sink about the occurrence an event in the environment. Packet-level reliability aims to ensure that all the packets carrying sensed data from all the sensor nodes are reliably transported to the sink. On the other hand, event level reliability aims to ensure that the sink gets enough information about the events happening such that at least one packet about each distinct event is reliably delivered to the sink instead of sending through all the sensed packets.

2.4 Techniques to Mitigate the Challenges

Schemes that only require event reliability can take a distributed approach in which nodes in the network can suppress or ignore redundant packets as long as some packets about each distinct event are forwarded to the sink. Such distributed approaches can reduce the network traffic and reduce the congestion around the sink and therefore achieve greater energy efficiency. The central challenge for distributed approaches is determining whether two or more packets are about the same event or not.

This section briefly describes the techniques that have been used to build a solution to the problems of energy efficiency and event reliability in WSN. This thesis will build on these techniques.

2.4.1 Event Identification with Spatio-temporal Correlation

If nodes can tell whether two packets contain information about the same event or not, then they can choose not to forward one of the redundant packets. This will reduce the network traffic and therefore the energy consumption and reduce the congestion and the consequent packet loss. Such a distributed approach would address the problems of the centralized event identification. In general, determining when two or more packets have information about the same event is not easy because it depends upon the particular application semantics; therefore, a major challenge for event reliability mechanisms is to determine when two or more packets are about the same event. Achieving energy efficient event reliability requires a solution to this challenge.

If an event is accurately identified, and then only a single packet carrying information about the event would be enough for the sink instead of receiving all the packets from the individual sensor nodes that detected the event. This approach allows the sensor nodes to suppress the transmission of packets that carry duplicate information about an event, typically those packets that have been generated at the same time by sensors in the same sensing region, by exploiting spatio-temporal correlation [19] among the closely located sensor nodes.

Some approaches assume that determining the event in a packet is trivial and ignore the problem. This is unrealistic, as the sensor information about an event does not usually uniquely identify it. Other approaches do not assume that the event in a packet is uniquely identified but assume that the location of every event is known. They then use this information of spatio-temporal correlation to identify packets about the same event. However, locations of events are generally not known beforehand and must be inferred from sensor information. Therefore, accurately identifying the event detected by the nodes is one of the most critical characteristics of event-driven applications in WSNs.

Estimating the location of the event can play a significant role in accurate event identification. This localization of the event can be performed with the help of geometric information about the network, where communication among two nodes can be used to gather information about their geometric relationship that is used to estimate the location information of the event detected by the nodes. Multilateration, which is further explained in the following section is one way of accomplishing this and can assist in filtering duplicate events by identifying whether multiple in-range nodes can detect the same event or not.

2.4.2 Location Identification with Multilateration

In WSNs, localization is the task of determining the physical coordinates of sensor nodes (or a group of sensor nodes) or the spatial relationships among objects. It comprises of techniques and mechanisms that allow a sensor to estimate its location based on information gathered from the sensor's environment [129]. Multilateration is the most well known and commonly used localization mechanism in WSNs [133]; however, it has never been used in performing event identification in WSNs. Existing literature uses multilateration technique for calculating the unknown location of a node in a network based on measured distances between itself and a number of anchor nodes with known locations [128]. Given the location of an anchor and a sensor's distance to the anchor, it is known that the sensor must be positioned somewhere along the circumference of a circle centered at the anchor's position with a radius (sensing range in this case) equal to or less than the sensor-anchor distance [130]. To estimate the location of the unknown node, distance measurements from at least three non-collinear anchor nodes are required. The mulitlateration process to calculate the location of an unknown node is illustrated as follows [128].

Let's consider the problem of calculating the unknown location **x** $= (x, y)$ of an node with the help of n anchor nodes with known locations which are given as $\mathbf{x}_i = (x_i, y_i)$ where $(i = 1...n)$. The distances between the unknown sensor location **x** and the anchor nodes $\mathbf{x}_i$ are given by $(r_i, i = 1...n)$. This information can be used to express the relationship between the event and nodes detected that event in a matrix notation as follows:

$$\begin{bmatrix} (x_1 - x)^2 + (y_1 - y)^2 \\ (x_2 - x)^2 + (y_2 - y)^2 \\ (x_3 - x)^2 + (y_3 - y)^2 \\ \vdots \\ (x_n - x)^2 + (y_n - y)^2 \end{bmatrix} = \begin{bmatrix} r_1^2 \\ r_2^2 \\ r_3^2 \\ \vdots \\ r_n^2 \end{bmatrix}$$

The above matrix equation can be linearized by subtracting the last matrix equation from all the previous ones, thus arriving at a proper system of linear equations given by the following matrix form:

$$A\mathbf{x} = b$$

with the coefficient matrix

$$A = \begin{bmatrix} 2(x_n - x_1) & 2(y_n - y_1) \\ 2(x_n - x_2) & 2(y_n - y_2) \\ 2(x_n - x_3) & 2(y_n - y_3) \\ \vdots & \vdots \\ 2(x_n - x_{n-1}) & 2(y_n - y_{n-1}) \end{bmatrix}$$

and the right side vector will be

$$b = \begin{bmatrix} r_1^2 - r_n^2 - x_1^2 - y_1^2 + x_n^2 + y_n^2 \\ r_2^2 - r_n^2 - x_2^2 - y_2^2 + x_n^2 + y_n^2 \\ r_3^2 - r_n^2 - x_3^2 - y_3^2 + x_n^2 + y_n^2 \\ \vdots \\ r_{n-1}^2 - r_n^2 - x_{n-1}^2 - y_{n-1}^2 + x_n^2 + y_n^2 \end{bmatrix}$$

It can now use the least square approach in order to calculate the estimated location of (x, y) event detected by using

$$\mathbf{x} = (A^T A)^{-1} A^T b,$$

with the assumption that $A^T A$ is non-singular, where the subscript T denotes the transpose.

This thesis uses multilateration mechanism in a novel context to ensure that multiple packets generated approximately at the same time and originated from the nodes in close vicinity are actually carrying information about the same event or not. It uses multilateration to estimate the location of events and spatio-temporal correlation to identify events, based on estimated locations and time. This combination of spatio-temporal correlation and multilateration strengthens the distributed event identification process which in turns achieves the overall goal of making hop-by-hop event reliability efficient. The spatio-temporal correlation technique is based on the location of the originator node that first sensed the event and the event time stamp. The approach in this thesis does not require the nodes to have the global knowledge; it only assumes that each node knows its own location, and there is a synchronized time.

2.4.3 Hop-by-hop Reliability

Most reliability schemes in WSNs use end-to-end mechanisms to maintain reliable transmission of data from source to destination, which requires only the end nodes to be responsible for achieving reliability. The schemes adopting hop-by-hop packet reliability perform better in terms of energy efficiency and reliability. This is due to the short transmission range of sensor nodes; data may have to travel several hops before it reaches the destination. Each hop introduces opportunities for errors resulting in data loss which needs to be recovered to maintain required reliability. In end-to-end mechanisms, not only the destination will have to wait longer for the lost data, but also the source node requires a longer time to receive the acknowledgments of the sent packet and even longer if the acknowledgment packet gets lost at an intermediate hop. When there is a low probability of any packet getting to the destination, there will be many requests for retransmissions (which are mostly lost themselves) each requiring multiple hops from the sink to the source and back.

This can be avoided to some extent by using a hop-by-hop (link-oriented) method, as it ensures reliability at each intermediate hop from the source to the destination. Therefore, instead of waiting for the destination's acknowledgment which is several hops away, every intermediate hop takes the responsibility of recovering the lost packet exploiting in-node processing capabilities of a sensor node. Another important consideration for hop-by-hop reliability is that it can be performed using implicit acknowledgments. Implicit acknowledgments further reduce the communication cost, where instead of transmitting the control packets between the two hops as acknowledgments, the nodes overhear the transmission of sent packets by the next hop nodes towards the destination by exploiting the broadcast nature of the wireless channel. The use of implicit acknowledgments makes it possible for the hop-by-hop reliability mechanism to maintain the required reliability from source to destination in an efficient manner. Further details on the methods and schemes aiming to ensure reliability in a wireless sensor network are presented in the following Chapter.

2.5 Functional Requirements

This section introduces the factors that are required for an energy-efficient event reliability mechanism in a WSN. The most relevant constraints of existing reliability schemes in WSNs and the techniques to mitigate these constraints are presented in Section 2.3 and 2.4 respectively. The respective techniques to mitigate these constraints of WSNs are thus the functional requirements for the design of an energy-efficient and reliable WSN. These requirements are:

Distributed event identification: Instead of relying on the sink, the sensor nodes should make their own decision about multiple packets containing information about the same events to minimize the unnecessary transmission of redundant events. The processing of data must be done at the sensor node level to limit the communication of unnecessary events and conserve the limited energy of the sensor nodes.

Link-oriented event reliability: Once the events are accurately identified at each hop, the event information should be reliably delivered to the destination by adopting a hop-by-hop mechanism. This requires the WSN application to perform both event identification and event reliability in a distributed fashion.

Therefore, the hypothesis is that inclusion of these factors as described above at the sensor node level is beneficial for achieving event reliability with reduced energy consumption in a WSN.

2.6 Summary

This background chapter provided a brief introduction of wireless sensor networks. It discussed a number of potential application areas of wireless sensor networks and their requirements. The main application areas discussed in this chapter were environmental monitoring, surveillance and health-care applications. The challenges involved in achieving energy efficient and reliable transmission are also discussed in detail. This chapter also examined critical requirements for event reliability. Event reliability and energy efficiency was a fundamental requirement for the application areas discussed and the chapter examined the key issues in event reliability. The next chapter analyses how a variety of existing systems that have attempted to achieve reliability in the wireless sensor network and its related areas.

Chapter 3

Literature Review and Analysis

This chapter describes how other researchers have addressed the factors involved in achieving energy efficient reliable transmission in WSNs with an event identification mechanism. It provides a qualitative analysis of the different schemes and identifies the strengths and weaknesses of existing works. This analysis forms the motivation and basis for the rest of the thesis.

This chapter is structured as follows. Section 3.1 explains the focus of this chapter, the scope of the analysis and the criteria used. Sections 3.2 and Section 3.3 then apply these criteria to approaches in event identification and reliability respectively. Section 3.4 discusses the implications of the research summarized in this chapter for this thesis and for wireless sensor networks in general. The chapter concludes with a summary in Section 3.5.

Most of this chapter has previously been published in [14] and currently has over 150 citations.

3.1 Focus

This chapter analyses the approaches involved in achieving reliable transmission in WSNs followed by a discussion of the existing schemes for event identification and reliability in WSNs. This section gives more details about the scope of the analysis, i.e., which areas are covered, and about the criteria used for the evaluation of related work.

3.1.1 Scope

Event identification and link-oriented event reliability have been introduced in the previous chapter. This chapter looks further into specific related work about the techniques mentioned above to see how other researchers have covered these areas. Given that we hypothesize (see Section 1.3) that the integration of these factors is vital for energy-efficient event reliability in a wireless sensor network, related literature is presented wherever possible that includes the integration of these techniques.

3.1.2 Criteria

The functional requirements for energy-efficient event reliability in wireless sensor networks described in Section 2.5 are used as criteria for the analysis of work in related areas. These are:

- *Distributed event identification:*

  To accurately identify the events uniquely by the sensor nodes, a mechanism is needed to differentiate events information contained in multiple packets based on the spatio-temporal correlation of the originator nodes that sensed the events. Existing work is examined on the basis of whether it assumes a pre-located event identified based on the sink-centric approach or random events identified by the nodes in real-time based on a distributed approach.

- *Link-oriented event reliability:*

  Hop-by-hop event reliability, as opposed to the other reliability mechanisms, is sufficient for most of the event-driven WSNs. Existing work is investigated based on the methods involved in creating the overall reliability technique and find the best possible combination of the available options by introducing a three-dimensional (3D) reference model to categorize the work done on providing reliability in WSNs, as shown in Section 3.3.

These requirements represent the choices that must be made in the design of link-oriented event reliability with accurate event identification in a sensor network. The analysis in this chapter describes what choices were made, implicitly or explicitly, for the analyzed systems.

3.2 Event Identification

To achieve event reliability in wireless sensor networks, it is essential for the nodes to identify that two or more packets contain information about the same event or not. Otherwise, it is impossible to suppress the unnecessary transmission of duplicate event data in a non-sink centric way, while making sure that only enough information about the event reaches the sink not all the packets from all the nodes. To the best of our knowledge, none of the existing event reliability schemes have performed link-oriented event reliability with distributed event identification. However, in this section, we examine some general schemes performing event identification, while their focus is not on maintaining reliability.

The hole detection algorithms have been proposed using computational geometry such as Voronoi diagrams and complex simplex methods; however, detecting the location of the hole has been ignored. Zhang *et al.* [132] propose a hole detection algorithm which detects the coverage hole (i.e., the area that is not covered by any node in the network) using existing methods while finding the exact location of the coverage holes using the proposed virtual edge based hole localization strategy. This algorithm works in two steps, it first uses Voronoi diagrams to detect the nodes at the boundary of the hole, following which the location of the hole is detected by using the virtual coverage edge method. The proposed algorithm is compared against Voronoi diagrams and complex simplex algorithms, where the authors claim that the proposed algorithm detects and localizes the coverage hole more accurately than the other two algorithms.

Vuran *et al.* [18] proposed a theoretical framework modeling spatio-temporal correlation for the sink to capture the signals from the sensor nodes within a particular distortion level and with minimum energy expenditure. The authors discussed some of the MAC and transport layer protocols and claimed that none of them had exploited the spatio-temporal correlation among the nodes to provide effective communication.

The sink is interested in estimating the event source (actual event) based on the data it receives from the nodes in the network. Once the sink collected the data from the sensor nodes for the specific time period, it estimates the event source with distortion constraint. It is assumed that the sensed data is sent to the sink by using a reporting frequency, which is controlled such that the distortion level is not exceeded in the estimation of event source at the sink. The authors discussed the possibilities of exploiting spatio-temporal correlation regarding improving the communication protocols at medium access and transport layer levels in a WSN.

At the medium access level, the proposed work makes a subset of all the nodes in the event area to be able to send their sensed data to the sink. This subset of the nodes is known as representative nodes, which are selected randomly in the increasing and decreasing order. It is observed that by decreasing the number of representative nodes, energy efficiency is increased, while the distortion level is maintained. Apart from the number of representative nodes, the distance between representative nodes also puts an effect, where increasing distance saves energy among representative nodes without degrading the distortion at the sink. Thus, the distortion can be minimized by calculating the minimum number of representative nodes, which are closer to the event source but located far from each other.

On the other hand, the importance of spatio-temporal correlation is discussed from the perspective of a reliable transport mechanism in WSNs.

The reliable event detection is performed at the sink while considering the congestion issue by minimizing the traffic as well as the energy expenditure. In the transport mechanism, the event features are estimated at the sink within a distortion bound, i.e., required reliability level. The observed event distortion is compared with the desired event distortion, which is the maximum distortion allowed to assure the reliable event detection and estimation performed at the sink. The reporting frequencies of the source nodes are controlled at the sink to maintain the desired reliability level.

The proposed scheme assumed that the nodes are one hop away from the sensor, which is not a practical approach regarding WSNs. The distances between nodes surrounding the event have been calculated, but it was not discussed that how the distance of the nodes from event source is calculated without knowing the location of the event. It is claimed that energy efficiency can be improved by shifting the burden to the high powered sink to conserve sensor's limited resources. However, this creates the extra communication overhead of all the data going to the sink for centralized decision making.

Villas *et al.* [19] proposed EAST (Efficient Data Collection Aware of Spatio-Temporal Correlation), a clustering-based algorithm for energy efficient data collection in WSNs, exploiting spatio-temporal correlation among the sensor nodes. A leader node aggregates the data sensed by the group of nearby representative nodes, sensing the same event. Spatial correlation occurs when an event is detected by a group of closely located nodes; whereas temporal correlation occurs based on the continuous data gathered by a node over a short period. Most of the previous work discussed on spatio-temporal correlation is based on clustering and does not consider the energy consumption, event characteristics, high delays and outdated data arriving at the sink.

Some applications require closer nodes to transmit the sensed data (correlation region is smaller) while others do not require them to report data (i.e., correlation region is bigger). Therefore, depending on the application requirements, in EAST, the sink dynamically changes the size of correlation region. The event area is divided into cells, where each cell represents a correlation region and nodes within each cell are assumed to be spatially correlated. Similarly, the degree of correlation (temporal correlation) among the continuous sensed data may vary according to the characteristics of the phenomenon. In this way, each source node (representative node) sends the data to the coordinator node only on a condition that the current sensed value is less than the temporal coherency tolerance (tct). Otherwise, it is dropped (i.e., temporal suppression) without communicating with its neighbors. The representative nodes are the nodes having higher residual energy among the rest of the nodes in a cell. The coordinator then sends the data towards the sink by creating a straight line between itself and the sink and choosing the nodes closest to the line as relay nodes towards the sink.

One of the objectives of this study is data accuracy, whereas during the data transmission only the latest sensed data is being forwarded. It might give a more accurate result if the latest and the previous data are efficiently aggregated before forwarding. Furthermore, this scheme proposed a sink centric approach, which can result in high energy consumption. The authors have ignored the overheads of clustering, where a high communication is required to set up and manage the clusters. This scheme assumes a single event occurrence which might not be a reasonable assumption regarding WSNs. Moreover, this algorithm also assumes that the communication link is ideal, which is also not valid in real sensor networks.

In wireless sensor networks, the measurements or sensed data that deviate from a standard pattern are known as outliers. Events are one of the main reasons for outliers, whereas, noise, errors and malicious attacks might also be the other reasons. Zhang *et al.* [20] present an overview of existing outlier detection techniques for WSNs based on the characteristics such as data type, outlier type, outlier identity and outlier degree. It is important to detect outliers to provide data reliability, event reporting and secure functioning of the network.

There are some differences between event and outlier detection, such that, outliers detection techniques do not hold any prior knowledge of trigger condition or semantics of an event, where event detection techniques do. Moreover, outlier detection aims at identifying the anomalous data by comparing the sensed data with each other, while event detection techniques aim at identifying the event by comparing the sensed data with the trigger condition. In addition to this, outlier techniques aim at preventing the standard data to be classified as an outlier, whereas, event detection techniques prevents erroneous data in order to maintain reliability. Besides the differences between outlier and event detection techniques, spatio-temporal correlation is commonly used by both techniques to differentiate between events and errors. This is based on the fact that the erroneous data is likely to be stochastically unrelated, whereas, event data is likely to be spatially correlated. So far outlier detection techniques have not been extensively used regarding event detection because not all outliers have to be identified in event detection applications.

It is a challenging task to make use of the conventional outlier detection techniques in a resource-constrained WSN due to the high computation and communication cost, processing of distributed streaming data online, network topology and heterogeneity of sensor nodes, large scale sensor networks and difficulty in distinguishing between errors, events, and malicious attacks. In addition to the challenges of using outlier detection techniques in WSNs, this work also presents a framework that categorizes several outlier detection techniques. Furthermore, the authors perform a qualitative analysis of some of the techniques based on the nature of sensor data, characteristics of outliers and outlier detection. This work presents an overview of outlier detection for WSNs which would be helpful in gathering some idea to use these techniques in sensor networks.

Yau *et al.* [21] present the analysis of periodically generated stochastic events while monitoring a point of interest (PoI). The analysis highlights the factors that affect the amount of information gathered at a PoI. The scheduling of the periodic sensing varies based on the static or mobile sensor nodes. The static nodes turn off to conserve energy; however, the mobile nodes move between the PoI, where sensors' coverage time, based on the importance of PoI is distributed. This results in the optimization of periodic mobile coverage schedules with equal sharing while improving the overall quality of monitoring (QoM).

The authors focus on the use of mobile sensors to cover a large area, where mobile sensor moves between the PoIs to collect the sensed data and carry it to the sink. The use of mobile nodes not only reduces the sending energy of sensors but also covers a broad network area. Furthermore, different PoI may vary in different regions; therefore, it is essential to evaluate the proportional sharing in terms of sensor coverage properly. Thus this research focuses on the problem of data collection about stochastic events at a set of PoIs, having different importance levels. In addition to data collection, the mobile node allocates its coverage time among the PoIs in proportion to the importance level.

3.3 Reliability

Various protocols have been proposed using different techniques to cope with the challenge of achieving reliability in wireless sensor networks. We investigate the methods involved in creating the overall reliability technique and find the best possible combination of the available options by introducing a three-dimensional (3D) structure to categorize the work done on providing reliability in WSNs, as shown in Figure 3.1. In later sections, the 3D reliability structure will be unfolded to perform in-depth analysis, pointing out the unexplored and efficient combination of the techniques to encounter the challenge of achieving reliability in WSNs. Existing methods to ensure reliability belong mainly to one of two techniques which are retransmission or redundancy. These techniques ensure reliability by recovering the lost data using a hop-by-hop or an end-to-end method, adopting either packet or event level reliability based on the application requirements.

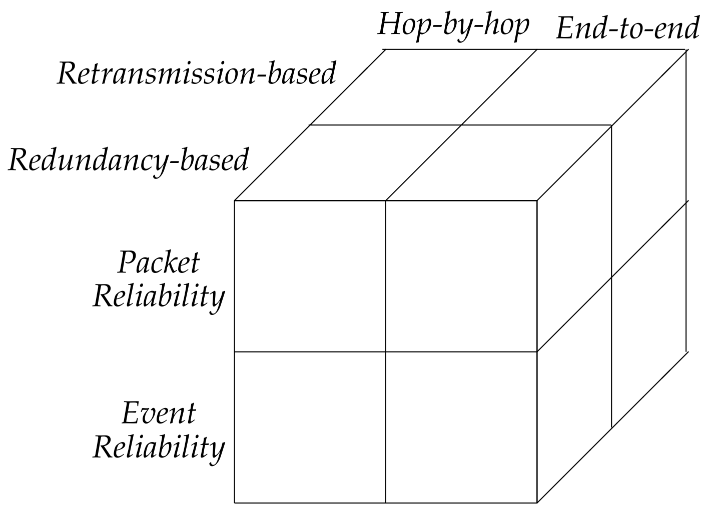


Figure 3.1: A Three-dimensional(3D) Reliability Structure in WSNs

The previous chapter presented an introduction of reliability in WSNs, the challenges involved in achieving reliability in WSNs and the terminology that will be used throughout this thesis. This sections presents an overview of retransmission and redundancy technique and discusses the existing retransmission redundancy based reliability protocols in WSNs. The existing schemes are compared and analyzed using different criterion for comparison by presenting a qualitative analysis.

3.3.1 Retransmissions-based Reliability

In a wired or wireless network, retransmission is the most commonly used technique in order to achieve data reliability. Retransmission is also widely used in a resource-constrained wireless sensor network for recovering the lost data and achieving reliability. Retransmission-based reliability can either be performed using an end-to-end or a hop-by-hop method. End-to-end retransmission requires only the source node that generated the packet to retransmit the lost packet, whereas hop-by-hop retransmission allows the intermediate nodes to perform retransmission of lost packets by caching the information of data packets in their local buffer. In a multi-hop WSN, retransmission-based reliability can be achieved through acknowledgments, where the sender node requires an acknowledgment of its data packets from the next hop relay node on the path to the sink.

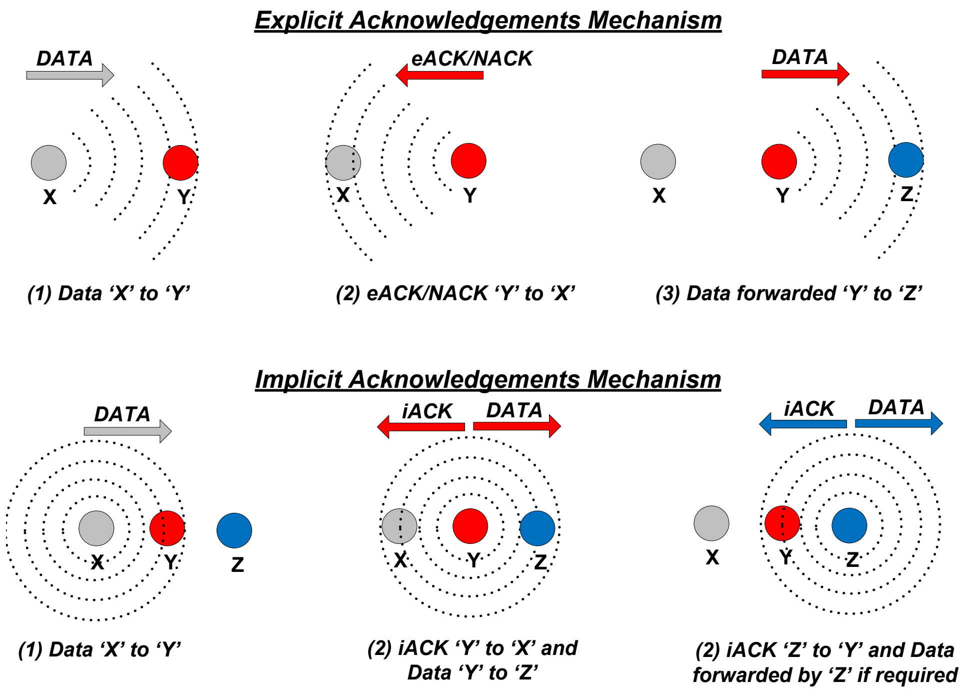

Explicit Acknowledgements Mechanism
DATA
X
Y
(1) Data 'X' to 'Y'
eACK/NACK
X
Y
(2) eACK/NACK 'Y' to 'X'
DATA
X
Y
Z
(3) Data forwarded 'Y' to 'Z'
Implicit Acknowledgements Mechanism
DATA
X
Y
Z
(1) Data 'X' to 'Y'
iACK
DATA
X
Y
Z
(2) iACK 'Y' to 'X' and Data 'Y' to 'Z'
iACK
DATA
X
Y
Z
(2) iACK 'Z' to 'Y' and Data forwarded by 'Z' if required

Figure 3.2: Implicit & Explicit Acknowledgement Mechanisms for Retransmissions [51]

There are two kinds of acknowledgment mechanisms, namely, explicit acknowledgment (eACK), which includes negative acknowledgment (NACK), and implicit acknowledgment (iACK) as shown in Figure 3.2. The eACK mechanism relies on a special control message which the receiving node, after successfully receiving a packet, sends back to the sender as a receipt of the sent packet as shown in Figure 3(a). Like eACK, NACK also corresponds to a special control message that allows the receiver to notify the sender to retransmit a particular missing sequence of packets. The special control messages transmitted in eACK and NACK mechanisms result in additional transmission overhead and energy consumption which may not be feasible for most WSN applications [52]. On the other hand, in the iACK mechanism, the sender after transmitting the packet, listens to the channel and interprets the forwarding of its sent packet by the next hop node as a receipt of the acknowledgment. In this way, iACK exploits the broadcast nature of the wireless channel, avoiding the additional transmission overhead and energy usage caused by the transmission of special control messages (eACK/NACK). A brief overview along with the shortcomings of the major retransmission-based reliability protocols in terms of providing event or packet reliability are presented as follows.

Packet Reliability

We have divided the retransmission-based reliability protocols in WSNs into two major categories based on the amount of data need to be transmitted in order to maintain reliability. Packet-level reliability is one of the categories where the application requires all the packets from all the sensor nodes to be reliably delivered to the sink. This can either be achieved using an end-to-end or a hop-by-hop method. The following subsections briefly describe the schemes within the two categories of end-to-end and hop-by-hop retransmission-based packet reliability.

End-to-End (Connection-based)

Iyer *et al.* [65] propose Sensor Transmission Control Protocol (STCP), a sink centric end-to-end reliability protocol with a congestion control mechanism. Congestion control is performed in such a way that the sink after receiving the information about congested paths, directs the downstream congested nodes to opt for the alternative paths. On the other hand, reliability is maintained by using ACK/NACK based end-to-end retransmission mechanisms, where each source node keeps the packets in its cache until it gets an acknowledgment from the sink. This might not be ideal for resource-constrained WSNs, as in the case of high hop counts from a source node to the sink, the long waiting time for an acknowledgment from the sink can cause high latency and cache overflow.

Like STCP, Zhou *et al.* [77] propose Reliable Transport with Memory Consideration (RTMC), an end-to-end packet reliability protocol addressing the limited memory constraint and unreliable links in WSNs. The source node transmits data segments until the memory of the relay node is full. To prevent memory overflow, the packet headers are loaded with the memory information which is shared among the neighbors. Reliability is maintained by keeping the forwarded segments at the sender until an acknowledgment is received; when the current node's memory is free, it will ask the previous node to send packets. In RTMC, the experiments are performed using only six sensor nodes deployed in a line. This raises the question of scalability, where the performance of RTMC might be significantly degraded in a dynamic topology with a large number of nodes. A large number of nodes would require more data segments to be stored at each hop which increases the possibilities of memory overflow, resulting in packet loss and thus affects reliability.

Unlike STCP and RTMC, Marchi *et al.* [114] propose Distributed Transport for Sensor Networks (DTSN), an energy efficient non-sink centric end-to-end packet reliability protocol that provides a full and a differentiated reliability mechanism. Full reliability is supported with retransmission based explicit acknowledgment mechanism; however, differential reliability is performed independently using the enhancement flow strategy. We will discuss the differential reliability part, involving erasure coding in the related section later. However, in the full reliability mechanism, the source node keeps transmitting the packets until the number of transmitted packets equals the size of the Acknowledgement Window (AW), following which the source node sends an explicit acknowledgment request (EAR) to the destination for the confirmation of message delivery (ACK or NACK). An ACK is sent if the sequence of the packets is in order and these in-sequence packets are removed from the output buffer of the source node. However, if a NACK is received then retransmissions of the missing sequence of packets are required.

The ACK/NACK mechanism results in high message overhead which affects the overall energy efficiency. An overhearing mechanism can be introduced to reduce this message overhead. DTSN provides a mechanism to transfer the core part of data, but does not provide details of how the size of the core data is determined or the reliability level is maintained in changing network conditions. However, unlike other end-to-end retransmission based packet reliability mechanisms, DTSN adopts a distributed approach instead of relying on the sink for the flow control of data making it a more energy efficient approach in a WSN.

Hop-by-hop(Link-based)

This section discusses schemes fall into the category of retransmission-based packet reliability using the hop-by-hop approach. Maroti [72] propose a Directed Flood-Routing Framework (DFRF) using Stop-and-Wait Implicit Acknowledgement (SWIA) approach as a reliable packet recovery mechanism in wireless sensor networks. SWIA makes use of the notion of implicit acknowledgment in WSNs, where a sending node, upon forwarding a packet, listens to the channel to see if the receiver has further forwarded the packet towards the destination. If the sender does not hear anything, the packet is considered as lost and to maintain reliability; every lost packet is retransmitted. This may aggravate the problem of channel contention and network congestion. Most sensor network applications can afford some degree of the packet loss and attempt to retransmit every single lost packet may be both unnecessary and detrimental to the reliability of the network.

Zhang *et al.* [73, 74] propose the Reliable Bursty Convergecast (RBC) protocol, designed for challenging situations where a large burst of packets from different locations needs to be reliably transported to the sink. The main benefits and design considerations for RBC are the need to improve channel utilization and alleviate retransmission incurred channel contention. It employs a windowless block acknowledgment scheme to improve channel utilization and implements an adaptive timer mechanism for retransmission timeouts to control the channel contention caused by the retransmissions. However, the use of priority schemes to schedule transmission could make packets with a lower priority wait forever in the queue, resulting in high latency and unnecessary queue occupancy. While RBC appears to provide good reliability, it lacks validation of its energy consumption, which is one of the most critical issues of WSNs.

Unlike others, Le *et al.* [75] focus on the energy constraint and propose ERTP, an energy efficient reliability protocol providing a sensor to sink reliability by maintaining reliability at each hop. ERTP provides statistical reliability determined by the quantity of data packets received at the sink for a WSN application producing streaming data. It uses the SWIA mechanism to recover the lost packets and utilizes a retransmission timeout estimation mechanism that dynamically minimizes the number of retransmissions to reduce energy consumption. ERTP has identified the two most essential issues in WSNs by focusing on both reliability and energy efficiency together. ERTP has been validated with a low transmission rate and a negligible amount of congestion. A higher transmission rate with more nodes should also be considered to check the overall performance. Moreover, it uses a low power listening (LPL) MAC protocol for implicit acknowledgments, which might result in low overhearing and consequently packet loss.

Shaikh *et al.* [76] propose Tuneable Reliability with Congestion Control for Information Transport (TRCCIT), aiming to provide reliable data transmission in varying network conditions and application requirements. This is achieved using a localized hybrid acknowledgment (HACK) mechanism and a timer management system, where HACK is a combination of implicit and explicit acknowledgment. Implicit acknowledgments are used as the main source of acknowledgment; however, the next hop, after receiving the information from the sender, can send an eACK to the sender to stop its unnecessary retransmissions. On the other hand, the adaptive retransmission timeout mechanism is based on the buffer occupancy and balancing the number of incoming and outgoing messages at each node.

Stann and Heidemann's [79] Reliable Multi-Segment Transport (RMST) is a NACK-based sink-centric scheme designed to make directed diffusion reliable. It has a caching and non-caching mode which decides whether or not the data segments need to be stored for retransmission purposes. One of the disadvantages of RMST is that it does not handle the possibility of NACK implosion when the downstream nodes issue a chain of NACK requests for gaps detected in the non-caching mode.

Even though downstream (sensor to sink) reliability is not the focus of this thesis, it is worthwhile to briefly discuss a few downstream reliability protocols in WSNs, as they also fall within the category of hop-by-hop packet reliability. Wan *et al.* [56] and Park *et al.* [57] propose hop-by-hop downstream reliability mechanisms, Pump Slowly Fetch Quickly (PSFQ) and GARUDA respectively. Both these protocols aim to achieve reliability guarantee to disseminate the control messages or software code segments towards downstream sensor nodes while performing loss detection and data recovery using retransmissions.

Event Reliability

Apart from packet reliability, event reliability is the other category based on the amount of data required to ensure the reliable transmission using retransmissions technique. In WSNs, event reliability, as opposed to packet reliability, should be sufficient for most of the event-driven applications. In this context of event reliability, reliability means that at least one of the many packets from the sensors that detected an event is delivered to the sink, instead of all the packets from all the sensors. The following discussion covers the major schemes that fall within this category of end-to-end or hop-by-hop retransmission-based event reliability.

End-to-End (Connection-based)

Akan *et al.* [62, 63] are the first to introduce the concept of event reliability by proposing the Event-to-sink reliability protocol (ESRT), where reliability is measured by the number of packets carrying information about a particular event that are delivered to the sink. ESRT is sink centric protocol, where the sink centrally controls the flow of data in such a way that when the required reliability is not achieved, the sink will increase the event reporting frequency *(f)* of the nodes. Conversely, it will reduce *f* when there is congestion in order to restore the reliability required at the sink. If the required reliability is observed at the sink, it will decrease *f* to reduce the nodes' energy consumption. A key assumption of ESRT is that the sink has a high power radio that can transmit the latest value of *f* to all sensor nodes in a single broadcast message, and each node will adjust its *f* accordingly. This is a severe limitation because conventionally the sink is not one hop away from all the sensor nodes. Furthermore, ESRT increases the source sending rate in order to guarantee event reliability; however, this central rate control method is not as energy efficient as hop-by-hop retransmission-based loss recovery schemes. This would further deteriorate the overall bandwidth utilization and introduce high energy wastage as different parts of the network have different network conditions.

Gungor and Akan [64] propose Delay Sensitive Transport (DST), an extended version of ESRT by adding application-specific delay bounds with a Time Critical Event First (TCEF) scheduling policy. In TCEF, the data packets within each node's queue with lower remaining time to the deadline are given a high priority for transmission. The remaining time to the deadline is calculated by piggybacking the elapsed time information of that event with the data packets, which eliminates the need for global time synchronization. Like ESRT, one of the major criticisms of DST is that the whole functionality is entirely dependent on the one point that is the sink to regulate the flow of data, which results in increased traffic and higher energy usage. Moreover, DST works around a single predefined event with the known location which is not practically sound, as, in a WSN, multiple concurrent events may happen in close proximity or all across the network.

Tezcan and Wang [67] propose Asymmetric and Reliable Transport (ART) by choosing some essential nodes (E-nodes) within the sensor network that work together as cluster heads to provide upstream end-to-end reliability and downstream query reliability. Thus, ART is one of the few protocols to provide bidirectional reliability. The E-nodes do most of the work; thus, all the other non E-nodes do not incur end-to-end communication overhead. The E-nodes are selected by looking at the remaining battery power of the nodes and picking those with the highest remaining energy. To achieve upstream reliability, ART uses the ACK mechanism between the E-nodes and the sink, while reliable downstream query propagation is performed using the NACK mechanism. E-nodes also regulate the data flow by restricting the neighboring non E-nodes from sending their data until the congestion is cleared. As the E-nodes perform the congestion control and event reliability, packet loss due to congestion at non-essential nodes cannot be recovered by this protocol. ART assumes that there is congestion when an ACK is not received within a timeout period, which is not a reasonable assumption as it is not necessary that the ACK is lost due to congestion; it might also be lost due to poor link quality or some other form of transmission errors. Moreover, clustering involves a lot of communication, which might not be suitable for a resource constraint WSN.

Following ART, Xue *et al.* [69] propose a Loss-Tolerant Reliable Event Sensing transport protocol (LTRES), an end-to-end event reliability protocol designed for applications with heterogeneous sensing fidelity requirements over different event areas within a WSN. It aims to provide source rate adaption based on network capacity awareness which is calculated by measuring the event goodput at the sink. In addition, it also employs a lightweight congestion control mechanism, where instead of relying on the nodes to report congestion, the sink itself suppresses the transmission of the aggressive nodes based on the data it receives. This protocol is entirely dependent on the sink, where the sink sends the updated source rates to the downstream E-nodes, but it does not employ any downstream reliability or congestion control mechanism. Lack of loss recovery mechanism and reliability considerations for the non-essential nodes is another critical issue that is ignored.

Recently, Hosung *et al.* [70] propose Quality-based Event Reliability Protocol (QERP), which unlike the existing quantity-based event reliability protocols emphasize on the quality of event data reported to the sink. The authors claim that other existing quantity-based event reliability protocols are not suitable for the resource-constrained wireless sensor networks, as they aim to achieve reliability by increasing or decreasing the frequency of data reporting rates. These varying frequencies sometimes increase the overall data rate more than the desired rate, which would aggravate the network conditions. QERP as an end-to-end event reliability protocol claims to provide better reliability than ESRT by using the concept of Contribution Degree (CD). This means that the difference of sensor data in CD for event detection is due to the different environmental conditions like varying distance of nodes from the events. The data packets are marked with a CD value, where those originating from the nodes located close to the event's location are assigned with higher CD values and assumed to be more critical and are transmitted on higher priority. The node closest to the event is selected as a leader node, while the rest of the nodes in the event's region report their data to the sink through the leader nodes. If the node density is high near the event's location, then the process of selecting the leader node would be longer, resulting in energy wastage. There is no packet loss detection mechanism, and the data flow mechanism is based on the event's location; however, there is no mention of the mechanism used to find the event's location, which is used to define the CD values.

The Price-Oriented Reliable Transport Protocol (PORT) proposed by Zhou *et al.* [66] aims to provide finer fidelity in event reliability as compared to ESRT, while minimizing energy consumption. Unlike ESRT [63], PORT performs the adjustment of frequency reporting rate in a different manner, by adjusting the different reporting rates of source nodes in different regions accordingly. In addition, PORT also provides energy efficient congestion control mechanism by avoiding the nodes along the paths having high loss rates based on the node price, which is defined as the number of transmission attempts made before the packet is successfully delivered. The energy cost for the communication is also evaluated with the help of the node price. The sink regulates the reporting rate of a particular source based on the communication cost information according to the changing network conditions, where the sink decreases the reporting rates of sources having higher communication cost and vice versa while maintaining reliability.

PORT has been shown to be more energy efficient compared to ESRT, and PORT works well for the networks having diversified reporting rates at different parts of the network. However, if the reporting rates of different areas of the network are similar, then PORT will also give similar results to the other schemes. Like the other event reliability schemes, PORT also requires a high powered sink to regulate the flow of data and does not provide any packet loss recovery mechanism which may be vital in high loss environments.

Hop-by-hop (Link-based)

Most recently Mahmood and Seah [71] propose the Event Reliability Protocol (ERP), an upstream hop-by-hop event reliability protocol which aims to deliver the information about event occurrences to the sink reliably. It suppresses the transmission of redundant packets containing information about similar events based on spatio-temporal correlation. This reliable transmission of unique events to the sink is performed with the help an intelligent region-based selective retransmission mechanism. Apart from minimizing the transmission of unnecessary duplicate packets containing similar event data, ERP further reduces the traffic by exploiting the broadcast nature of the wireless channel through the use of implicit acknowledgments (iACK). Instead of relying on the sink to control the data flow, ERP performs in-network data processing that saves the overall network cost in terms of energy, data flow, and congestion. The proposed approach highlights the importance of in-node data processing while minimizing the dependability on the sink. A more robust event identification mechanism would further strengthen its performance. However, the use of eACKs might also be a suitable option, since eACKs do not force the sender to listen on the channel for possibly a long time before it can overhear the packet transmission.

Open Issues

We have briefly described retransmission-based reliability, followed by the most important schemes in WSNs under this category. We have further subdivided this category into sub-categories based on a different criterion to achieve reliability, such as end-to-end or hop-by-hop, packet or event level, types of acknowledgments and sink or non-sink centric approaches. Each of the schemes uses a combination of this criterion to achieve reliability, but the existing research still has open issues to be dealt with. One of the most critical issues in most of the retransmission based reliability schemes is the dependency on the sink to control the data flow and maintain reliability. This is not an appropriate approach to be used in WSNs, as this requires the nodes to first transmit the data towards the sink, where the sink adjusts the data rate for the downstream nodes to achieve the required reliability. Therefore, even the most efficient of the schemes using sink-centric approach would require a lot of communication, which in turn affects the energy efficiency. It would be better to use a distributed approach in which nodes perform in-node processing to maintain the reliability, as communication is more costly than processing. Likewise, schemes using clustering also incur additional communication overhead for cluster management.

An important issue and challenge in event reliability schemes is the accurate identification of the event, which has been overlooked by the existing research. Most, if not all, of the existing schemes use pre-defined events and their locations are already known, which is not realistic. In a wireless sensor network, events can randomly happen anywhere, and the locations should be identified on the fly, using some appropriate methods. Scalability is another critical issue, as most of the schemes have only been validated using a limited number of scenarios with fewer nodes. For a retransmission based reliability scheme, it is essential to choose the most efficient acknowledgment mechanism for the recovery of the lost packets, but many of the schemes have not emphasized on this issue. Moreover, introducing an intelligent retransmission timeout mechanism also has considerable influence in preventing the stale packets occupying the sensors' limited buffers. The following section will provide a brief overview of redundancy-based reliability mechanisms and discuss the published work on this topic.

3.3.2 Redundancy-based Reliability

This section addresses some of the existing protocols that aim to achieve reliability through the use of information redundancy. Redundancy-based reliability in WSNs is not the focus of this thesis; however, the related work is discussed for the complete understanding of overall reliability in WSNs. In retransmission-based reliability, a bit lost within a packet is treated as a loss of the entire packet and requires the retransmission of that packet to achieve reliable data transmission. This motivates the concept of redundancy-based reliability where the lost or corrupted bits within the packet can be recovered through the use of coding techniques [94, 95]. This reduces the transmission overhead caused by the retransmission of the entire packet [97]. Keeping in view the resource-constrained nature of WSNs, employing redundancy techniques can reduce the overall network cost in terms of energy, communication, memory usage and increased network lifetime at the cost of adding information redundancy.

Among the coding techniques, Forward Error Correction (FEC) techniques are used for error detection and correction at the lower layers of the protocol stack for telecommunication and computer communication (e.g., CRC in computer communication) systems [53]. However, when it comes to the upper layers (e.g. link, transport and application layers), erasure codes are needed to deal with the missing packets, since it is easier to determine the exact position of the lost data. Rizzo [54] implemented and evaluated erasure coding for a desktop computer network. By carefully choosing the optimization parameters, erasure coding can also be implemented in a resource-constrained WSN.

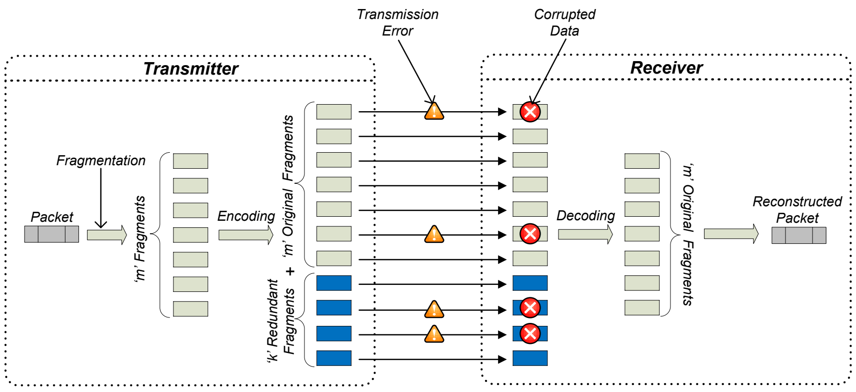


Figure 3.3: Redundancy Mechanism using Erasure Coding (adapted from [116])

In erasure coding, the sender divides the packet into m fragments, which are encoded, adding another k fragments. These m+k fragments are transmitted to the sink which can reconstruct the original data packet out of it. The receiver (i.e., the sink) is able to reconstruct the original data packet only on the condition that the number of fragments it has received i.e., m' is equal to or higher than the number of original fragments, i.e. m as shown in Figure 3.3. Like retransmissions, redundancy-based reliability can also be achieved on an end-to-end (connection) or hop-by-hop (link) level. In end-to-end erasure coding, the encoding/decoding of data is only performed at the source and the sink nodes, and the intermediate nodes simply relay the packets. However, in hop-by-hop erasure coding, encoding/decoding of data is performed at every intermediate node through to the sink. Thus, unlike end-to-end erasure coding, hop-by-hop erasure coding refreshes the data at each hop, reducing the chance of losing more than the desired amount of fragments on the way to the sink. Therefore, end-to-end erasure coding is not as reliable as hop-by-hop erasure coding in a lossy environment with a higher number of hops from the source node to the sink. The existing research employing information redundancy to achieve reliability in WSNs is discussed below.

Packet Reliability

Like retransmissions, redundancy-based reliability protocols in WSNs are also further subdivided into two major categories based on the quantity of data packets required by the application to maintain reliability. First, we discuss packet level reliability that aims to ensure the delivery of every data packet to the sink. This can either be achieved by performing encoding/decoding only at the source and the destination node or at every hop from the source to the destination.

End-to-End (Connection-based)

Ali *et al.* [112] propose Optimum Reed-Solomon Erasure Coding (OREC) to achieve reliability in a distributed wireless sensor network. The source node creates and transmits the fragments of a packet using Reed-Solomon codes, a class of erasure codes, along with multiple paths towards the sink where they are reconstructed. The cost of transmitting these fragments under different network conditions is computed and Genetic Algorithms (GAs) are used to determine the optimized number of fragments to be transmitted accordingly. This scheme assumed a query-based WSN where the sink sends the query to the prongs which are nodes one hop from the sink. The prongs further broadcast the queries throughout the network, and the relay nodes keep track of the path through which they received the queries. This helps the relay nodes to forward the response of the queries back to the sink through the prongs while selecting the path with short hop counts or high reliability.

Theoretical analysis of OREC shows that by increasing the number of parity fragments (additional redundant fragments), reliability is increased at the cost of traffic overhead. Therefore, intelligent selection of the number of redundant bits becomes the optimization factor in Reed-Solomon codes. However, the use of genetic algorithms requires intensive processing and might not be suitable for a resource-constrained WSN. Also not considered is the case of large networks with a high hop count between the source node and the sink which may affect the performance of OREC, as the encoding/decoding is performed only at the source node and the sink, introducing many entry points for errors. Moreover, the proposed scheme uses downstream query processing without introducing any downstream reliability mechanism. The idea of introducing prongs does not seem to be suitable, as it will create an extra processing overhead on the nodes acting as prongs while wasting the enhanced processing abilities of the sink.

Distributed Transport for Sensor Networks (DTSN) proposed by Marchi *et al.* [114] provides an energy efficient non-sink centric end-to-end packet reliability protocol with a full and a differentiated reliability mechanism. We have already discussed full reliability mechanism in Section 3.3.1 under the retransmissions category. However, in cases where full reliability is not required, a differential reliability mechanism is used separately with an addition of a Forward Error Correction (FEC) technique using erasure coding. Their proposed Enhancement Flow strategy supports differential reliability. This is used to transfer a block of data, say an image, where the basic image (i.e., an image with minimum resolution) is reliably transferred using FEC, performed independently; however, FEC support is not provided to the rest of the image (i.e., an image with maximum resolution). Enhancement Flow along with the FEC technique is implemented separately, independent of retransmission-based full reliability mechanism. This results in high reliability and improved throughput as compared to full reliability strategy with the complete block transfer. DTSN employs FEC on an end-to-end basis, where a high hop count from source to a destination increases the loss probability of data fragments, and consequently the destination will not be able to perform error correction entirely. DTSN provides a mechanism to transfer the core part of data, but does not provide details of how the size of the core data is determined or the reliability level is maintained under changing network conditions.

Unlike other redundancy-based reliability schemes in WSNs, Kim *et al.* [115] propose Reliable Transfer on Wireless Sensor Networks (RTSN) by experimentally validating the use of erasure codes on sensor test beds. RTSN is a packet level reliability mechanism using systematic codes, where the node does not need to perform a great deal of computation for the reconstruction of the original message. Systematic codes are implemented by labeling the transmitted packets either as an original or a redundant packet. This is because, if a part of the encoded message is the original message itself, then the receiving node can reconstruct it without performing any decoding operation. This will not only reduce the processing cost and memory usage of the nodes but also improve reliability and energy efficiency. The RTSN reliability mechanism has been compared with simple retransmission based reliability where retransmission results in high latency and buffer overflow, which was alleviated by introducing erasure coding at the cost of traffic overhead. The performance of erasure coding deteriorates in a lossy environment, therefore finding alternative routes when there is a continuous packet loss at the current path would be a better option.

Hop-by-Hop (Link-based)

Wen *et al.* [99] propose Transmission Reliability in Sensor Networks (TRSN), an energy efficient reliability mechanism, where reliability is evaluated in terms of packet arrival probability by checking it against required reliability, while energy efficiency is evaluated in terms of energy consumed in the successful delivery of one packet. The authors modeled the loss behavior using the Gilbert model [100], which is basically designed for a one-hop channel. However, for a multi-hop network, it is assumed that the two-state Gilbert model can be adopted independently at each hop of a multi-hop transmission. The successful packet arrival probability through multiple (n) hops and average energy consumption for one successful packet arrival is theoretically calculated and analyzed for simple transmissions, retransmissions and erasure coding mechanisms. The packet arrival probability and energy consumption are mainly compared against the unconditional loss probability (ulp), which is the probability that the next packet will be lost based on the condition that the previous packet was lost. Finally, erasure coding is evaluated to be better than the hop-by-hop retransmission mechanism in terms of energy efficiency only on the condition that the ulp value is below a certain threshold. However, it is pertinent to mention that the performance of erasure codes might degrade in an environment with high loss rate and have a negative effect on both reliability and energy efficiency. Therefore, it is essential to evaluate the performance of these mechanisms with varying loss rates to make a robust network capable of performing well in a lossy environment. In addition, the proposed analysis should also be validated experimentally or through simulation.

Most recently, Srouji *et al*. [116] propose the Reliable erasure-coding based Data Transfer Scheme (RDTS) that is an efficient hop-by-hop data transfer reliability mechanism using erasure coding. Unlike end-to-end redundancy based schemes, RDTS uses erasure coding at each hop to attain hop-by-hop reliability. In addition to erasure coding, RDTS applies partial coding to reduce the computational overhead triggered by performing erasure coding at each hop. The partial coding mechanism further reduces the cost of performing the complete erasure coding process at each intermediate node. This ensures that the receiver has received and transmitted enough data fragments required at the next hop for the reconstruction of the original message. Therefore, the encoding/decoding process is performed only when one or more original fragments are lost in transmission. Thus, the overall cost of transmission is reduced due to lower processing overheads and reduced delay. RDTS calculates the number of fragments required at the next hop on the basis of the probability of successful data arrival, which is assumed to be a random value predefined from 0.7 to 0.95. However, the overall performance of RDTS could be improved by calculating this probability dynamically, according to the varying network conditions.

Most of the schemes discussed so far focus on upstream reliability guarantees implemented using erasure codes; however, Kumar *et al.* [117] propose one of the very few schemes to provide redundancy-based downstream data transmission reliability in a large WSN. The context for Kumar's work was pushing out software updates to WSN nodes. When the software codes need to be updated for the downstream nodes, the sink is required to broadcast the software updates throughout the sensor network. This broadcast results in excessive overhead which may become the cause of data loss and affect overall reliability. Unfortunately, a partial software update is unacceptable as all nodes must be updated for the whole network to function correctly. To provide 100% reliability, Kumar *et al.* [117] propose FBcast, a redundancy-based downstream reliability protocol, using FEC techniques.

Kumar *et al.* use fountain codes [120], a class of erasure codes, producing an encoded block of data on-the-fly without any dependence on the limited buffer space. The encoded data is broadcast to the neighbors, which is further rebroadcast when new fragments are received. This protocol is based on the assumption that all the nodes in the network are one hop away from the sink, which broadcasts the updated codes in a single broadcast message. This is not a feasible solution for a network where the sink is multiple hops away from the downstream nodes. Thus Kumar *et al.* propose an extended version of FBcast protocol with the concept of repeaters. The repeater functionality is applied on the sensor nodes, where the sensor nodes, after receiving enough data fragments, reconstruct the original data. This reconstructed data is further re-encoded and rebroadcasted to these sensors' neighbors and so on. In another variation of the scheme, a probabilistic broadcast scheme is used; however, this does not provide good reliability. The idea of implementing erasure codes for ensuring downstream reliability might be new, but performing encoding/decoding at every downstream node of the network to successfully broadcast the data throughout the network is an expensive choice in a dense network.

Event Reliability

To the best of our knowledge, the existing research on reliability in WSNs ignores the use of redundancy-based event reliability. The interdependent nature of different techniques and methods discussed to achieve reliability has led us to develop the 3D reliability structure presented earlier. This framework highlights the existence of an unexplored area that seems to be promising in improving the energy efficient reliable transmission of events' information towards the destination. This is highlighted in Figure 3.4 by a blank square showing that existing research does not exploit redundancy-based event reliability by using either an end-to-end or a hop-by-hop method.

Achieving redundancy-based event reliability involves the challenge of accurate identification of the event and its location to achieve a high degree of efficiency in monitoring and control systems. We argue that information redundancy mechanism could be used to achieve event reliability. We reason that the event data from closely located nodes in a sensor network tends to be highly correlated. Therefore, transmitting as few as one single

	Hop-by-Hop	End-to-End		
Retransmission based	PORT [66];ERP [71] SWIA [72];ERTP [75] TRCCIT [76];RBC [74] RMST [79];PSFQ [56] GARUDA [57]	ESRT [63];DST [64] ART [67];LTRES [69] RTMC [77];DTSN [114] STCP [65];QERP [70]		
Redundancy based	RDTS [116] FBcast [117]	TRSN [99] OREC [112] DTSN [114] RTSN [115]		
Packet Reliability	RDTS [116],ERTP [75] FBcast [117];SWIA [72] GARUDA [57] TRCCIT [76];RBC [74] RMST [79];PSFQ [56]	TRSN [99];OREC [112] DTSN [114];RTSN [115] RTMC [77] STCP [65]	TRSN [99];RDTS [116] OREC [112];FB-cast [117] DTSN [114] RTSN [115]	TRCCIT [76];RBC [74] DTSN [114];ERTP [75] GARUDA [57];SWIA [72] STCP [65];RTMC [77] RMST [79];PSFQ [56]
Event Reliability	ERP [71]	ESRT [63];DST [64] ART [67];LTRES [69] STCP [65];QERP [70]		ESRT [63];DST [64] PORT [66];ART [67] LTRES [69];ERP [71] STCP [65];QERP [70]
			Redundancy based	Retransmission based

Figure 3.4: Unfolded 3D Reliability Structure

packet carrying information about the event instead of multiple copies of the same packet would not only reduce the transmission overhead but also save the sensors' limited energy. Also, it is more favorable to perform information redundancy in a hop-by-hop fashion instead of end-to-end. This is because hop-by-hop information redundancy is better than the end-to-end approach, as discussed in the earlier sections. This would not only take away the additional cost of retransmissions but will also improve energy efficiency and further strengthen the reliable transmission of data.

Open Issues

This section has highlighted the importance of redundancy-based reliability techniques in WSNs, followed by a brief overview of how the encoding/decoding is performed. We provide an overview and critical analysis of the existing schemes that use information redundancy. One of the important issues that need attention is that the performance of most of the existing schemes has only been assessed theoretically. Ideally, these theoretical performance evaluations should further be validated through simulation and/or experimentation. This would verify not only in-depth implementation details but also the robustness of the respective schemes under varying network conditions. Another important issue that needs to be explored is how to intelligently select the number of redundant bits to be added to the packet to reconstruct the original packet. This intelligent mechanism should be able to select the minimum amount of redundant bits according to the current network's conditions. In this way, it will not only decrease the overall network load but will also save sensors' limited energy and bolster overall reliability. Above all, a significant gap in this research area as highlighted in this thesis is to investigate the redundancy-based event reliability mechanism further. The next section gives a brief introduction to the hybrid mechanism, another approach that this thesis proposes to achieve reliability.

3.4 Qualitative Analysis

Tables 3.1 and 3.2 present the comparisons of the schemes discussed in the previous sections. Each scheme aims to achieve reliable transmission of data through the use of retransmission and redundancy techniques. The criteria used to compare are support for packet or event reliability, traffic flow, hop-by-hop or end-to-end methods, loss recovery, use of acknowledgment mechanisms and level of dependency on the sink to control the flow of data. Each scheme combines these factors in different ways to implement reliable data transmission.

Retransmissions-based Schemes						
Protocol	*Traffic Flow*	*Reliability Level*	*Loss Recovery*	*ACK Mechanism*	*Sink Centric*	*Coding Scheme*
ERP [71]	Up	Event	Hop-by-hop	iACK	No	No
ESRT [63]	Up	Event	End-to-end	–	Yes	No
DST [64]	Up	Event	End-to-end	–	Yes	No
PORT [66]	Up	Event	End-to-end	–	Yes	No
ART [67]	Up/ Down	Event	End-to-end	ACK/NACK	Yes	No
LTRES [69]	Up	Event	End-to-end	ACK/NACK	Yes	No
QERP [70]	Up	Event	End-to-end	–	Yes	No
STCP [65]	Up	Event/ Packet	End-to-end	ACK/NACK	Yes	No
SWIA [72]	Up	Packet	Hop-by-hop	iACK	No	No
RBC [74]	Up	Packet	Hop-by-hop	iACK	Yes	No
ERTP [75]	Up	Packet	Hop-by-hop	iACK/ACK	Yes	No
TRCCIT [76]	Up	Packet	Hop-by-hop	iACK/ACK	Yes	No
RTMC [77]	Up	Packet	End-to-end	–	Yes	No
RMST [79]	Up	Packet	Hop-by-hop	NACK	Yes	No
DTSN [114]	Up	Packet	End-to-end	ACK/SACK	Yes	No
PSFQ [56]	Down	Packet	Hop-by-hop	NACK	Yes	No
GARUDA [57]	Down	Packet	Hop-by-hop	NACK	Yes	No

Table 3.1: Comparison of Retransmission-based Reliability Schemes

Table 3.1 presents the retransmission based reliability schemes that we have discussed, where the major portion focuses on achieving packet level reliability based on the application requirements. These schemes have been developed to support a wide range of applications: e.g., TRCCIT and STCP may be suitable for heterogeneous and concurrent multiple application, ESRT, LTRES and QERP for event detection applications like signal estimation or tracking, ART and DST may be useful for mission critical applications like country border security and intrusion detection, and ERTP for data streaming applications such as weather and habitat monitoring. Some other protocols like ERP is suitable for event-driven applications, RBC for high volume bursty traffic applications, RMST for multimedia applications as it requires fragmentation/reassembly and others used for general purpose applications. Therefore, it is worth mentioning that all the schemes can be used for a wide range of applications based on their functionalities discussed.

Most, if not all the existing schemes make two design assumptions. First, reliance on the sink to impose some form of control over the flow of data in the network in order to maintain the required reliability level. Second, a reliable downstream communication mechanism is available. With regard to the sink-reliance assumption, ERP is the exception; it uses in-network data processing where the data packets are processed at the nodes before forwarding them further. This in-node processing not only reduces unnecessary retransmission of the packets but also saves the overall network cost of in term of energy consumption and data flow controlled by the sink. Among the existing schemes, the majority recover the lost packets using explicit acknowledgments; however, only a few exploit the broadcast characteristics of a wireless channel by using implicit acknowledgments. Some of them like ESRT, DST, PORT and QERP do not explicitly mention any loss detection and notification mechanism.

Event reliability tends to be a more energy efficient strategy than packet reliability when performed with hop-by-hop retransmissions but requires an efficient cache management system at each hop. ESRT, DST, STCP, ART, LTRES, PORT and QERP aim to achieve event reliability while recovering the lost packets based on the end-to-end retransmissions, where the rest of the schemes use the hop-by-hop method. The end-to-end retransmission based loss recovery strategy, in comparison to hop-by-hop loss recovery, introduces high delay and energy wastage especially in a network with a high hop count. In end-to-end retransmissions, the packet stays in the cache of the source node till it is acknowledged by the sink which consumes more energy at the node and introduces delays especially when the sink is multiple hops away from the source node. Thus, hop-by-hop performs better than end-to-end reliability in terms of less energy consumption, low communication cost, and quick loss recovery and fast congestion alleviation.

Among the existing event reliability schemes, PORT aims to achieve end-to-end event reliability by relying on the sink for the centralized control without providing any loss detection and notification mechanism. On the other hand, STCP, ART and LTRES use explicit acknowledgments as a loss detection and notification mechanism, introducing extra message overhead over the network. This requires a hop-by-hop event reliability mechanism that performs loss detection by using implicit acknowledgments, saving the overhead caused by using explicit acknowledgments mechanisms. Moreover, instead of relying on the sink to attain certain application-specific reliability level, this event reliability mechanism needs to perform the decisions at the node level using in-node data processing to save the overall communication cost of the network.

Similarly, the packet reliability schemes using hop-by-hop loss recovery mechanism also perform better in terms of energy efficiency and reliability in comparison to the end-to-end mechanism. The performance of packet reliability schemes can also be improved by exploiting the broadcast nature of a wireless channel using implicit acknowledgment. This can minimize the use of explicit acknowledgments which adds to the traffic overhead. Like upstream reliability, downstream reliability also plays an important role when the software updates need to be distributed to sensor nodes; however, this topic of distributing software updates to sensor nodes is out of the scope of this thesis. Among the schemes studied, only ART performs both upstream and downstream reliability, whereas STCP is the most flexible regarding providing both the event and packet reliability depending on the application requirement.

Redundancy-based Schemes						
Protocol	***Traffic Flow***	***Reliability Level***	***Encoding/ Decoding***	***Coding Scheme***	***Evaluation***	***Supportive Mechanism***
TRSN [99]	Up	Packet	hop-by-hop	Erasure Codes(EC)	Theoretical Analysis	ULP
OREC [112]	Up	Packet	End-to-end	RS-Codes	Theoretical Analysis	Genetic Algorithms
DTSN [114]	Up	Packet	End-to-end	Erasure Codes	Simulations	Enhancement Flow
RTSN [115]	Up	Packet	End-to-end	Systematic & EC	Real test-beds	Alternate Route-fix
RDTS [116]	Up	Packet	Hop-by-hop	Erasure Codes	Simulations	Partial Coding
FBcast [117]	Down	Packet	Hop-by-hop	Erasure Codes	Simulations	FB Cast

Table 3.2: Comparison of Redundancy-based Reliability Schemes

Apart from retransmissions, few schemes rely on adding information redundancy to achieve data transport reliability in wireless sensor networks as shown in Table 3.2. We have discussed the redundancy-based schemes in Section 3, where Rizzo *et al.* [54] first validated the erasure codes in computer communication which motivated the idea of implementing erasure codes in resource-constrained WSNs. Among the existing redundancy based schemes, TRSN, OREC, DTSN, RTSN and RDTS aim to achieve upstream packet reliability, while only FBcast focuses on achieving downstream reliability. All these protocols claim to achieve high reliability at the cost of message overhead.

TRSN and OREC performed theoretical analysis, while DTSN, RDTS and FBcast performed simulations to evaluate their performance. Only

RTSN experimentally validated erasure codes on testbeds and claimed to achieve close to 100% reliability. Thus, the erasure coding mechanism should be a good candidate for further enhancing the reliability in the field of WSNs. More recently, RDTS proposes the idea of performing erasure coding in a hop-by-hop fashion and achieved better performance than end-to-end erasure coding in terms of reduced coding overhead and increased network lifetime. This motivates the need to investigate the event reliability mechanism using information redundancy for reliable transport of event data from the source to the sink in a distributed fashion. There is also a need to introduce a distributed redundancy-based event reliability mechanism based on a non-sink centric approach which, to the best of our knowledge, has not been researched in the existing literature.

3.5 Discussion

WSNs hold the promise of many new applications in the field of monitoring and control systems. With the advent of cheap and small sensors, monitoring systems can make use of them to monitor numerous characteristics of the environment. All these applications are built for specific purposes, where maintaining reliable transmission of data from source to destination is one of the most important challenges. To address this issue, we surveyed the various existing protocols, where each of them has its approach to ensure reliability. Some of them require full end-to-end packet reliability while others can tolerate some degree of packet loss depending on the nature of the application. The most commonly used techniques to ensure reliability includes the use of retransmissions, while few use information redundancy in WSNs.

We surveyed, compared and critically analyzed several WSNs reliability protocols, where high loss rate is one of the challenges which is better dealt with using hop-by-hop mechanisms than end-to-end mechanisms. The high hop count in large WSNs introduces more entry points for errors which become the cause of packet loss, thus affects reliability. This can be avoided to some extent by using a hop-by-hop method, as it performs loss recovery at each intermediate hop from source to the destination. Another challenge is the provisioning of a dynamic retransmission timeout and an efficient buffer management system for preventing stale packets from occupying the queue. This would be beneficial to save sensors' limited memory resources.

Another major challenge for the event reliability protocols is an accurate event identification mechanism, which is the primary characteristic of an event-driven wireless sensor network. Once the events are accurately identified, then only a single packet carrying information about the event would be enough for the sink instead of receiving all the packets from all the nodes carrying information about the same event. This requires the nodes to suppress the transmission of the packets that are found to be in the same sensing region by exploiting spatio-temporal correlation [15] among the closely located sensor nodes.

Most of the existing reliability protocols have been evaluated by simulation, where real-time implementations are challenging but should be performed to have a more realistic evaluation of the protocols. Developments in cross-layer designs is another research area in order to achieve reliability, where interaction with the MAC layer would help in solving the channel contention problems while network layer would help to find the best route or the alternative route in case of congestion. In addition to reliability, congestion control also plays an essential role in minimizing the packet loss which in turn strengthens reliability. All WSNs reliability protocols consider applications are working in hostile environments; however, all of them have ignored considering security risks due to malicious attacks. Furthermore, the existing reliability protocols lack the use of in-node data processing [16]; instead, they rely on the sink to regulate the flow of data to achieve reliability. This increases the communication overhead which is the highest power consumer in a wireless network; it is worthwhile to note that sensors receiving packets from the sink can consume as much energy as transmitting packets themselves [17]. We believe that introducing a hybrid mechanism with an intelligent combination of retransmissions and redundancy techniques tends to be the center of future research, as discussed in previous sections.

Event identification is one of the major challenges for event reliability protocols in WSNs. Among the existing event reliability protocols, only ERP [71] and EERP [78] raised this challenge of performing event identification before reliably transmitting it towards the sink.

To achieve event reliability in a WSN, it is essential to perform event identification. In this context, event identification focuses on identifying if two or more packets contain information about the same event or not. Otherwise, it is impossible to suppress the unnecessary transmission of duplicate event data in a non-sink centric way, while making sure that only enough information about the event reaches the sink not all the packets from all the nodes. Among the existing event reliability protocols, only ERP and EERP raised this challenge of performing event identification before reliably transmitting it towards the sink.

There are a number of schemes that perform event detection or outlier detection, but these concepts are different from event identification. Event detection is the process of defining the event of interest based on some threshold values or it is the process of identifying if an event of interest has occurred or not. For example, a fire has no meaning to a sensor node; therefore, it is required to have a criterion (e.g., based on a certain threshold) to describe events of interest in a way that sensor nodes can understand them. In contrast, event identification is the process that enables the sensor nodes to identify whether two or more packets are carrying information about the same event or not, such as a fire event detected by multiple sensor nodes in a vicinity.

On the other hand, outlier detection is a learning process based on historical values, where these historical values are compared if they lie within a certain range or not. If they are outside a certain range or sort of abnormal values, then they are considered to be outliers. But it all depends on the underlying algorithms proposed by the research community. Several researchers calls them events as well. So, both event and outliers are similar kind of concepts but different from event identification.

This chapter is based on the already published work [14]. This section has been updated to include newer work. Shahid et al. [101, 102] propose the mechanisms based on the identification of outlier type as events or errors can also be made on the basis of the observation that erroneous measurements are likely to be spatially unrelated, while event measurements are likely to be spatially correlated. Shami et al. [103] present the classification between malicious events and important emergency events by comparing between the transmitted aggregated values with normal range and having neighbour's opinion to decide whether the event is malicious or correct event. K. Kapitanova at el. [104] use fuzzy values to perform event description and improve event detection accuracy. T. K. Xuan at el. [105] also utilizes fuzzy logic to evaluate the credibility of clusters where the decision about event presence or absence is made at the cluster heads (based on signal strengths) and transmitted to the fusion centre to decide (fuzzy decision) the credibility of a cluster (based on detection probability and false alarm). K. Ali at el. [106] performs outlier detection by computing the eigen vectors and the corresponding eigen values of the covariance matrix of the outliers in the detected event. A number of other researchers proposes event/outlier detection mechanisms utilising the existing algorithms to enhance the detection process [107–111]. However, to the best of our knowledge, none of the schemes has proposed an energy efficient event reliability by performing event identification in a distributed fashion.

3.6 Summary

This chapter focused on recent research work related to wireless sensor networks. It examines how others have addressed the issue of energy efficient reliable in WSNs. We presented an introduction of reliability in WSNs, the challenges involved in achieving reliability in WSNs and the terminology that will be used throughout this thesis. Following this, an overview of retransmission technique and the existing protocols based on retransmission reliability in WSNs. We addressed some of the existing protocols that aim to achieve reliability through the use of information redundancy. We further compared and analyzed the existing schemes using different criterion for comparison by presenting a qualitative analysis. Finally, we highlighted some research areas to be addressed in the future and conclude the chapter in this section.

Chapter 4

System Design

This chapter presents the design of the basic event reliability protocol (ERP) and the enhanced event reliability protocol (EERP) based on the arguments and requirements presented in the previous chapters. The design takes into consideration challenges due to the resource-constrained nature of a WSN (Section 2.3) and techniques to mitigate those constraints (Section 2.4). The in-depth study and detailed analysis of existing literature in Chapter 3 identified the shortcomings in existing work and their possible solutions. This analysis gives a solid base for architecting the design of ERP and EERP. This Chapter contributes towards building the basic architecture and design of the mechanisms proposed in this thesis by using step-by-step process flows that in turn provide the grounds for the implementation of the second and third contribution of this thesis (Section 1.3).

Section 4.1 provides an overview of the objectives for the design of the proposed mechanism. Section 4.2 provides an overall understanding of the conceptual architecture. Section 4.3 gives the design and process flow of ERP and EERP. Section 4.4 presents a scenario-based example to better understand the functionality of the proposed mechanism. Section 4.5 presents the limitations of the system design and Section 4.6 concludes the chapter with a summary.

Parts of Sections 4.2, 4.3 and 4.4 have been published in [71], [78] and [93] respectively.

4.1 Design Requirements

The objective of this thesis, as introduced in Chapter 1, is to develop an energy efficient and reliable event delivery mechanism that efficiently performs distributed event identification for the reliable delivery of event information from nodes in the network to the sink. The mechanism should exploit the in-node processing capabilities of sensor nodes by performing decision making at the node level instead of relying on the sink's centralized control and decision making. Furthermore, the mechanism should only require the sink to receive sufficient packets carrying information about a particular event instead of all the packets from all the sensor nodes that sensed the event.

This thesis introduces a novel solution that builds on the spatio-temporal information of the originating nodes and employs an efficient combination of a greedy strategy with the use of a multilateration algorithm to perform event identification before reliably transmitting the event information towards the sink node. This thesis applies multilateration in a novel way; it is the first to use multilateration for event identification instead of traditional node localization in a WSN.

Since communication is the highest consumer of energy in a resource-hungry WSN, communicating all the packets to and from the sink is always more expensive than performing in-node processing that transmits only the necessary packets to the sink. Therefore, this architecture employs a non-sink-centric approach to provide hop-by-hop event reliability while saving the unnecessary transmission of additional packets with redundant event information with the help of distributed event identification.

The functional requirements for distributed event identification and reliable transmission of event information were discussed in Section 2.5. To achieve the goal of this thesis, the first step is to look at the operations performed by the nodes at different levels and categorize the nodes accordingly. Each node performs a process based on the category of the node it belongs. Keeping in mind the aim of this thesis, the functional requirements, and the operations performed within each nodes' category, the design requirements are:

- A distributed mechanism to enable the nodes to identify if two or more packets are carrying information about the same event or not. This requires a region-based selective transmission mechanism to minimize the transmission of packets with redundant event information while providing complete coverage of the events throughout the network.

- A method to maintain reliable transmission of event data by exploiting the broadcast nature of wireless channel while ensuring that as few packets as possible should be transmitted while still providing sufficient information about the events happening across the network.

To meet these requirements the design uses a hop-by-hop event reliability mechanism that provides distributed event identification based on spatio-temporal information and implicit acknowledgments.

4.2 Overview of the Conceptual Design

Keeping in view the issues and challenges discussed in the previous chapters; this section provides an overview of the conceptual design of an efficient hop-by-hop event reliability protocol. The initial design, Event Reliability Protocol (ERP), combines the benefits of in-network processing capabilities of a WSN with implicit acknowledgments and a region-based selective retransmission strategy to provide a distributed event reliability in an energy efficient manner. The second design, the Enhanced Event Reliability Protocol (EERP), builds on ERP by introducing an efficient combination of multilateration mechanism with a greedy approach to improves the overall efficiency in terms of accurate event identification for more than two packets and hop-by-hop event reliability. The use of region-based selective retransmission and implicit acknowledgments is the second contribution. The EERP which uses multilateration to perform event identification is the third contribution of this thesis.

In WSNs, a large number of densely deployed sensor nodes continuously sense events and send the data packets containing event information towards the sink. Typically, these data packets from the sensor nodes converge towards the sink which often results in congestion causing packet loss. To achieve event reliability such that only the necessary packets carrying information about a particular event reach the sink while minimizing congestion becomes a challenging task. Another challenge is that packets are also lost due to transmission errors and signal interference when multiple nodes within range of one another transmit simultaneously. Maintaining reliability requires the retransmission of lost packets, but retransmitting every lost packet can aggravate the problem of lost packets.

The event data from closely located sensor nodes in a network tend to contain information about the same event. This allows the ERP to selectively tolerate some packet loss if there is another packet with information about the same event rather than aiming to achieve full hop-by-hop packet reliability. This means the ERP can avoid some unnecessary retransmissions that aggravate the network congestion and hence negatively affect the reliable transmission of event information.

ERP design builds on the spatial locality condition and introduces a region-based selective retransmission mechanism with implicit acknowledgments to achieve hop-by-hop event reliability in WSNs. This mechanism will retransmit the lost packets only when no other packet from the same region of the lost packet is present in the node's queue. The ERP mechanism is a distributed approach which uses in-node processing to perform simple node-level decisions instead of relying on the sink for centralized decision making. The ERP prevents retransmissions of unnecessarily lost packets, but it transmits every packet, even unnecessary ones, the first time. It also limits its event identification to two packets at a time based on their spatio-temporal correlation.

The EERP design extends the ERP to perform event identification process on more than two packets at the same time. This allows EERP to further improve the efficiency of ERP in terms of accurate event identification, reliability, energy, event coverage and traffic overhead. EERP also employs an efficient combination of a greedy strategy with the use of a multilateration algorithm in a novel way to perform event identification before reliably transmitting event data towards the sink node which avoids any transmissions of unnecessary packets with duplicate event.

The ERP and EERP divide nodes into three categories based on their capabilities and the type of operation performed: (i) originator nodes, (ii) intermediate nodes and (iii) the sink nodes. Each category of the node is briefly described below.

4.2.1 Originator Nodes

Any node in the network that senses the events happening within its sensing range is described as an originator node. An originator node is denoted by N_O, and its sensing range is denoted by R. After sensing an event, an originator node creates a packet, adds to the packet header its own location information along with the information about the sensed event required by the underlying scheme, and sends the packet to the next hop node towards the destination (i.e., the sink node).

4.2.2 Intermediate Nodes

All the nodes that relay a packet between an originator node and the sink node are considered intermediate nodes and are denoted by N_I throughout the thesis. After receiving a packet from a previous node, an intermediate node performs some processing tasks and may forward the data packet towards the sink node. Note that a particular node may act as an originator node of some packets, and as an intermediate node for other packets.

4.2.3 The Sink Node

The sink node acts as a gateway between the sensor network and the end user. The sink node receives all the information about the events sensed by the originator nodes that is being processed and forwarded by a set of intermediate nodes. The sink node is denoted by N_S throughout this thesis.

4.3 Design & Process Flow

This section presents the design of ERP and EERP by discussing in detail the process flows involved at different stages of the proposed schemes. These process flows are based on the nodes' categorization as discussed above.

4.3.1 Network Model

The network model is a typical homogeneous WSN having a set of sensor nodes and a sink node. An event-driven model is assumed: upon sensing an event, an originator node N_O generates a packet containing event data and transmits it to the sink through intermediate nodes. It is assumed that the nodes are randomly deployed and that the routes are computed for the entire network by a WSN routing algorithm. Routing is not the focus of this thesis; therefore, we will not go into the routing details as the proposed mechanism is compatible with any underlying routing protocol. It is assumed that the sensor nodes and sink are static and placed according to pre-configured location information. The distances between the nodes and the sensing range of the nodes also remain static. It is also assumed that the nodes know their own locations, but do not have global knowledge such as the number of nodes in the network and the size of the network. The model uses a flat network topology, where all the sensor nodes are equal in terms of their sensing and communication range.

The sensor nodes are placed in such a way that they cover the whole network area in terms of sensing events. An event can be detected by multiple sensor due to the overlap of the sensory region which is necessary to provide full sensing coverage. Therefore, an event is likely to sensed by multiple nodes and multiple packets will be created containing information about the same event.

4.3.2 ERP Design

This section presents the conceptual design of basic ERP by explaining the processes involved in different node categories. ERP design provides a solid foundation that leads towards achieving the overall aim of this thesis.

4.3.2.1 Process Flow at Originator Node

When an event occurs in the network, each sensor node in the vicinity of the event that successfully detects the event creates a packet, puts the event information in the packet header, temporarily places the packet in its queue for further processing and transmission as shown in Figure 4.1. The nodes send each packet in their queue to the next hop node that lies towards the sink as defined by the default routing protocol. After sending a packet, a node starts its timer and listens to the wireless channel to see if the next hop node has further forwarded its sent packet or not. When the node overhears the next hop node transmitting the packet that it has recently sent, it treats this as an implicit acknowledgment (iACK) that the next hop node has reliably received and forwarded the packet. The node then removes the packet from its queue to avoid unnecessary stale packets occupying the queue and processes the next packet in the queue.

In the case the node does not hear any iACKs, it assumes that its sent

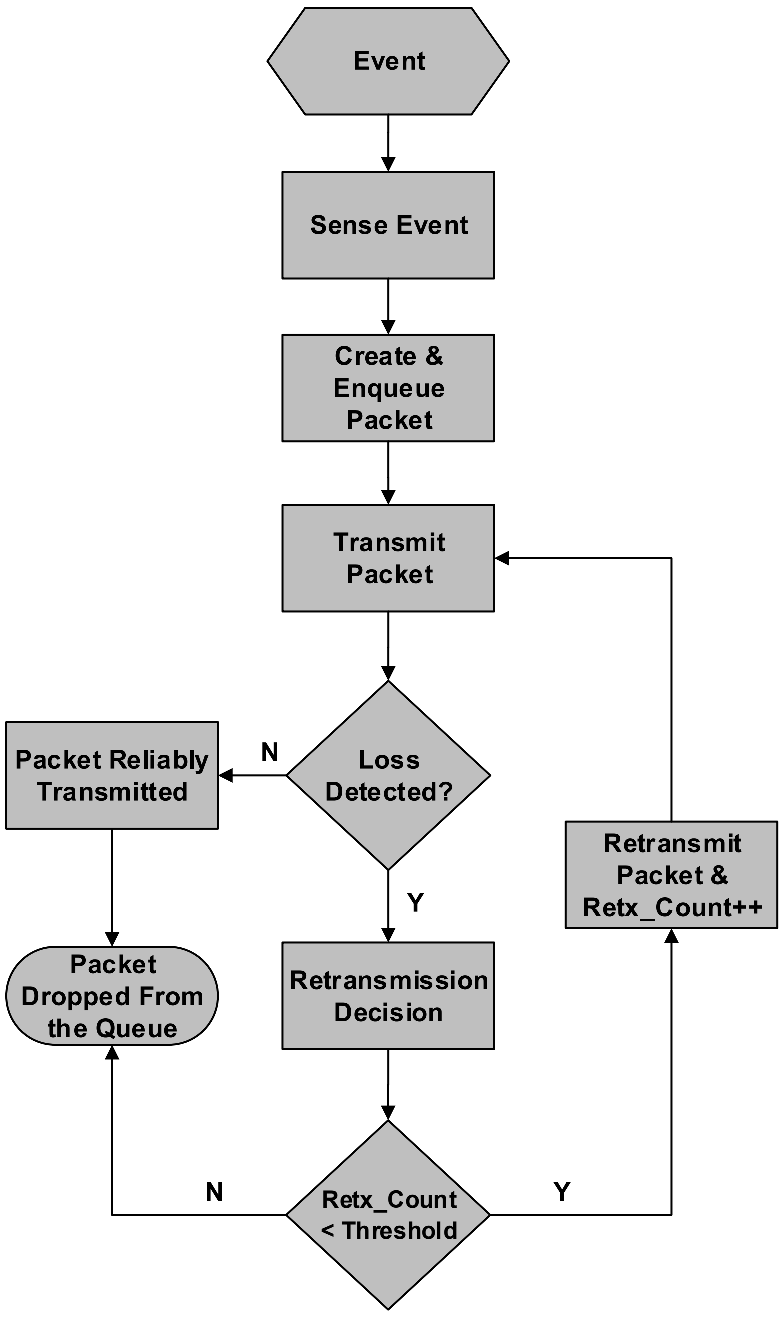


Figure 4.1: High Level Flow Chart of the Originator Node

packet has been lost and needs to be retransmitted to maintain reliability. It would be possible to keep retransmitting the lost packet until it is acknowledged, but this can overload the network in some situations. Instead, the proposed mechanism limits the number of retransmissions to a threshold value (currently two) in order to reduce the overall network transmission overhead and avoid the network becoming congested. Once the packet's retransmission counter exceeds the threshold, the node gives up trying to retransmit the packet; the packet is then dropped from the queue, and the next packet in the queue is processed. The node gives up on retransmitting the packet because it assumes that the next hop node is too busy sending other packets that are probably related to the same event. Underlying this mechanism is an assumption that the network is reasonably reliable so that a packet will be received at least once in three attempts, and therefore going over the retransmission threshold is always due to an overload of traffic.

4.3.2.2 Process Flow at Intermediate Node

This section describes the process flow at the intermediate node as shown in Figure 4.2. When an intermediate node receives a packet, it will temporarily place the packet in its queue for further processing and transmission. When a node has packets in its queue, the packet at the head of the queue is transmitted to the next hop as defined by the routing protocol but is not yet removed from the queue. To maintain reliability, the intermediate node after transmitting the packet, waits for the iACK of its sent packet. If the node overhears the next hop node transmitting the packet that it has sent, it assumes that the packet is forwarded successfully, so it removes the packet from the queue and processes the next packet in the queue.

In the case the node does not detect an iACK for the packet, the process enters the next phase of ERP, that is, the region-based selective retransmission mechanism. This requires the intermediate node to take a retransmission decision based on the spatio-temporal information contained in the header of the packet and the headers of all the other packets in the queue.

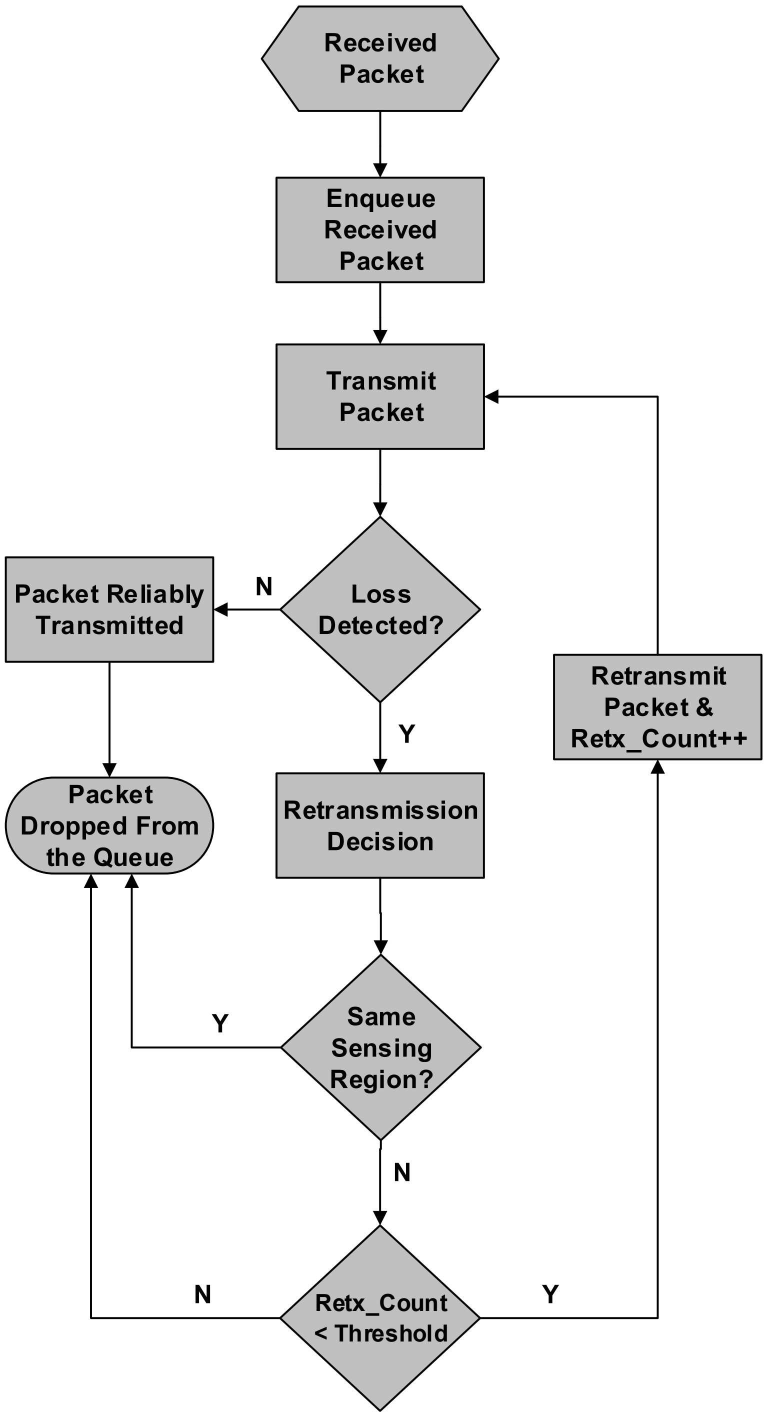


Figure 4.2: High Level Flow Chart of the Intermediate Node in ERP

For each other packet on the queue, it calculates the spatial correlation and the temporal correlation between the head packet and the other packets. The spatial correlation is whether two events are spatially correlated or not is determined by calculating the Euclidean distance between the originator node's location of the head packet and the originator node's location of another packet; the temporal correlation is calculated by matching the event generation time of the head packet with the event generation time of the other packets. If there exists another packet in the queue, whose spatial correlation with the head node is less than twice the sensing range, and whose temporal correlation with the head node is close, then the node assumes, based on temporal and spatial locality, that both packets contain information about the same event. Hence, it is now possible to suppress its retransmission and drop the head packet from the queue with minimal loss of event information, as another packet containing the same event is present in the queue.

However, if there is no such highly correlated packet found in the node's queue, then it retransmits the packet and repeats the process until the retransmission threshold is reached; at which point it drops the packet from the queue to prevent stale packets from occupying the queue. Meanwhile, if another packet from the same event as the head packet arrives and is added to the current node's queue, then the lost packet will be dropped the next time a retransmission decision is needed.

4.3.2.3 Process Flow at the Sink Node

The sink node as the final destination collects all the essential information about the events happening all over the network. After receiving a packet, it sends an explicit acknowledgment (eACK) to the previous hop node as a confirmation that the packet is reliably received (see Figure 4.3). It also acts as a bridge between the sensor network and the end user, where the end user collects all the monitoring data required as specified by the application.

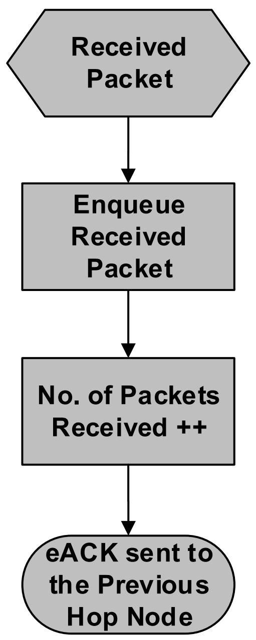

Received Packet
Enqueue Received Packet
No. of Packets Received ++
eACK sent to the Previous Hop Node

Figure 4.3: High Level Flow Chart of the Sink Node

4.3.3 EERP Design

This section provides design details of EERP, which is built on top of the ERP design by further enhancing the overall performance in terms of improving event identification, event information delivery and energy efficiency while maintaining good coverage of events throughout the network. Like ERP, EERP uses a hop-by-hop reliability mechanism which takes advantage of the broadcast nature of wireless channels by using implicit acknowledgments (iACK) to maintain reliability. However, EERP introduces an improved event identification mechanism and a more effective event reliability mechanism. Like ERP, event identification exploits the fact that sensor data tends to exhibit a high-level of spatial locality but EERP introduces a greedy approach with the use of a multilateration algorithm in a novel way to efficiently determine whether multiple packets contain the same event or not. It also makes a more sophisticated decision about when to forward or retransmit than ERP's decisions. First, it makes a decision about a packet before sending it the first time, not just at retransmission. Secondly, it keeps and uses a history of recently sent packets as well as a queue of packets to be sent and a history of past decisions to not to send a packet. In addition to all these features, EERP also compares multiple packets at once (using multilateration) to determine whether they are all carrying the same event, not just pairs of packets.

Multilateration is a commonly used localization mechanism to estimate the location of unknown node; however, it has never been used to estimate an event's location in WSNs. This thesis uses a multilateration algorithm to estimate if multiple nodes could have detected the same event or not and to perform distributed event identification to minimize the transmission of redundant packets carrying information about the same event.

The standard multilateration algorithm details given in Section 2.4.2 use the estimated distances from the known anchor nodes to the unknown node. In the context of this thesis, the distance to an event from an originator node transmitting the information about the event must be less than the sensing range of the node. EERP makes an assumption that the event is at maximum sensing range for the purpose of the multilateration algorithm. Given several candidate nodes, the multilateration process estimates the position of an event if it were detected by all the nodes. The EERP then checks this estimated point against every candidate node to see if it lies within the sensing range of all the nodes or not. The rest of this section discusses the processes involved at each category of the node to achieve enhanced event reliability in WSNs.

4.3.3.1 Process Flow at Originator Node

In EERP, the process at the originator node is the same as for ERP, as described in Section 4.3.2.1. Packets are sent and retransmitted if no iACKs are detected up to a retransmission threshold.

4.3.3.2 Process Flow at Intermediate Node

Unlike ERP, EERP uses a different data structures in intermediate nodes as shown in Figure 4.4 that makes EERP a more aggressive and efficient approach. A node in EERP has a queue, a sent list and a collection of drop lists. The queue is where the incoming packets reside temporarily. The sent list keeps a history of packets that the node has reliably sent. Each drop list is associated with a particular packet in the sent list and contains a history of packets that were dropped and not sent further because they contain the same event as that of the associated packet in the sent list.

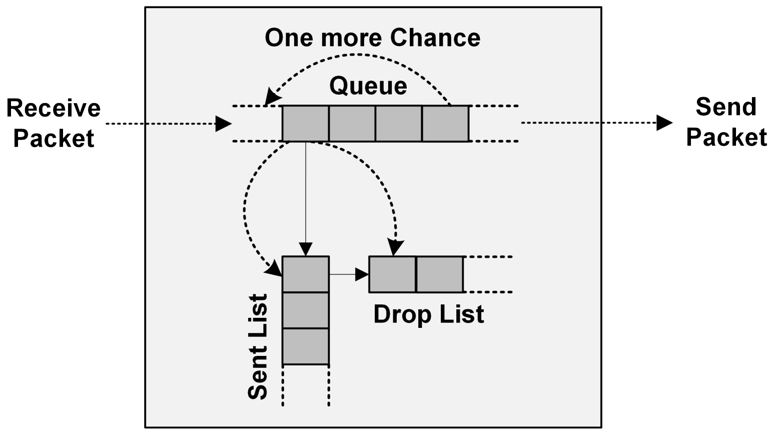


Figure 4.4: Data Structure of a Node in EERP

In EERP, when an intermediate node receives a packet, it temporarily places the packet in the queue for further processing and transmission. The node grabs the packet at the head of its queue and decides whether or not to further forward it to the next hop node that lies towards the sink, as depicted in Figure 4.5. This decision is taken based on the history of already sent packets having the same event. The node checks if there exists any already sent packet in the sent list whose event generation time is approximately same as the head packet and whose originator node lies within the vicinity of the head packet's originator node. If it finds such a packet in its sent list and there are no packets in the associated drop list, then the head packet is dropped from the queue and placed in the drop list by suppressing its transmission.

However, if there is at least one packet in the drop list, then it collects the originator node of the head packet, the originator node of the sent packet with the same event, and all the originator nodes of the packets in the drop list. It checks if all these nodes have sensed the event at the same time, then performs multilateration on all these nodes. The multilateration process is successful if the event location estimated by multilateration lies within the sensing range of all the nodes involved in the process. This confirms that all these packets are originating from the same region and could have detected the same event. If multilateration is successful, the packet is dropped from the head of the queue and added to the associated drop list, an explicit acknowledgment (eACK) is sent to the previous hop node and the next packet in the queue is processed. The eACK is sent because the previous hop not might be listening for the transmission of its sent packet. However, if the multilateration process concludes that the originator nodes of the packets could not have detected the same event then the node will transmit the head packet to the next hop node lying towards the sink. This is greedy because it does not go back to check other packets on the sent list. Although this might save sending some redundant packets, the cost of checking all sent packets is considered to be too high. Chapter 6 reports on the experiments that show that the greedy approach performs better.

After transmitting the packet, the node monitors the wireless channel for an iACK to see if the next hop node has reliably received and forwarded the packet or not. Upon detecting the iACK, the node adds the packet to the sent list and drops it from the queue to avoid unnecessary packets occupying the queue.

In the case of no iACKs detected, the node assumes that the packet might be lost and decides whether to retransmit this packet or not. The retransmission decision process starts by checking if the retransmission

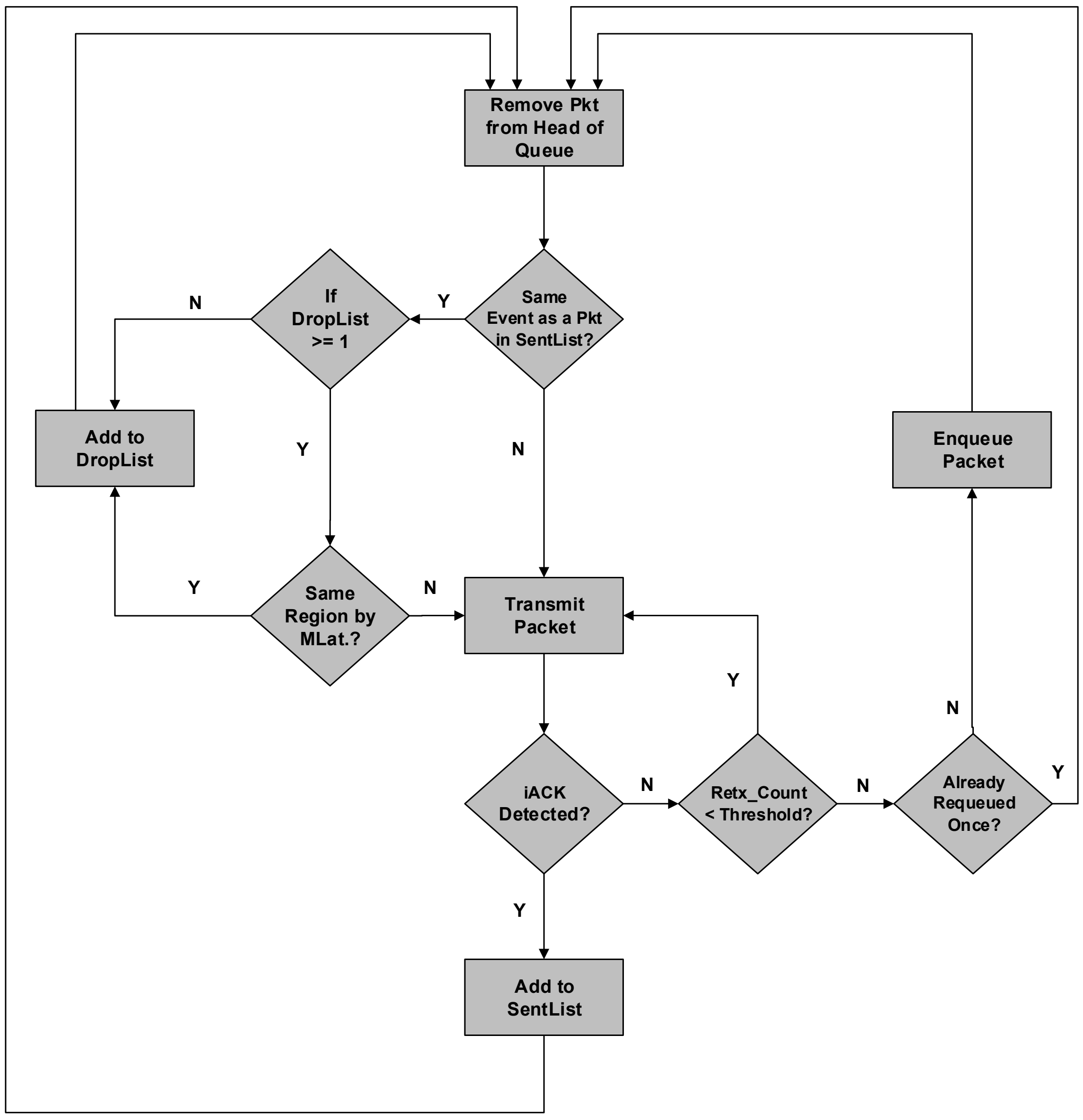


Figure 4.5: High Level Flow Chart of the Intermediate Node in EERP

counter of the packet has exceeded the retransmission threshold. If not, it retransmits the packet and starts a timer to wait for iACK. If the iACK is received, the packet is considered to be successfully forwarded to the next hop node and thus added to the sent list. However, if the retransmission counter has reached its limit, then the node checks if this packet has previously been moved to the back of the queue or not. If the packet has not yet been re-queued then it is moved to the back of the queue to attempt retransmission later. This will give the packet another chance to be forwarded towards the sink when it again reaches the head of the queue. However, if the packet has previously been re-queued then the node gives up on retransmitting the packet and abandons it based on the assumption that the next hop node has already sent another packet related to the same event. This enables information about successfully transmitted events to propagate backward in the network and reduce unnecessary transmissions.

4.3.3.3 Process Flow at the Sink Node

The operations performed at the sink node are the same in both ERP and EERP, as described in Section 4.3.1.3 and illustrated in Figure 4.3.

4.4 Scenario-based Conceptual Model

In order to illustrate the overall distributed event identification and hop-by-hop event reliability components, we work through a simple example scenario with a small-sized sensor network of eight nodes (three originators, four intermediate and a sink node). The overall process is divided into four main phases: (A) Network Setup, (B) Event Sensing, (C) Processing at Intermediate Nodes and (D) Recieving Event at the Sink Node. Each phase is described in detail below with the help of corresponding figures.

(A) Network Setup:

In this example, we consider an event-driven WSN with a flat network topology, where all the nodes are equal in terms of their sensing range (i.e. R) as shown in Figure 4.6. It is assumed that all the sensor nodes are placed within a finite area with a small overlap in sensing regions. Each node has a static position with pre-configured location information. The routes are computed for the entire network by an underlying WSN routing algorithm and the nodes do not have global knowledge about the number of nodes and the network area.

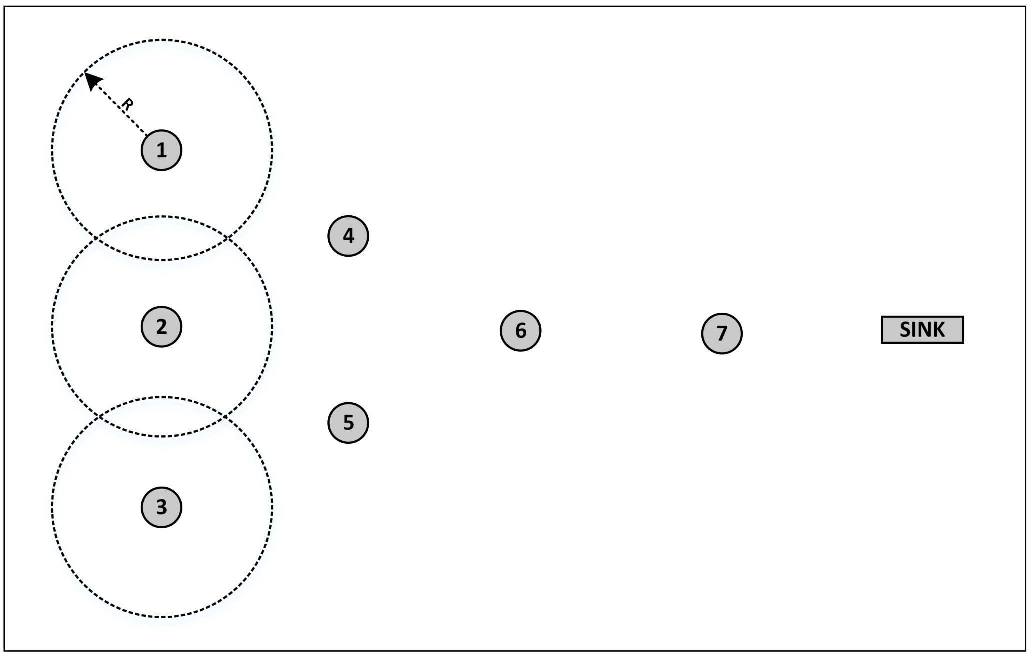


Figure 4.6: Network Setup

(B) Event Sensing:

When an event occurs in the network, each originator node that detected the event sends the event's information towards the sink. In this case, two events (i.e., E_A and E_B) occurred one after the other at approximately the same time, as shown in Figure 4.7. E_A is observed by nodes 1 and 2 leading to packets P_1 and P_2. E_B is observed by nodes 2 and 3 leading to packet P_3 and P_4. Let's assume the nodes transmit the packets in the order P_1, P_4, P_2 and P_3. Nodes 1 and 2 send packets P_1 and P_2 to the intermediate node 4; nodes 2 and 3 send packets P_3 and P_4 to the intermediate node 5 lying towards the sink as shown in Figure 4.7.

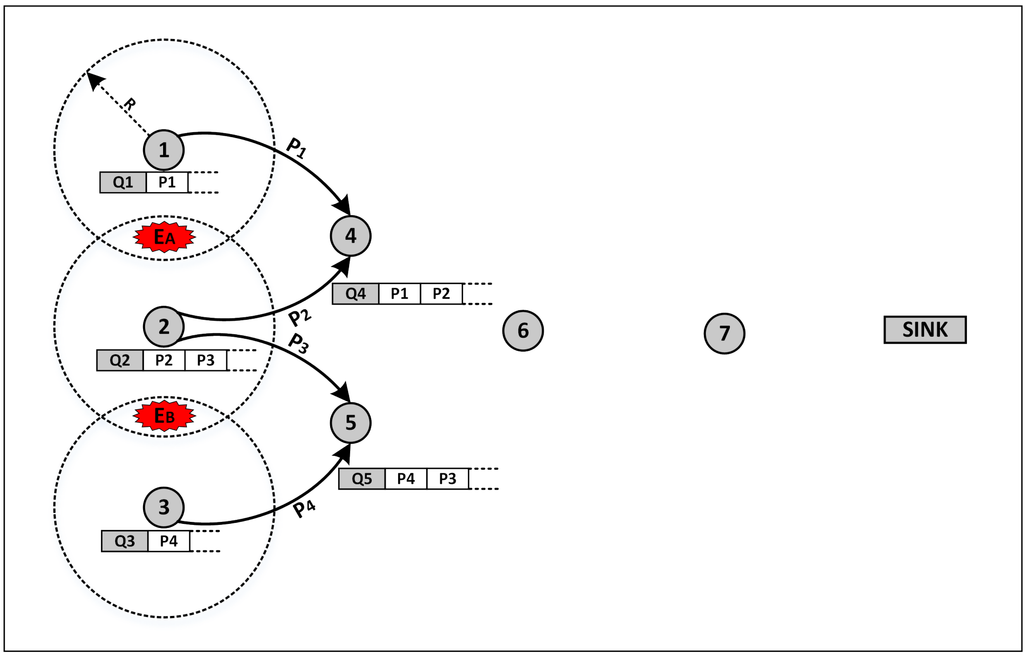


Figure 4.7: Event Sensing

(C) Processing at Intermediate Nodes:

The intermediate nodes (Nodes 4, 5, 6 and 7) before further forwarding the packets, apply the process mentioned earlier in Section 4.3.3.2 and shown in Figures 4.8 and 4.9. Node 4 receives packets P_1 and P_2, and node 5 receives packets P_3 and P_4 respectively. Node 4 takes the head packet P_1, checks the sent list to see if there is any other packet with the same event and it finds nothing, as P_1 is the first packet. So, node 4 simply forwards P_1, listens to the channel and receives the iACK for P_1. Upon receiving the iACK, node 4 drops P_1 from its queue and adds it to its sent list. Soon after, it receives another packet P_2 and before forwarding P_2 it finds packet P_1 in its sent list that is likely to have the same event as P_2. Therefore, node 4 suppresses the transmission of P_2 by dropping it from the queue and adding it to the drop list associated with P_1. Node 4 also sends an eACK for P_2 back to node 2, as it is not further transmitted and node 2 might be listening to the channel for an iACK of P_2. Similarly, node 5 forwards P_4, while suppressing the transmission of P_3. Node 6 and 7 forward both P_1 and P_4 towards the sink because they could not be describing the same event, as shown in Figure 4.9.

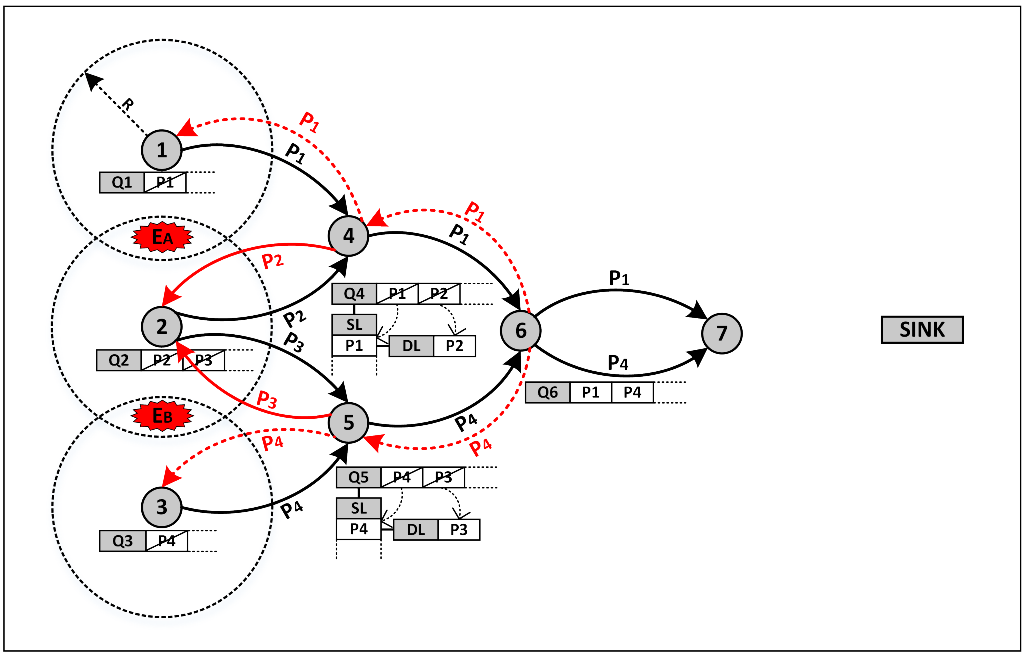


Figure 4.8: Processing at Intermediate Nodes

(D) Receiving Event at the Sink Node:

When packets P_1 and P_4 reach the sink, the sink node sends an explicit acknowledgment to the previous hop node to maintain reliability because there are no iACKS, as shown in Figure 4.9.

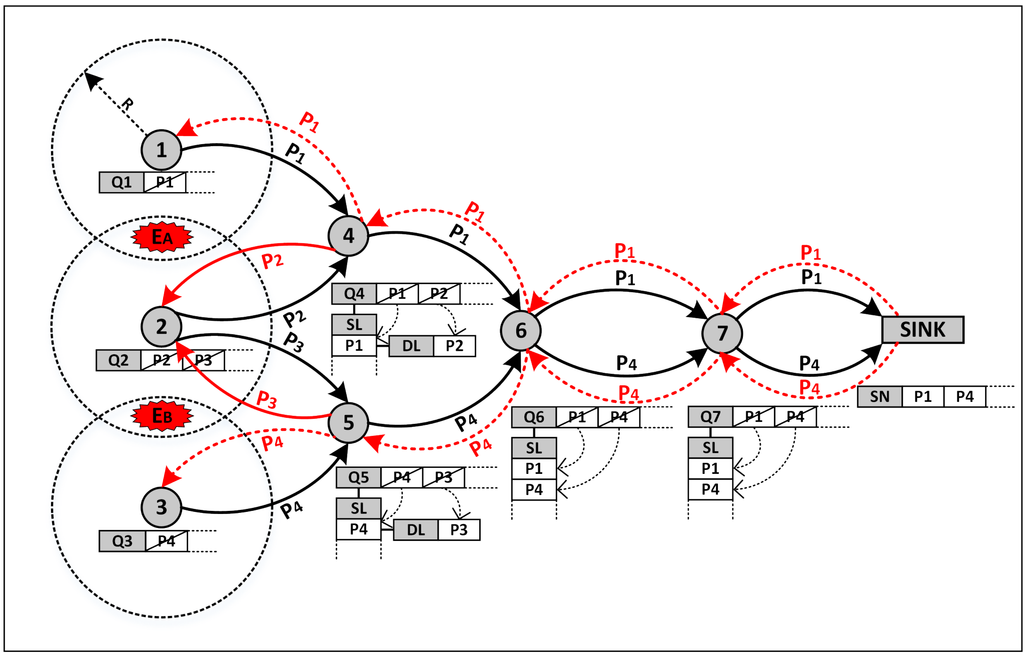


Figure 4.9: Receiving Event at the Sink Node

Figure 4.10 presents a sequence diagram that illustrates the same example scenario in terms of all the processes performed in parallel with a timeline. This is just one sample scenario to illustrate the whole process. In this scenario, all the events reached the sink with no duplicate packet or event information. However, the number of events uniquely received at the sink node depends on the order in which packets are sensed and sent towards the sink. If the packets had been sent in a different order, different packets might have reached the sink. For example, if the packets had been sent in the order P_2, P_3, P_4 and P_1, then packets P_2 and P_3 would have reached the sink, where both P_2 and P_3 might have sensed the same event. EERP can be aggressive in some circumstances with a high density of events. Chapter 6 describes the experiments where the proposed schemes will be thoroughly tested and evaluated with a variety of realistic WSN scenarios comprising of a large number of nodes.

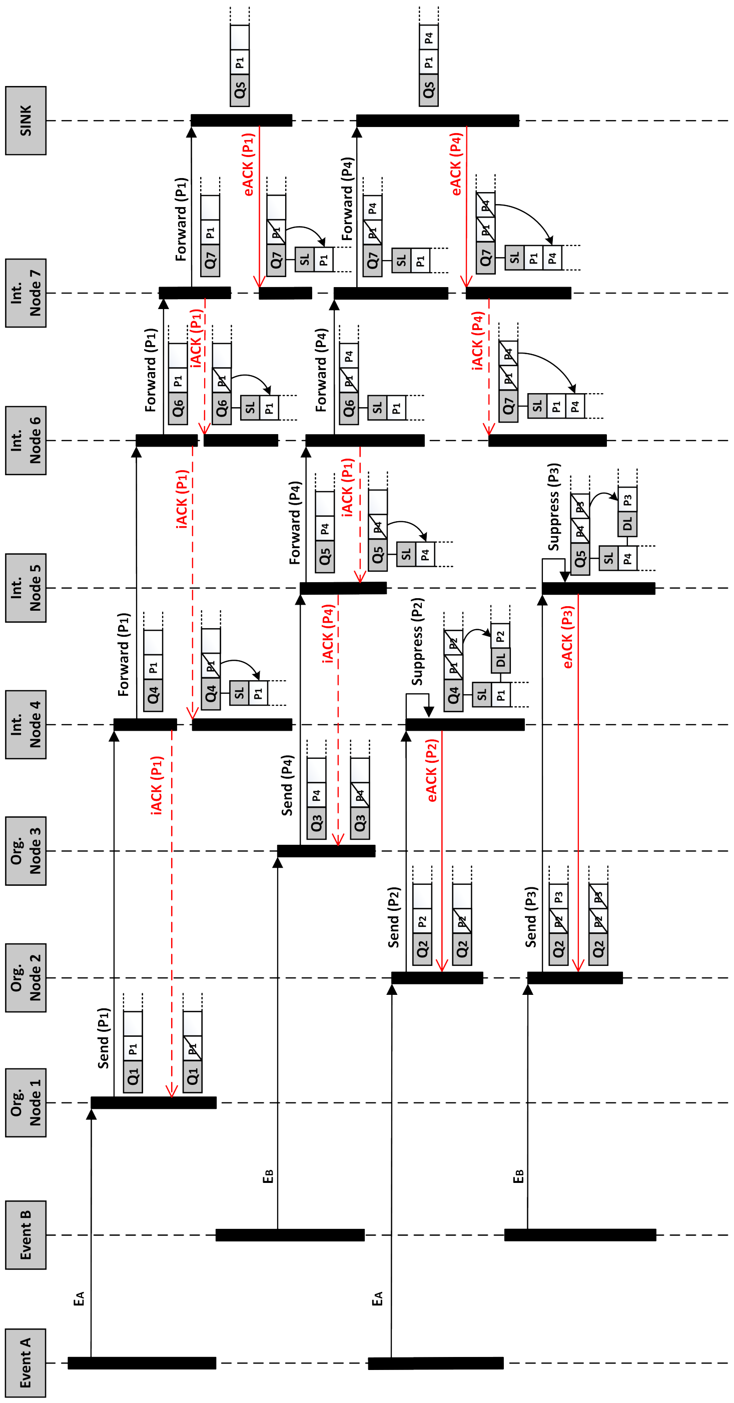


Figure 4.10: EERP Sequence Diagram

4.5 Limitations of the System Design

The functional requirements for this thesis as mentioned in Section 2.5 are distributed event identification and hop-by-hop event reliability for an energy efficient and reliable wireless sensor network. For distributed event identification, the focus is on providing a non-sink centric approach where the processing of data must be done at the sensor node level instead of relying on the sink to perform all the decision making that can be done more efficiently at the node level. The basic and enhanced ERP both provide distributed event-to-sink reliability with improved scalability of event detection in a WSN by minimizing the unnecessary retransmission/transmission of data packets coming from multiple nodes in an event's locality. ERP takes advantage of the broadcast nature of wireless channel by using an iACK mechanism with region-based selective retransmission of packets. The region-based selective retransmission mechanism introduced in ERP can only perform event identification on two packets. EERP overcomes this limitation by introducing multilateration process allowing it to perform event identification process on more than two packets at the same time.

ERP performs the event identification process only when the packet is lost and needs to be retransmitted. However, EERP performs event identification before transmitting the packets that not only reduces the overall transmissions and retransmissions of packets reducing the overall traffic. This thesis uses spatio-temporal techniques to perform event identification; therefore, it is assumed that all the nodes in the network have prior knowledge of their own location. This location information is disseminated to all the nodes during the network setup phase. Also, the design developed in this thesis addresses and focuses on a wireless sensor network where the nodes are static. This design is not appropriate for WSN where the nodes are mobile, until and unless the location information of the nodes get continuously updated which is not an efficient approach for the resource hungry WSN as it involves additional communication which is the highest consumer of energy. Therefore this thesis only focuses on the WSNs with the static nodes, as this is the most common type of WSNs while disseminating the local location information to the nodes only once during the network setup phase is feasible and common to do in a WSN.

4.6 Summary

This chapter contributes by providing a solution that how a wireless sensor network can provide hop-by-hop event reliability based on a novel event identification mechanism in an energy efficient manner. This chapter builds on the recommendations discussed in Section 2.5, keeping in view the design requirements mentioned in Section 4.1. The design fulfills the recommendations by providing support for a hop-by-hop event reliability mechanism while providing distributed event identification initially by introducing a region-based selective retransmission mechanism in ERP while enhancing it by introducing a novel combination of a greedy-based approach and multilateration mechanism in EERP. The design for both ERP and EERP caters for the resource constrained WSNs by using a non-sink centric approach and efficiently making decisions based on spatio-temporal information while minimizing the communication overhead and saving limited energy of a sensor node.

This chapter explains how the design efficiently achieved the aim of this thesis by giving sensor nodes specific roles and performing operations accordingly. For a better understanding of the system design and operation, this chapter also presented a scenario-based conceptual model with a simple example scenario and a sequence diagram explaining the details of the proposed mechanisms.

In the proposed design, by processing event data locally, the sensor nodes can make the event identification and retransmissions decisions efficiently without the need of any flow control from the sink node to maintain reliability. This makes the proposed design appropriate for most of the monitoring applications operating remotely in environments with minimal human intervention while requiring reliable transmission of event information from the source node to the sink. The following chapter presents the implementation details all the operations performed on the different level of the node categories as presented in the conceptual design to achieve the aims and objectives of this thesis.

Chapter 5

Prototypical Implementation of ERP

The previous chapter presented the design and scenario-based conceptual model for the Event Reliability Protocol (ERP) and Enhance Event Reliability Protocol (EERP), event-to-sink reliability protocols that use a novel distributed event identification mechanisms to reliably deliver event information from all over the network to the sink with minimum communication cost. This chapter presents a prototypical implementation of basic and enhanced ERP that exploits the in-node processing capabilities of a sensor node to reduce the communication cost. This is done by implementing different roles associated with each node category. The implementation details presented in this chapter addresses how to perform hop-by-hop event reliability in combination with an efficient event identification mechanism in a resource-constrained wireless sensor network.

This chapter is structured as follows: Section 5.1 presents the focus of the implementation in terms of the nodes categorization presented in the previous chapter. Section 5.2 briefly describes the implementation environment. Section 5.3 explains the implementation details with the help of the pseudo codes for the functions and methodologies used throughout this thesis. Section 5.4 discusses the limitations of the current implementation and Section 5.5 concludes the chapter with a summary.

Parts of Section 5.2 and 5.3 are previously published in [78] and [93].

5.1 Focus of Implementation

This section explains how the implementation of ERP and EERP accomplishes the design requirements presented in the preceding chapter. This chapter focuses on the implementation details of both ERP and EERP. ERP provides the foundations of this work that leads to the development of enhanced ERP, thus EERP includes most of the basic operations involved in ERP. Therefore, the implementation details for EERP primarily focuses on the additional methodologies introduced in EERP to enhance the efficiency of ERP. Following the design presented in the previous chapter, the implementation of basic and enhanced ERP is also based on having the nodes play different roles to achieve the overall aim of this thesis. These roles are (i) originator nodes, (ii) intermediate nodes, and (iii) the sink node. Any node in the network, apart from the sink node, can adopt the role of an originator node or an intermediate node at a particular time and act according to the procedures designed in the previous chapter and implementation details discussed in this chapter.

Originator Nodes: Any node in the network that first senses an event is considered to be an originator node. The originator node adds its own location information along with the event generation time and other related information to the packet header and transmits it to the next hop node towards the sink. It waits for the acknowledgment from the next hop node because it is also responsible for maintaining reliability

Intermediate Nodes: The next hop intermediate node is responsible for performing event identification process and maintaining hop-by-hop event reliability until the event information is delivered to the sink. Both ERP and

EERP uses different techniques to perform the event identification process. ERP uses a region-based selective retransmission procedure, where EERP efficiently uses multilateration mechanism and a greedy approach to perform event identification. However, hop-by-hop reliability is maintained by both ERP and EERP using implicit acknowledgments.

Sink Nodes: Finally, the event information from all over the network reaches the sink.

Details of the implementation according to the procedures required for different roles of nodes is provided in the subsequent sections of this chapter.

5.2 Implementation Environment

The implementation of both ERP and EERP was done using QualNet 5.2 simulator [137] with all the nodes based on IEEE 802.15.4 compliant motes (i.e., sensor nodes) running as full-function devices. QualNet Simulator is a state-of-the-art simulator for large, heterogeneous networks and the distributed applications that execute on those networks [137]. Other event reliability schemes assume that an event has occurred when a node generates the packet without actually sensing an event that can also be sensed by the nearby nodes. Therefore, to implement the design of the proposed schemes and analyze them in a realistic WSN environment, an event generation application is developed in QualNet (see Appendix A). This contributes towards providing a realistic event-driven WSN environment where other protocols developed in QualNet can also use it.

The event generation application simulates an event that is sensed by the nearby nodes and makes the simulation as realistic as possible. This makes any node that detected an event to send a UDP unicast packet towards the sink. This makes ERP and EERP more realistic because other schemes ignore the importance of event generation based on the assumption that the node sending a packet must have sensed an event without simulating any event occurred within the node's sensing range. In a realistic wireless sensor network, the originator node only generates a packet and sends it towards the sink when an event has occurred within its sensing range.

Three major components that were developed in the QualNet are (i) Mote, (ii) Basestation, and (iii) Generator, for more details see Appendix A. All the sensor nodes are motes and sensor nodes can either play the role of an originator or an intermediate node, where they maintain the transmission and forwarding of packets. The basestation component is meant to act as the sink node that implements the reception of packets. The Generator component simulates events that are being sensed at regular intervals and notifies all the in-range motes that the event has occurred. The motes then send the packet with the event information towards the basestation. The generator is created as a non-functioning node, which means it never acts like a typical networking node performing different operations, instead acts as an event generator while notifying the node of events happening. Appendix A shows the steps involved in the development and implementation of this event generation application in QualNet. The following sections discuss in detail the implementation of each component.

5.3 Pseudo Code For Implementation

This section presents the implementation details of ERP and EERP by using pseudo code to perform the procedures involved at different node levels such as originator node, intermediate node, and the sink node. The originator node (identified as N_O) senses an event E happening within its sensing range R, it generates a packet P containing the event's information, and sends it towards the destination node (the sink node N_S) through a set of intermediate nodes (identified as N_I). As explained in the previous chapter, each node performs a set of operations based on its current role (i.e., originator, intermediate or the sink node) according to the procedures defined by the underlying schemes (i.e., ERP and EERP). The processes involved in ERP and EERP, their differences and how they operate in order to efficiently perform event identification and provide reliability is explained in Section 4.3.2 and 4.3.3 respectively. However, the following sections discuss the implementation details of both ERP and EERP with the help of pseudo code.

5.3.1 Implementation of ERP

The ERP operations start with the session initialization phase where the entire network configures itself upon deployment and various parameter values, such as the number of nodes, packet size, network area, individual nodes' locations, routes, etc. are set up accordingly. When an event occurs in the network, each sensor node in the vicinity of the event that successfully detected it puts the event information in a packet and sends the packet towards the sink. The next hop node towards the sink, after receiving the packet, places the packets in its queue and the packet at the head of the queue is transmitted to the next hop node. When the node hears the next hop node transmitting the packet that it has sent, it is treated as an implicit acknowledgment (iACK) that the head packet is forwarded successfully. The node removes the packet from the head of the queue and processes the next packet in the queue.

However, in case no iACK is detected, a retransmission decision is taken on the basis of spatio-temporal information contained in the packet header. In ERP the packet is retransmitted only if there exists no other packet in the node's queue that is judged as originating from the same sensing area and have the same event generation time as that of the lost packet. Otherwise, if another similar packet is found in the queue, then the retransmission of the lost packet can be suppressed by dropping it from the queue and the next available packet in the queue is processed. Finally, the sink receives the packet and sends an explicit acknowledgment to the previous hop node.

The implementation details of these processes associated with the current node's role are explained as follows.

5.3.1.1 Originator Node

The implementation details and procedures involved at the originator node in ERP are as follows.

1. The originator node (N_O) senses an event of interest E that occurred within its sensing range R.

Algorithm 1 Sense Event

```
1: procedure $Sense(E)$
2:     $E \leftarrow (N_O \rightarrow Sense(event))$
3: end procedure
```

2. N_O generates a packet P and adds its own location along with the event's information to the packet header.

Algorithm 2 Generate Packet

```
1: procedure $GeneratePacket(Node's\ coordinates, Event's\ information)$
2:     $GeneratePacket(N_O \rightarrow \text{x}-axis, N_O \rightarrow \text{y}-axis, E \rightarrow timestamp, E \rightarrow id)$
3: end procedure
```

3. N_O temporarily stores the P in its queue and transmits it towards the sink node (N_S) through the intermediate nodes (N_I) based on the information from the underlying routing protocol.

Algorithm 3 Transmit Packet

```
1: procedure $Transmit(P)$
2:     $N_O \rightarrow EnQueue(P)$
3:     $Transmit(N_O, Route(N_I, N_S, P)$
4: end procedure
```

4. N_O starts its timer and listen to the channel for an iACK of its sent packet P, only if $N_O's$ next hop node is not the sink node. As the sink node is the final destination, thus no iACKs.

Algorithm 4 Start Timer

```
1: procedure $StartTimer(P)$
2:     if $N_O \rightarrow NextHopNode \neq N_S$ then
3:         $timer \leftarrow StartTimer(P)$
4:     end if
5: end procedure
```

5. N_O promiscuously listens to the channel until the timer expires (i.e., α number of ticks after the start timer).

Algorithm 5 Overhearing the Channel

```
1: function $PromiscousListen(P)$
2:     do
3:         $PromiscousListen(P)$
4:         $timer \leftarrow timer + ticks$
5:     while $(PromiscousListen(P) \neq TRUE)$ &&
6:         $checkTimer(timer \leq ExpireTimer(P, \alpha)) == TRUE$
7: end function
```

6. If N_O overhears the transmission of its packet P, it assumes that P has been successfully forwarded by the next hop intermediate node towards the sink and thus N_O drops P from its queue.

Algorithm 6 Packet Reliably Received & Forwarded by Next-hop Node

```
1: procedure $DeQueue(P)$
2:     if $PromiscousListen(P) == TRUE$ then
3:         $N_O \rightarrow DeQueue(P)$
4:     end if
5: end procedure
```

7. Otherwise, N_O assumes that P is lost and thus it needs to be retransmitted for β number of times to maintain reliability before dropping it altogether from the queue.

Algorithm 7 Retransmit The Lost Packet

```
1: procedure $ReTransmit(P)$
2:     if $(PromiscousListen(P) == FALSE)$ && $(P \rightarrow RetxCount < \beta)$ then
3:         $P \rightarrow RetxCount \leftarrow (P \rightarrow RetxCount + 1)$
4:         $ReTransmit(N_O, Route(N_I, N_S), P)$
5:     else
6:         if $(PromiscousListen(P) == FALSE)$&&$(P\rightarrow RetxCount > \beta)$ then
7:             $N_O \rightarrow DeQueue(P)$
8:         end if
9:     end if
10: end procedure
```

5.3.1.2 Intermediate Node

This section explains in detail the implementation steps involved in ERP when a node plays the role of an intermediate node.

1. The Intermediate Node (N_I) receives a packet P to be forwarded to the Sink Node (N_S). N_I temporarily stores P in its queue and forwards it towards the Sink Node (N_S) through other Intermediate Nodes (N_I).

Algorithm 8 Receive, Store and Forward

```
1: $N_I \rightarrow ReceivePacket(P)$
2: $N_I \rightarrow EnQueue(P)$
3: $Forward(N_I, Route(N_I, N_S), P)$
```

2. N_I starts its timer and listens to the channel to get an implicit acknowledgment of its forwarded packet. N_I will not wait for the implicit acknowledgment if sink node is the next hop node, because it is the final node.

Algorithm 9 Start Timer

```
1: procedure $StartTimer(P)$
2:     if $N_I \rightarrow NextHopNode \neq N_S$ then
3:         $timer \leftarrow StartTimer(P)$
4:     end if
5: end procedure
```

3. N_I listens to the channel until the timer expires (i.e. α number of ticks after the start timer).

Algorithm 10 Overhearing the Channel

```
1: function $PromiscousListen(P)$
2:     do
3:         $PromiscousListen(P)$
4:         $timer \leftarrow timer + ticks$
5:     while $PromiscousListen(P) \neq TRUE$ &&
6:           $checkTimer(timer \leq ExpireTimer(P, \alpha)) == TRUE)$
7: end function
```

4. If N_I hears to the transmission of P, it assumes that P has been reliably received and forwarded by the next hop intermediate node and thus N_I drops P from its queue.

Algorithm 11 Packet Reliably Received & Forwarded by Next-hop Node

```
1: procedure $DeQueue(P)$
2:     if $PromiscousListen(P) == TRUE$ then
3:         $N_I \rightarrow DeQueue(P)$
4:     end if
5: end procedure
```

5. If N_I does not overhear P then it assumes that P has been lost; therefore, a retransmission decision needs to be taken by N_I. However, if N_I has reached the retransmission threshold then N_I drops P from its queue and starts processing the other packets in the queue.

Algorithm 12 Packet Lost, Retransmit Or Not

```
1: procedure $RetransmissionRule(N_I,\ P)$
2:     if $PromiscousListen(P) == FALSE\ \ \&\&\ \ RetxCount < \beta$ then
3:         $RetxCount\ \leftarrow\ RetxCount + 1$
4:         $RetxDecisionProcess(N_I,\ P)$
5:     else
6:         $ProcessCount \leftarrow ProcessCount + \infty$
7:     end if
8: end procedure
```

6. The retransmission decision process starts by checking the lost packet P against the other packets in $N_I's$ queue that have same event generation time as the lost packet P. If the event generation time of P matches with another packet (let's assume packet P') then N_I calculates the Euclidean distance between the originator node's locations of packet P and P'. If the distance is less than the twice of sensing range R then N_I assumes based on both temporal and spatial locality that P contains data that is strongly correlated to P' data. Hence, N_I drops P from its queue by suppressing its retransmission.

   However, if no spatiotemporal match is found for the lost packet P in $N_I's$ queue and $P's$ retransmission counter is less than the threshold then N_I retransmits P.

Algorithm 13 Region-based Selective Retransmission Decision Process

```
1: procedure $RetxDecisionProcess(N_I,\ P)$
2:     function $CreateQueueLocArray()$
3:         for $temp\ <\ N_I \rightarrow Queue.Size - 1$ do
4:             $temp \leftarrow temp + 1$
5:             if $N_I \rightarrow Queue\ [temp] \rightarrow timestamp\ ==\ P \rightarrow timestamp$ then
6:                 $QueueLocArray\ [j] \leftarrow\ temp$
7:                 $j\ \leftarrow\ j + 1$
8:             end if
9:         end for
10:     end function
```

Algorithm 13 Region-based Selective Retx Decision Process (Cont.)

```
11:     if $QueueLocArray.Size \geq 1$ then
12:         $dist \leftarrow EDistance(P.loc,\ P'.loc)$
13:         if $dist \leq 2R$ then
14:             $N_I \rightarrow DeQueue(P)$
15:             $ProcessCount \leftarrow ProcessCount + \infty$
16:         else
17:             if $(P \rightarrow RetxCount < \beta)$ then
18:                 $P \rightarrow RetxCount \leftarrow (P \rightarrow RetxCount + 1)$
19:                 $ReTransmit(N_I, Route(N_I, N_S), P)$
20:             else
21:                 $N_I \rightarrow DeQueue(P)$
22:                 $ProcessCount \leftarrow ProcessCount + \infty$
23:             end if
24:         end if
25:     else
26:         if $(P \rightarrow RetxCount < \beta)$ then
27:             $P \rightarrow RetxCount \leftarrow (P \rightarrow RetxCount + 1)$
28:             $ReTransmit(N_I, Route(N_I, N_S), P)$
29:             $ProcessCount \leftarrow ProcessCount + \infty$
30:         end if
31:     end if
32: end procedure
```

5.3.1.3 Sink Node

The sink node (N_S), being the final destination, after receiving packet P, sends an explicit acknowledgment to the previous hop node in order to notify that P has been reliably received and to avoid unnecessary retransmissions from the previous hop node as shown in Algorithm 17.

Algorithm 14 Packet Reliably Received, Stored & Ack Sent

```
1: $N_I \rightarrow ReceivePacket(P)$
2: $N_I \rightarrow EnQueue(P)$
3: $N_S \rightarrow eACK(N_S \rightarrow PrevHopNode,\ P)$
```

5.3.2 Implementation for EERP

This section provides the implementation details of EERP by presenting the pseudo code and processes performed based on what role the current node is playing. EERP builds on top ERP; therefore, EERP includes most of the basic operations involved in ERP. Both ERP and EERP acts in a similar fashion when a node is playing the role of an originator node or the sink node. However, all the major functionalities are added at the intermediate node level in EERP to enhance its overall efficiency in terms of event identification, energy and reliable transmission of event information to the sink. EERP introduces a greedy approach with the use of a multilateration algorithm in a novel context to efficiently determine whether multiple packets contain the same event or not. EERP makes a decision about a packet before sending it the first time, not just at retransmission. EERP also keeps and uses a history of sent packets and a history of dropped packets for each packet in the sent list. EERP also compares multiple packets at once to confirm if they all have the same event, not just the pair of packets as done in ERP. Based on these justifications, this section explains the implementation details of EERP when the current node is acting as an intermediate node. For further details of the processes involved at the originator and the sink node level, please see Section 5.3.1.1 and 5.3.1.3.

5.3.2.1 Intermediate Node

The implementation steps and pseudo codes involved at the intermediate node level in EERP are as follows.

1. An intermediate node (N_I) receives a packet P to be forwarded to the sink node (N_S). N_I temporarily stores P in its queue for further processing. Before forwarding P to the next hop node towards the sink N_S, N_I first checks its sent list to see if it has already sent a packet similar to P based on the spatiotemporal information in the packet's header. This spatiotemporal comparison is performed by comparing the event generation time and location of $P's$ originator node with the other packets in the $N_I's$ sent list. If N_I finds another similar packet in the sent list and there exists at least one packet in its associated drop list then the event identification process is performed. The event identification process first compares the event generation time of P with the packet in sent list and the other packets in its associated drop list. If the time does not match, then P needs to be transmitted. However, if the time matches then multilateration process is performed on all these packets to see if they can detect the same even or not. The multilateration process is considered successful if the estimated event location is less than double the sensing range of the originator nodes of all the packets (P, packets in the sent list, packets in the drop list).

   If the event identification process is successful, that involves the event generation time checking and multilateration of all the packets, N_I suppresses P's transmission by dropping P from its queue and adding it in the drop list associated to that same packet in the sent list. N_I sends an explicit acknowledgment (eACK) to the previous hop node as there will be not iACKs because N_I has suppressed the transmission of P. Otherwise, N_I forwards packet P to the next hop node towards the sink N_S while maintaining reliability.

Algorithm 15 Receive, Store and Forward

```
1: $N_I \rightarrow ReceivePacket(P)$
2: $N_I \rightarrow EnQueue(P)$
3: if $(EventIdentification(P) == TRUE)$ then
4:     $N_I \rightarrow AddToSentList(P)$
5:     $N_I \rightarrow eACK(P)$
6:     $N_I \rightarrow DeQueue(P)$
7: else
8:     $Forward(N_I, Route(N_I, N_S), P)$
9: end if
```

2. One of the important aspects of EERP is to identify if two or more packets are carrying information about the same event or not and forward only those packets which are not yet sent by the current node. For this purpose, our implementation is divided into the following steps:

   N_I checks the packet P against each packet in the list of its successfully sent packets $(i.e.\ N_I \rightarrow SentPktsList[\,])$ that whether there is any other sent packet originated at a similar time as of P and lies within the vicinity of P's originator node.

Algorithm 16 Event Identification Process - Matching of Events

```
1: procedure $Create\ \ QueueLocArray()$
2:     for $temp < N_I \rightarrow SentPktsList.Size - 1$ do
3:         $temp \leftarrow temp + 1$
4:         if $(N_I \rightarrow SentPktsList[temp] \rightarrow timestamp \approx P \rightarrow timestamp)$ && $(Distance(N_I \rightarrow SentPktsList[temp] \rightarrow OriginaorNodeLoc, P \rightarrow OriginaorNodeLoc)) \leq 2R)$ then
5:             $QueueLocArray[j] \leftarrow SentPktsList[temp]$
6:             $j \leftarrow j + 1$
7:         end if
8:     end for
```

Algorithm 16 Event Identification Process - Matching of Events (Cont.)

```
9: end procedure
```

3. If there are any packets similar to P in N_I's sent packet list ($i.e.\ N_I \rightarrow SentPktsList[\,]$), then it checks P's event generation time against the group of dropped packets in its associated drop list. If there are no packets in the drop list, then N_I adds P in this drop list, sends an eACK to the previous hop node as there will be no iACKs for the previous hop node for P, and drops P from N_I's queue. However, if there are packets found in the dropped list that have similar event generation time, then multilateration process is performed on P, P' that is the identical packet found in sent list and the packets in the drop list.

Algorithm 17 Event Identification Process - Match Found

```
1: procedure $EventIdentification(P)$
2:     if $QueueLocArray.Size == 1$ AND
   $N_I \rightarrow SentPktsList[temp] \rightarrow DroppedPktsList.Size == NULL$ then
3:         $AddToDropList(N_I \rightarrow SentPktsList[temp], P)$
4:         $N_I \rightarrow eACK(P)$
5:         $N_I \rightarrow DeQueue(P)$
6:         $ProcessCount \leftarrow ProcessCount + \infty$
7:     end if
8:     if $QueueLocArray.Size == 1$ AND
   $N_I \rightarrow SentPktsList[temp] \rightarrow DroppedPktsList.Size \geq 1$ then
9:         $mLat \leftarrow multilaterationProcess(P,$
   $N_I \rightarrow SentPktsList[temp],$
   $N_I \rightarrow SentPktsList[temp] \rightarrow DroppedPktsList[\,])$
10:    end if
11:    if $mLat == Success$ then
12:        $AddToDropList(N_I \rightarrow SentPktsList[temp], P)$
13:        $N_I \rightarrow eACK(P)$
14:        $N_I \rightarrow DeQueue(P)$
15:        $ProcessCount \leftarrow ProcessCount + \infty$
16:    else
```

Algorithm 17 Event Identification Process - Match Found (Cont.)

```
17:         $N_I \rightarrow Forward(P)$
18:         $ProcessCount \leftarrow ProcessCount + \infty$
19:     **end if**
20: **end procedure**
```

4. However, if there are no packets found similar to P in $N_I's$ sent list then N_I needs to forward P towards the sink, as N_I has not sent any packet that has same event as that of P's event.

Algorithm 18 Forward Packet

```
1: **procedure** $Forward(P)$
2:     $N_I \rightarrow EnQueue(P)$
3:     $Forward(N_I,\ Route(N_I,\ N_S),\ P)$
4: **end procedure**
```

5. N_I starts its timer and listens to the channel to see if the next hop node has further forwarded its sent packet P or not only on a condition if the next hop node is not the sink node. As sink node is the final destination, it does not further forwards the packet and sends an eACK back to the previous hop node after successfully receiving the packet.

Algorithm 19 Start Timer

```
1: **procedure** $StartTimer(P)$
2:     **if** $N_I \rightarrow NextHopNode \neq N_S$ **then**
3:         $timer \leftarrow StartTimer(P)$
4:     **end if**
5: **end procedure**
```

6. N_I promiscuously listens to the channel until the timer expires (i.e., α number of ticks after the start timer).

Algorithm 20 Overhearing the Channel

```
1: function $PromiscousListen(P)$
2:     do
3:         $PromiscousListen(P)$
4:         $timer \leftarrow timer + ticks$
5:     while $PromiscousListen(P) \neq TRUE$ &&
6:           $checkTimer(timer \leq ExpireTimer(P, \alpha)) == TRUE)$
7: end function
```

7. If N_I hears the transmission of P, it assumes that P has been successfully forwarded by the next hop node towards the sink. Thus, N_I adds P to its sent list and drops P from its queue.

Algorithm 21 Packet Reliably Received & Forwarded by Next-hop Node

```
1: procedure $DeQueue(P)$
2:     if $PromiscousListen(P) == TRUE$ then
3:         $N_I \rightarrow AddToSentList(P)$
4:         $N_I \rightarrow DeQueue(P)$
5:     end if
6: end procedure
```

8. If N_I does not overhear P then it assumes that P has been lost; therefore, a retransmission decision needs to be taken by N_I. N_I first checks if it has not yet reached the retransmission threshold for P then it will repeat the process as from step 4 to 7 (Algorithm 11 to 14) until it reaches the retransmission threshold.

   In case, N_I reaches its retransmission threshold and P has not been moved to the end of the queue earlier than N_I moves P to the end of its queue for further processing and continue processing other packets in its queue. However, if P have already been moved to the end of the queue earlier then P is finally abandoned from the queue and N_I continues to process other packets in its queue.

Algorithm 22 Retransmission Decision @ Intermediate Node

```
1: procedure $RetxDecisionProcess(N_I,\ P)$
2:     if $(PromiscousListen(P) == FALSE)\ \&\&\ (P \rightarrow RetxCount < \beta)$ then
3:         $P \rightarrow RetxCount \leftarrow (P \rightarrow RetxCount + 1)$
4:         $ReTransmit(N_I, Route(N_I, N_S), P)$
5:     else
6:         if $(PromiscousListen(P) == FALSE)\ \&\&\ (P \rightarrow RetxCount > \beta)\ \&\&\ (P \rightarrow MoveToEndCounter > \gamma)$ then
7:             $N_O \rightarrow DeQueue(P)$
8:         else
9:             $MovePktToEndOfQueue(P)$
10:        end if
11:    end if
12: end procedure
```

9. The process of moving the packet P to the end of the queue and continue processing other packets in the queue.

Algorithm 23 Move Packet to the End of Queue

```
1: procedure $MovePktToEndOfQueue(P)$
2:     if $QueueLocArray.Size - 1 == NULL$ then
3:         $P \rightarrow MoveCounter \leftarrow P \rightarrow MoveCounter + 1$
4:         $MovePktToEndOfQueue(N_I, P)$
5:         $ProcessCount \leftarrow ProcessCount + \infty$
6:     end if
7: end procedure
```

5.4 Limitations of Implemented Mechanisms

The implementation of region-based selective retransmission mechanism introduced in ERP identifies only two packets if they are carrying information about the same event or not. As it uses the Euclidean distance to measure the spatial correlation between the two packets, this limits the identification through spatial correlation to only two packets and affects the accuracy of event identification process. This is because there might be more than two packets in the queue that originated as a result of the same event, but ERP can check only the correlation between the two packets. However, the implementation of EERP overcomes this limitation by introducing the multilateration-based event identification mechanism which improves the accuracy of event identification by calculating the spatial correlation of multiple packets at the same time.

Since the focus of this thesis is to perform its operations in a distributed fashion; therefore, in-network data processing is used throughout the implementation of the proposed work. In order to perform processing at the nodes on the data readily available to it, the data needs to be temporarily stored in the node's queue for processing. Both basic and enhanced ERP maintains the nodes' queues by dropping a significant amount of stale packets occupying the queue. However, the proposed methodologies do not have any additional queue management mechanism to further improve the limited memory and storage of the nodes. Also, the proposed work uses both fixed retransmission timeout and the threshold for the number of times a packet can be retransmitted before it is considered to be lost. However, an intelligent retransmission timeout mechanism that dynamically adjusts the retransmission timeouts and the threshold depending on the network traffic load might benefit in terms of saving node's space and reducing the overall packet delivery delay.

5.5 Summary

This chapter contributes towards the second, third and fourth contributions along with the overall aim of the thesis by defining how the proposed event reliability protocol can be implemented with an efficient event identification mechanism. The prototypical implementation of EERP divides the overall event identification and reliability process into three main categories of nodes, namely originator, intermediate and the sink node. The originator node senses the event occurred within its sensing range, it generates a packet, adds the event information in the packets, sends it towards the sink and uses an iACK mechanism to maintain reliable transmission of its sent packet through a set of intermediate nodes. The intermediate nodes receive the packet, perform event identification process and forwards the packets towards the sink node accordingly while maintaining event reliability. The sink node receives the packet and acknowledges the previous hop node about it.

For the realistic implementation of ERP and EERP, a distributed event generation application is developed and integrated with the implementation of the proposed protocols are implemented using QualNet simulator. To further improve the performance of ERP and EERP, the proposed work is implemented on a network providing 100% connectivity and coverage guarantee to improve the scalability of event detection. The processes and procedures involved in the implementation of the proposed work are explained in detail in the form of pseudo codes according to each category of nodes. This conceptual design is implemented for a static sensor network with flat network topology. However, these limitations do not hamper the functionality and viability of the conceptual design for the purpose of the objectives mentioned in Section 1.2. The next chapter evaluates the performance of the implementation. The evaluation is based on the simulation results in various scenarios.

Chapter 6

Performance Evaluation

The previous chapters (Chapter 4 and Chapter 5) presented the conceptual design and a prototypical implementation of ERP and EERP. This thesis aims to provide a hop-by-hop event reliability mechanism that efficiently performs distributed event identification based on spatiotemporal correlation phenomena using in-node data processing for reliability and energy benefits. The proposed design and implementation need to be evaluated to find out if it successfully achieves the aim and objectives of this thesis. Therefore, this chapter further adds towards achieving the aim of this thesis by evaluating the proposed mechanisms in terms of achieving event reliability, event identification, energy efficiency and event coverage throughout the network.

The remaining chapter is organized as follows. Section 6.1 presents the details of the simulation environment used for the evaluation of the proposed work. Section 6.2 presents the experimental design that discusses the comparison criteria and how the proposed schemes will be assessed. Section 6.3 and Section 6.4 discuss a range of experiments performed using grid and random deployments respectively to evaluate the proposed schemes with the others. Section 6.5 further evaluates the multilateration process that is used in performing event identification. Section 6.5 concludes the chapter with a summary.

Parts of this chapter are previously published in [71], [78] and [93].

6.1 Simulation Environment

The simulation-based experiments to evaluate ERP and EERP are implemented in QualNet 5.2 [137] with all the nodes configured as IEEE 802.15.4 compliant nodes running as full-functioning devices. QualNet's sensor networks model library is used to implement the proposed work. This model library provides specifications for a suite of high-level communication protocols using small, low-power digital radio based on IEEE 802.15.4 standard for Wireless Sensor Networks. It targets Radio Frequency (RF) applications that require a low data rate, long battery life, and secure networking. These networks are aimed at automation, remote control, and other WSN applications. The QualNet's IEEE 802.15.4 standard defines the physical layer (PHY) and Medium Access Control sublayer (MAC) specifications as the wireless communication standard for low-power consumption, Low-Rate WPAN (LR-WPANs).

The effectiveness of the proposed work is assessed through simulations developed in QualNet for distributed networks of sensor nodes. Sensor nodes are deployed in a network area of 100m$\times$100m. A sink is placed at the center of the network area. Sensor nodes are deployed with 100% network connectivity and coverage. This 100% network connectivity and coverage guarantee is provided by exploiting the idea presented by Xing et al. [138], where if a network is proven to provide 100% sensing coverage for a uniform sensing range R, then the network will be connected if the common transmission range is at least $2R$. The network may be connected if the transmission range is less than $2R$, but that is more difficult to prove; so this thesis uses this simplification. Thus, to make a network with 100% connectivity and coverage, the antenna height and transmission power of the sensor motes used in the simulation have been chosen to achieve a sensing and radio range of 20m and 40m. Further details regarding the grid and random deployments with 100% connectivity and coverage are given in Appendix B and Appendix C.

The nodes are deployed in a flat sensor area using two different deployments, i.e., grid and random. For each grid and random topologies, planned deployments of increasing densities are generated. The sensor nodes are deployed for ten different node densities (i.e., number of nodes per square meter). As density increases, each node has more overlapping regions with its neighbors sensing areas from near perfect to heavily overpopulated sensor networks are generated, as indicated by the increasingly dark areas in Figure 6.1 to Figure 6.4. The node densities generated for grid deployments are 0.0025, 0.0030, 0.0035, . . . , 0.0070; and random deployments are 0.0040, 0.0045, 0.0050, . . . , 0.0085. For the grid deployment, node densities from 0.0025 to 0.0070 represent a low to highly dense deployment of nodes as shown in Figure 6.1 and Figure 6.2. Whereas, for the random deployments, node densities from 0.0040 to 0.0085 represent low to highly dense deployments as shown in Figure 6.3 and Figure 6.4.

For each deployment, 10 random sensing events are generated periodically. The regular interval between the events is uniformly distributed between 3 to 10 simulated seconds apart from each other at random locations within the network. Each simulation is executed with 10 uniquely random seeds for each scheme in each of the random and grid-based node deployment scenarios. The two-ray path loss model is used at a frequency of 2.4GHz, where the packet reception model is the Signal-to-Noise Ratio (SNR) bounded with a threshold of 10dB. As routing is not the focus of this thesis; therefore, it is assumed that the underlying routing protocol has already computed the routes. Also, the simulations performed do not consider mobility as the nodes are considered to be static.

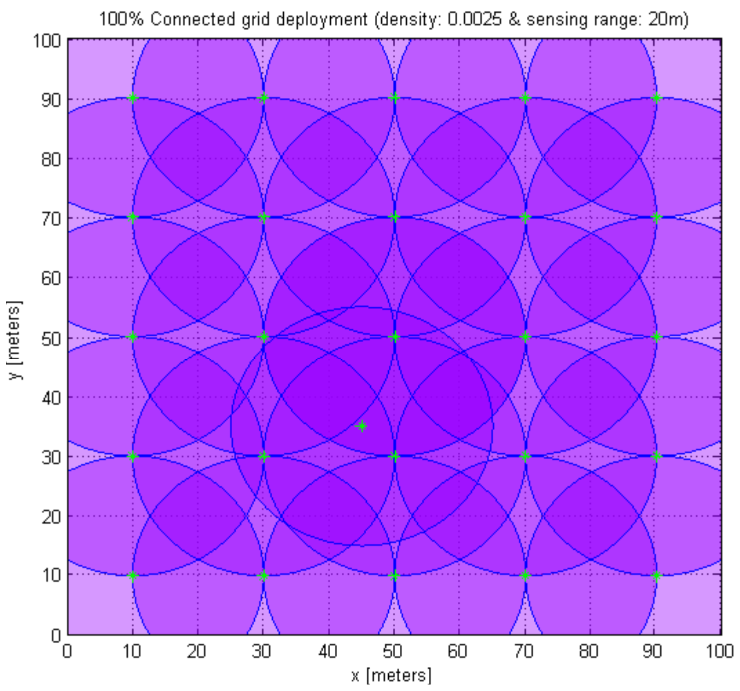


Figure 6.1: A grid deployment for density 0.0025

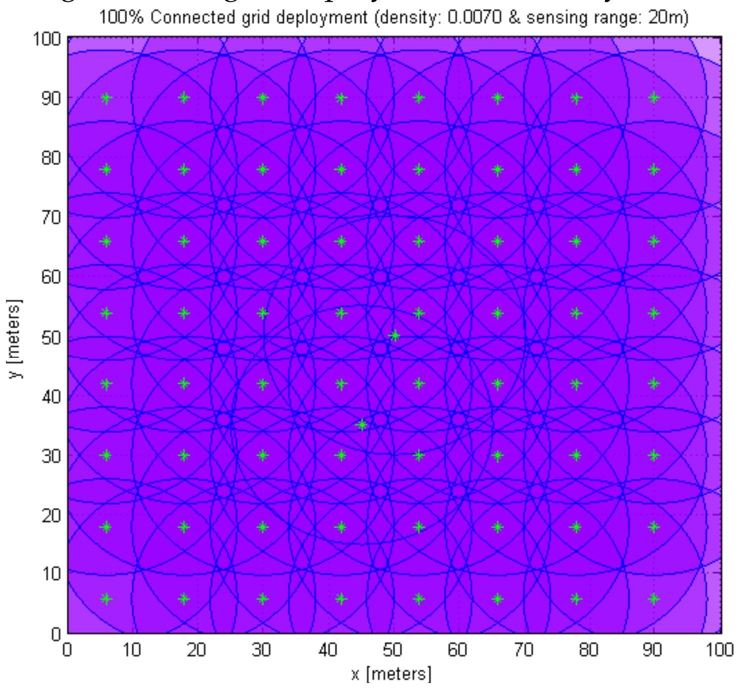


Figure 6.2: A grid deployment for density 0.0070

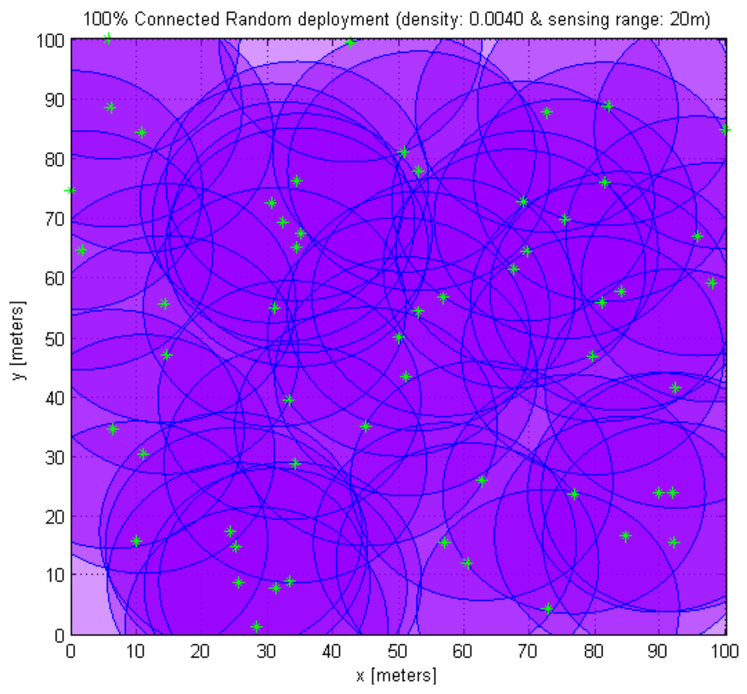


Figure 6.3: A random deployment for density 0.0040

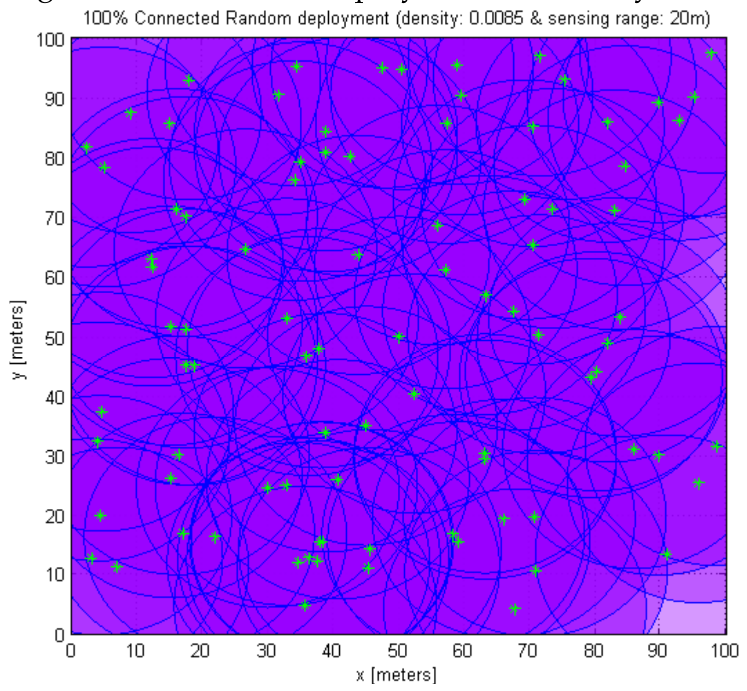


Figure 6.4: A random deployment for density 0.0085

6.2 Experimental Design

This thesis focuses on efficiently performing event identification and reliable delivery of event information from the source to the sink node. Therefore, keeping in view the simulation environment as discussed in the previous section, the proposed enhanced Event Reliability Protocol (EERP) [78] is assessed by how well event occurrences are reported at the sink from all over the network as compared to Event Reliability Protocol (ERP) [71], Stop-and-Wait Implicit Acknowledgment (SWIA) [72], and a BaseModel (with no reliability) in the grid and the random deployments. ERP and EERP being the basic and enhanced versions of event reliability protocols proposed in this thesis have been thoroughly discussed throughout this thesis in the preceding chapters. SWIA (Stop-and-Wait Implicit Acknowledgement) as discussed in Section 3.3.1 makes use of implicit acknowledgments in a distributed hop-by-hop fashion to maintain reliability by retransmitting the lost packet until it is implicitly acknowledged. The BaseModel presents a simple packet forwarding protocol where nodes transmit packets without considering any reliability or event identification mechanism. They simply forward the packets one after another without retransmitting any of the packets that are lost due to transmission errors, low connectivity, collisions or any other foreseeable reasons.

Comparison with BaseModel shows the benefits of EERP over a scheme which does not use any sophisticated mechanism for event identification or reliability. Whereas, SWIA is one of the very few schemes that perform hop-by-hop reliability while using implicit acknowledgments (iACKs) for maintaining reliability. ERP is the only scheme performing event identification process in a distributed fashion with hop-by-hop event reliability using iACKs. EERP further enhances ERP in terms of a more sophisticated event identification mechanism with reliable delivery of event information from all over the network.

The focus of performing the simulations is to evaluate how efficiently each scheme performs in terms of energy efficiency, reliable delivery of event information and event coverage from all over the network.

6.3 Grid Deployment

This section presents and discusses the experimental results based upon deploying the nodes using grid topology with 100% connectivity and coverage. The performance EERP is evaluated against ERP, SWIA and BaseModel based on the following set of experiments.

6.3.1 Unique Event Reported at the Sink

Figure 6.5 shows the results for the number of unique events successfully received at the sink node out of 10 actual events. Comparing EERP with BaseModel, SWIA, and ERP, it can be noted that EERP shows better performance as compared to the other protocols. In the Basemodel, nodes

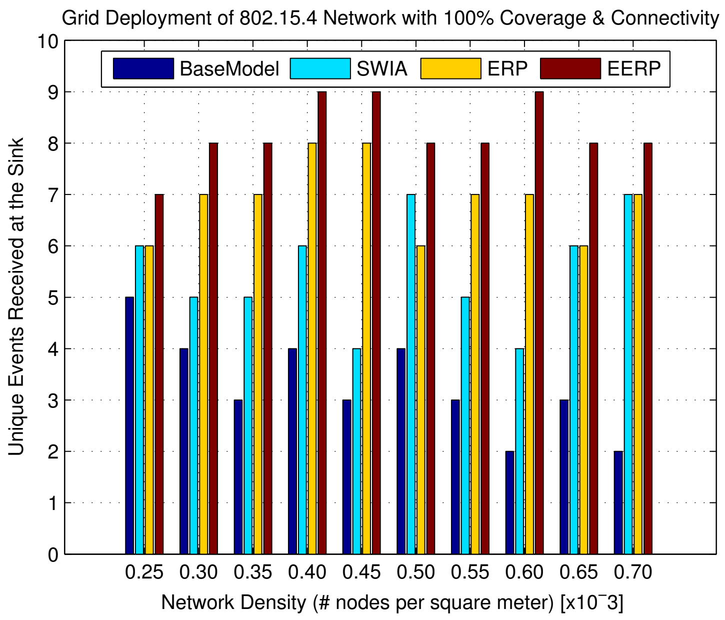


Figure 6.5: Unique Event Reported at the Sink (Average)

transmit packets without considering any reliability or acknowledgment mechanism. They forward the packets one-after-another without retransmitting any of the packets that are lost. Without a reliability mechanism, it is not surprising to see that the BaseModel reports only a small amount of unique events. This is because, when the network is too dense then the packets are lost due to collisions and congestion, and with no reliability mechanism, a minimal amount of packets reach the sink.

The SWIA approach is simple and straightforward; it does not perform the efficient event identification process as done by EERP and thus able to deliver fewer unique events than EERP. On the other, hand ERP performs better than SWIA in terms of unique events reported to the sink, as ERP uses its region-based selective retransmission mechanism to reduce the unnecessary transmission of duplicate event data. However, EERP also performs than ERP, as it uses a more efficient event identification mechanism than ERP.

6.3.2 Percentage of Duplicate Events Reported at the Sink

Calculating the number of duplicate events reported at the sink shows that SWIA produces the largest percentage of duplicate events due to its retransmission mechanism (see Figure 6.6). We can also see the benefits provided by EERP in minimizing the number of duplicate event packets. The number of duplicate events reported at the sink in EERP is much lower as compared to that in SWIA, ERP and BaseModel. This means that the actual number of unique events reported at the sink for EERP is significantly higher than other protocols. Therefore, EERP is significantly better in terms of successfully transmitting useful event information to the sink. This is further evident from the unique events out of the total packets received at the sink node as shown in the other results presented, respectively.

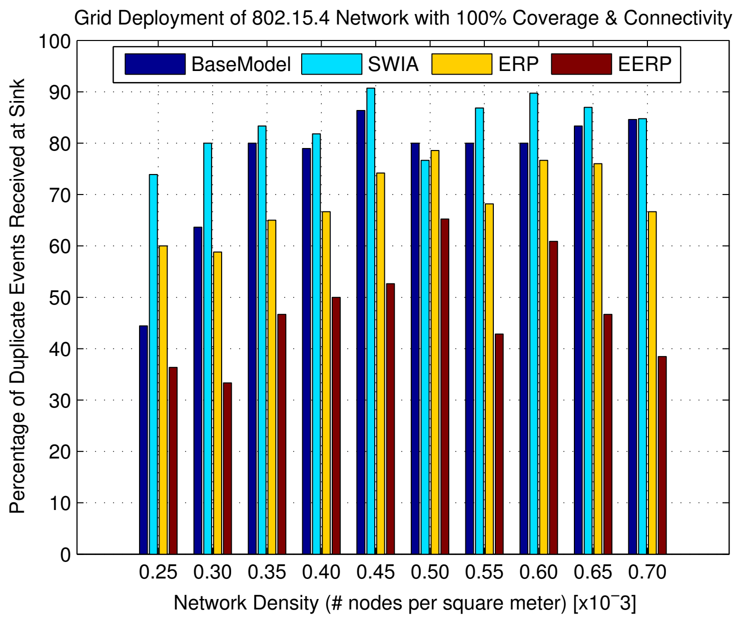


Figure 6.6: Percentage of Duplicate Events Reported at the Sink

6.3.3 Total Packets Received at the Sink

Since energy consumption is one of the major issues in WSNs and communication being the highest consumer of energy, especially in a densely populated sensor network, where nodes are continuously transmitting the event data from originator nodes to the sink. The high number of packets received at the sink means that an even higher number of packets were transmitted across the network. This is because, if three nodes sense an event then at least three packets will be transmitted about the same event, where this redundant transmission might also multiply along the intermediate hops towards the sink. This will increase the transmission of packets with duplication events that increases the overall network traffic and thus results in more energy consumption. Therefore, we take this measure of average total packets received at the sink as an indicator of energy consumed based on the amount of communication occurring in the network.

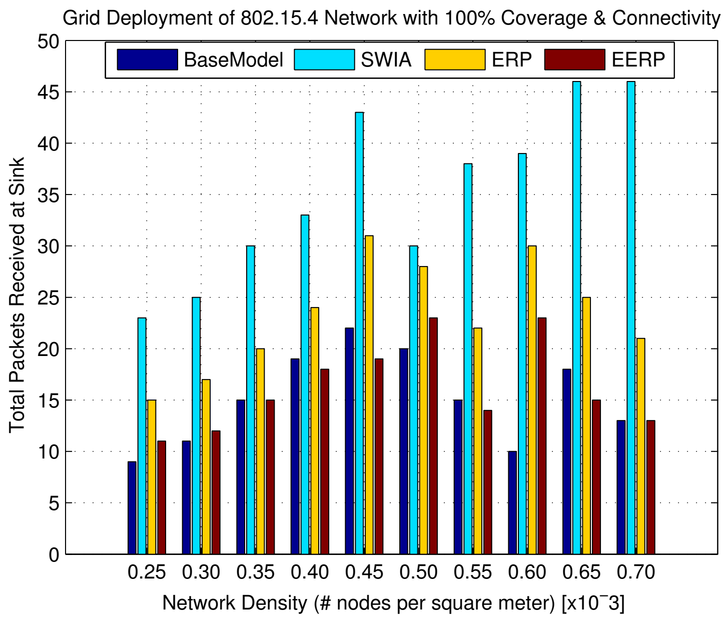


Figure 6.7: Total Packets Received at the Sink (Average)

Based on this justification, Figure 6.7 shows that SWIA has the highest transmission overhead and therefore the highest energy consumption. This high transmission could be due to the continuous retransmissions in case of packet loss, which would further increase in a high-density network. ERP performs comparatively better than SWIA, because of its efficient region-based selective retransmission mechanism. BaseModel shows the lowest energy consumption in Figure 6.7, but it does not mean that it is the best scheme, because due to having no loss recovery or event identification mechanism, the packets received are mostly duplicates. However, EERP significantly reduces the energy consumption as compared to the other schemes by transmitting only the sufficient packet to the destination while maintaining event reliability and network scalability with increasing node density.

6.3.4 Average Battery Consumed by the Nodes

In order to further verify the results shown and justifications given in the previous section (Section 6.3.3), we used QualNet's battery model library to produces the results for average battery consumed by the nodes during the simulation time. Battery provides voltage and current for the components attached to the battery such as radio interfaces, CPU, Memory blocks, sensing core, etc. The total battery consumed by the system per cycle is the sum of energies consumed by the radio transceivers (E_{Trans}), protocol processor (E_{CPU}), the DC-DC converter (E_{DC}) and the efficiency losses in the battery (E_{Bat}) [137].

$$E_{Cycle} = E_{Trans} + E_{CPU} + E_{DC} + E_{Bat}$$

Battery model is defined for a node where the total battery consumed by a node is reported in mAhr. Figure 6.8 shows the average battery consumed by a node for varying numbers of node densities in various protocols.

It can be seen in Figure 6.8 that SWIA consumed more battery for all network densities than either the base model, ERP and EERP. This is due to the retransmission mechanism employed by the SWIA that keeps retransmitting the packets until the node detects an iACK. This creates a high transmission overhead resulting in higher battery consumption. ERP employs an efficient retransmission mechanism that also limits the retransmissions to a certain threshold, and thus, sends/forwards fewer packets than SWIA and hence consumes less battery power at the node level. Since, EERP employs event filtering (based on the identification of packets with the same event) and limits the overall transmissions and retransmissions, thus, resulting in better performance than the others for all node densities. Also, the base model just performs forwarding of packets from the sensor nodes in the network and does not consider reliability; therefore, the battery consumed at the node is almost equivalent to that of

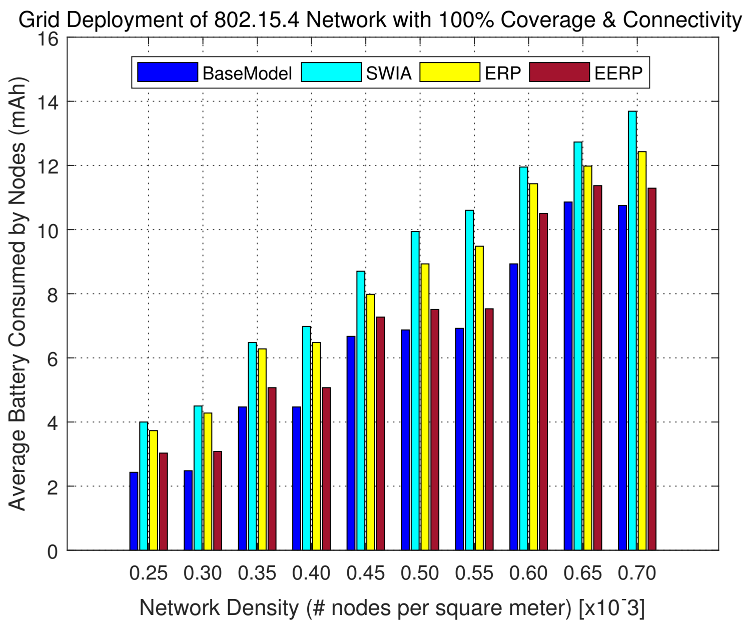


Figure 6.8: Average Battery Consumed by the Nodes

ERP and SWIA in most cases.

6.3.5 False Positive

The false positive measure is the probability of a node concluding that two or more packets contain data of the same event when in actuality the packet contain data of different events, hence termed as false positive in our scenario. This incorrect decision results in discarding packets that need to be transmitted to the sink node thereby resulting in the loss of useful event information.

The false positives are measured as the ratio of total events generated in the network to the total number of events reported at the sink node. Since events are detected at the source and our method performs filtering of similar events on their way to the sink. Thus, the ratio of the events reported to the events generated will provide a measure of events that have been dropped and not reported based on the false assumption that they

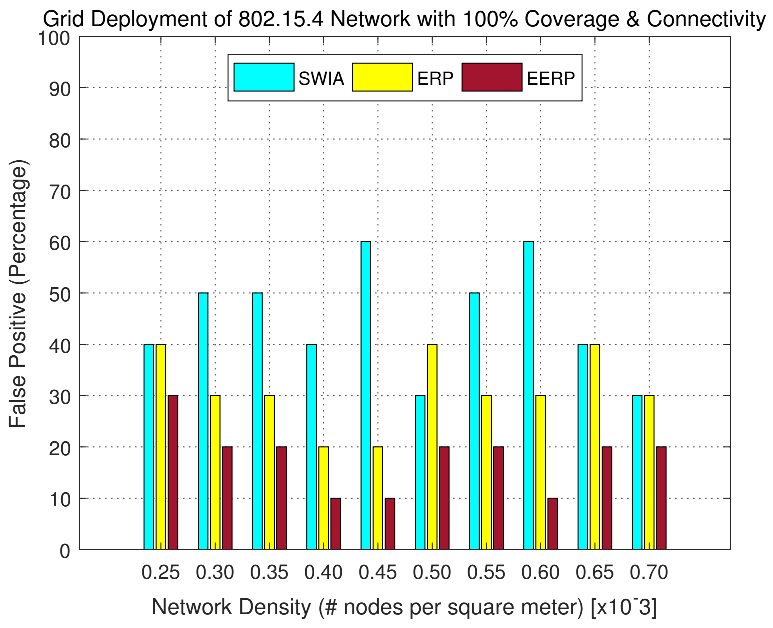


Figure 6.9: False Positives Percentage for Grid Deployments

contain similar event information.

Figure 6.9 shows the false positive percentage for various protocols in a network with varying node densities. It can be seen that EERP produces the least number of false positives as compared to all other protocols. The information loss due to false positives in EERP is around 10%-20% as compared to 20%-40% for ERP and 30%-60% for SWIA. The results show that the event identification decision making in EERP is performed more accurately in terms of identifying the packets with the same event as compared to ERP. There is no reliability or event identification decision making involved in the BaseModel; therefore, it is not considered in the false positive results.

6.3.6 False Negative

The false negative measure is the probability of a node concluding that two or more packets contain data of different events when in actual the data are of the same events, hence called false negative in our scenario. Again, this incorrect decision will make the nodes transmit the duplicate packet towards the sink node. This increases the network traffic that can lead to multiple problems (such as congestion in the network, queuing and processing at the intermediate nodes and consumption of energy on transmitting unnecessary packets).

False negatives are measured by considering the number of duplicate packets received at the sink node (see Figure 6.6). It can be seen that EERP is more efficient in identifying unique events and thus the percentage of duplicates packets (i.e., false negative) is smaller than the other protocols. EERP shows an improvement of 53% over SWIA and 41% over ERP in identifying events correctly.

6.4 Random Deployment

This section discusses the experimental results performed by deploying the nodes using random topology with 100% connectivity and coverage. The number of nodes in the deployments for different densities in the random topology is comparatively higher than that of grid deployment. The densities were increased to maintain 100% coverage and connectivity of the sensor network (Appendix B and Appendix C). Following experiments are performed to evaluate the performance of EERP with ERP, SWIA and BaseModel.

6.4.1 Unique Event Reported at the Sink

To find out the unique event delivery from all over the network to the sink in randomly deployed nodes, we calculate the average number of unique events that were successfully delivered to the sink. This performance metric is measured by comparing EERP against ERP, SWIA and the Basemodel, as shown in Figure 6.10. Like grid deployments, similar trends are observed for random deployment, where EERP outperforms the other protocols in terms of number of unique events reported to the sink. Although ERP performs better than SWIA, it is observed that SWIA shows comparable performance to ERP in some of the scenarios with the high node densities. The protocols (SWIA and ERP) themselves might not cause this; rather,

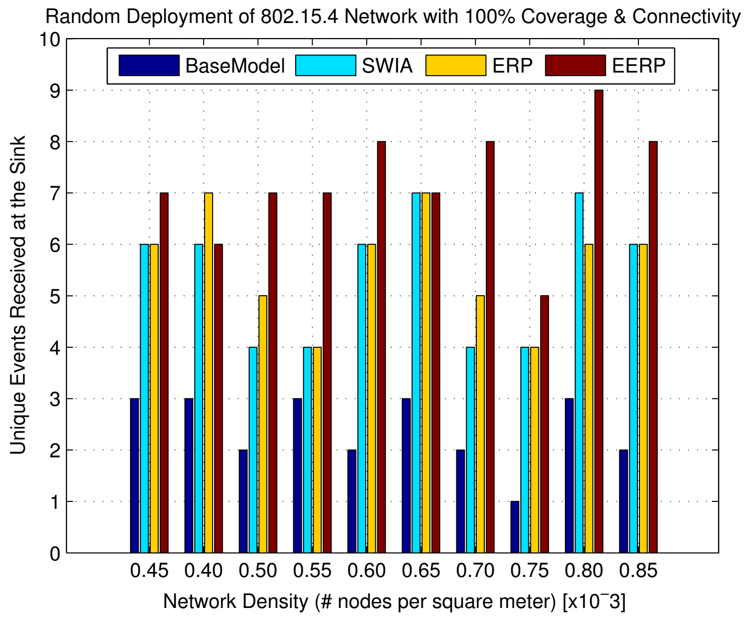


Figure 6.10: Unique Event Reported at the Sink (Average)

this could be due to the randomness of the network topology and node placement. Without a reliability mechanism, it is not surprising to see that the BaseModel reports only a small amount of unique events. This is because the BaseModel simply forwards the packets without retransmitting any of the packets that are lost during the transmission.

6.4.2 Percentage of Duplicate Events Reported at the Sink

Figure 6.11 shows the percentage of duplicate events reported to the sink node for random deployments with 100% connectivity and coverage. SWIA reports the largest percentage of duplicated events, which is due to its higher number of retransmissions. The duplicate event percentage for SWIA increases with the increase in node density as altogether more retransmissions and transmissions are preformed. It can also be seen the ben-

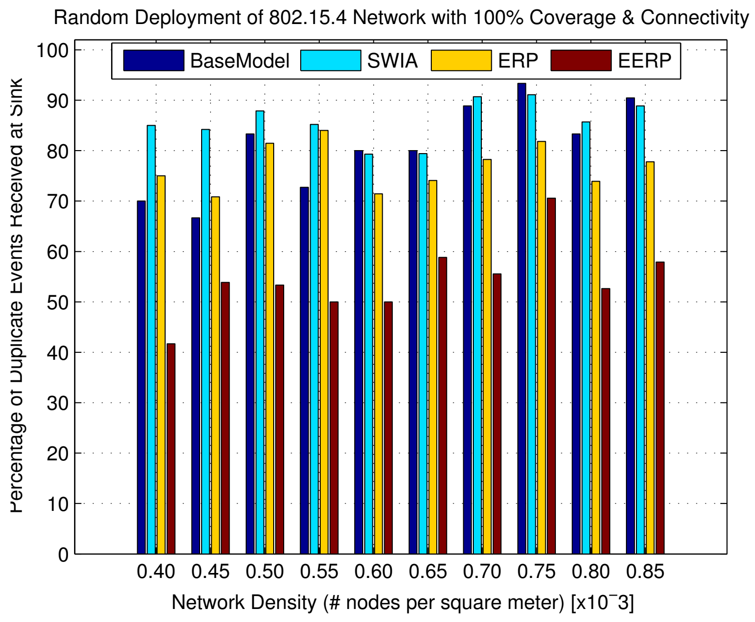


Figure 6.11: Percentage of Duplicate Events Reported at the Sink

efits provided by EERP in minimizing the percentage of duplicate events in all node densities. EERP shows the lowest number of duplicate events as compared to SWIA, ERP and the BaseModel. The better performance of EERP in terms of duplicate events transmitted is further demonstrated when we calculate the unique events reported. This means that the percentage of unique events in those duplicate packets are much better in EERP than the other protocols. Therefore, EERP performs significantly better in successfully transmitting useful even information to the sink with a smaller number of duplicate events as compared to the other protocols.

6.4.3 Total Packets Received at the Sink

As discussed earlier, that communication is one of the major factors that affects energy efficiency, so one of the ways we calculated energy consumption is by taking the average of total packets delivered at the sink node for different node densities, as shown in Figure 6.12. A large number of packets received at the sink is the result of higher number packets transmitted by the nodes, which in turn means more communication has happened at the node level. Based on this assumption, EERP has the lowest transmission overhead which means it is the most energy efficient as compared to the other protocols. Average packets received at the sink not only shows the energy efficiency of the protocol but also shows that how efficiently EERP delivers the information about the events from all over the network with a minimum amount of packets transmitted. In this context, BaseModel's performance is comparable to EERP, but looking at the overall performance, EERP performs better in terms of reliably delivering the maximum information about all the events happening in the network with minimum communication cost. However, the BaseModel reports the lowest unique events so that it could be energy efficient but the overall performance is not comparable to other protocols. On the other hand, SWIA performs the worst in terms of communication cost, as the number of retransmissions is too high in case of a packet loss, especially in a high-density network. In SWIA, the total packets received at the sink node significantly increases as the node density increases, this might be due to the randomness in the node deployments. However, ERP turns out to be the third in best in terms of packets transmitted to the sink, but second best in terms of event information delivery to the sink.

High node density increases the number of packets transmitted across the network therefore decreasing energy efficiency. Based on this fact, it can be clearly seen that EERP also outperforms other protocols in terms of energy efficiency by transmitting only the most significant packets towards

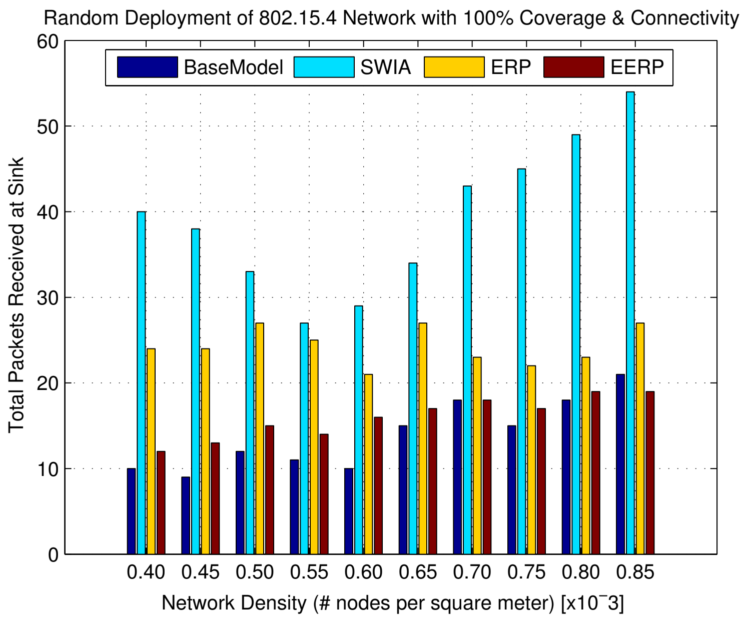


Figure 6.12: Total Packets Received at the Sink (Average)

the sink while maintaining event coverage. Thus, the overall performance of EERP (despite randomness in topology and increased node densities) is much better than that of BaseModel, SWIA and ERP.

6.4.4 Average Battery Consumed by the Nodes

For further verification of the results discussed in the previous section (Section 6.4.3), the amount of average battery consumed by the nodes is calculated using QualNet's battery model, as shown in Figure 6.13. Like grid deployments, similar trends are observed in random deployments, but marginally higher battery power is consumed in the random deployments for all the node densities across all the protocols. This is because, in random deployments, a larger number of nodes detect and report the events producing more communication and thus consuming more battery at the nodes. However, in both random and grid deployments, EERP again outperforms all the other protocols in terms of average battery consumed

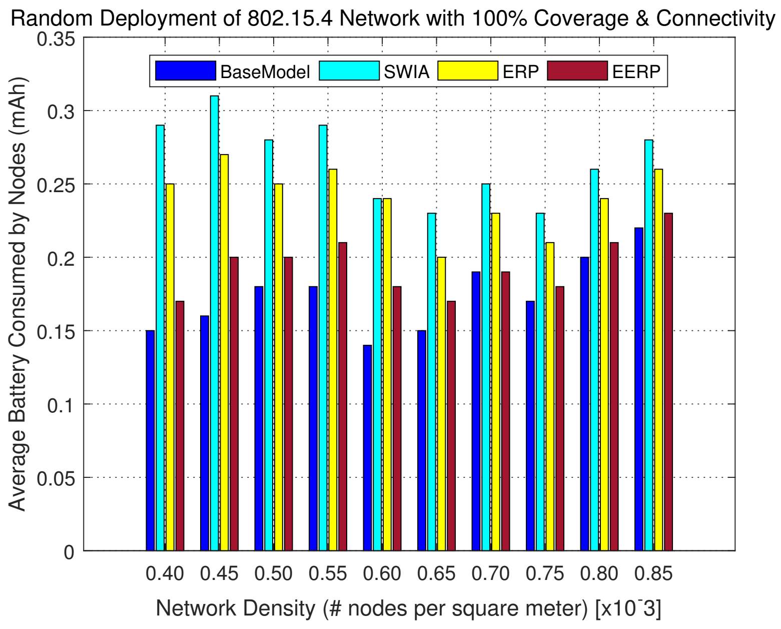


Figure 6.13: Average Battery Consumed by the Nodes

by the nodes.

6.4.5 False Positive

As explained earlier in Section 6.3.5, the false positive measure is the probability of a node concluding that two or more packets contain data of the same event when in fact the packets contain data representing different events.

Figure 6.14 shows the false positive ratio of various protocols in the random topologies for varying node densities. It can be seen that EERP produces the least number of false positives as compared to all other protocols. Also, comparing the results of random deployment to grid deployment (see Figure 6.9), it can be seen that results for grid deployment are more stable as compared to the random deployment. This is because as compared to the random deployments, the nodes in grid deployments are plotted at specific locations with fewer nodes overlapping the same region depending

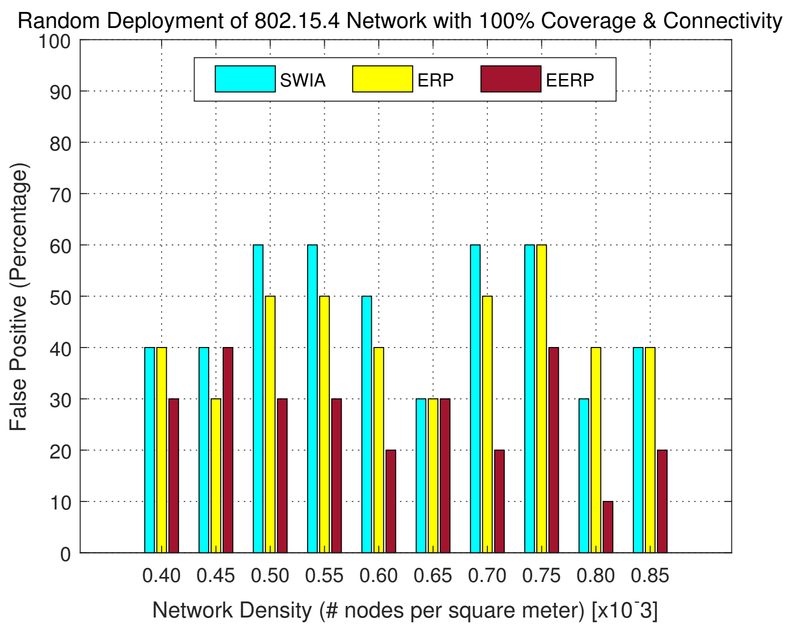


Figure 6.14: False Positives Percentage for Random Deployments

on the particular density values.

6.4.6 False Negative

The false negative measure has been already described in Section 6.3.6. The false negatives are measured by the number of duplicate packets received at the sink node in a random deployment (see Figure 6.11). It is observed that EERP produces the least number of duplicate packets (i.e., false negative) as compared to the other protocols. Again, in comparison to the false negative ratio in grid deployments (see Figure 6.6), random deployments have a similar trend in the results for smaller densities. However, as the node densities increases, the ratio of false negative also increases. This is because, the increased number of overlap mean a larger number of nodes detecting and reporting the events, the higher number of packets containing similar event information are reported to sink.

6.5 Error Estimation in Multilateration

In this section, we evaluate the performance of multilateration by how accurately it identifies an event's location in a wireless sensor network. To test the functionality of multilateration, we picked a grid and a random deployment scenario with the increasing node densities for each scenario and the same network configurations as all the other experiments performed above. For each node density in the grid and the random deployments, events are randomly generated all over the network and multilateration is executed only if three or more nodes sense the event. Each multilateration execution estimates the unknown event location; however, the record of these randomly generated events are kept to calculate the average error between the actual and the estimated locations of the events. The average of the estimation error is calculated for each case that is 3 to 10 nodes surrounding the event.

Figure 6.15 shows the average error in estimating the location of the event for the grid deployment. It can be seen that the error rate decreases with the increase in node density, where event locations can be estimated more accurately with a larger number of nodes surrounding the event. The estimation errors are calculated in meters, and it noticed in Figure 6.15 that the differences between the actual and estimated values starts approximately from 8m and goes down to 1m ranges with increasing densities. It is also noted in Figure 6.15 that for some node densities like 0.0025 and 0.0030, the maximum number of nodes surrounding the event does not go beyond 5; however, some scenarios go until 6 and 7 nodes. This is purely due to the grid-based deployments, as the nodes are placed on specific locations with similar overlapping in each density ensuring 100% connectivity and event coverage throughout the network.

Figure 6.16 shows the average error in estimating the location of the event using multilateration for the random deployment with increasing

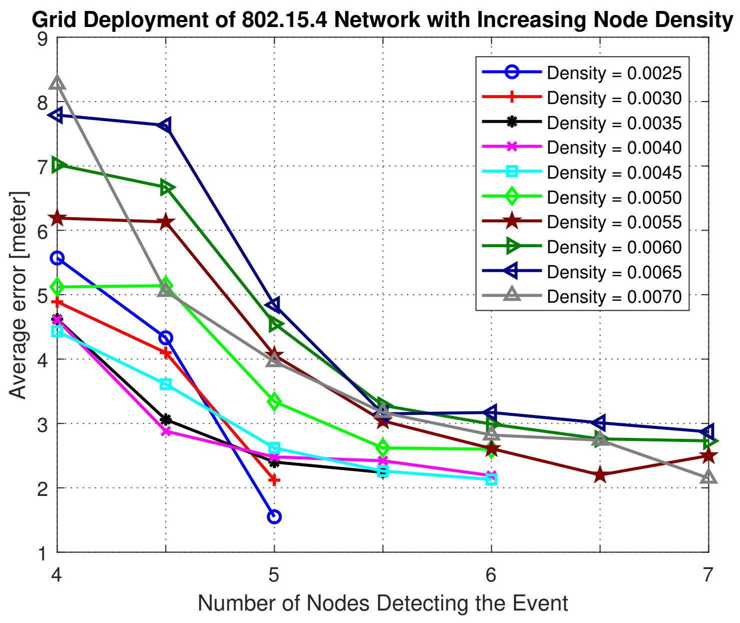


Figure 6.15: Error Estimation for Grid Deployments

node densities. A similar trend can be seen in the random deployments, where the margin of error between the actual and estimated event location is reducing with the increase in node density. Unlike grid deployments, in random deployments, the number of nodes detecting the events are from 3 to 10, even for the least dense deployment. However, the error margin is higher as compared to the grid deployments. This is because the nodes are randomly deployed while considering the 100% connectivity and coverage require more and more nodes until the network becomes 100% connected. Therefore all the cases from 3 to 10 nodes detecting an event can be found.

This is evident from the results of both random and grid deployments that error rate in estimating the event's location decreases with more nodes surrounding the event. Therefore, event identification process will get more accurate as the number of nodes surrounding the event increases. This in turns improves the event reliability process in a highly dense network, as a greater number of redundant packets will be eliminated.

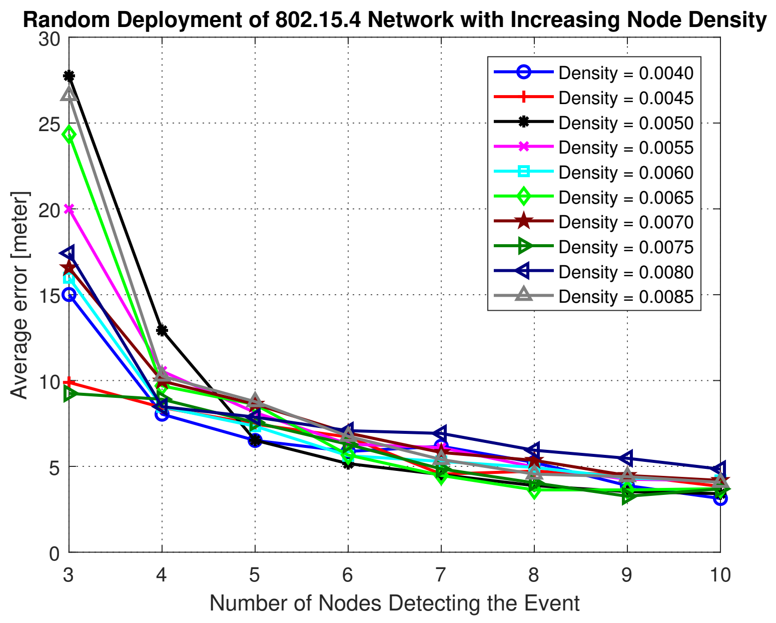


Figure 6.16: Error Estimation for Random Deployments

6.6 Summary

This chapter presented a performance analysis of the EERP as compared to BaseModel, SWIA and ERP. The analysis was conducted to observe the benefits of identifying the same events at the nodes to reduce transmission of unnecessary packets to the sink node.

The simulation results shown in this thesis were performed on Qualnet for varying node densities in a grid and random deployment. Seven different performance metrics were measured for each grid and random deployments. The first metric measures the number of unique events received at the sink node. The second metric establishes the benefits of our protocol in terms of minimizing the duplicate packets. These results confirmed that by identifying events at the node level significantly improves the overall event reliability in a WSN. The third and the fourth metrics are the transmission overhead and average battery consumed by the nodes, respectively. It was observed that energy consumption could be significantly reduced at the sensor nodes in a WSN. It was observed that the energy benefits in EERP could be achieved only if the nodes limit the communication of packets containing the same event information. The fifth metrics measure results provided the number of false positives and false negatives respectively. It was shown that EERP performs better in terms of false positive and false negative ratios amongst all the tested protocols. Measures specific to the event identification process used (i.e., multilateration technique) were presented last. The demonstrated the effectiveness of multilateration process when multiple nodes in vicinity are observing an event.

Overall, the findings from the experiments demonstrate that EERP achieves its objective (see Section 1.2) of utilizing sensor nodes' processing capabilities to improve event reliability in a WSN. The sensor nodes processed events at the node level and reduced the number of packets with redundant event information while maintaining event reliability. This resulted in the more reliable WSN with reduced energy consumption at the sensor nodes.

The next chapter concludes the research. It presents a summary of the thesis. The contributions and limitations of the research work are discussed, and future research directions are suggested.

Chapter 7

Conclusion and Future Work

This chapter provides the overall conclusion of the research work presented in this thesis. It will summarize the thesis, briefly discuss the achieved objectives, contributions and the future work.

7.1 Summary

Wireless Sensor Networks (WSNs) hold the promise of many new applications in the field of monitoring and control systems. With the advent of cheap and small sensors, monitoring systems can make use of them to monitor numerous characteristics of the environment (Chapter 2). All these applications, while built for a range of purposes, require energy efficiency and reliable transmission of data from source to destination. To address this issue, we surveyed the research and state-of-art in reliability schemes and protocols for WSNs (Chapter 3). These were analyzed based on our proposed 3D reference model that establishes a common taxonomy to evaluate research in WSN reliability. The model involves different combinations of either retransmission or redundancy techniques that aim to ensure event or packet level reliability by using either end-to-end (connection-oriented) or hop-by-hop (link-oriented) methods.

Retransmission and redundancy are classified as the two main approaches to achieve data transmission reliability in WSNs (Chapter 3). Retransmission is the traditional way of ensuring reliability and performs well in high loss environments at the cost of high communication overheads and memory occupancy. By comparison, redundancy-based mechanisms achieve reliability with reduced communication and memory requirements but require high processing overhead. However, the performance of redundancy-based reliability mechanisms degrades in high loss environments.

Packet or event-level reliability is concerned with how much sensed data is required by the WSN application to be delivered to the sink in order to achieve the required reliability. Event-level reliability, as opposed to packet level reliability, is sufficient for most event-driven WSNs, where a certain level of packet loss is acceptable as long as sufficient information about events is delivered to the sink. However, packet-level reliability is most commonly used for the WSN applications such as surveillance that require the delivery of all the sensed packets though this is at the cost of high energy consumption and possible network congestion.

Both retransmission and redundancy techniques aiming to achieve either packet or event level reliability perform better when using the hop-by-hop method as compared to the end-to-end method. The high hop count in large-scale WSNs introduces more opportunities for errors which becomes the cause of packet loss, thus affecting the reliability. This can be avoided to some extent by using a hop-by-hop method, as it ensures reliability at each intermediate hop from the source to the destination. However, this requires in-node processing and may increase the overall latency in reporting event information to the sink.

Accurately identifying the uniqueness of the event detected by the nodes is one of the most important characteristics of event-driven applications in WSNs (Chapter 4 and Chapter 5). The thesis introduced a distributed event identification mechanism in a novel context by using a multilateration technique. This allows estimating the location of the event to accurately identify if multiple packets contain the same event or not. Estimating event location plays a significant role in accurate event identification. This event localization can be performed with the help of geometric information about the nodes in the network that might have detected the event.

The thesis proposed the Event Reliability Protocol (ERP), an event-to-sink reliability protocol that serves to improve the scalability of event detection in a WSN by minimizing the unnecessary retransmission of data packets coming from multiple nodes in an event's locality. ERP takes advantage of the broadcast nature of the wireless channel by using an implicit acknowledgment (iACK) mechanism to maintain reliability. It builds on the spatial locality condition and employs a region-based selective retransmission mechanism to identify if two packets relate to the same event. ERP is the first protocol to introduce a distributed event reliability approach whereby, instead of relying on the sink, each node makes simple decisions using information readily available to it to maintain event reliability and does not incur additional network traffic.

This thesis also presented the Enhanced Event Reliability Protocol (EERP), that enables reliable transmission of event information to the sink node while minimizing redundant packets from nodes in the close vicinity of one another (Chapter 4 and Chapter 5). EERP builds on the spatio-temporal information of the originating nodes and employs an efficient combination of a greedy strategy with the use of a multilateration algorithm in a unique way to perform event identification before reliably transmitting the event information towards the sink node. Both ERP and EERP do away with the previous sink-centric approach used by other protocols by introducing in-network data processing, wherein ERP packets are processed during the retransmission decision process. In EERP, the packets are pre-processed at the nodes before forwarding them further. This in-node processing reduces the transmission of unnecessary packets and saves the overall network cost in terms of energy, congestion and data flow.

The performance of EERP was evaluated and compared against a Base

Model, ERP and a commonly used scheme (SWIA) (Chapter 6). Our results show that EERP significantly improves reliable delivery of unique event information from all over the network in an energy efficient manner.

7.2 Research Objectives Achieved

The thesis accomplished its objectives (Section 1.2) as discussed in detail below:

Objective 1: Design a methodology that can identify whether two or more packets are carrying information about the same events or not, based on spatio-temporal correlation phenomena by performing in-node data processing. The thesis presented new methods to identify packets containing duplicate event data, and a retransmission timeout mechanism to further improve the energy resource utilization. The methods used multilateration at the sensor node level to accurately identify the same events detected by multiple nodes in an event's locality. The thesis also provided event identification algorithms (Section 4.3.3) and their implementations (Section 5.3.2). The results show energy benefits for a sensor node in a WSN by efficiently performing distributed event identification based on spatio-temporal correlation phenomena using in-node data processing.

Objective 2: Design a mechanism that ensures the reliable delivery of information about the events' occurrence from the source to the sink node. The thesis designed and introduced hop-by-hop event reliability protocols that serve to improve the scalability of event detection in a WSN by minimizing the unnecessary retransmission and transmissions of data packets coming from multiple nodes in an event's locality. They take advantage of the broadcast nature of the wireless channel by using an implicit acknowledgments (iACKs) mechanism with region-based selective retransmission and multilateration-based forwarding of packets with unique events in a distributed fashion.

7.3 Contributions

This thesis explored aspects of performing event identification and reliable delivery of event information in WSNs. Research contributions have been made in multiple areas and are restated here:

3D-Reliability reference model: The first contribution is a novel 3D Reliability reference model that establishes a common taxonomy to evaluate the research in WSN reliability (Chapter 3). It provides a detailed analysis and comparison of state-of-the-art schemes and protocols that aim to achieve reliable transmission of data in WSNs based on the 3D-Reliability reference model. This model provides detailed guidelines and directions to the researchers for classifying the research in the area of reliability in WSNs, which will be used to perform in-depth analysis of the unexplored areas.

Region-based Selective Retransmission Mechanism: The second contribution is the idea of performing hop-by-hop event reliability and introduced a region-based selective retransmission mechanism based on the high level of spatial locality exhibited by sensor data (Chapter 4, 5 and 6). This mechanism can be beneficial when a retransmission decision is required whether to retransmit the unique events and introduced the concept of distributed event identification. It also exploits the broadcast nature of wireless channel by using iACKs and reduces the communication overhead caused by the transmission of other control packets like explicit and negative acknowledgments. This provides energy efficient and reliable delivery of event information and minimizes the overall network traffic by reducing the unnecessary retransmissions of redundant events.

Multilateration-based Distributed Event Reliability Mechanism: The t--hird contribution is a novel multilateration-based distributed event identification mechanism that uses a greedy strategy to allow energy efficient and reliable transmission of event information from all over the network to the sink (Chapter 4, 5 and 6). This approach performs event identification before transmitting packets to avoid the unnecessary transmission of the duplicate events. This significantly reduces the overall communication overhead and further improves the event reliability, energy efficiency and network lifetime.

An Event Generation Application for IEEE 802.15.4 networks: The four--th contribution is developing an event generation application in QualNet for IEEE 802.15.4 networks. Using this event generation application will help the research community in WSNs using the QualNet simulation environment to evaluate their proposed schemes more realistically (chapter 6). Unlike existing reliability schemes in WSNs, which assume that event generation has happened when a packet is transmitted, this application actually simulates the events all over the network that are sensed and transmitted by the in-range nodes. This makes the performance evaluation more realistic for event-driven WSN scenarios.

7.4 Future Work

This section briefly discusses the future work arising out of this research by identifying a number of open areas for extension and further investigation.

Dynamic Retransmission Threshold: Both ERP and EERP use retransmission techniques to maintain reliability. If a packet is lost, the nodes in the network are required to retransmit that packet to achieve reliability. In some situations, the retransmitted packet is also lost, so it needs to be retransmitted again. These continuous retransmissions make the network congested and negatively affect the reliable transmission and overall performance. To avoid such a situation, both ERP and EERP uses a static retransmission threshold that is set for all the sensor nodes in advance. Although static retransmission thresholds are commonly used in WSNs, they result in delayed delivery times.

Reducing these delayed delivery times requires a dynamic retransmission threshold mechanism that dynamically adjusts the retransmission threshold value based on the link quality and delay requirements of the underlying application.

Efficient Queue Management: Both ERP and EERP need to continuously drop packets from the queue to avoid irrelevant packets occupying the limited memory available for the queues. However, they need to keep sufficient history of sent and dropped packets to be able to identify unnecessary redundant packets correctly. The correct trade-off between these two factors is completely dependent on the type of application on how much and how frequently the data is required from the sensor nodes.

In WSN, the frequency of required data is usually known as sampling frequency (i.e. sensing frequency) requirements, where the application needs to identify that how frequently the data is needed to be sensed. A higher sampling frequency requires data to be sensed more frequently and thus needs to maintain more data storage. For example, a surveillance application would require the sensor to sense data every other second, so it fills up the storage very quickly; however, in environmental applications, the sampling frequency might be lower (such as in minutes). However, in these application areas, the data needs to be removed from the node regularly. This requires a sophisticated queue management technique that can identify and remove the garbage to avoid the unnecessary packets occupying the sensor's limited storage but retain all the data that may still be needed.

This will be a good addition to make both ERP and EERP work more efficiently for a variety of applications.

Adaptable Retransmission Timeout: When a node transmits a packet towards the sink, it waits for a period to detect the implicit acknowledgment of its sent packet by listening to the wireless channel. If the node does not detect the iACK, then the packet is considered lost and the node must make decisions to maintain reliability. Both ERP and EERP use a fixed time interval for the nodes to wait for an iACK. We chose this interval based on the time a node takes to receive the packet, perform required processing and transmit it to the next hop node towards the sink. However, this static timeout mechanism may affect latency and energy efficiency. In some situations, for example, heavy traffic where the nodes are busy, the timeout is too short and the node will make a false assumption that the packet is lost and retransmit it, adding to the traffic. On the other hand, there will be situations when the timeout is too long, the packet is already lost, but the node is still waiting for an acknowledgment using unnecessary energy and causing extra delay. Therefore, it is important to investigate an adaptable retransmission timeout mechanism that dynamically adjusts the timeout based on the estimated processing times of the nodes and network conditions.

Real World Deployment: In this thesis, we evaluated the effectiveness of the proposed schemes by implementing them in QualNet simulation environment. The simulation environment does not always reflect real-world scenarios precisely, as many real-world deployment issues such as degradation in performance of the nodes and anomalous sensor data are not considered. Therefore, to fully confirm the effectiveness of the proposed schemes, ERP and EERP would need to be examined using real-world testbeds.

7.5 Concluding Remarks

The key contributions of this research in the area of WSNs is enabled the sensor nodes to efficiently perform event identification and reliably deliver the event information towards the sink. The mechanisms were introduced to provide support for the nodes to identify if multiple packets have the same event or not and provide maximum information with minimum packets transmissions. This work is an initial step towards the realization of identifying events at the sensor nodes for energy and reliability benefits. Some steps for extending this work further have been suggested.

Appendix A

Event Generation Application

An event generation sensing application was developed for IEEE 802.15.4 networks. This Appendix explains the steps involved in the implementation of this event generation application in QualNet 5.2. Three major components that were developed in the QualNet are (i) Mote, (ii) Basestation, and (iii) Generator.

All the sensor nodes are motes and sensor nodes can either play the role of an originator or an intermediate node, where they maintain the transmission and forwarding of packets. The basestation component is meant to act as the sink node that implements the reception of packets. The Generator component simulates events that are being sensed at regular intervals and notifies all the in-range motes that the event has occurred. Motes send a UDP unicast packet to the basestation whenever a sensing event is detected. Motes are positioned in such a way that they are only in communication range of the mote next closest to the basestation based on the information provided by the underlying routing protocol, hence the requirement for a multi-hop network.

To elaborate the process let's take a simple example of a sensing application, where motes maintain a count of packets sent and the basestation maintains a count of packets received. A third component, the ″generator″, simulates sensing events every 10 seconds and the notifies all in-range motes that the event has occurred. The motes then each send a message to the basestation saying so. The payload of the messages is, for no particular reason, the node identifier and an indication of how much energy the mote has remaining in it's battery.

Having the generator as a non-functioning node in Qualnet is just a simple of way of hooking into the Qualnet event dispatch loop. So long as you keep it out of the way of the sensing nodes, it will not interfere with their communications and it should be fine. The following steps are needed to run the event generation application.

The following source code and configuration changes are needed for a very basic event generation application. The source code files and the QualNet configuration changes are for the simple application described above and needs to be modified according to the requirements.

A.1 Source Code for Even Generation Application

Following is the main source code developed for the basic event generation application.

```

emph# emphinclude "api.h"
emph# emphinclude "app_util.h"
emph# emphinclude "network_ip.h"
emph# emphinclude "partition.h"

emph# emphinclude "app_sensor.h"

emph# emphdefine SENSOR_GENERATOR_TIMER    1
emph# emphdefine MSG_SENSOR_SensingEvent 1001

/********************** Utility **********************/

```

```
static void sendUdpPacket(Node *node, Address sourceAddr,
    Address destAddr, AppType appType, int sourcePort, char *
    payload, int payloadLength) {
     APP_UdpSendNewDataWithPriority(
                node,
                appType, // (used as destination port)
                sourceAddr,
                (short) sourcePort,
                destAddr,
                0, // outgoingInterface,
                payload,
                payloadLength,
                0, // priority,
                0, // delay,
                TRACE_BROADCAST, // traceProtocol,
                FALSE, // isMdpEnabled,
                0 // uniqueId
        );
}

static void printApplicationStat(Node *node, char *buf, int
    index, const char *app) {
     IO_PrintStat(
                node,
                "Application",
                app,
                ANY_DEST,
                index,
                buf);
}

static void startTimer(Node *node, AppType appType, int
    timerType, clocktype startTime) {
     Message *timerMsg = MESSAGE_Alloc(node, APP_LAYER, appType,
    MSG_APP_TimerExpired);
     MESSAGE_InfoAlloc(node, timerMsg, sizeof (AppTimer));
     AppTimer *timer = (AppTimer *) MESSAGE_ReturnInfo(timerMsg);
     timer->type = timerType;
    MESSAGE_Send(node, timerMsg, startTime);
}

/********************* Mote *********************/

static void printMoteStat(Node *node, char *buf) {
    printApplicationStat(node, buf, -1, "SENSOR_MOTE");
}

static SensorMoteAppData *newSensorMote(Node *node) {
    SensorMoteAppData *smad =
            (SensorMoteAppData *) MEM_malloc(sizeof (
   SensorMoteAppData));
    memset(smad, 0, sizeof (SensorMoteAppData));
    APP_RegisterNewApp(node, APP_SENSOR_MOTE, smad);
    return smad;
}

static SensorMoteAppData *getSensorMote(Node *node) {
    AppInfo *appList = node->appData.appPtr;
    SensorMoteAppData *smad = NULL;
    for (; appList != NULL; appList = appList->appNext) {
        if (appList->appType == APP_SENSOR_MOTE) {
            smad = (SensorMoteAppData*) appList->appDetail;
            break;
        }
    }
    return smad;
}

static void sendMoteUdpPacket(Node *node, Address sourceAddr,
   Address destAddr, char *payload, int payloadLength) {
    sendUdpPacket(node, sourceAddr, destAddr,
   APP_SENSOR_BASESTATION, APP_SENSOR_MOTE, payload,
   payloadLength);
}

static void sendSensingNotification(Node *node) {
    SensorMoteAppData *smad = getSensorMote(node);

    char payload[32] = {0};
    sprintf(payload, "%d %f", node->nodeId,
   BatteryGetRemainingCharge(node));

    sendMoteUdpPacket(
            node,
            smad->localAddress,
            smad->remoteAddress,
            (char *) payload,
            32
            );

    smad->numPktsSent += 1;
}

void AppSensorMoteInit(Node *node, Address localAddress, Address
    remoteAddress) {
    SensorMoteAppData *smad = newSensorMote(node);
    smad->localAddress = localAddress;
    smad->remoteAddress = remoteAddress;
    smad->numPktsSent = 0;
}

void AppSensorMoteProcessEvent(Node *node, Message *msg) {
    switch (msg->eventType) {
        case MSG_SENSOR_SensingEvent:
        {
            sendSensingNotification(node);
            break;
        }
        default:
        {
            char buf[MAX_STRING_LENGTH];
            TIME_PrintClockInSecond(getSimTime(node), buf, node)
   ;
            printf("######## SensorMote: at time %sS, node %d "
                    "received message of unexpected type %d\n",
                    buf, node->nodeId, msg->eventType);
            }
        }

    MESSAGE_Free(node, msg);
}

void AppSensorMoteFinalize(Node *node) {
    SensorMoteAppData *smad = getSensorMote(node);
    char buf[MAX_STRING_LENGTH];
    sprintf(buf, "Packets Sent = %u", smad->numPktsSent);
    printMoteStat(node, buf);
}

/********************* Basestation ********************/

static void printBasestationStat(Node *node, char *buf) {
    printApplicationStat(node, buf, -1, "SENSOR_BASESTATION");
}

static SensorBasestationAppData *newSensorBasestation(Node *node
    ) {
    SensorBasestationAppData *sbad =
      (SensorBasestationAppData *) MEM_malloc(sizeof (
    SensorBasestationAppData));
    memset(sbad, 0, sizeof (SensorBasestationAppData));
    APP_RegisterNewApp(node, APP_SENSOR_BASESTATION, sbad);
    return sbad;
}

static SensorBasestationAppData *getSensorBasestation(Node *node
    ) {
    AppInfo *appList = node->appData.appPtr;
    SensorBasestationAppData *sbad = NULL;
    for (; appList != NULL; appList = appList->appNext) {
        if (appList->appType == APP_SENSOR_BASESTATION) {
            sbad = (SensorBasestationAppData*) appList->
    appDetail;
            break;
        }
    }
    return sbad;
}

void AppSensorBasestationInit(Node *node) {
    SensorBasestationAppData *sbad = newSensorBasestation(node);
    sbad->numPktsRcvd = 0;
}

void AppSensorBasestationProcessEvent(Node *node, Message *msg)
    {
    switch (msg->eventType) {
        case MSG_APP_FromTransport:
        {
            SensorBasestationAppData *sbad =
    getSensorBasestation(node);
            sbad->numPktsRcvd += 1;
            break;
        }
    }

    MESSAGE_Free(node, msg);
}

void AppSensorBasestationFinalize(Node *node) {
    SensorBasestationAppData *sbad = getSensorBasestation(node);
    char buf[MAX_STRING_LENGTH];
    sprintf(buf, "Packets Received = %u", sbad->numPktsRcvd);
    printBasestationStat(node, buf);
}

/********************* Generator *********************/

static void printGeneratorStat(Node *node, char *buf) {
    printApplicationStat(node, buf, -1, "SENSOR_GENERATOR");
}

static void startGeneratorTimer(Node *node, int type, clocktype startTime) {
    startTimer(node, APP_SENSOR_GENERATOR, type, startTime);
}

static SensorGeneratorAppData *newSensorGenerator(Node *node) {
    SensorGeneratorAppData *sgad =
      (SensorGeneratorAppData *) MEM_malloc(sizeof (SensorGeneratorAppData));
    memset(sgad, 0, sizeof (SensorGeneratorAppData));
    APP_RegisterNewApp(node, APP_SENSOR_GENERATOR, sgad);
    return sgad;
}

static SensorGeneratorAppData *getSensorGenerator(Node *node) {
    AppInfo *appList = node->appData.appPtr;
    SensorGeneratorAppData *sgad = NULL;
    for (; appList != NULL; appList = appList->appNext) {
        if (appList->appType == APP_SENSOR_GENERATOR) {
            sgad = (SensorGeneratorAppData*) appList->appDetail;
            break;
        }
    }
    return sgad;
}

static void notifySensor(Node *node) {
    Message *msg = MESSAGE_Alloc(node, APP_LAYER, APP_SENSOR_MOTE, MSG_SENSOR_SensingEvent);
    MESSAGE_Send(node, msg, 0);
}

static void notifySensors(Node *node) {
    Node *n;
    for (unsigned i = 2; i < node->nodeId; i++) {
            n = MAPPING_GetNodePtrFromHash(node->partitionData->
    nodeIdHash, i);
            notifySensor(n);
        }
    SensorGeneratorAppData *sgad = getSensorGenerator(node);
  sgad->numEventsGenerated += 1;
}

void AppSensorGeneratorInit(Node *node) {
    SensorGeneratorAppData *sgad = newSensorGenerator(node);
    sgad->numEventsGenerated = 0;
    startGeneratorTimer(node, SENSOR_GENERATOR_TIMER,
    TIME_ConvertToClock("10S"));
}

void AppSensorGeneratorProcessEvent(Node *node, Message *msg) {
    switch (msg->eventType) {
        case MSG_APP_TimerExpired:
        {
            AppTimer *timer = (AppTimer *) MESSAGE_ReturnInfo(
    msg);

            if (timer->type == SENSOR_GENERATOR_TIMER) {
                notifySensors(node);
                startGeneratorTimer(node, SENSOR_GENERATOR_TIMER
    , TIME_ConvertToClock("10S"));
            }
            break;
        }
    }

    MESSAGE_Free(node, msg);
}

void AppSensorGeneratorFinalize(Node *node) {
    SensorGeneratorAppData *sgad = getSensorGenerator(node);
    char buf[MAX_STRING_LENGTH];
    sprintf(buf, "Events Generated = %u", sgad->
   numEventsGenerated);
    printGeneratorStat(node, buf);
}
```

A.2 Header File for Even Generation Application

Following is the basic header file for the simple event generation application.

```

emph# emphifndef APP_SENSOR_H
emph# emphdefine APP_SENSOR_H

typedef
struct struct_sensor_mote_app_data
{
    Address  localAddress;
    Address  remoteAddress;
    UInt32   numPktsSent;
}
SensorMoteAppData;

typedef
struct struct_sensor_basestation_app_data
{
    UInt32   numPktsRcvd;
}
SensorBasestationAppData;

typedef
struct struct_sensor_generator_app_data
{
    UInt32   numEventsGenerated;
}
SensorGeneratorAppData;

```

```
void AppSensorMoteInit(Node *node, Address localAddress, Address
    remoteAddress);
void AppSensorMoteProcessEvent(Node *node, Message *msg);
void AppSensorMoteFinalize(Node *node);

void AppSensorBasestationInit(Node *node);
void AppSensorBasestationProcessEvent(Node *node, Message *msg);
void AppSensorBasestationFinalize(Node *node);

void AppSensorGeneratorInit(Node *node);
void AppSensorGeneratorProcessEvent(Node *node, Message *msg);
void AppSensorGeneratorFinalize(Node *node);

emph# emphendif
```

A.3 Modifications in QualNet's Configuration Code for Event Generation Application

Once we have the source files, then we need to make changes inside QualNet's configuration files to make this event generation application a part of QualNet Simulator. In this way, other researchers using QualNet to evaluate their proposed work in event-driven wireless sensor networks will be able to use this event generation application to simulate close to real-time event generation mechanism. At the end of the following steps, one needs build QualNets as usual and the application will start working.

Header and Source Files

Copy the header and source files to

`$QUALNET_HOME/libraries/user_models/src$`

Edit Makefile

Edit the following makefile as shown:

`$QUALNET_HOME/libraries/user_models/src/Makefile-common`

```

USER_MODELS_SRCS = \
  $(USER_MODELS_DIR)/app_sensor.cpp \
  $(USER_MODELS_DIR)/app_hello.cpp
    $...
```

Edit Trace File

Edit the following file as shown:

`$QUALNET_HOME/include/trace.h`

```

    TRACE_PHY_LTE,

    TRACE_SENSOR,

    // Must be last one!!!
    TRACE_ANY_PROTOCOL
};
```

Edit Application Header File

Edit the following file as shown:

`$QUALNET_HOME/include/application.h`

```

    APP_HELLO,

    APP_SENSOR_MOTE,
    APP_SENSOR_BASESTATION,
    APP_SENSOR_GENERATOR,

    APP_PLACEHOLDER
};
```

Edit Application Source File

Edit the following file as shown:

`$QUALNET_HOME/main/application.cpp`

There are four blocks that need editing.

```

/*******************************/
/*********** Block 1 ***********/
/*******************************/

emph# emphifdef USER_MODELS_LIB
emph# emphinclude "app_hello.h"
emph# emphinclude "app_sensor.h"
emph# emphendif /* USER_MODELS_LIB */

/*******************************/
/*********** Block 2 ***********/
/*******************************/

      case APP_HELLO:
      {
        AppHelloProcessEvent(node, msg);
        break;
      }
        case APP_SENSOR_MOTE:
      {
        AppSensorMoteProcessEvent(node, msg);
        break;
      }
      case APP_SENSOR_BASESTATION:
      {
        AppSensorBasestationProcessEvent(node, msg);
        break;
      }
      case APP_SENSOR_GENERATOR:
      {
        AppSensorGeneratorProcessEvent(node, msg);
        break;
      }

/*********************************/
/************ Block 3 ************/
/*********************************/

emph# emphifdef DEBUG
        printf("Parsing for application type %s\n", appStr);
emph# emphendif /* DEBUG */

        if (strcmp(appStr, "HELLO") == 0)
        {
            char sourceString[MAX_STRING_LENGTH];
            NodeAddress sourceNodeId;
            Address sourceAddress;
          numValues = sscanf(appInput.inputStrings[i], "%*s %s",
    sourceString);
          IO_AppParseSourceString(
              firstNode,
                  appInput.inputStrings[i],
                  sourceString,
                  &sourceNodeId,
                  &sourceAddress);
            node = MAPPING_GetNodePtrFromHash(nodeHash,
   sourceNodeId);
          AppHelloInit(node, nodeInput);
        }

        else if (strcmp(appStr, "SENSOR_MOTE") == 0) {
            char sourceString[MAX_STRING_LENGTH];
            char destString[MAX_STRING_LENGTH];
            NodeAddress sourceNodeId;
            Address sourceAddr;
            NodeAddress destNodeId;
            Address destAddr;

            numValues = sscanf(appInput.inputStrings[i],
                            "%*s %s %s",
                            sourceString,
                            destString);

            if (numValues != 2)
            {
                char errorString[MAX_STRING_LENGTH];
                sprintf(errorString,
                        "Wrong SENSOR_MOTE configuration format !\n"
                        "SENSOR_MOTE   \n");
                ERROR_ReportError(errorString);
            }

            IO_AppParseSourceAndDestStrings(
                firstNode,
                appInput.inputStrings[i],
                sourceString,
                &sourceNodeId,
                &sourceAddr,
                destString,
                &destNodeId,
                &destAddr);

            node = MAPPING_GetNodePtrFromHash(nodeHash, sourceNodeId);
            AppSensorMoteInit(node, sourceAddr, destAddr);
        }

        else if (strcmp(appStr, "SENSOR_BASESTATION") == 0) {
            char sourceString[MAX_STRING_LENGTH];
            NodeAddress sourceNodeId;
            Address sourceAddress;
          numValues = sscanf(appInput.inputStrings[i], "%*s %s",
    sourceString);
          IO_AppParseSourceString(
              firstNode,
                  appInput.inputStrings[i],
                  sourceString,
                  &sourceNodeId,
                  &sourceAddress);
            node = MAPPING_GetNodePtrFromHash(nodeHash,
   sourceNodeId);
            AppSensorBasestationInit(node);
        }

        else if (strcmp(appStr, "SENSOR_GENERATOR") == 0) {
            char sourceString[MAX_STRING_LENGTH];
            NodeAddress sourceNodeId;
            Address sourceAddress;
          numValues = sscanf(appInput.inputStrings[i], "%*s %s",
    sourceString);
          IO_AppParseSourceString(
              firstNode,
                  appInput.inputStrings[i],
                  sourceString,
                  &sourceNodeId,
                  &sourceAddress);
            node = MAPPING_GetNodePtrFromHash(nodeHash,
   sourceNodeId);
            AppSensorGeneratorInit(node);
        }


/********************************/
/************ Block 4 ************/
/********************************/

```

```
                case APP_HELLO:
                {
                    AppHelloFinalize(node);
                    break;
                }
                case APP_SENSOR_MOTE:
                {
                    AppSensorMoteFinalize(node);
                    break;
                }
                case APP_SENSOR_BASESTATION:
                {
                    AppSensorBasestationFinalize(node);
                    break;
                }
                case APP_SENSOR_GENERATOR:
                {
                    AppSensorGeneratorFinalize(node);
                    break;
                }

};
```

Appendix B

Random Deployments

B.1 QualNet: Randomly Deploying 802.15.4 Motes to Form a Connected Network with 100% Coverage

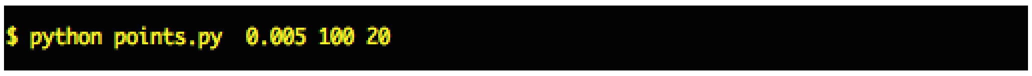
In a sensor network where each node has an identical radio range, the network is guaranteed to be connected if every point within the sensing area is no more than half that common radio range from at least one node. To put it another way, if a network is proven to provide 100% sensing coverage for a uniform sensing range R, then the network will be connected if the common transmission range is at least 2R, as shown by Xing et al. [138]. The network may be connected if the transmission range is less than 2R, but that's a lot harder to prove.

The python script in Listing B.1 first generates a uniformly distributed network of nodes based a Poisson Point Process configured with the parameters you supply. The script then checks to see if the generated network provides 100% coverage. If it does, the script prints the node locations in QualNet format for you to include in your simulations. I would like to acknowledge that these node deployment scripts were developed with one of my ex-colleague, David Harrison.

Following parameters are needed to to execute the scripts:

- Node density (float) i.e., Number of nodes per square meter
- Size of the network area in meters (integer)
- The sensing range of the nodes in meters (integer)

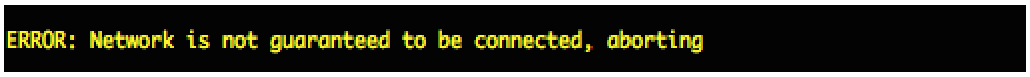
For example (as shown in Figure B.1):



```
$ python points.py  0.005 100 20
```

Figure B.1: Parameters for running the python script

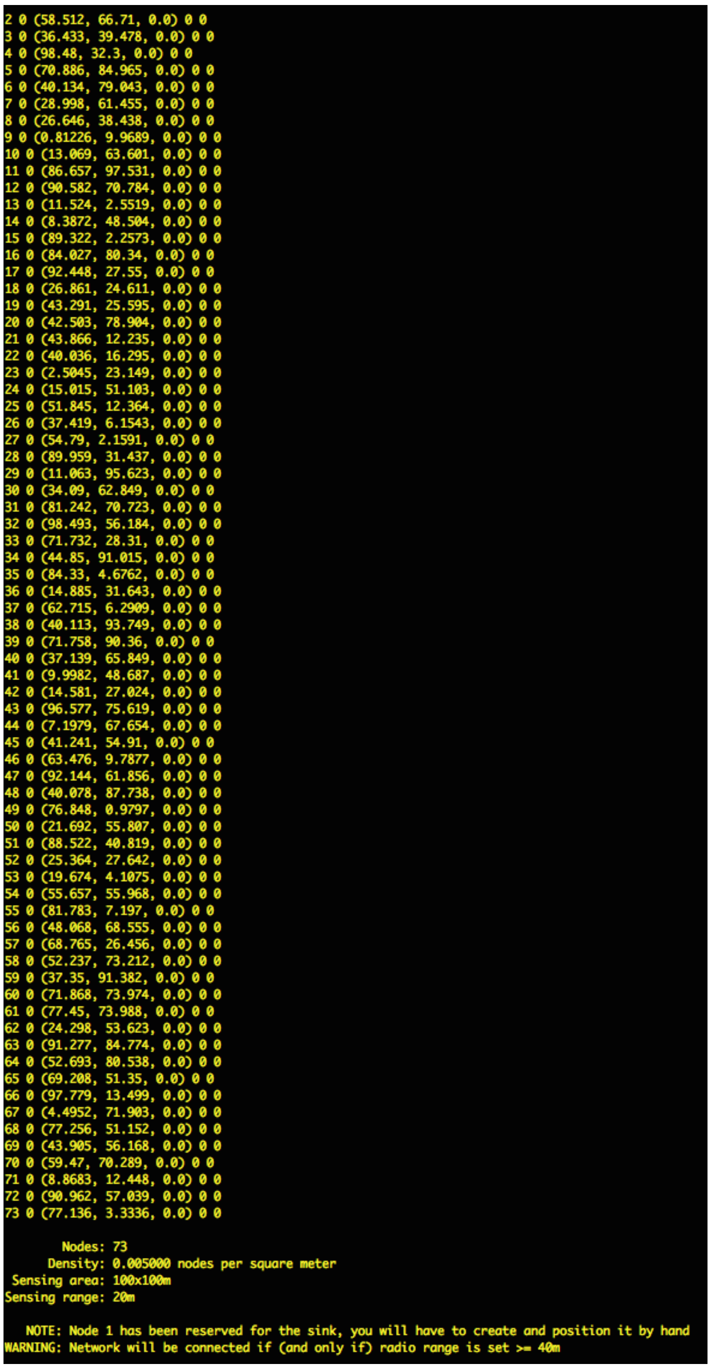
If those parameters did not create a potentially connected network, the script will say so (as shown in Figure B.2).



```
ERROR: Network is not guaranteed to be connected, aborting
```

Figure B.2: Error Message

If a potentially connected network is generated, the script will print the node locations in Qualnet format and make a few pertinent observations (see Figure B.3).



```
2 0 (58.512, 66.71, 0.0) 0 0
3 0 (36.433, 39.478, 0.0) 0 0
4 0 (98.48, 32.3, 0.0) 0 0
5 0 (70.886, 84.965, 0.0) 0 0
6 0 (40.134, 79.043, 0.0) 0 0
7 0 (28.998, 61.455, 0.0) 0 0
8 0 (26.646, 38.438, 0.0) 0 0
9 0 (0.81226, 9.9689, 0.0) 0 0
10 0 (13.069, 63.601, 0.0) 0 0
11 0 (86.657, 97.531, 0.0) 0 0
12 0 (90.582, 70.784, 0.0) 0 0
13 0 (11.524, 2.5519, 0.0) 0 0
14 0 (8.3872, 48.504, 0.0) 0 0
15 0 (89.322, 2.2573, 0.0) 0 0
16 0 (84.027, 80.34, 0.0) 0 0
17 0 (92.448, 27.55, 0.0) 0 0
18 0 (26.861, 24.611, 0.0) 0 0
19 0 (43.291, 25.595, 0.0) 0 0
20 0 (42.503, 78.904, 0.0) 0 0
21 0 (43.866, 12.235, 0.0) 0 0
22 0 (40.036, 16.295, 0.0) 0 0
23 0 (2.5045, 23.149, 0.0) 0 0
24 0 (15.015, 51.103, 0.0) 0 0
25 0 (51.845, 12.364, 0.0) 0 0
26 0 (37.419, 6.1543, 0.0) 0 0
27 0 (54.79, 2.1591, 0.0) 0 0
28 0 (89.959, 31.437, 0.0) 0 0
29 0 (11.063, 95.623, 0.0) 0 0
30 0 (34.09, 62.849, 0.0) 0 0
31 0 (81.242, 70.723, 0.0) 0 0
32 0 (98.493, 56.184, 0.0) 0 0
33 0 (71.732, 28.31, 0.0) 0 0
34 0 (44.85, 91.015, 0.0) 0 0
35 0 (84.33, 4.6762, 0.0) 0 0
36 0 (14.885, 31.643, 0.0) 0 0
37 0 (62.715, 6.2909, 0.0) 0 0
38 0 (40.113, 93.749, 0.0) 0 0
39 0 (71.758, 90.36, 0.0) 0 0
40 0 (37.139, 65.849, 0.0) 0 0
41 0 (9.9982, 48.687, 0.0) 0 0
42 0 (14.581, 27.024, 0.0) 0 0
43 0 (96.577, 75.619, 0.0) 0 0
44 0 (7.1979, 67.654, 0.0) 0 0
45 0 (41.241, 54.91, 0.0) 0 0
46 0 (63.476, 9.7877, 0.0) 0 0
47 0 (92.144, 61.856, 0.0) 0 0
48 0 (40.078, 87.738, 0.0) 0 0
49 0 (76.848, 0.9797, 0.0) 0 0
50 0 (21.692, 55.807, 0.0) 0 0
51 0 (88.522, 40.819, 0.0) 0 0
52 0 (25.364, 27.642, 0.0) 0 0
53 0 (19.674, 4.1075, 0.0) 0 0
54 0 (55.657, 55.968, 0.0) 0 0
55 0 (81.783, 7.197, 0.0) 0 0
56 0 (48.068, 68.555, 0.0) 0 0
57 0 (68.765, 26.456, 0.0) 0 0
58 0 (52.237, 73.212, 0.0) 0 0
59 0 (37.35, 91.382, 0.0) 0 0
60 0 (71.868, 73.974, 0.0) 0 0
61 0 (77.45, 73.988, 0.0) 0 0
62 0 (24.298, 53.623, 0.0) 0 0
63 0 (91.277, 84.774, 0.0) 0 0
64 0 (52.693, 80.538, 0.0) 0 0
65 0 (69.208, 51.35, 0.0) 0 0
66 0 (97.779, 13.499, 0.0) 0 0
67 0 (4.4952, 71.903, 0.0) 0 0
68 0 (77.256, 51.152, 0.0) 0 0
69 0 (43.905, 56.168, 0.0) 0 0
70 0 (59.47, 70.289, 0.0) 0 0
71 0 (8.8683, 12.448, 0.0) 0 0
72 0 (90.962, 57.039, 0.0) 0 0
73 0 (77.136, 3.3336, 0.0) 0 0

        Nodes: 73
      Density: 0.005000 nodes per square meter
 Sensing area: 100x100m
Sensing range: 20m

   NOTE: Node 1 has been reserved for the sink, you will have to create and position it by hand
WARNING: Network will be connected if (and only if) radio range is set >= 40m
```

Figure B.3: Node locations

Following the Python code for generating the above mentioned deployments:

```
import scipy.stats
import numpy
import sys
import math

if len(sys.argv) != 4:
  print('Usage: python %s <density (nodes per square meter)> <size (meters)> <
    sensing range (meters)>' \
    % sys.argv[0])
  sys.exit(0)

density = float(sys.argv[1])
L = int(sys.argv[2])
r = int(sys.argv[3])
N = scipy.stats.poisson(density*L*L).rvs()

points = numpy.random.uniform(low = 0, high = L, size = (N,2))

def distance(p1, p2):
  dx = p1[0] - p2[0];
  dy = p1[1] - p2[1];
  return math.sqrt(dx * dx + dy * dy)

x = 0.5
origin = [0,0]
while x < L:
  y = 0.5
  while y < L:
    p = [x,y]
    found = 0
    for point in points:
      if distance(p,point) < r:
        found = 1
        break
    if found < 1:
            print 'ERROR: Network is not guaranteed to be connected, aborting'
            sys.exit(0)
        y += 1
    x += 1

node = 2
for point in points:
    print '{:d} 0 ({:.5}, {:.5}, 0.0) 0 0'.format(node, point[0], point[1])
    node += 1

node -= 1
print ''
print '      Nodes: %d' % node
print '    Density: %f nodes per square meter' % density
print ' Sensing area: %dx%dm' % (L,L)
print 'Sensing range: %dm' % r
print ''
print '   NOTE: Node 1 has been reserved for the sink, you will have to create and position it by hand'
print 'WARNING: Network will be connected if (and only if) radio range is set >= %dm' % (r*2)
print ''
```

Listing B.1: Python script to generate a connected network of 802.15.4 nodes for Qualnet

B.2 QualNet: Generate Multiple 802.15.4 Connected Networks with 100% Coverage

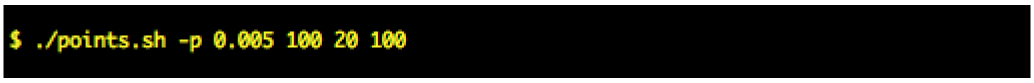
If you have used python script in Listing B.1 to generate connected networks and want to generate multiple deployments, then follow the steps below.

Parameters you need to supply are:

- Whether you want the PDFs generated (optional, default is to NOT produce them)

- Node density (float) i.e. number of nodes per square meter
- Size of the network area in meters (integer)
- The sensing range of the nodes in meters (integer)
- How many networks you want generating (integer)

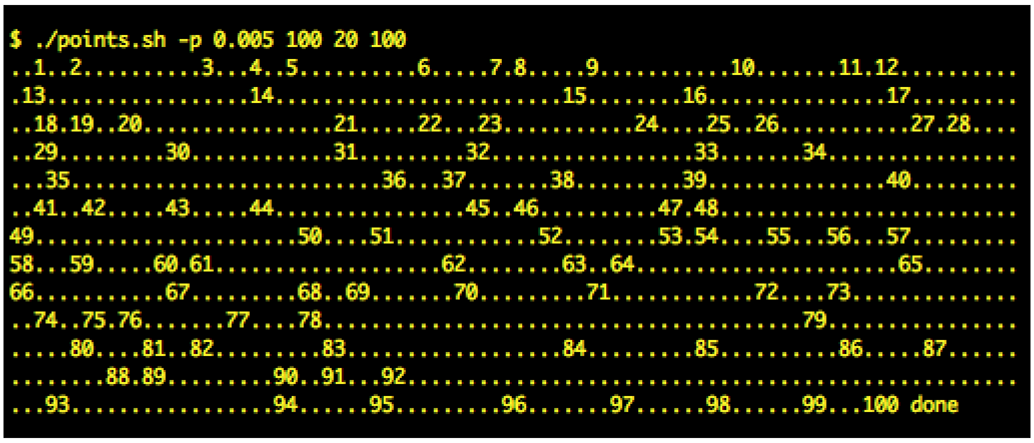
For example to run the script, you will need to provide the command as shown in Figure B.4.



```
$ ./points.sh -p 0.005 100 20 100
```

Figure B.4: Parameters for running the python script

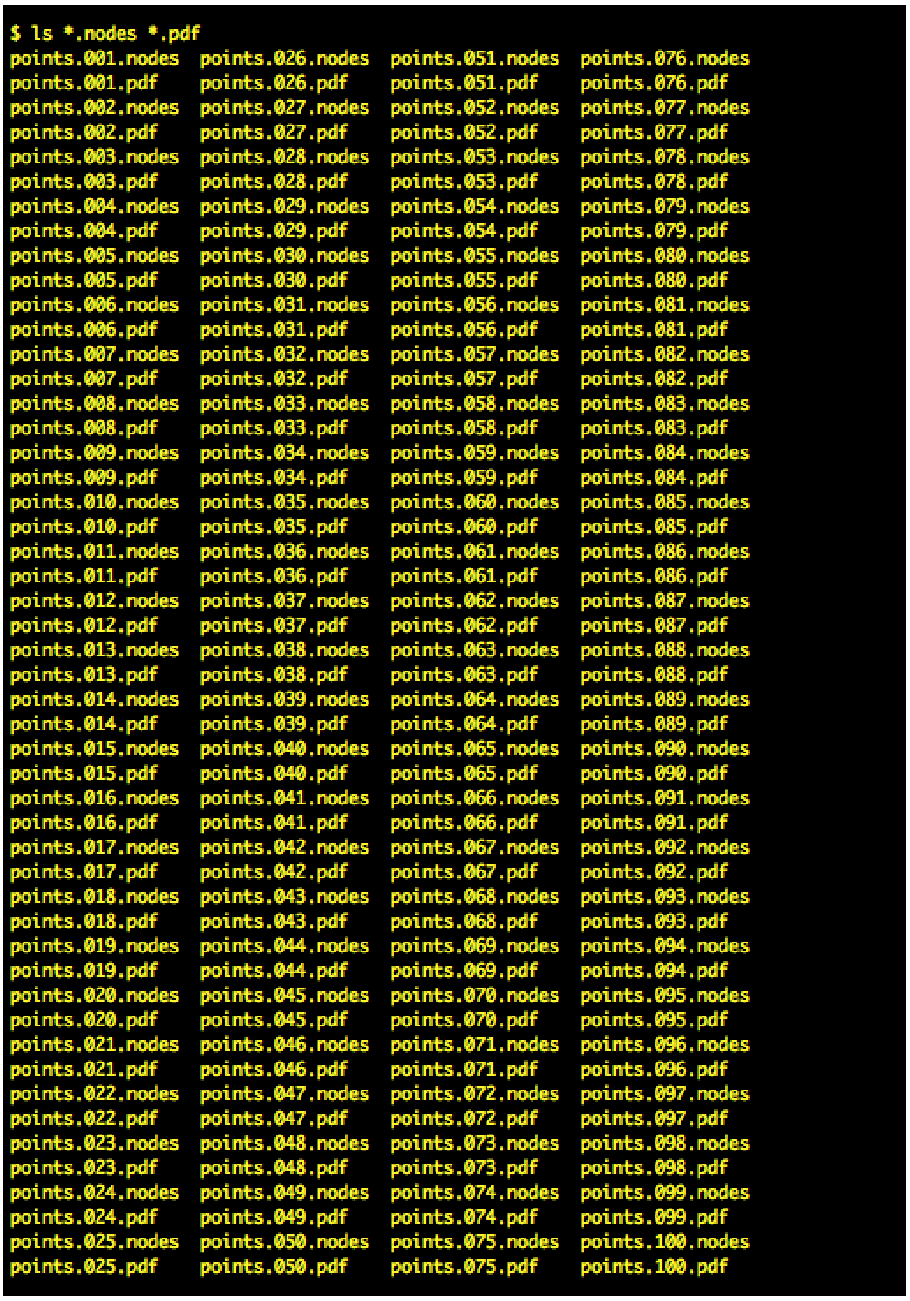
Which will produce output as shown in Figure B.5, but remember, this is a (pseudo) random process, so your output will almost certainly be different:



```
$ ./points.sh -p 0.005 100 20 100
..1..2..........3...4..5..........6.....7.8.....9...........10.......11.12..........
.13.................14........................15........16...............17.........
..18.19..20................21.....22...23...........24....25..26...........27.28....
..29.........30............31.........32.................33.......34................
...35..........................36...37.......38.........39...............40.........
..41..42.....43.....44................45..46..........47.48.........................
49......................50....51............52........53.54....55...56...57.........
58...59.....60.61...................62........63..64......................65........
66...........67.........68..69.......70.........71............72....73..............
..74..75.76.......77....78........................................79................
.....80....81..82.........83..................84.........85..........86.....87......
........88.89.........90..91...92...................................................
...93.................94......95.........96.......97......98......99...100 done
```

Figure B.5: Output as shown on the screen

Once the script has completed (see Figure B.6), you'll have (in this case) 100 QualNet node files and 100 PDFs:



```
$ ls *.nodes *.pdf
points.001.nodes  points.026.nodes  points.051.nodes  points.076.nodes
points.001.pdf    points.026.pdf    points.051.pdf    points.076.pdf
points.002.nodes  points.027.nodes  points.052.nodes  points.077.nodes
points.002.pdf    points.027.pdf    points.052.pdf    points.077.pdf
points.003.nodes  points.028.nodes  points.053.nodes  points.078.nodes
points.003.pdf    points.028.pdf    points.053.pdf    points.078.pdf
points.004.nodes  points.029.nodes  points.054.nodes  points.079.nodes
points.004.pdf    points.029.pdf    points.054.pdf    points.079.pdf
points.005.nodes  points.030.nodes  points.055.nodes  points.080.nodes
points.005.pdf    points.030.pdf    points.055.pdf    points.080.pdf
points.006.nodes  points.031.nodes  points.056.nodes  points.081.nodes
points.006.pdf    points.031.pdf    points.056.pdf    points.081.pdf
points.007.nodes  points.032.nodes  points.057.nodes  points.082.nodes
points.007.pdf    points.032.pdf    points.057.pdf    points.082.pdf
points.008.nodes  points.033.nodes  points.058.nodes  points.083.nodes
points.008.pdf    points.033.pdf    points.058.pdf    points.083.pdf
points.009.nodes  points.034.nodes  points.059.nodes  points.084.nodes
points.009.pdf    points.034.pdf    points.059.pdf    points.084.pdf
points.010.nodes  points.035.nodes  points.060.nodes  points.085.nodes
points.010.pdf    points.035.pdf    points.060.pdf    points.085.pdf
points.011.nodes  points.036.nodes  points.061.nodes  points.086.nodes
points.011.pdf    points.036.pdf    points.061.pdf    points.086.pdf
points.012.nodes  points.037.nodes  points.062.nodes  points.087.nodes
points.012.pdf    points.037.pdf    points.062.pdf    points.087.pdf
points.013.nodes  points.038.nodes  points.063.nodes  points.088.nodes
points.013.pdf    points.038.pdf    points.063.pdf    points.088.pdf
points.014.nodes  points.039.nodes  points.064.nodes  points.089.nodes
points.014.pdf    points.039.pdf    points.064.pdf    points.089.pdf
points.015.nodes  points.040.nodes  points.065.nodes  points.090.nodes
points.015.pdf    points.040.pdf    points.065.pdf    points.090.pdf
points.016.nodes  points.041.nodes  points.066.nodes  points.091.nodes
points.016.pdf    points.041.pdf    points.066.pdf    points.091.pdf
points.017.nodes  points.042.nodes  points.067.nodes  points.092.nodes
points.017.pdf    points.042.pdf    points.067.pdf    points.092.pdf
points.018.nodes  points.043.nodes  points.068.nodes  points.093.nodes
points.018.pdf    points.043.pdf    points.068.pdf    points.093.pdf
points.019.nodes  points.044.nodes  points.069.nodes  points.094.nodes
points.019.pdf    points.044.pdf    points.069.pdf    points.094.pdf
points.020.nodes  points.045.nodes  points.070.nodes  points.095.nodes
points.020.pdf    points.045.pdf    points.070.pdf    points.095.pdf
points.021.nodes  points.046.nodes  points.071.nodes  points.096.nodes
points.021.pdf    points.046.pdf    points.071.pdf    points.096.pdf
points.022.nodes  points.047.nodes  points.072.nodes  points.097.nodes
points.022.pdf    points.047.pdf    points.072.pdf    points.097.pdf
points.023.nodes  points.048.nodes  points.073.nodes  points.098.nodes
points.023.pdf    points.048.pdf    points.073.pdf    points.098.pdf
points.024.nodes  points.049.nodes  points.074.nodes  points.099.nodes
points.024.pdf    points.049.pdf    points.074.pdf    points.099.pdf
points.025.nodes  points.050.nodes  points.075.nodes  points.100.nodes
points.025.pdf    points.050.pdf    points.075.pdf    points.100.pdf
```

Figure B.6: Generated QualNet node files and PDFs

Following is the bash script to generate a set of connected networks of 802.15.4 nodes for Qualnet

```
USAGE="Usage: $0 [-p] <density> <size> <range> <count>"

if [ $# -lt 1 ]
then
   echo "$USAGE"
   exit 1
fi

plot=false
if [ $1 == "-p" ] || [ $1 == "--plot" ]
then
  plot=true
  shift
fi

if [ $# -lt 4 ]
then
   echo $USAGE
   exit 1
fi

density=$1
size=$2
range=$3
count=$4
found=0

while true
do
  python points.py $density $size $range > points.nodes
  l=$(wc -l points.nodes | awk '{print $1}')
  if (( $l > 1 ))
  then
      found=$((found+1))
     set `printf "%03d" $found`
```

```
      ext=$1
    cp points.nodes points.$ext.nodes
    echo -n ".$found"
    if [ "$plot" = true ]
      then
     gnuplot -e "size=$size" -e "range=$range" points.dem
     cp points.pdf points.$ext.pdf
      fi
    if [ $found -ge $count ]
    then
     echo " done"
     break
    fi
  else
    echo -n "."
  fi
done

rm points.nodes points.dat points.pdf
```

Listing B.2: Shell script (points.sh)

Also, if you want to generate the PDFs, you need to save the gnuplot command file shown in Listing B.3 into the same directory as the python and shell scripts.

```
reset
set term pdf size 5,5 font "Times-Roman, 16"
set output "points.pdf"
set size square
set xrange [-range:size+range]
set yrange [size+range:-range]
set object 1 rect from 0.0,0.0 to size,size \
  fc rgb "black" fs transparent solid 0.1 border rgb "black"

unset border
unset xtics
unset ytics

set lmargin 0
set rmargin 0
set tmargin 0
set bmargin 0

system("cat points.nodes | sed 's/(//' | sed 's/,//g' | awk '{print $1, $3, $4}' >
    points.dat")

set style fill transparent solid 0.1
plot "points.dat" using 2:3:(range) title '' with circles
```

Listing B.3: Gnuplot Command File (points.dem)

Here's an example of a generated PDF (Figure B.7):

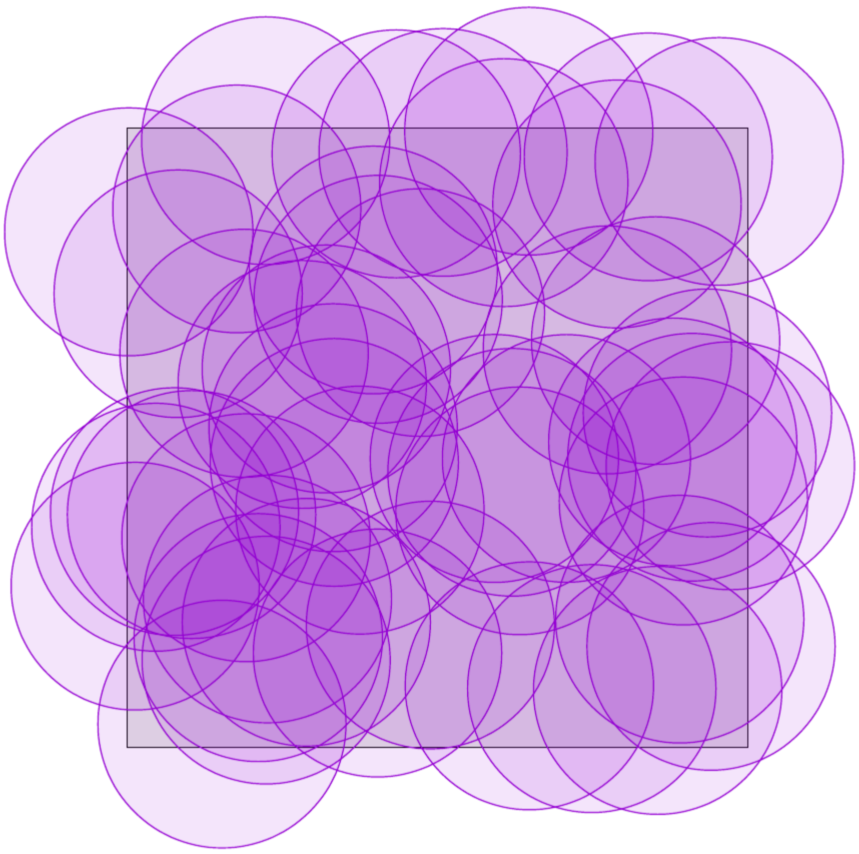


Figure B.7: Example of a generated PDF

Appendix C

Grid Deployments

C.1 QualNet : Generate 802.15.4 Grid Network with 100% Coverage and Connectivity

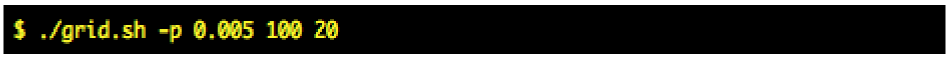
The python scripts (shown in Listing B.1 and Listing B.2, respectively) were developed to generate randomly distributed connected networks, the next step is to generate uniformly distributed grid deployments of 802.15.4 network for QualNet simulations.

The shell script and python script (shown in Listing C.1 and Listing C.2, respectively). If you want to generate the PDFs, you need to save the gnuplot command file (as shown in Listing C.3) into the same directory as the python and shell scripts.

Parameters you need to supply are:

- If the PDFs are needed (optional, default is to NOT produce them)
- Node density (float) i.e. Number nodes per square meter
- Size of the network area in meters (integer)
- The sensing range of the nodes in meters (integer)

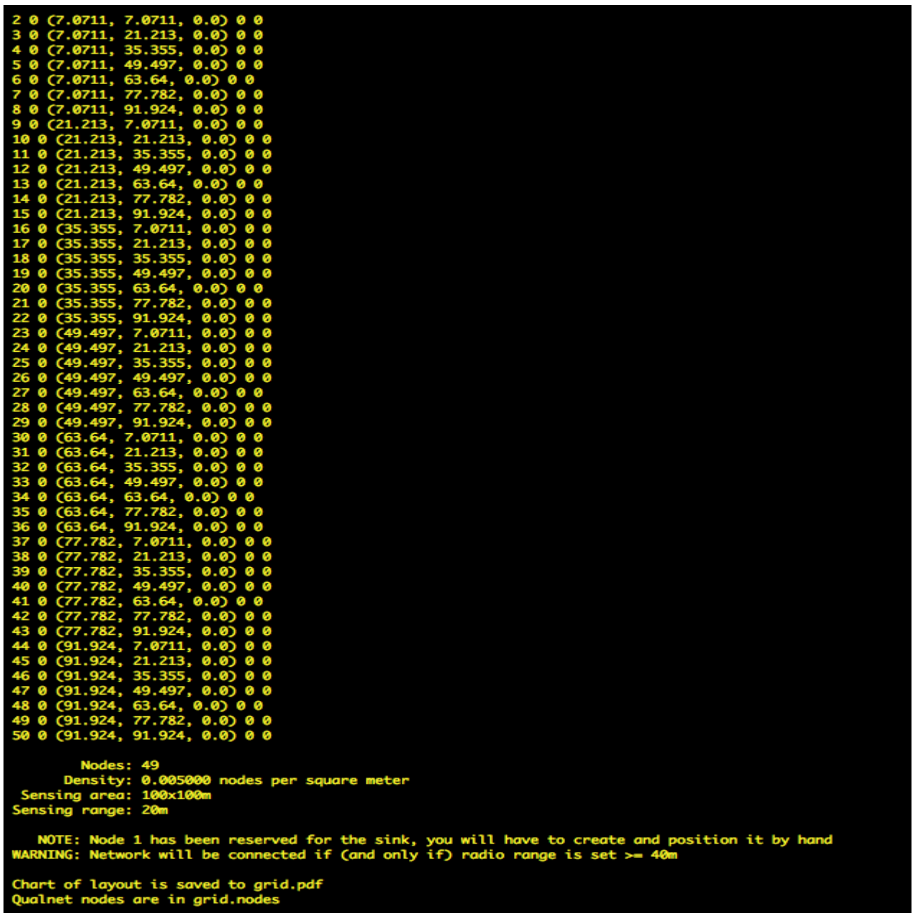
For example (as shown in Figure C.1):



```
$ ./grid.sh -p 0.005 100 20
```

Figure C.1: Parameters for running the python script

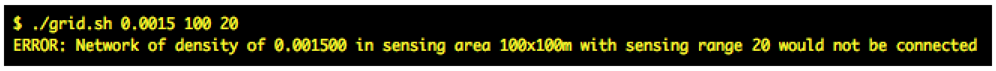
Which will produce output along these lines as shown in Figure C.2.



```
2 0 (7.0711, 7.0711, 0.0) 0 0
3 0 (7.0711, 21.213, 0.0) 0 0
4 0 (7.0711, 35.355, 0.0) 0 0
5 0 (7.0711, 49.497, 0.0) 0 0
6 0 (7.0711, 63.64, 0.0) 0 0
7 0 (7.0711, 77.782, 0.0) 0 0
8 0 (7.0711, 91.924, 0.0) 0 0
9 0 (21.213, 7.0711, 0.0) 0 0
10 0 (21.213, 21.213, 0.0) 0 0
11 0 (21.213, 35.355, 0.0) 0 0
12 0 (21.213, 49.497, 0.0) 0 0
13 0 (21.213, 63.64, 0.0) 0 0
14 0 (21.213, 77.782, 0.0) 0 0
15 0 (21.213, 91.924, 0.0) 0 0
16 0 (35.355, 7.0711, 0.0) 0 0
17 0 (35.355, 21.213, 0.0) 0 0
18 0 (35.355, 35.355, 0.0) 0 0
19 0 (35.355, 49.497, 0.0) 0 0
20 0 (35.355, 63.64, 0.0) 0 0
21 0 (35.355, 77.782, 0.0) 0 0
22 0 (35.355, 91.924, 0.0) 0 0
23 0 (49.497, 7.0711, 0.0) 0 0
24 0 (49.497, 21.213, 0.0) 0 0
25 0 (49.497, 35.355, 0.0) 0 0
26 0 (49.497, 49.497, 0.0) 0 0
27 0 (49.497, 63.64, 0.0) 0 0
28 0 (49.497, 77.782, 0.0) 0 0
29 0 (49.497, 91.924, 0.0) 0 0
30 0 (63.64, 7.0711, 0.0) 0 0
31 0 (63.64, 21.213, 0.0) 0 0
32 0 (63.64, 35.355, 0.0) 0 0
33 0 (63.64, 49.497, 0.0) 0 0
34 0 (63.64, 63.64, 0.0) 0 0
35 0 (63.64, 77.782, 0.0) 0 0
36 0 (63.64, 91.924, 0.0) 0 0
37 0 (77.782, 7.0711, 0.0) 0 0
38 0 (77.782, 21.213, 0.0) 0 0
39 0 (77.782, 35.355, 0.0) 0 0
40 0 (77.782, 49.497, 0.0) 0 0
41 0 (77.782, 63.64, 0.0) 0 0
42 0 (77.782, 77.782, 0.0) 0 0
43 0 (77.782, 91.924, 0.0) 0 0
44 0 (91.924, 7.0711, 0.0) 0 0
45 0 (91.924, 21.213, 0.0) 0 0
46 0 (91.924, 35.355, 0.0) 0 0
47 0 (91.924, 49.497, 0.0) 0 0
48 0 (91.924, 63.64, 0.0) 0 0
49 0 (91.924, 77.782, 0.0) 0 0
50 0 (91.924, 91.924, 0.0) 0 0

        Nodes: 49
      Density: 0.005000 nodes per square meter
 Sensing area: 100x100m
Sensing range: 20m

   NOTE: Node 1 has been reserved for the sink, you will have to create and position it by hand
WARNING: Network will be connected if (and only if) radio range is set >= 40m

Chart of layout is saved to grid.pdf
Qualnet nodes are in grid.nodes
```

Figure C.2: Output as shown on the screen

If the parameters are supplied that would leave the network uncovered, the script will tell you so (as shown in Figure C.3).



```
$ ./grid.sh 0.0015 100 20
ERROR: Network of density of 0.001500 in sensing area 100x100m with sensing range 20 would not be connected
```

Figure C.3: Error Message

Here's an example of a generated PDFs (see Figure C.4).

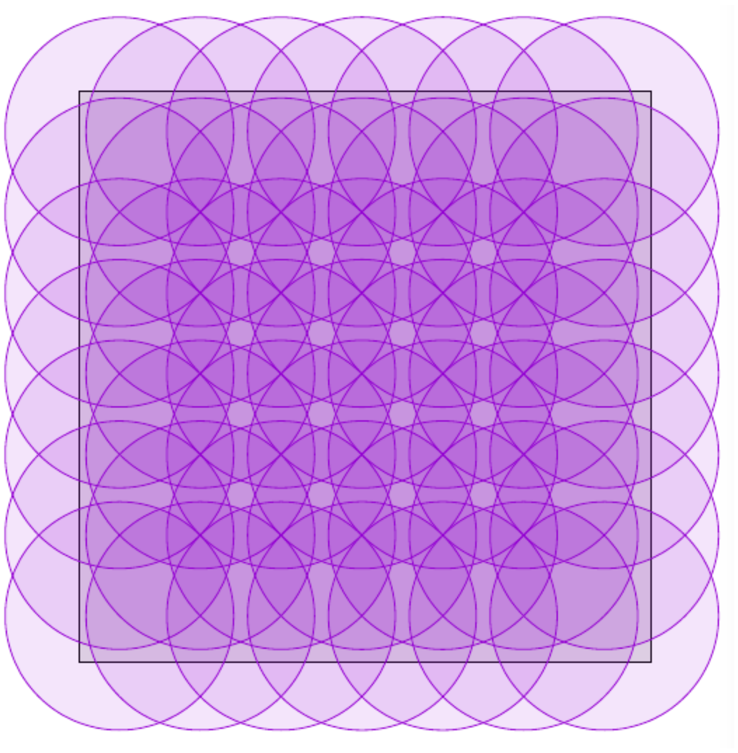


Figure C.4: Node locations

Following is the bash script to generate a set of connected networks of 802.15.4 nodes for Qualnet.

```
USAGE="Usage: $0 [-p] <density> <size> <range>"

if [ $# -lt 1 ]
then
    echo "$USAGE"
    exit 1
fi

plot=false
if [ $1 == "-p" ] || [ $1 == "--plot" ]
then
  plot=true
  shift
fi

if [ $# -lt 3 ]
then
    echo $USAGE
    exit 1
fi

density=$1
size=$2
range=$3

python grid.py $density $size $range | tee grid.nodes
l=$(wc -l grid.nodes | awk '{print $1}')
if (( $l > 1 ))
then
    if [ "$plot" = true ]
    then
    cp grid.nodes points.nodes
    gnuplot -e "size=$size" -e "range=$range" points.dem
    mv points.pdf grid.pdf
    rm points.nodes
    echo "Chart of layout is saved to grid.pdf"
    echo "Qualnet nodes are in grid.nodes"
    echo ""
  fi
fi
```

Listing C.1: Grid Shell script (grid.sh)

Following the Python code for generating the grid based deployment for a network of 802.15.4 nodes for QualNet:

```
import sys
import math

if len(sys.argv) != 4:
    print('Usage: python %s <density (nodes per square meter)> <size (meters)> <
        sensing range (meters)>' \
        % sys.argv[0])
    sys.exit(0)

density = float(sys.argv[1])
L = int(sys.argv[2])
r = int(sys.argv[3])

nodes = int(L*L*density)

X = L/math.sqrt(nodes)
Y = X

#print 'nodes = %d, X = %f' % (nodes,X)

if X > r:
    print 'ERROR: Network of density of %f in sensing area %dx%dm with sensing
        range %d would not be connected' \
    % (density, L, L, r)
    sys.exit(0)

node = 2
```

```
row = 0
x = X/2
origin = [0,0]
while x < L:
    col = 0
    y = Y/2
    while y < L:
        print ’{:d} 0 ({:.5}, {:.5}, 0.0) 0 0’.format(node, x, y)
        y += Y
        col += 1
        node += 1
    x += X
    row += 1

node −= 2
print ’’
print ’        Nodes: %d’ % node
print ’      Density: %f nodes per square meter’ % density
print ’ Sensing area: %dx%dm’ % (L,L)
print ’Sensing range: %dm’ % r
print ’’
print ’   NOTE: Node 1 has been reserved for the sink, you will have to create and
        position it by hand’
print ’WARNING: Network will be connected if (and only if) radio range is set >= %
        dm’ % (r∗2)
print ’’
```

Listing C.2: Python script to generate a connected grid network of 802.15.4 nodes for Qualnet (grid.py)

```
reset
set term pdf size 5,5 font ”Times−Roman, 16”
set output ”points.pdf”
set size square
set xrange [−range:size+range]
set yrange [size+range:−range]
set object 1 rect from 0.0,0.0 to size,size \
  fc rgb ”black” fs transparent solid 0.1 border rgb ”black”
```

```

unset border
unset xtics
unset ytics

set lmargin 0
set rmargin 0
set tmargin 0
set bmargin 0

system("cat points.nodes | sed 's/(//' | sed 's/,//g' | awk '{print $1, $3, $4}' >
    points.dat")

set style fill transparent solid 0.1
plot "points.dat" using 2:3:(range) title '' with circles
```

Listing C.3: Gnuplot Command File (points.dem)

C.2 QualNet: Generate Multiple 802.15.4 Connected Networks with 100% Coverage

For generating multiple connected networks along with their pdf layout please follow the instructions given in Listing B.2.

Bibliography

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393-422, 2002.

[2] J. Yick, B. Mukherjee and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292-2330, 2008.

[3] S. Nikoletseas, "Models and Algorithms for Wireless Sensor Networks (Smart Dust)," *invited paper in the Proceedings of the 32nd SOFSEM Conference on Current Trends in Theory and Practice of Computer Science*, ser. Lecture Notes in Computer Science (LNCS), Eds. Springer Verlag, vol. 3831, pp. 64-83, 2006.

[4] T. He, S. Krishnamurthy, J. A.Stankovic, T. F. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui and B. Krogh, "An energy-efficient surveillance system using wireless sensor networks," in *Proceedings of the 2nd international conference on Mobile Systems, Applications, and Services* (MobiSys), Boston, Massachusetts, USA, pp. 270–283, 6-9 June 2004.

[5] S. Nikoletseas and J. D. P. Rolim, *Theoretical Aspects of Distributed Computing in Sensor Networks,* Springer Verlag, 2011.

[6] A. Boukerche and S. Nikoletseas, "Protocols for Data Propagation in Wireless Sensor Networks: A Survey," in *Handbook of Wireless Communi-*

cations Systems and Networks, Mohsen Guizani (Ed), Kluwer Academic Publishers, pp. 23–51, 2004.

[7] L. Krishnamachari, D. Estrin and S. Wicker, "The impact of data aggregation in wireless sensor networks," in *Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops* (ICDCSW), Vienna, Austria, pp. 575–578, 23-29 April 2006.

[8] K. A. Bispo, L. H. A. De Freitas, N. S. Rosa and P. R. F. Cunha, "A Reconfiguration Approach Using Ontologies to Save Energy on WSNs," in *Proceedings of the Third International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies* (UBICOMM), Sliema, Malta, pp. 182–187, 11-16 October 2009.

[9] A. Silberstein, R. Braynard, G. Filpus, G. Puggioni, A. Gelfand, K. Munagala, J. Yang, B. Kamachari, D. Estrin and S. Wicker, "Data Driven Processing in Sensor Networks," in *Proceedings of the Third Biennial Conference on Innovative Data Systems Research* (CIDR), Asilomar, California, USA, pp. 10–21, 7-10 January 2007.

[10] Y. Chen, H. V. Leong, M. Xu, J. Cao, K. Chan and A. Chan, "In-Network Data Processing for Wireless Sensor Networks," in *Proceedings of the 7th International Conference on Mobile Data Management* (MDM), Nara, Japan, pp. 26, 10-12 May 2006.

[11] X. Tang and J. Xu "Extending Network Lifetime for Precision-Constrained Data Aggregation in Wireless Sensor Networks," in *Proceedings of the 25th IEEE International Conference on Computer Communications* (INFOCOM), Barcelona, Catalunya, Spain, pp. 01–12, 23-29 April 2006.

[12] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson and D. Culler, "An analysis of a large scale habitat monitoring application," in *Pro-*

ceedings of the 2nd international conference on Embedded Networked Sensor Systems (SenSys), Baltimore, Maryland, pp. 214–226, 2-5 July 2002.

[13] M. A. Mahmood, W. K. G. Seah and I. Welch, "Reliability in Wireless Sensor Networks: Survey and Challenges Ahead," *Technical Report ECSTR12-13*, School of Engineering & Computer Science, Victoria University of Wellington, New Zealand, 6 April 2012.

[14] M. A. Mahmood, W. K. G. Seah and I. Welch, "Reliability in wireless sensor networks: A survey and challenges ahead," in *Computer Networks*, vol. 79, pp. 166-187, 2015, `https://doi.org/10.1016/j.comnet.2014.12.016`.

[15] M. C. Vuran, O. B. Akan and I. F. Akyildiz, "Spatio-temporal correlation: theory and applications for wireless sensor networks," in *Computer Networks*, vol. 45, no. 03, pp. 245–259, 2004.

[16] M. K. Kasi and A. M. Hinze, "Cost analysis for complex in-network event processing in heterogeneous wireless sensor networks," in *Proceedings of the 5th ACM International Conference on Distributed Event-Based Systems* (DEBS), New York, USA, pp. 385-386, 11-14 Jul 2011.

[17] Q. Wang and W. Yang, "Energy Consumption Model for Power Management in Wireless Sensor Networks," *Proceedings of the 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks* (SECON), San Diego, California, USA, 18-21 June 2007.

[18] M. C. Vuran, O. B. Akan and I. F. Akyildiz, "Spatio-temporal correlation: theory and applications for wireless sensor networks," in *Computer Networks*, vol. 45, no. 03, pp. 245–259, 2004.

[19] L. A. Villas, A. Boukerche, D. L. Guidoni, H. A. B. F. de Oliveira, R. B. de Araujo and A. A. F. Loureiro, "An energy-aware spatio-temporal correlation mechanism to perform efficient data collection

in wireless sensor networks," in *Computer Communications*, 2012, doi:10.1016/j.comcom.2012.04.007.

[20] Y. Zhang, N. Meratnia and P. Havinga, "Outlier Detection Techniques for Wireless Sensor Networks: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 12, no. 2, pp. 159-170, 2010.

[21] D. K. Y. Yau, N. K. Yip, C. Y. T. Ma, N. S. Rao and M. Shankar, "Quality of monitoring of stochastic events by periodic & proportional-share scheduling of sensor coverage," in *Proceedings of the 4th International ACM Conference on emerging Networking EXperiments and Technologies* (CoNEXT), Madrid, Spain, pp. 1-12, 9-12 December 2008.

[22] P. Rawat, K. D. Singh, H. Chaouchi and J. M. Bonnin, "Wireless sensor networks: a survey on recent developments and potential synergies," in *The Journal of Supercomputing*, vol. 68, no. 01, 2014.

[23] D. Uttamchandani, *Wireless MEMS Networks and Applications*, Woodhead Publishing, 2017.

[24] M. F. Othman and K. Shazali, "Wireless Sensor Network Applications: A Study in Environment Monitoring System," in *Elsevier Journal of Procedia Engineering*, vol. 41, pp. 1204-1210, 2012.

[25] J. K. Hart and K. Martinez, "Environmental Sensor Networks: A revolution in the earth system science," in *Earth-Science Reviews*, vol. 78, no. 03-04, pp. 177–191, 2006.

[26] K. Martinez, R. Ong and J. Hart, "Glacsweb: a sensor network for hostile environments," in *Proceedings of the First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks* (SECON), Santa Clara, CA, USA, pp. 81-87, 2004.

[27] J. Burrell, T. Brooke and R. Beckwith, "Vineyard computing: Sensor networks in agricultural production," in *IEEE Pervasive Computing*, vol. 03, no. 01, pp. 3845, 2004.

[28] G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz and J. Lees, "Deploying a wireless sensor network on an active volcano," in *IEEE Internet Computing*, vol. 10, no. 02, pp. 1825, 2006.

[29] B. O. Flyrm, R. Martinez, J. Cleary, C. Slater, F. Regan, D. Diamond and H. Murphy, "Smartcoast: A wireless sensor network for water quality monitoring," in *Proceedings of the 32nd IEEE Conference on Local Computer Networks* (LCN), pp. 815816, 2007.

[30] M. P. Durisic, Z. Tafa, G. Dimic and V. Milutinovic, "A survey of military applications of wireless sensor networks," in *Proceedings of the Mediterranean Conference on Embedded Computing* (MECO), pp. 196-199, 2012.

[31] T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic, T F. Abdelzaher, J. Hui and B. Krogh, "Vigilnet: An integrated sensor network system for energy-efficient surveillance," in *ACM Transactions on Sensor Networks*, vol. 02, no. 01, pp. 1-38, 2006.

[32] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora and M. Miyashita, "A line in the sand: a wireless sensor network for target detection, classification, and tracking," in *Computer Networks*, vol. 46, no. 05, pp. 605634, 2004.

[33] H. Alemdar, C. Ersoy, "Wireless sensor networks for healthcare: A survey," in *Computer Networks*, vol. 54, no. 15, pp. 2688-2710, 2010.

[34] D. Malan, T. Fulford-Jones, M. Welsh and S. Moulton, "Codeblue: An ad hoc sensor network infrastructure for emergency medical care," in *Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks* (WIBSN), 2004.

[35] S. Movassaghi, M. Abolhasan, J. Lipman, D. Smith and A. Jamalipour, “Wireless body area networks: A survey,” in *IEEE Communications Surveys Tutorials*, vol. 16, no. 03, pp. 16581686, 2014.

[36] B. Moulton, J. Chen, G. Croucher, S. Lal, E. Lawrence, L. Mahendran and A. Varis, “Ambulatory health monitoring and remote sensing systems to be used by outpatients and elders at home: User-related design considerations,” in *Proceedings of the 11th International Conference on e-Health Networking, Applications and Services* (Healthcom), pp. 4853, December 2009.

[37] V. Shnayder, B. Chen, K. Lorincz, T. R. F. F. Jones and M. Welsh, “Sensor networks for medical care,” in *Proceedings of the 3rd international conference on Embedded networked sensor systems* (SenSys), New York, NY, USA, pp. 314314, 2005.

[38] P. Corke, T. Wark, R. Jurdak, W. Hu, P. Valencia and D. Moore, “Environmental Wireless Sensor Networks,” in *IEEE Proceedings*, vol. 98, no. 11, pp. 1903-1917, 2010.

[39] X. Xiong, M. Zhang, B. Gao and Z. Wang, “Study of the communication module of wireless sensor networks node platform,” in *Proceedings of the IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems* (CYBER) Kunming, China 2011, pp. 226-229, 20-23 March 2011.

[40] M. A. Rahman, A. E. Saddik and W. Gueaieb, “Wireless sensor network transport layer: State of the art,” in *Sensors,* ser. Lecture Notes in Electrical Engineering, S. Mukhopadhyay and R. Huang, Eds. Springer Berlin Heidelberg, vol. 21, pp. 221-245, 2008.

[41] C. Wang, K. Sohraby, B. Li, M. Daneshmand and Y. Hu, “A survey of transport protocols for wireless sensor networks,” *IEEE Network*, vol. 20, no. 03, pp. 34 – 40, 2006.

[42] C. Wang, K. Sohraby, Y. Hu, B. Li and W. Tang, "Issues of transport control protocols for wireless sensor networks," in *Proceedings of the International Conference on Communications, Circuits and Systems* (ICC-CAS), Hong Kong, China, pp. 422–426, 27-30 May 2005.

[43] K. Sohraby, D. Minoli and T. Znati, "Transport Control Protocols for Wireless Sensor Networks" in *Wireless Sensor Networks Technology, Protocols, and Applications*, John Wiley & Sons, Inc., 2007.

[44] A. J. D. Rathnayaka, V. M. Potdar, A. Sharif, S. Sarencheh and S. Kuruppu, "Wireless Sensor Networks: Challenges Ahead," in *Proceedings of the fifth International Conference on Broadband, Wireless Computing, Communication and Applications* (BWCCA), Fukuoka, Japan, pp. 824–829, 4-6 November 2010.

[45] C. K. Rupani and T. C. Aseri, "An improved transport layer protocol for wireless sensor networks," *Computer Communications*, vol. 34, no. 06, pp. 758 – 764, 2011.

[46] L. Liang, X. Yang, L. Zhang, D. Gao and H. Zhang, "Issues for event monitoring in event-driven wireless sensor networks," in *Proceedings of 7th International Conference on Wireless Communications, Networking and Mobile Computing* (WiCOM), Wuhan, China, pp. 1–4, 23-25 September 2011.

[47] L. Buttyan and L. Csik, "Security analysis of reliable transport layer protocols for wireless sensor networks," in *Proceedings of 8th IEEE International Conference on Pervasive Computing and Communications Workshops* (PERCOM Workshops), Mannheim, Germany, pp. 419–424, 29 March - 2 April 2010.

[48] F. Yunus, N. Ismail, S. Ariffin, A. Shahidan, N. Fisal and S. Syed-Yusof, "Proposed transport protocol for reliable data transfer in wireless sensor network (WSN)," in *Proceedings of the 4th International Conference*

on Modeling, Simulation and Applied Optimization (ICMSAO), Kuala Lumpur, Malaysia, pp. 1 –7, 19-21 April 2011.

[49] S. Sridevi and M. Usha, "Taxonomy of transport protocols for Wireless Sensor Networks," in *Proceedings of the International Conference on Recent Trends in Information Technology* (ICRTIT), Chennai, India, pp. .467–472, 3-5 June 2011.

[50] G. Anastasi, M. Conti, M. D. Francesco and A. Passarella, "Energy conservation in wireless sensor networks: A survey," in *Ad Hoc Networks*, vol. 07, no. 03, pp 537568, 2009.

[51] R. Gonzalez and M. Acosta, "Evaluating the impact of acknowledgment strategies on message delivery rate in wireless sensor networks," in *Proceedings of the IEEE Latin-American Conference on Communications* (LATINCOM), Bogota, Colombia, pp. 1 –6, 15-17 September 2010.

[52] S. -J. Park, R. Vedantham, R. Sivakumar, and I. Akyildiz , "A scalable approach for reliable downstream data delivery in wireless sensor networks," in *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing* (MobiHoc), Tokyo, Japan, pp. 78–89, 24-26 May 2004.

[53] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, Prentice Hall, 1995.

[54] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," *ACM Computer Communication Review*, vol. 27, no. 02, pp. 24 – 36, 1997.

[55] J. Postel, "Transmission Control Protocol," *IETF RFC 793*, September 1981.

[56] C.-Y. Wan, A. Campbell and L. Krishnamurthy, "Pump-slowly, fetch-quickly (PSFQ): a reliable transport protocol for sensor networks,"

IEEE Journal on Selected Areas in Communications, vol. 23, no. 04, pp. 862 – 872, 2005.

[57] S.-J. Park, R. Sivakumar, I. Akyildiz and R. Vedantham, "GARUDA: Achieving effective reliability for downstream communication in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 07, no. 02, pp. 214 –230, 2008.

[58] P. Levis, N. Patel, D. Culler and S. Shenker, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks," in *Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation* (NSDI), San Francisco, California, USA, pp. 15–28, 29-31 March 2004.

[59] R. Vedantham, R. Sivakumar and S.-J. Park, "Sink-to-sensors congestion control," in *Proceedings of the IEEE International Conference on Communications* (ICC), Seoul, Korea, pp. 3211–3217, 16-20 May 2005.

[60] S. Talik, N. B. Abu-Ghazaleh and W. Heinzelman, "Infrastructure tradeoffs for sensor networks," in *Proceedings of the 1st ACM international workshop on Wireless Sensor Networks and Applications* (WSNA), Atlanta, Georgia, USA, pp. 49–58, 28 September 2002.

[61] C. Cimen, E. Cayirci and V. Coskun, "Querying sensor fields by using quadtree based dynamic clusters and task sets," in *Proceedings of the IEEE Military Communications Conference* (MILCOM), Boston, Massachusetts, USA, pp. 561–566, 13-16 October 2003.

[62] Y. Sankarasubramaniam, O. B. Akan and I. F. Akyildiz, "ESRT: event-to-sink reliable transport in wireless sensor networks," in *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing* (MobiHoc), Annapolis, Maryland, USA, pp. 177–188, 1-3 June 2003,

[63] O. B. Akan and I. F. Akyildiz, “ESRT: Event-to-sink reliable transport in wireless sensor networks,” *IEEE/ACM Transactions on Networking*, vol. 13, no. 05, pp. 1003–1016, 2005.

[64] V. C. Gungor and O. B. Akan “DST: delay sensitive transport in wireless sensor networks,” in *Proceedings of the 7th International Symposium on Computer Networks* (ISCN), Istanbul, Turkey, pp. 116-122, 16-18 June 2006.

[65] Y. Iyer, S. Gandham and S. Venkatesan, “STCP: a generic transport layer protocol for wireless sensor networks,” in *Proceedings of 14th International Conference on Computer Communications and Networks* (IC-CCN), San Diego, California, USA, pp. 449 – 454, 17-19 October 2005.

[66] Y. Zhou, M. Lyu, J. Liu and H. Wang, “PORT: a price-oriented reliable transport protocol for wireless sensor networks,” in *Proceedings of 16th IEEE International Symposium on Software Reliability Engineering* (ISSRE), Chicago, Illinois, USA, pp. 10 pp. –126, 8-11 November 2005.

[67] N. Tezcan and W. Wang, “ART: an asymmetric and reliable transport mechanism for wireless sensor networks,” *International Journal of Sensor Networks*, vol. 02, no. 3-4, pp. 188–200, 2007.

[68] Y. Xue, B. Ramamurthy and Y. Wang, “Providing Reliable Data Transport for Dynamic Event Sensing in Wireless Sensor Networks,” in *Proceedings of IEEE International Conference on Communications* (ICC), Beijing, China, pp. 3146–3150, 19-23 May 2008.

[69] Y. Xue, B. Ramamurthy and Y. Wang, “LTRES: A loss-tolerant reliable event sensing protocol for wireless sensor networks,” *Computer Communications*, vol. 32, no. 15, pp. 1666 – 1676, 2009.

[70] H. Park, J. Lee, S. Oh, Y. Yim, S. Kim and K. Nam, “QERP: Quality-based event reliability protocol in wireless sensor networks,” in *Pro-*

ceedings of IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, Nevada, USA, pp. 730–734, 9-12 January 2011.

[71] M. A. Mahmood and W. K. G. Seah, "Event Reliability in Wireless Sensor Networks," in *Proceedings of the seventh International Conference on Intelligent Sensors, Sensor Networks and Information Processing* (ISSNIP), Adelaide, Australia, pp. 01– 06, 6-9 December 2011.

[72] M. Maróti, "Directed flood-routing framework for wireless sensor networks," in *Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware* (Middleware), Toronto, Canada, pp. 99–114, 18-22 October 2004.

[73] H. Zhang, A. Arora, Y.-ri. Choi and M. G. Gouda, "RBC: Reliable bursty convergecast in wireless sensor networks," in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing* (MobiHoc), Urbana-Champaign, Illinois, USA, pp. 266–276, 25-28 May 2005.

[74] H. Zhang, A. Arora, Y-ri. Choi and M. G. Gouda, "RBC: Reliable bursty convergecast in wireless sensor networks," *Computer Communications*, vol. 30, no. 13, pp. 2560-2576, 2007.

[75] T. Le, W. Hu, P. Corke and S. Jha, "ERTP: Energy-efficient and reliable transport protocol for data streaming in wireless sensor networks," *Computer Communications*, vol. 32, no. 7-10, pp. 1154 – 1171, 2009.

[76] F. Shaikh, A. Khelil, A. Ali and V. Suri, "Trccit: Tunable reliability with congestion control for information transport in wireless sensor networks," in *Proceedings of the 5th Annual International ICST Wireless Internet Conference* (WICON), Singapore, pp. 1 – 9, 1-3 March 2010.

[77] H. Zhou, X. Guan and C. Wu, "RTMC: Reliable transport with memory consideration in wireless sensor networks," in *Proceedings of the*

IEEE International Conference on Communications (ICC), Beijing, China, pp. 2819 –2824, 19-23 May 2008.

[78] M. A. Mahmood, P. Andreae and I. Welch, "Enhanced Event Reliability in Wireless Sensor Networks," in *Proceedings of the 32nd IEEE International Conference on Advanced Information Networking and Applications* (IEEE AINA-2018), Krakow, Poland, 16-18 May 2018.

[79] F. Stann and J. Heidemann, "RMST: reliable data transport in sensor networks," in *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications* (SNPA), Anchorage, Alaska, USA, pp. 102 – 112, 11 May 2003.

[80] Z. Rosberg, R. Liu, A. Dong, L. Tuan and S. Jha, "ARQ with implicit and explicit acks in wireless sensor networks," in *Proceedings of the IEEE Global Telecommunications Conference* (GLOBECOM), New Orleans, Loaisiana, USA, pp. 1 –6, 30 November - 4 December 2008.

[81] K. -K. Yap and M. Motani, "An opportunity to acknowledge in wireless multi-hop networks," technical report, Stanford University, National University of Singapore, July 2007.

[82] G.-W. Lee and E.-N. Huh, "Reliable data transfer using overhearing for implicit ack," in *Proceedings of the ICCAS-SICE International Joint Conference* (ICCAS-SICE), Fukuoka, Japan, pp. 1976 –1979, 18-21 August 2009.

[83] A. Dunkels, J. Alonso, T. Voigt and H. Ritter, "Distributed TCP Caching for Wireless Sensor Networks," in *Proceedings of 3rd Annual Mediterranean Ad-Hoc Networks Workshop* (Med-Hoc-Net), Bodrum, Turkey, 27-30 June 2004.

[84] K. Sundaresan, V. Anantharaman, H. Hsieh and A. R. Sivakumar, "ATP: a reliable transport protocol for ad-hoc networks," in *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking &*

computing (MobiHoc), Annapolis, Maryland, USA, pp. 64–75, 1-3 June 2003.

[85] K. Sundaresan, V. Anantharaman, H. Hsieh and A. R. Sivakumar, "ATP: a reliable transport protocol for ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 04, no. 06, pp. 588–603, 2005.

[86] E. Giancoli, F. Jabour and A. Pedroza, "CTCP: Reliable Transport Control Protocol for sensor networks," in *Proceedings of the International Conference on Intelligent Sensors, Sensor Networks and Information Processing* (ISSNIP), Sydney, Australia, pp. 64–75, 15-18 December 2008.

[87] V. C. Gungor, O. B. Akan and I. F. Akyildiz, "A Real-Time and Reliable Transport $(RT)^2$ Protocol for Wireless Sensor and Actor Networks," *IEEE/ACM Transactions on Networking*, vol. 16, no. 02, pp. 359–370, 2008.

[88] J. Paek and R. Govindan, "RCRT: rate-controlled reliable transport for wireless sensor networks," in *Proceedings of the 5th ACM international conference on Embedded networked sensor systems* (SenSys), Sydney, Australia, pp. 305–319, 6-9 November 2007.

[89] A. Dunkels, J. Alonso and T. Voigt, "Making TCP/IP Viable for Wireless Sensor Networks," in *Proceedings of First European Workshop on Wireless Sensor Networks* (EWSN), Berlin, Germany, 19-21 January 2004.

[90] A. Dunkels, T. Voigt, J. Alonso, H. Ritter and J. Schiller, "Connecting Wireless Sensornets with TCP/IP Networks," in *Proceedings of Second International Conference on Wired/Wireless Internet Communications* (WWIC), Frankfurt, Germany, 4-6 February 2004.

[91] T. Braun, T. Voigt and A. Dunkels, "TCP support for sensor networks," in *Proceedings of the Fourth Annual Conference on Wireless on Demand Network Systems and Services* (WONS), Obergurgl, Tyrol, Austria, pp. 162–169, 24-26 January 2007.

[92] H. Park, T. Kim, J. Lee, M. Jin and S. Kim, "Hop-by-hop control for reliable data dissemination in wireless sensor networks," in *Proceedings of the 9th International Symposium on Autonomous Decentralized Systems* (ISADS), Athens, Greece, pp. 1–6, 23-25 March 2009.

[93] M. A. Mahmood, I. Welch and M. K. Kasi, "Multilateration-based Event Identification in a Wireless Sensor Network," in *Proceedings of the ACM International Conference on Engineering & MIS* (ICEMIS), Istanbul, Turkey, 19-21 Jun 2018.

[94] G. Clark, *Error-Correction Coding for Digital Communications*, Plenum Press, 1981.

[95] B. Sklar, *Digital Communications: Fundamentals and Applications*, 2nd Edition, Prentice Hall, 2001.

[96] S. L. Howard, C. Schlegel and K. Iniewski, "Error control coding in low-power wireless sensor networks: when is ECC energy-efficient?," *EURASIP Journal on Wireless Communications and Networking*, vol. 02, pp. 29–29, 2006.

[97] J. Jeong and C. Ee, "Forward Error Correction in Sensor Networks," in *Proceedings of the The First International Workshop on Wireless Sensor Networks* (WWSN), Marrakesh, Morocco, pp. 1–6, 4-8 June, 2007.

[98] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics* (SIAM), vol. 08, no. 02, pp. 300 – 304, 1960.

[99] H. Wen, C. Lin, F. Ren, Y. Yue and X. Huang, "Retransmission or Redundancy: transmission reliability in wireless sensor networks," in *Proceedings of the IEEE 4th Internatonal Conference on Mobile Adhoc and Sensor Systems* (MASS), Pisa, Italy, pp. 1 – 7, 8-11 October 2007.

[100] J. C. Bolot, "End-to-end packet delay and loss behavior in the internet," in *Proceedings of the ACM SIGCOMM Conference on Communications Architectures, Protocols and Applications* (SIGCOMM), San Francisco, California, USA, pp. 289–298, 13-17 September 2005.

[101] N. Shahid, I. H. Naqvi and S. B. Qaisar, "Quarter-Sphere SVM: Attribute and Spatio-Temporal correlations based Outlier and Event Detection in wireless sensor networks," in *proceedings of IEEE International Conference on Wireless Communications and Networking Conference* (WCNC), Paris, France, pp. 2048-2053, 1-4 April 2012. doi: 10.1109/WCNC.2012.6214127

[102] N. Shahid, S. B. Ali, K. Ali, M. A. Lodhi, O. B. Usman and I. H. Naqvi, "Joint Event Detection and Identification: A Clustering based approach for Wireless Sensor Networks," in *proceedings of IEEE International Conference on Wireless Communications and Networking Conference* (WCNC), Shanghai, China, pp.2333-2338, 7-10 April 2013. doi: 10.1109/WCNC.2013.6554925

[103] K. S. Shimi, "Event Identification for Wireless Sensor Network," *International Journal of Scientific and Research Publications,* Vol 4, Issue 11, 2014.

[104] K. Kapitanova, S. H. Son and K. D. Kang, "Using fuzzy logic for robust event detection in wireless sensor networks," *Elsevier International Journal of Ad Hoc Networks*, Vol 10, No. 4, pp. 709-722, 2012. http://dx.doi.org/10.1016/j.adhoc.2011.06.008

[105] T. K. Xuan and I. Koo, "A collaborative event detection scheme using fuzzy logic in clustered wireless sensor networks," *Elsevier AEU - International Journal of Electronics and Communications*, Vol 65, No. 5, pp. 485-488, 2011. http://dx.doi.org/10.1016/j.aeue.2010.05.002.

[106] K. Ali, T. Anwaro, I. H. Naqvi and M. H. Jafry, "Composite Event Detection and Identification for WSNs using General Hebbian Algorithm," in *proceedings of IEEE International Conference on Communications* (ICC), London, UK, pp. 6463-6468, 8-12 June 2015. doi: 10.1109/ICC.2015.7249354

[107] S. Kundu and N. Das, "Event boundary detection and gathering in wireless sensor networks," in proceedings of IEEE International conference on Applications and Innovations in Mobile Computing (AIMoC), Kolkata, India, pp. 62-67, 12-14 Feb 2015. doi: 10.1109/AIMOC.2015.7083831

[108] J. K. Min, T. N. Raymond and K. Shim, "Aggregate query processing in the presence of duplicates in wireless sensor networks," in *Elsevier International Journal of Information Sciences*, Vol 297, pp. 1-20, 2015. http://dx.doi.org/10.1016/j.ins.2014.11.021.

[109] N. Sarwade and D. Patil, "EESET: Energy Efficient Secure Event Transmission Protocol in Wireless Sensor Network," in *proceedings of IEEE International Conference on Devices, Circuits and Communications* (ICDCCom), Ranchi, India, pp. 1-5, 2014.

[110] L. Liang, D. Gao, H. Zhang and O. W. W. Yang, "Efficient Event Detecting Protocol in Event-Driven Wireless Sensor Networks," in *IEEE Sensors Journal*, vol. 12, No. 6, pp. 2328-2337, 2012.

[111] F. Lin, F. Pei, D. Zhang and W. Li, "Quality of Service Support for Event Detection in Wireless Sensor Networks," in *proceedings of IEEE International Conference on Wireless Communications, Networking and Mobile Computing* (WiCOM), pp. 1-4, 23-25 September 2011.

[112] S. Ali, A. Fakoorian and H. Taheri, "Optimum Reed-Solomon erasure coding in fault tolerant sensor networks," in *Proceedings of the 4th*

International Symposium on Wireless Communication Systems (ISWCS), Trondheim, Norway, pp. 6–10, 16-19 October 2007.

[113] P. J. M. Hayinga, "Energy efficiency of error correction on wireless systems," in *Proceedings of the IEEE Wireless Communications and Networking Conference* (WCNC), New Orleans, Louisiana, USA, pp. 616–620, 21-24 September 1999.

[114] B. Marchi, A. Grilo and M. Nunes, "DTSN: Distributed transport for sensor networks," in *Proceedings of the 12th IEEE Symposium on Computers and Communications* (ISCC), Aveiro, Portugal, pp. 165–172, 1-4 July 2007.

[115] S. Kim, R. Fonseca and D. Culler, "Reliable transfer on wireless sensor networks," in *Proceedings of the 1st Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks* (IEEE SECON), Santa Clara, California, USA, pp. 449–459, 4-7 October 2004.

[116] M. S. Srouji, Z. Wang and J. Henkel, "RDTS: A Reliable Erasure-Coding Based Data Transfer Scheme for Wireless Sensor Networks," in *Proceedings of the 17th International Conference on Parallel and Distributed Systems* (ICPADS), Tainan, Taiwan, pp. 481–488, 7-9 December 2011.

[117] R. Kumar, A. Paul, U. Ramachandran and D. Kotz, "On improving wireless broadcast reliability of sensor networks using erasure codes," in *Mobile Ad-hoc and Sensor Networks* (MSN), ser. Lecture Notes in Computer Science, J. Cao, I. Stojmenovic, X. Jia and S. K. Das, Eds. Springer Berlin Heidelberg, vol. 4325, pp. 155-170, 2006.

[118] K. Obraczka, K. Viswanath and G. Tsudik, "Flooding for reliable multicast in multi-hop ad hoc networks," *Wireless Networks*, vol. 7, no. 6, pp. 627-634, 2001.

[119] W. Peng and X.-C. Lu, "On the reduction of broadcast redundancy in mobile ad hoc networks," in *Proceedings of the 1st ACM international*

symposium on Mobile ad hoc networking & computing (MobiHoc), Boston, Massachusetts, USA, pp. 129-130, 11 August 2000.

[120] M. Luby, "LT Codes," in *Proceedings of the 43rd Symposium on Foundations of Computer Science* (FOCS), Vancouver, BC, Canada, pp. 271-280, 16-19 November 2002.

[121] Y. Xue, B. Ramamurthy and Y. Wang, "Locating the nodes: cooperative localization in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 54-69, 2005.

[122] S. De, "On hop count and euclidean distance in greedy forwarding in wireless ad hoc networks," *IEEE Communications Letters*, vol. 09, no. 11, pp. 1000-1002, 2005.

[123] L. A. Anto Gracious and T. C. Subbulakshmi, "Multihop distance estimation with Greedy approach in wireless sensor networks," in *Proceedings of the International Conference on Emerging Trends in Electrical and Computer Technology* (ICETECT), Nagercoil, India, pp. 901-905, 23-24 March 2011.

[124] E. Ekici, J. Mcnair and D. Al-Abri, "A Probabilistic Approach to Location Verification in Wireless Sensor Networks," in *Proceedings of the IEEE International Conference on Communications* (ICC), Istanbul, Turkey, pp. 3485-3490, 11-15 June 2006.

[125] Y. He and H. Li, "A Distributed Node Localization Algorithm Based on Believable Factor for Wireless Sensor Network," in *Proceedings of the 5th International Conference on Wireless Communications, Networking and Mobile Computing* (WiCom), Beijing, China, pp. 01-04, 24-26 September 2009.

[126] D. Niculescu and B. Nath, "Ad hoc positioning system (APS)," in *Proceedings of the IEEE Global Telecommunications Conference* (GLOBECOM), San Antonio, Texas, USA, pp. 2926-2931, 25-29 November 2001.

[127] L. E. Miller, "Distribution of Link Distances in a Wireless Network," *Journal of Research of the National Institute of Standards and Technology*, vol. 106, no. 2, pp. 401-412, 2001.

[128] W. Dargie and C. Poellabauer, *Fundamentals of Wireless Sensor Networks Theory and Practice*, John Wiley & Sons Ltd, 2011.

[129] A. K. Paul and T. Sato, "Localization in Wireless Sensor Networks: A Survey on Algorithms, Measurement Techniques, Applications and Challenges," in *Journal of Sensor and Actuator Networks*, vol. 06, no. 04, 2017.

[130] I. F. Akyildiz and M. C. Vuran, *Wireless Sensor Networks*, John Wiley & Sons Ltd, pp. 265-285, 2010, doi: 10.1002/9780470515181.ch12

[131] R. Verdone, D. Dardari, G. Mazzini and A. Conti, *Wireless Sensor and Actuator Networks: Technologies, Analysis and Design*, Academic Press, 2008.

[132] Y. Zhang, X. Zhang, Z. Wang and H. Liu, "Virtual Edge Based Coverage Hole Detection Algorithm in Wireless Sensor Networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference* (WCNC), Shanghai, China, pp. 1488-1492, 07-10 April 2013.

[133] A. Savvides, C. C. Han and M. B. Srivastava, "Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors," in *Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking* (MobiCom), Rome, Italy, pp 166-179, 1621 July, 2001.

[134] S. N. Chiu, D. Stoyan, W. S. Kendall and J. Mecke, *Stochastic Geometry and Its Applications*, John Wiley & Sons Ltd, 2015.

[135] M. W. Marcellin and T. R. Fischer, "Trellis coded quantization of memoryless and Gauss-Markov sources," in *IEEE Transactions of Communication*, vol. 38, no. 01, pp. 8293, 1990.

[136] X. Zeng, R. Bagrodia and M. Gerla, “GloMoSim: a library for parallel simulation of large-scale wireless networks,” in *Proceedings of the 12th workshop on Parallel and Distributed Simulation* (PADS), Banff, Alberta, Canada, pp. 154-161, 26-29 May 1998.

[137] Scalable Network Technologies, “The QualNet Communications Simulation Platform,” 2011. [Online]. Available: `http://web.scalable-networks.com/content/qualnet`

[138] G. Xing, X. Wang, Y. Zhang, C. Lu, R. Pless, and C. Gill, “Integrated coverage and connectivity configuration for energy conservation in sensor networks,” ACM Transactions on Sensor Networks, vol. 01, no. 01, pp. 3672, 2005.