School of Electrical
Engineering, Computing and
Mathematical Sciences

Coverage-Oriented Reliability of Wireless Sensor Networks with Multistate Nodes

Suparna Chakraborty

0000-0001-9779-4027

This thesis is presented for the Degree of
Doctor of Philosophy
of
Curtin University

September 2021

Declaration

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgment has been made.

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

Signature: ………………………………………….

Date: 18/11/2021

ABSTRACT

The current era of the Internet of Things (IoT) has led us to a world of automation in almost every sphere of life, such as healthcare, home security, precision agriculture, etc. Wireless Sensor Networks (WSNs) serve as an important implementation tool to actualize the IoT in the real world. WSNs comprise a large number of inexpensive sensor nodes, each capable of sensing, processing, and transmitting environmental information to the sink node. These sensor nodes are vulnerable to failure due to external causes such as variability in environmental conditions, including rainfall, humidity, foliage, and internal reasons like noise, lack of battery power, hardware failure, random duty-cycle, etc. These internal causes enable a sensor node to exist in multistate such as ACTIVE, RELAY, SLEEP, IDLE, and FAIL in its entire lifecycle. To gauge the performance of such networks, reliability analysis of these networks becomes extremely important.

A WSN is an event-driven network that depends on the collective data provided by the sensor nodes monitoring a particular phenomenon. In such cases, the network is considered successfully operating if and only if a certain minimum aggregate amount of information is delivered to a given sink node. Thus, WSN reliability is defined to be the probability that the network can successfully transmit the application-specific required amount of flow to the sink node under such multistate nature of each sensor node. Further, owing to such multistate nature of sensor nodes, it is also important to quantify the capability of a WSN to provide adequate coverage of the region of interest. To quantify the flow-oriented reliability and coverage-oriented reliability of WSNs with multistate nodes, this thesis proposes new metrics: WSN Reliability ($WSNRel$), Area-Coverage Reliability (ACR) and Coverage-oriented Reliability ($CORE$). $WSNRel$ provides a minimal-path based approach to evaluate flow-oriented reliability of WSNs with multistate nodes. The proposed approach includes enumeration of shortest minimal paths from application-specific flow satisfying sensor nodes (source nodes) to the sink node. It then proposes a modified sum-of-disjoint products approach to evaluate WSN reliability in the presence of multistate nodes from the enumerated shortest minimal paths. Computing $WSNRel$ has been shown to be NP-Hard.

The metrics, ACR, and $CORE$, aim at quantifying the coverage-oriented capability of a WSN. To quantify ACR and $CORE$, Monte Carlo simulation approaches that utilize an energy matrix to check the capability of a WSN in satisfying the application-specific coverage-oriented requirement are presented. The energy matrix reflects the residual energy of sensors, the energy required to transmit data to the neighboring nodes, connectivity, and the sensors' multistate nature. A Discrete-Time Markov Chain model is presented to study the multistate behavior of sensor nodes while evaluating $CORE$. ACR and $CORE$ bring together WSN reliability, area coverage, energy efficiency, mobility of data collector or sink, random duty-cycle of nodes, and multistate nature of sensor nodes under a common umbrella. It is noteworthy to mention here that ACR considers sensors with 4 states and $CORE$ considers nodes with 10 states.

Lastly, to preserve the limited energy resources of sensor nodes, and maximize $CORE$, a multi-path split-flow routing scheme is presented. Depending on the sensor nodes' distance and residual energies, few rules that govern the splitting of flow through the multi-paths are formulated.

Simulations on benchmark networks as well as random networks are performed for each work showing the effectiveness of all algorithms proposed in this thesis. The proposed metrics will help the network designers to consider the metrics as a part of sensor network design and determine reliability quickly. The proposed metrics are advantageous in the sense that they pave the way towards a more realistic aspect of wireless sensor network reliability.

Keywords— Coverage area reliability, Markov chain, multistate nodes, network reliability, node energy, random duty-cycle, shortest minimal paths, sum-of-disjoint products, wireless sensor network.

ACKNOWLEDGMENT

It gives me an immense sense of gratification as I retrospect my expedition through the doctoral research program to acknowledge all the people from different spheres of my life who have helped and contributed to the realization of this dissertation. I would never have succeeded in completing my task without the cooperation, encouragement, and help provided to me by various personalities.

With a deep sense of gratitude, I express my sincere thanks to my esteemed and worthy supervisors, Prof. Sieteng Soh from School of Electrical Engineering, Computing and Mathematical Sciences, Curtin University, Australia, Prof. Neeraj Kumar Goyal from Subir Chowdhury School of Quality and Reliability, Indian Institute of Technology Kharagpur, India, and Prof. Sudipta Mahapatra from the department of Electronics and Electrical Communication Engineering, Indian Institute of Technology Kharagpur, India, for their valuable guidance, encouragement, and cooperation while carrying out this research work. Without their guidance and persistent help, this work would not have been in its shape today. Their relentless advice and innovative conceptions helped the accomplishment of the present dissertation.

I would take this opportunity to express my heartfelt indebtedness towards Prof. Prasanta Kumar Das from the department of Mechanical Engineering and the current Dean (PGS&R) of Indian Institute of Technology Kharagpur, India, for his immense help and guidance. I shall be failing in my duties if I do not express my deep sense of gratitude to the Doctoral Scrutiny Committee members, Prof. V. N. A. Naikan, Prof. C. S. Kumar, Prof. S. S. Pathak, Prof. S. K. Chaturvedi, Prof. Wan-Quan Liu, and Prof. Mihai Lazarescu for helping me shape up my research aptitude.

I express my thankfulness to my friends Artha, Tomi, Rebecca, Sanyu, Rupa, Keerthi, Deepanshu, Blossom, Tony, Lely, Tuli, Subhra, and Mrunmai for helping and guiding me throughout my stay at Australia. Without their help and support, the work and life at Australia would not have been so beneficial and enjoyable.

I would also like to thank the non-teaching staffs of my school at Indian Institute of Technology Kharagpur, and Curtin University for their help and cooperation at times of need. I convey my deepest regards to Steven Cheah and Ryan Liang of Curtin University, and Mr. A. K. Naskar, Deputy Registrar, Indian Institute of Technology Kharagpur, for their generosity, sound advice, and helpful responses at times of my need. It is a lifetime opportunity for me to get an opportunity to study in world-class renowned institutes like the Indian Institute of Technology Kharagpur and Curtin University. I owe my love and respect to these organizations.

Any attempt at any level would have been incomplete without the guidance and encouragement of my parents, Mr. Shanti Ranjan Chakraborty, and Mrs. Maya Chakraborty, and my brother Souvik Chakraborty. I empathy them for their support and encouragement. I am also thankful to my husband Dr. Santanu Prasad Datta and my daughter Sauraseni Datta for their continuous love and unwavering support throughout this journey. I am obligated to my in-laws for their patience and inspiration over the last few years.

My greatest thanks are to all who wished me success. Above all, I render my gratitude to the ALMIGHTY**,** who bestowed self-confidence, ability, and strength in me to complete this work.

COPYRIGHT

CONTENTS

Chapter 1

Introduction and Literature Review

Keywords
Internet of Things,
Wireless Sensor Networks,
Network Reliability,
Literature map,
Contributions.

In this chapter, a brief description of network reliability measures is provided. The investigations reported in the open literature in the broad area of Wireless Sensor Network reliability evaluation techniques are summarized, and the critical observations are brought out. Based on these observations, few research problems have been identified for further exploration. The major objectives and the expected contributions of the thesis have also been brought out.

The desire for anytime, anywhere, and any device connectivity along with the unprecedented development in wireless technologies has led us to the era of the Internet of Things (IoT). “The IoT is the general idea of things especially everyday objects, that are readable, recognizable, locatable, addressable, and controllable via the Internet - whether via RFID, wireless LAN, wide-area network, or other means (US National Intelligence Council 2008)”. The rapid growth of the IoT paradigm has linked almost every industry (see Figure 1.1), connecting over 50 billion devices together, which is equivalent to almost seven times the world’s population (see Figure 1.2).

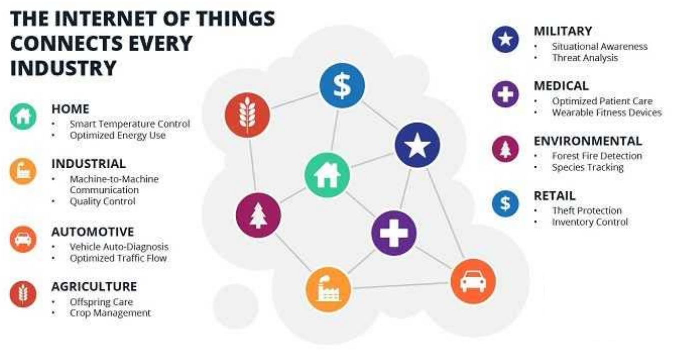

Figure 1.1 Industries connected by the Internet of Things[1]

[1] https://www.martechadvisor.com/articles/iot/by-2030-each-person-will-own-15-connected-devices-heres-what-that-means-for-your-business-and-content/

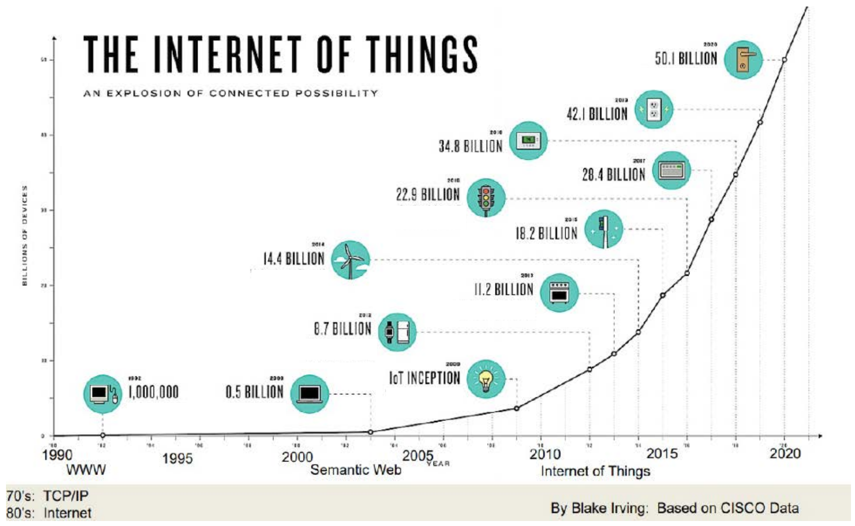

Figure 1.2 Device connectivity possibility[2]

This rapid growth of the IoT has been possible due to an unparalleled development in the field of Wireless Sensor Networks (WSNs). WSNs serve as an important implementation tool to actualize the IoT in the real-world. Design and development of one of the first WSN can be found in the middle of the '70s by the defense and military industries: during the Vietnam War, WSNs were used to detect enemies in remote jungles (Pantazis *et al.* 2013). Consequently, amongst issues like the availability of the internet, miniaturization of devices, etc., the IoT is majorly dependent on the development and proper functioning of WSNs.

WSNs can be defined as self-configuring, infrastructure-less wireless networks that monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion, or pollutants, and act cooperatively to transfer their data through the network to the data collector called sink. WSNs consist of several tiny, inexpensive sensor nodes, each capable of sensing, transmitting, and receiving environmental information. Due to their ease of deployment and multi-functionality nature of sensor nodes, WSNs have been utilized for a variety of applications such as healthcare (Baali et al. 2018; Alemdar and Ersoy 2010; Ko et al. 2010), target tracking (Ren et al. 2011; Sweidan and Havens 2018), precision agriculture (Gutierrez *et al.* 2014), precision aquaculture (Parra *et al.* 2018), smart homes, traffic surveillance and control (Chinrungrueng *et al.* 2006),

[2] http://publications.computer.org/cloud-computing/wp-content/uploads/sites/31/2018/03/ARL-pic-1.jpg

industrial automation (Silva *et al.* 2012; Stoianov *et al.* 2007), military operations, environment monitoring (Jelicic *et al.* 2011), and so on. This versatile nature of applications has increased dependence on WSNs for day-to-day activity and information exchange. With the growing dependency on such networks, quantifying the ability of the WSNs to perform such tasks has become one of the primary concerns. The degree to which a network is capable of providing the required services is quantified in terms of network reliability measures. Since long, reliability analysis of communication networks has focused on connectivity among the deployed nodes (Hardy *et al.* 2007; Cook and Ramirez-Marquez 2007; Cook and Ramirez-Marquez 2008; Cook and Ramirez-Marquez 2009; Chakraborty *et al.* 2014), and data handling capacity of binary state networks (Chakraborty and Goyal 2015a; Chakraborty and Goyal 2015b; Chakraborty and Goyal 2017; Kabadurmus and Smith 2018) as well as multistate networks (Ramirez-Marquez *et al*. 2006; Shrestha *et al*. 2010; Xing and Dai 2009; Zuo *et al*. 2007). Various performance metrics such as *2*-Terminal Reliability (2TR) (Cook and Ramirez-Marquez 2007), *All*-Terminal Reliability (ATR), *K*-terminal Reliability (*K*TR) (Hardy *et al.* 2007) to measure the connectivity reliability; and Capacity Related Reliability (CRR) (Hardy *et al.* 2007; Chakraborty and Goyal 2015a; Chakraborty and Goyal 2015b), and Multi-Node Pair Capacity Related Reliability (MNPCRR) (Chakraborty and Goyal 2017) to measure the flow-oriented reliability of binary state networks and multistate networks (Ramirez-Marquez *et al*. 2006; Shrestha *et al*. 2010; Xing and Dai 2009; Zuo *et al*. 2007) have been reported in the literature. However, the metrics 2TR, ATR, *K*TR, CRR, and MNPCRR are not directly applicable to WSNs owing to their dynamic characteristics such as ad-hoc topology of the network, the dependency of topology formation on the energy level of nodes, noise, software and hardware problems, and multiple operational states of nodes. Further, the approaches (Ramirez-Marquez *et al*. 2006; Shrestha *et al*. 2010; Xing and Dai 2009; Zuo *et al.* 2007) that deals with multistate networks are not applicable to WSNs as they focus on evaluating two-terminal reliability of multistate networks.

This chapter, divided into six main sections, provides the preliminaries and reviews of the literature that focus on reliability evaluation and are pertinent to this thesis. Section 1.1 describes the network model and notations that are used throughout the thesis. Additional notations that are used only in specific chapters are described in the corresponding chapters. Section 1.2 discusses the existing WSN reliability measures and evaluation techniques. Section 1.3 is dedicated towards the coverage aspects of WSNs. Section 1.4 brings out the energy-saving techniques in WSNs. Based on the observations made through Section 1.2 to Section 1.4, the objectives of the thesis are formulated in Section 1.5, and finally, Section 1.6 presents the organization of the thesis.

1.1 Network Model

A WSN is represented by a probabilistic graph $G = ((v) \cup N, L)$ where N represents a set of sensor nodes, v represents a sink node, and L represents a set of communication links of the network. In Chapter 2, we consider a static sink, while in Chapter 3, Chapter 4, and Chapter 5, the sink is mobile and moves along a predefined path. Owing to a sensor node's hardware components' working/failure states, random duty-cycle, energy availability, etc., a sensor node $s_i \in N$ may exist in various states in its entire lifecycle. This multistate behavior of sensor nodes is discussed in corresponding chapters of the thesis. As sensors are inexpensive low-powered devices, they are hardly repairable. In such cases, they are replaceable, and the meantime to replace them is assumed to be large. Thus, during any message transmission, any network state is solely dependent on its node-states.

In this thesis, a link's existence is governed not only by the Euclidian distance between sensors but also by the energy availability of the sensors as well as link reliability. To be more specific, in Chapter 2, the existence of a link $L_{ij} \in L$ between any two sensor nodes s_i and s_j is solely dependent on the Euclidian distance between them, while in Chapter 3, Chapter 4, and Chapter 5, the existence of a link $L_{ij} \in L$ between any two sensor nodes s_i and s_j is dependent both on the Euclidian distance as well as each node's energy.

In the following sections, we overview some of the approaches that focus on reliability and coverage of WSNs, followed by the node energy-saving strategies existing in the literature.

1.2 WSN Reliability Measures and Evaluation

In this section, we review the reliability measures applicable to WSNs and their evaluation techniques. There exists an exhaustive amount of research in the area of Wireless Sensor Network RELiability ($WSNRel$), classified into two broad categories, *connectivity*-oriented $WSNRel$ and *flow*-oriented $WSNRel$.

1.2.1 Connectivity-oriented Reliability of WSN

AboElFotoh *et al.* (2005) evaluated the reliability and message delay for a sensor network, in which the network is organized in clusters. Reliability in (AboElFotoh *et al.* 2005) is defined as the probability that there exists a connectivity-oriented operating path between at least one operational sensor and the sink node. Kharbash and Wang (2007) focused on evaluating connectivity-oriented two-terminal reliability for mobile ad hoc networks with unreliable devices and dynamic network connectivity. Egeland and Engelstad (2009) analyzed the connectivity-oriented two-terminal reliability of WSN with stochastic link-failures. Their work further forecasts the influence of adding redundant nodes to wireless networks. Considering network connectivity as the success criteria for evaluating network reliability, Cook and Ramirez-Marquez (2007) proposed methods for two-terminal reliability evaluation of mobile ad hoc networks that considered the effect of mobile nodes and continuous change in network connectivity due to node mobility.

Apart from these, to evaluate the connectivity-oriented reliability of WSNs, various other methods like enhanced factoring method (Xiao *et al.* 2008), binary decision diagram (BDD) based model (Wang *et al.* 2012), and ordered BDD (Xiao *et al.* 2009) based methods can be found in contemporary literature. Such works concentrated on mere connectivity between the sensors and the sink. As WSNs are application-oriented networks, where a minimum amount of information is required by the application to fulfill its purpose, these connectivity-oriented approaches do not provide a realistic aspect of WSNs. Thus, we now move on to have an overview of the approaches that focus on flow-oriented reliability evaluation of WSNs.

1.2.2 Flow-oriented Reliability of WSN

A WSN is an event-driven network that depends on the collective data provided by the sensor nodes monitoring a particular phenomenon. In such cases, the network is considered successfully operating if and only if a certain minimum aggregate amount of information, A_{req}, is delivered to a given sink node. Thus, WSN reliability is defined as the probability that an application-specific required amount of data, A_{req}, collected by the sensors, is successfully delivered to the sink node (AboElfotoh *et al.* 2006). The approach presented by AboElfotoh *et al.* (2007) evaluates flow-oriented WSN reliability considering various network states wherein each node has only two states, viz., working, or failure. Any network state contributes for being a successful state if it contains a successful operational tree that satisfies the flow demand of the application, i.e., A_{req}. The approach utilizes the *factoring* approach to evaluate $WSNRel$.

It is to be noted that owing to the hardware architecture and working principle of sensors, a sensor node is enabled to participate in the network not only by generating its own traffic (data) but also forwarding neighboring nodes' traffic to the sink node. Due to this property, the contribution of a sensor node in the RELAY state needs to be considered. The RELAY state of a sensor is a state in which a sensor node does not add to the total flow, but only forwards data from other sensors towards the sink node. A recent approach (Shazly *et al.* 2010) has put some light in this direction. Shazly *et al.* (2010) stated that the reliability evaluation problem for WSNs with multistate nodes is #P-hard. For a particular class of WSNs topology, called Diagonalized grids (D-grid), they (Shazly *et al.* 2010) proposed a polynomial-time algorithm to solve the problem. However, for arbitrary networks with $|N| + 1$ nodes, Shazly *et al.* (2010) used a Brute-Force (Complete State Enumeration) approach that generates all possible ($3^{|N|}$) network states. Any network state contributes to the reliability of the WSN, iff it can successfully transmit A_{req} to the sink node. Reliability is then evaluated by adding the probabilities of such successful network states. Due to the requirement of the generation of a huge number of states and their evaluation, these approaches are computationally costly and infeasible to solve moderate-sized networks with arbitrary topology. Most of the approaches that evaluate the flow-oriented reliability of WSNs with multistate nodes focus on non-pathset approaches viz., finding the entire network state, factoring, decomposition, etc., for evaluating the reliability of WSNs. Therefore, there is a need to work towards reducing the number of state generation and their evaluation along with developing efficient methods for reliability evaluation of such networks.

Besides network connectivity and data-driven capacity (AboElfotoh *et al.* 2007), coverage serves as one of the major service metrics of sensor networks. The next section gives more details about some of the related work in this direction.

1.3 Coverage in WSNs

Coverage (Mohamed *et al.* 2017) is concerned with how well the region of interest is being monitored by a WSN in the presence of sensor node failures. Various types of coverage problems have been studied and reported in the literature:

- Target coverage (Mostafaei and Meybodi 2013; Mostafaei *et al.* 2015): these types of problems aim to surveille a set of given targets in the monitoring field.
- Barrier coverage (Mohamed *et al.* 2017; Yang *et al.* 2014; Mostafaei *et al.* 2018; Dong *et al.* 2020): these problems aim at minimizing the probability of undetected penetration through a sensor barrier. They are concerned with border surveillance aiming to recognize the intruder's invalid intervention.
- Area coverage (Fan and Jin 2010; Liu and Liang 2005; Ma and Yang 2007; Mostafaei *et al.* 2016; Mohamed *et al.* 2017; Mostafaei *et al.* 2017; Yang *et al.* 2015): these problems are concerned with monitoring the whole region under consideration. Two types of area coverage problems have been studied: full area coverage (Ma and Yang 2007), and partial area coverage (Liu and Liang 2005; Mostafaei *et al.* 2016; Mostafaei *et al.* 2017). Full area coverage problems require monitoring of every point in the simulation region, i.e., 100% of the region of interest needs to be monitored, whilst partial area coverage problems are concerned with monitoring a limited percentage (θ%) of the entire simulation region.
- Point coverage (Chen *et al.* 2010; Huang and Tseng 2005): these problems are concerned with monitoring some important points corresponding to a set of defined targets.

- K-coverage (Shrestha *et al.* 2007; Shrestha and Xing 2008; Wang *et al.* 2017): these problems are concerned with monitoring each physical point in the area by at least K distinct (K ≥ 1) working (or active) sensors.
- m-connected K-coverage (Kumar *et al.* 2016; Li and Liu 2009): these problems are concerned with discovering a minimum number of sensors that should work such that each physical point in the area is monitored by at least K active sensors and the K active sensors form an m-connected graph. In a communication graph of $|N|$ sensors, the graph is said to be m-connected if, for any two vertices in N, there are m vertex-disjoint paths between the two vertices.

Irrespective of the type of coverage problems considered, reliable monitoring of a phenomenon depends not only on providing adequate sensing coverage but also on successful transmission of the sensed data to the sink node. In other words, for successful operation, a WSN is required to provide a guaranteed sensing coverage that exceeds a given application-specific coverage-area requirement. Further, it also requires to quantify how well the WSN provides coverage of the region of interest in the presence of multistate nodes.

In the near past, Shazly *et al.* (2011) proposed an approach to quantify the area coverage reliability of WSNs with multistate nodes. They (Shazly *et al.* 2011) considered only hardware failure of nodes to evaluate WSN node-states and WSN reliability. They failed to focus on the fact that the node-states are highly affected by a node's energy along with the node's hardware. Men and Chen (2016) presented an approach to identify the states of nodes based on node energy, but they did not address the reliability evaluation of such WSNs. WSN reliability depends on factors like network connectivity in the presence of multistate nodes, transfer of collected data to the sink node (fixed or mobile), adequate coverage of the monitoring area, and residual energy of nodes. Although the network connectivity problems (Chakraborty *et al.* 2014; Cook and Ramirez-Marquez 2007; Cook and Ramirez-Marquez 2009; Cook and Ramirez-Marquez 2008), the data collection problems (Chakraborty and Goyal 2015a; Chakraborty and Goyal 2015b; Chakraborty and Goyal 2017; Kabadurmus and Smith 2018), and the coverage problems (Mostafaei *et al.* 2017; Tsai 2008; Mohamed *et al.* 2017) have been studied individually, only a few approaches have studied these concepts unitedly (Shazly *et al.* 2010; Shrestha *et al.* 2007; Shrestha and Xing 2008). More specifically, approaches in (Shrestha *et al.* 2007; Shrestha and Xing 2008) evaluated connectivity-oriented K-coverage reliability for WSNs with binary-state sensors, and Shrestha and Xing (2008) evaluated flow-oriented reliability of WSNs with three-state nodes. As such, the approaches in (Shazly *et al.* 2010; Shrestha *et al.* 2007; Shrestha and Xing 2008) do not consider the multistate nature of sensors, the coverage-oriented reliability, and nodes' energy unitedly in one frame.

WSNs are essentially resource-constrained networks. Besides providing reliability and coverage, efficient and optimal use of the limited resources of the WSN is also a major concern. The next section, therefore, discusses the techniques that are readily available in the literature to preserve the limited resources of sensor nodes.

1.4 Energy Preserving in WSNs

Being resource-constrained networks, a thrust area of research in WSNs is on optimal use of sensors' limited resources. Yang *et al.* (2015) presented an energy-efficient approach to address the area-coverage problem. However, it does not put light on the border nodes' residual energy. Due to the limited transmission range, sensors, which are far away from the sink, use multi-hop communication to deliver the collected data. Thus, sensors near the sink are heavily used. Due to excessive usage, the energy of border nodes depletes quickly, leading to the failure of these nodes. This, in turn, leads to network disconnection and hence network failure. To address this issue, the concept of multiple static sinks was introduced (Lee *et al.* 2010; Luo *et al.* 2006). It decreased the average energy dissipation per node as compared to single static sink WSN. However, optimal placements of the multiple static sinks became a major problem. To cope with this problem, the concept of mobile WSN (mWSN) was introduced (Mohamed *et al.* 2017). In an mWSN, the data collector or the sink moves in and around the monitoring region to collect data from on-field sensor nodes.

Further, methods like optimizing duty-cycle (Jurdak *et al.* 2010), use of multiple sink nodes (Lee *et al.* 2011), use of optimal number of multiple sink nodes and deciding their geographical location (Lee *et al.* 2010), replacing multiple sink nodes by a mobile sink node (Di Francesco *et al.* 2011), optimizing trajectory movement of mobile sink node

(Habib *et al.* 2018), optimal deployment pattern of the sensors (Deif and Gadallah 2017; Liu 2015), network coding techniques (Cristescu and Beferull-Lozano 2006; Rout and Ghosh 2013), routing, etc. can be found in contemporary literature that aimed at optimally using the network resources. Amongst all, routing attracts a major concern. Several routing approaches based on different criteria (Sara and Sridharan 2014) such as network structure, state of information, mobility, and energy-efficiency have been proposed in the literature. Amid others, the most commonly used approach is *energy-efficient routing*. While designing energy-efficient routing, various aspects like controlling the transmission power (Kim *et al.* 2009), distribution of load (Huang *et al.* 2008; Choi *et al.* 2010), minimizing the energy due to overhearing (Park and Kim 2010), and idle listening (In and Hoc 2006) have been taken care through single-path routing (Zonouz *et al.* 2014). However, these approaches are applicable to networks having binary state nodes and fail to put light on WSNs with multistate nodes.

The optimization framework through routing can be broadly categorized as single-path routing and multi-path routing. In single-path routing, a node sends data through a single path to the sink, whereas in multi-path routing, a node has multiple paths to send its data to the sink. It is well established that multi-path routing approaches achieve higher performance than single-path approaches (Waharte and Boutaba 2006). Multi-path routing provides advantages (Sha *et al.* 2013) like reduced latency, reduced delay, increased network lifetime, fault tolerance, load balancing, and lower power consumption (Waharte and Boutaba 2006).

A plethora of approaches can be found in the literature that focuses on multi-path enumeration (Marina and Das 2001; Periyasamy and Karthikeyan 2017; Sha *et al.* 2013; Morino *et al.* 2009). Such approaches contribute to either improving connectivity through the establishment of redundant/backup paths (Perkins and Royer 1999; Marina and Das 2001) or on optimizing network resource utilization via load distribution among several paths (Lee and Gerla 2001; Leung *et al.* 2001). The main goal of the load distribution method is to balance the energy usage among the nodes and to maximize the network lifetime by avoiding over-utilized nodes when selecting a routing path. This can be achieved by transmitting data through multiple-paths (Lee and Gerla 2001; Leung *et al.* 2001; Morino *et al.* 2009; Zhao *et al.* 2015; Kabadurmus and Smith 2018). However, these approaches consider either node-disjoint or link-disjoint multi-paths utilization while transmitting data to the sink, and thus are not applicable for multi-source single-sink networks. Further, they deal with binary-state sensor nodes and therefore fail to optimize energy-oriented aspects of WSNs with multistate nodes.

A new paradigm of approaches focuses on load distribution through multi-paths to optimize communication energy. However, such approaches are applicable to networks having binary-state nodes, i.e., working or failure, and single-source single sink networks, and they do not focus on (i) reliability, i.e., the capability of a network to successfully deliver an application-specific required amount of data to the sink, (ii) multistate nature of sensor nodes, (iii) coverage, i.e., the capability of a WSN to provide the required application-specific coverage of the region of interest, and (iv) the most important aspect that WSNs are essentially multi-source single-sink networks.

1.5 Objectives

In view of the above discussions, the author is motivated to formulate the following research objectives:

- To develop a reliability model for wireless sensor networks with multistate nodes.
- To develop approaches for evaluating coverage-oriented reliability of mobile wireless sensor networks with multistate nodes under link reliability constraints.
- To develop an approach that maximizes the coverage-area reliability of mobile wireless sensor networks with multistate nodes.

1.6 Organization of the Thesis

The thesis is documented in six chapters, organized as follows:

Chapter 1: Introduction

This chapter presents the motivation of the thesis by offering an overall view of the different reliability measures and their evaluation techniques to assess the performance of WSNs with binary states and multistate nodes. It also covers coverage-oriented aspects and energy-saving approaches in WSNs.

Based on the literature survey a few shortcomings are identified. These, in turn, facilitate the formulation of the objectives of the present thesis. An organization of the thesis is also presented in this chapter.

Chapter 2: Minimal Path-based Reliability Model for Wireless Sensor Networks with Multistate Nodes

This chapter formalizes a WSN reliability ($WSNRel$) problem in the presence of multistate nodes. It proposes a minimal path-based approach, called as the Multi Node-State Reliability Evaluator (MNRE) approach, to solve the $WSNRel$ problem and compute exact solutions of the $WSNRel$ measure on arbitrary networks. The method is supported by well-explained examples at each step.

Chapter 3: Area Coverage Reliability Evaluation of Mobile Wireless Sensor Networks with Multistate Nodes under Link Reliability constraints

This chapter formalizes a second WSN reliability problem, referred to as the area coverage reliability (ACR) problem. It proposes a disjoint area coverage scheme for each sensor node to avoid loss of energy due to sensing of overlapping areas by two or more sensors. In this chapter, the topology of the mobile WSN is accounted by a connectivity matrix that accounts for residual energies of all nodes, node states, as well as node and link reliability. This chapter assumes a 4-state node reliability model. The effectiveness of our proposed approach is shown through several performance comparisons on mWSNs with various sizes.

Chapter 4: A Monte-Carlo Markov Chain Approach for Coverage-area Reliability of Mobile Wireless Sensor Networks with Multistate Nodes

This chapter formalizes a new WSN reliability index, referred to as the COverage-area REliability ($CORE$), that accounts for providing application-specific coverage of the monitoring region, ensuring reliable data delivery in the presence of multistate nodes. A Discrete-Time Markov Chain approach that models the hardware, random duty-cycle, and energy-dependent multistate nature of a sensor node is presented in this chapter. This chapter considers sensor nodes to exist in any of ten different states.

Chapter 5: On Maximizing Coverage-oriented Reliability of Mobile Wireless Sensor Networks with Multistate Nodes

This chapter aims at maximizing $CORE$ by minimizing the energy spent in each transmission round. It presents a smart multi-path split-flow routing approach to maximize $CORE$. This chapter investigates on enumerating multi-paths for multi-source single sink WSNs with multistate nodes and focusses on determining the amount of traffic (data) to be split through multiple paths.

Chapter 6: Conclusions and Scope of Future Work

This chapter summarizes the work presented in this thesis, reiterates the major contributions, and proposes some directions for future work.

In addition to these chapters, the thesis includes Appendix A that provides the complexity analysis and correctness of the approach presented in Chapter 2. Every method proposed in this thesis has been suitably exemplified. The algorithms developed are coded in MATLAB and implemented on a computer with 4.00 GHz Intel(R) Core(TM) 2 Duo processor and 4 GB RAM, running Windows 7 operating system.

Chapter 2

Minimal Path-based Reliability Model for Wireless Sensor Networks with Multistate Nodes

This chapter studies the reliability of Wireless Sensor Networks (WSNs), where each sensor node has multiple operational states. It proposes an approach to evaluate the flow-oriented reliability of WSNs consisting of multistate sensor nodes. The proposed approach enumerates the shortest minimal paths from application-specific flow satisfying sensor nodes (source nodes) to the sink node. It then proposes a modified sum-of-disjoint products approach to evaluate WSN reliability from the enumerated shortest minimal paths. Simulations are performed on WSNs of various sizes to show the applicability of the proposed approach on such WSNs with arbitrary topology.

Keywords
multistate nodes,
network reliability,
shortest minimal paths,
sum-of-disjoint products,
wireless sensor network.

2.1 Introduction

Wireless Sensor Networks (WSNs) have emerged as a promising technology towards the development of Next Generation Networks such as the Internet of Things. WSNs consist of a large number of tiny low-powered sensor devices that are spatially distributed over an area of interest to perform an application-oriented task. These tiny sensor devices, typically known as sensors, perform the task of capturing and revealing real-world physical phenomena such as temperature, sound, light, moisture, etc., in almost all types of environments (industrial, agriculture, military, volcanic, etc.). Further, a sensor node exhibits multistate behavior during its lifecycle due to its hardware components working/failure states. Owing to their adaptability of operating in almost every type of environment and the existence of multistate sensors, performance quantification of such networks in terms of reliability becomes challenging. To mention here, communication within a WSN can be categorized (Shrestha *et al.* 2007; Shrestha and Xing 2008; Shrestha *et al.* 2012; Zhu *et al.* 2016; Zonouz *et al.* 2013) as *uplink* or *application communication* and *downlink* or *infrastructure communication*. In uplink mode, the sensed data from the sensors is transferred to the sink node, whilst in the downlink mode the sink node sends control messages to the sensors. This chapter focusses on the reliability of application communication in WSN. Recalling from Chapter 1, WSN reliability is defined as the probability that an application-specific required amount of data, A_{req}, collected by the sensors, is successfully delivered to the sink node (AboElfotoh *et al.* 2006).

As discussed in Section 1.2, most of the approaches have concentrated on finding WSN reliability based on methods such as factoring, decomposition, etc. However, WSNs follow a multipath strategy wherein all the source nodes have to be simultaneously connected to the sink node to fulfill the flow criteria A_{req}. To the best of our knowledge, no such approach exists in the literature till date that evaluates path-based reliability of WSNs with multistate nodes. Aiming to evaluate the reliability of WSNs with multistate nodes, this chapter presents an algorithm for finding the set of shortest minimal paths in order of their lengths for a WSN with multistate nodes. It then presents a modified sum-of-disjoint products approach for calculating the reliability of the network in terms of the probabilities of success of the nodes present in the enumerated minimal paths.

This chapter is organized as follows. Section 2.2 presents the assumptions made during the work. Section 2.3 discusses the multistate node reliability model considered for this study. Section 2.4 introduces the reliability measure and defines the problem formally. Section 2.5 describes the details of our solution. The proposed algorithms are given in Section 2.6. An illustrative example is provided in Section 2.7. Some benchmark networks are taken from the literature where some values of required aggregate flow A_{req} are assumed, and the results are discussed in Section 2.8. Finally, Section 2.9 concludes this chapter.

2.2 Assumptions

The assumptions considered for this study are:

- No limitation is imposed on the capacity of nodes and links for the transfer of data.
- Links are perfectly reliable.
- The flow from each sensor is not divisible.
- Nodes are statistically independent of each other.
- No loss of data is considered on the path from the multi-sources to the sink node.

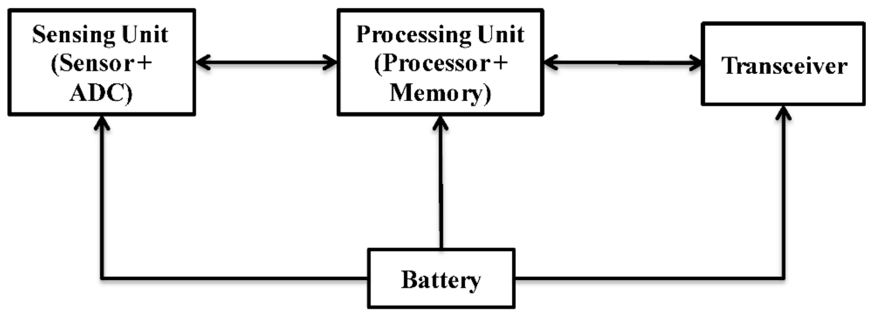
Sensing Unit (Sensor+ ADC)
Processing Unit (Processor+ Memory)
Transceiver
Battery

Figure 2.1 Inner architecture of a sensor node (Butenko *et al.* 2014)

- Node mean-time-to-failure (MTTF) is relatively large compared to the time required for transmission of messages and the time required by the network to self-configure due to node failures.

2.3 The Multistate Node Reliability Model

A WSN is represented by a probabilistic graph $G = ((v) \cup N, L)$ where N represents the set of sensor nodes, v represents a sink node, and L represents the set of communication links of the network. A communication link exists between two nodes if they are within the radio transmission range of each other. As sensors are inexpensive low-powered devices, they are hardly repairable. In such cases, they are replaceable, and the meantime to replace is assumed to be large. Thus, during any message transmission, any network state is solely determined in terms of its node-states.

As depicted in Figure 2.1, each sensor node is composed of four components, viz., a sensing unit, a powering unit (the battery), a microcontroller (processing unit), and a transceiver unit (AboElfotoh *et al.* 2006). Due to the harsh nature of WSN's application environment, sensor nodes are subjected to random failures. Failure of either the powering unit or microcontroller leads to a complete node failure. Therefore, in this work, it is assumed that the powering unit and the microcontroller are always operational; hence various states of a sensor node correspond to failure of either the sensing unit or the transceiver unit or both. Failure of the transceiver unit is assumed to be independent of the sensing unit. Thus, each sensor node can operate in one of the following states:

a) ACTIVE: The state in which a sensor, $s_i \in N$, can perform both communication and sensing tasks as both its sensing and transceiver units are working. As given in (Shazly *et al.* 2010), this state probability is evaluated as:

$$p_A(s_i) = p_{cu}(s_i) \times p_{su}(s_i) \tag{2.1}$$

where $p_{cu}(s_i)$ denotes the probability that the transceiver unit (communication module) of any sensor s_i is working and $p_{su}(s_i)$ denotes the probability that the sensing unit of any sensor s_i is working.

b) RELAY: The state in which a sensor, $s_i \in N$, can communicate but not sense the environment. As given in (Shazly *et al.* 2010), this state probability is given by

$$p_R(s_i) = p_{cu}(s_i) \times (1 - p_{su}(s_i)) \tag{2.2}$$

c) FAIL: The state in which a sensor, $s_i \in N$, is unable to communicate as the transceiver unit has failed. The working or failure of the sensing unit makes no difference as the data cannot be sent to the neighbors due to failure of the transceiver unit. As given in (Shazly *et al.* 2010), this state probability is given by

$$p_F(s_i) = 1 - p_A(s_i) - p_R(s_i) = 1 - p_{cu}(s_i). \tag{2.3}$$

2.4 Reliability Measure and Problem Formulation

Our problem considers a multi-source single-sink WSN for reliable monitoring of any event or phenomenon. Let λ_i be the amount of information collected (sensed) by any sensor $s_i \in N$; λ_i is also called as the flow rate of sensor s_i. The problem requires that the WSN's multiple-source node(s) can successfully collect and transfer a minimum amount of information of the event, A_{req}, to the sink node. In other words, the WSN is functioning if (i) there is at least one set of operational source node(s) that together can collect a minimum

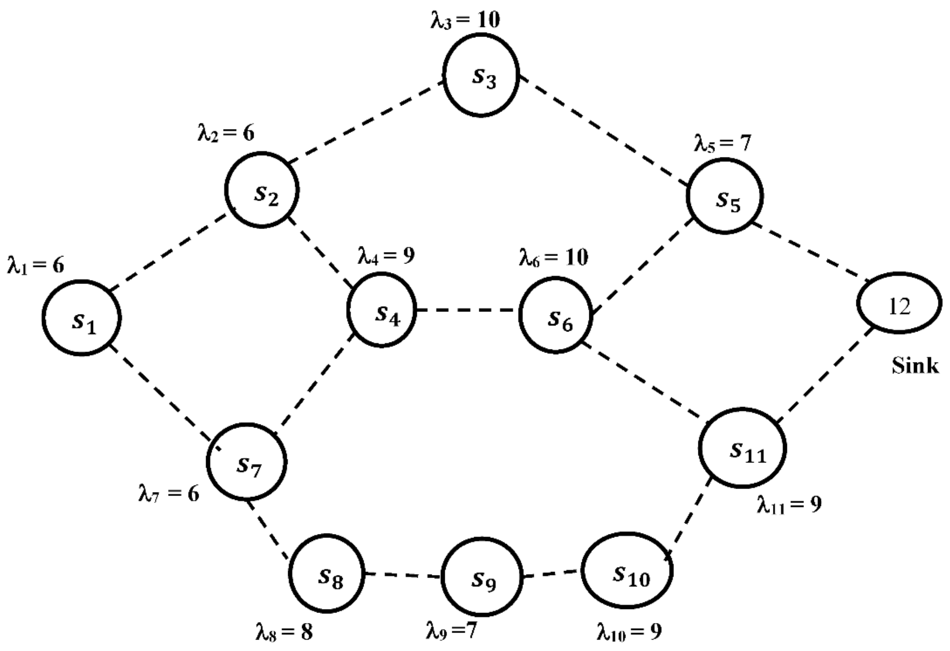

Figure 2.2 A typical wireless sensor network (12 nodes) (AboElFotoh *et al.* 2005). (Traffic flow rates of each node is indicated by λ_i).

information A_{req}, and (ii) the collected information can be successfully transmitted to the sink node. Note that both conditions (i) and (ii) consider, at any point of time, each sensor node s_i can be in any of the three states, i.e., ACTIVE, RELAY, or FAIL, with a probability of $p_A(s_i)$, $p_R(s_i)$, or $p_F(s_i)$, respectively.

Let T_m be the mth set of sensor nodes whose gathered information capacity, $C(T_m)$, satisfies A_{req}. Such node-sets are termed as valid node-sets, i.e., each T_m is a valid node-set. The total flow rate of T_m is computed as:

$$C(T_m) = \sum_{s_i \in T_m} \lambda_i. \tag{2.4}$$

As an example, consider Figure 2.2 with A_{req} = 20. One possible T_m is $\{s_3, s_6\}$, and corresponding individual information gathering rates are λ_3 = 10 and λ_6 = 10. Therefore, its computed total flow is $C(T_m) = \lambda_3 + \lambda_6 = 10 + 10 = 20$.

A WSN is said to be performing reliably only when it successfully delivers the collected data $C(T_m) \geq A_{req}$ to the sink node. In this paper, we consider data to be transmitted via Shortest Minimal Paths (*SMPs*). *SMPs* are the set of minimal paths (*MPs*) from each T_m to sink v arranged in order of their *lengths*. The *length* of an *MP* is determined by the minimum number of hops required by any sensor $s_i \in T_m$ to reach the sink node v, and the *length* of an *SMP* is the maximum among the *length* of its *MPs*. It is worth mentioning that on the failure of any *SMP*, the network chooses the next available *SMP* to transmit its collected data to the sink. Therefore, it is essential to enumerate the set of *SMPs* in order of their *lengths*. Let $SMP(T_m)$ denote the set of all *SMPs* from each T_m to v, $SMP_n(T_m)$ denote the *n*th *SMP* from T_m to the sink node v, and K_m denote the total number of $SMP(T_m)$s. Thus,

$$SMP(T_m) = \{SMP_n(T_m) | \, n = 1, 2, \ldots, K_m\}. \tag{2.5}$$

As an example, considering the WSN in Figure 2.2, for $T_m = \{s_3, s_6\}$, we have $SMP(\{s_3, s_6\}) = \{s_3, s_6, s_5, 12\}, \{s_3, s_2, s_4, s_6, s_{11}, 12\}$, and $\{s_3, s_2, s_1, s_6, s_7, s_8, s_9, s_{10}, s_{11}, 12\}$. Note that, $\{s_3, s_6, s_5, s_{11}, 12\}$ and $\{s_3, s_2, s_4, s_6, s_7, s_8, s_9, s_{10}, s_{11}, 12\}$ could also be the paths from $T_m = \{s_3, s_6\}$ to $v = 12$, but they are exempted from inclusion in $SMP(T_m)$ as they are not the shortest paths. Let T be a set containing each T_m in $G = ((v) \cup N, L)$, and $\mathbb{Q}$ be the total number of *SMPs* in $G = ((v) \cup N, L)$. Thus, the *SMPs* for the entire WSN are given as,

$$SMP(T) = \{SMP(T_m) | m = 1, 2, \ldots, \mathbb{Q}\}. \tag{2.6}$$

Note that each *SMP* in (2.6) is unique, i.e., $SMP(T_m) \cap SMP(T_n) = \varphi$, for any pair $T_m \neq T_n$. Let $SMP_q(T)$ be the q^{th} *SMP* in $SMP(T)$, for $q = 1,2, \ldots, \mathbb{Q}$. Given the enumerated *SMP*s in (2.6), the goal now is to compute $WSNRel$, i.e., the probability that there exists at least one operational $SMP_q(T)$. In other words, we need to compute

$$\begin{aligned} WSNRel = {} & P\big(SMP_1(T)\big) + P\big(\overline{SMP}_1(T)\, SMP_2(T)\big) + \cdots \\ & + P\left(\overline{SMP}_1(T)\ \overline{SMP}_2(T) \cdots \overline{SMP}_{\mathbb{Q}-1}(T)\, SMP_{\mathbb{Q}}(T)\right) \end{aligned} \tag{2.7}$$

where, $\overline{SMP}_q(T)$denotes the complement of $SMP_q(T)$, and $P(.)$ is the probability function.

In a nutshell, our solution to compute $WSNRel$ contains the following main challenges: (i) To generate each T_m; (ii) To enumerate $SMP(T_m)$ for each T_m as per (2.5) and $SMP(T)$ as per (2.6); and (iii) To compute the probability in (2.7). Notice that, in (i), there can be an exponential number of such set in terms of N. Further, generating $SMP(T_m)$ in (ii) requires enumerating all (s_i, v) minimal paths, for each sensor $s_i \in T_m$. Finally, for (iii), the existing *sum-of-disjoint products* (SDP) approach (Chaturvedi and Misra 2002) is not applicable to compute the probability value in (2.7) because the problem deals with sensors working in multistates. Thus, one needs to design a solution that takes into account the node-state information while enumerating the disjoint-sets to compute $WSNRel$. Note that the solution in (iii) is at least as hard as the problem to compute SDP for a binary state system, which has been proved #NP-hard in (Provan and Ball 1984). Concisely, this paper proposes a new approach to solve (i), (ii), and (iii) that can be used for WSNs with arbitrary topology.

2.5 Proposed Methodology and Approach

We propose the following four-step approach, named Multi Node-State Reliability Evaluator (MNRE), to solve the problem.

Step 1. Enumerate each valid node-sets, T_m, whose $C(T_m) \geq A_{req}$, and each T_m contains only nodes in the ACTIVE state.

Step 2. Enumerate *SMPs* from each $T_m \in T$ to the sink node. These *SMPs* consist of nodes in either ACTIVE or RELAY states.

Step 3. Enumerate the symbolic reliability expression from the *SMPs* (node-states) by disjointing them with each other.

Step 4. Evaluate the $WSNRel$ by supplying the node-state reliability values in the reliability expression.

Step #1 of MNRE contributes towards identifying the multiple sources, T_m, which, when simultaneously connected to the sink node, satisfactorily fulfills the requirement of A_{req}. If any of the source nodes ($s_i \in T_m$) is not connected (directly or via intermediate nodes) to the sink node, then the A_{req} is not fulfilled. Therefore, in order to have a reliable flow-oriented connectivity between a T_m and sink node v, there should always be a path (minimal) connecting all nodes in T_m and v. These *SMPs*, in order of their lengths, are enumerated through Step #2. In any *SMP*, any node $s_i \in T_m$ must be in the ACTIVE state to contribute towards fulfilling the A_{req}. Source nodes that are one-hop away from the sink node can directly transfer the sensed data to the sink node. However, the source nodes that are not directly connected to the sink node transfer their data via intermediate nodes to the sink node. These intermediate nodes must work in RELAY state to form a path between these source nodes and the sink node. This node-state information about each node present in each *SMPs* is stored as *NdState*, described later. The *NdState* that accounts for the node-state information in each *SMP* is used in Step #3 to enumerate the symbolic reliability expression. Once the symbolic reliability expression is enumerated, the node reliability values are supplied to evaluate $WSNRel$ in Step #4.

Sections 2.5.1, 2.5.2 and 2.5.3 elaborate each of the steps with proper illustrations.

2.5.1 Enumeration of T_m

Various order combinations of nodes are made and checked to ascertain whether they satisfy A_{req}. Recall that the *m*th node-set that satisfies the requirement A_{req} is termed as a valid node-set, denoted by T_m. Each T_m consists of the source nodes, which should work together in an ACTIVE state to satisfy the flow-requirement. If any lower-order combination of nodes satisfies the application-specific flow requirement, it is refrained from generating any higher-order combination, as it will result in redundancy.

Figure 2.2 shows a heterogeneous WSN with 12 nodes, where the generated flow rate of each node is indicated by λ_i. The minimum application-specific flow requirement for the WSN in Figure 2.2 is assumed to be A_{req} = 20. It can be seen that no single node satisfies the flow requirement. Therefore, by following *Lines 2-22* of **Algorithm 1**, presented in Section 2.6, various order combinations of nodes are made and checked against A_{req} = 20 for flow validation. Each enumerated T_m and its total flow $C(T_m)$ for the WSN in Figure 2.2 are shown in Table 2.1.

If any pth order node-set satisfies the requirement, A_{req}, then it is not considered further for higher-order generation as it results in supersets. For example, as shown in

Table 2.1 Flow satisfying node-sets for the WSN in Figure 2.2

T_m	$C(T_m)$	T_m	$C(T_m)$	T_m	$C(T_m)$	T_m	$C(T_m)$	T_m	$C(T_m)$
$\{s_3, s_6\}$	20	$\{s_1, s_7, s_8\}$	20	$\{s_2, s_6, s_{10}\}$	25	$\{s_3, s_{10}, s_{11}\}$	28	$\{s_5, s_8, s_9\}$	22
$\{s_1, s_2, s_3\}$	22	$\{s_1, s_7, s_{10}\}$	21	$\{s_2, s_6, s_{11}\}$	25	$\{s_4, s_5, s_6\}$	26	$\{s_5, s_8, s_{10}\}$	24
$\{s_1, s_2, s_4\}$	21	$\{s_1, s_7, s_{11}\}$	21	$\{s_2, s_7, s_8\}$	20	$\{s_4, s_5, s_7\}$	22	$\{s_5, s_8, s_{11}\}$	24
$\{s_1, s_2, s_6\}$	22	$\{s_1, s_8, s_9\}$	21	$\{s_2, s_7, s_{10}\}$	21	$\{s_4, s_5, s_8\}$	24	$\{s_5, s_9, s_{10}\}$	23
$\{s_1, s_2, s_8\}$	20	$\{s_1, s_8, s_{10}\}$	23	$\{s_2, s_7, s_{11}\}$	21	$\{s_4, s_5, s_9\}$	23	$\{s_5, s_9, s_{11}\}$	23
$\{s_1, s_2, s_{10}\}$	21	$\{s_1, s_8, s_{11}\}$	23	$\{s_2, s_8, s_9\}$	21	$\{s_4, s_5, s_{10}\}$	25	$\{s_5, s_{10}, s_{11}\}$	25
$\{s_1, s_2, s_{11}\}$	21	$\{s_1, s_9, s_{10}\}$	22	$\{s_2, s_8, s_{10}\}$	23	$\{s_4, s_5, s_{11}\}$	25	$\{s_6, s_7, s_8\}$	24
$\{s_1, s_3, s_4\}$	25	$\{s_1, s_9, s_{11}\}$	22	$\{s_2, s_8, s_{11}\}$	23	$\{s_4, s_6, s_7\}$	25	$\{s_6, s_7, s_9\}$	23
$\{s_1, s_3, s_5\}$	23	$\{s_1, s_{10}, s_{11}\}$	24	$\{s_2, s_9, s_{10}\}$	22	$\{s_4, s_6, s_8\}$	27	$\{s_6, s_7, s_{10}\}$	25
$\{s_1, s_3, s_7\}$	22	$\{s_2, s_3, s_4\}$	25	$\{s_2, s_9, s_{11}\}$	22	$\{s_4, s_6, s_9\}$	26	$\{s_6, s_7, s_{11}\}$	25
$\{s_1, s_3, s_8\}$	24	$\{s_2, s_3, s_5\}$	23	$\{s_2, s_{10}, s_{11}\}$	24	$\{s_4, s_6, s_{10}\}$	28	$\{s_6, s_8, s_9\}$	25
$\{s_1, s_3, s_9\}$	23	$\{s_2, s_3, s_7\}$	22	$\{s_3, s_4, s_5\}$	26	$\{s_4, s_6, s_{11}\}$	28	$\{s_6, s_8, s_{10}\}$	27
$\{s_1, s_3, s_{10}\}$	25	$\{s_2, s_3, s_8\}$	24	$\{s_3, s_4, s_7\}$	25	$\{s_4, s_7, s_8\}$	23	$\{s_6, s_8, s_{11}\}$	27
$\{s_1, s_3, s_{11}\}$	25	$\{s_2, s_3, s_9\}$	23	$\{s_3, s_4, s_8\}$	27	$\{s_4, s_7, s_9\}$	22	$\{s_6, s_9, s_{10}\}$	26
$\{s_1, s_4, s_5\}$	22	$\{s_2, s_3, s_{10}\}$	25	$\{s_3, s_4, s_9\}$	26	$\{s_4, s_7, s_{10}\}$	24	$\{s_6, s_9, s_{11}\}$	26
$\{s_1, s_4, s_6\}$	25	$\{s_2, s_3, s_{11}\}$	25	$\{s_3, s_4, s_{10}\}$	28	$\{s_4, s_7, s_{11}\}$	24	$\{s_6, s_{10}, s_{11}\}$	28
$\{s_1, s_4, s_7\}$	21	$\{s_2, s_4, s_5\}$	22	$\{s_3, s_4, s_{11}\}$	28	$\{s_4, s_8, s_9\}$	24	$\{s_7, s_8, s_9\}$	21
$\{s_1, s_4, s_8\}$	23	$\{s_2, s_4, s_6\}$	25	$\{s_3, s_5, s_7\}$	23	$\{s_4, s_8, s_{10}\}$	26	$\{s_7, s_8, s_{10}\}$	23
$\{s_1, s_4, s_9\}$	22	$\{s_2, s_4, s_7\}$	21	$\{s_3, s_5, s_8\}$	25	$\{s_4, s_8, s_{11}\}$	26	$\{s_7, s_8, s_{11}\}$	23
$\{s_1, s_4, s_{10}\}$	24	$\{s_2, s_4, s_8\}$	23	$\{s_3, s_5, s_9\}$	24	$\{s_4, s_9, s_{10}\}$	25	$\{s_7, s_9, s_{10}\}$	22
$\{s_1, s_4, s_{11}\}$	24	$\{s_2, s_4, s_9\}$	22	$\{s_3, s_5, s_{10}\}$	26	$\{s_4, s_9, s_{11}\}$	25	$\{s_7, s_9, s_{11}\}$	22
$\{s_1, s_5, s_6\}$	23	$\{s_2, s_4, s_{10}\}$	24	$\{s_3, s_5, s_{11}\}$	26	$\{s_4, s_{10}, s_{11}\}$	27	$\{s_7, s_{10}, s_{11}\}$	24
$\{s_1, s_5, s_8\}$	21	$\{s_2, s_4, s_{11}\}$	24	$\{s_3, s_7, s_8\}$	24	$\{s_5, s_6, s_7\}$	23	$\{s_8, s_9, s_{10}\}$	24
$\{s_1, s_5, s_9\}$	20	$\{s_2, s_5, s_6\}$	23	$\{s_3, s_7, s_9\}$	23	$\{s_5, s_6, s_8\}$	25	$\{s_8, s_9, s_{11}\}$	24
$\{s_1, s_5, s_{10}\}$	22	$\{s_2, s_5, s_8\}$	21	$\{s_3, s_7, s_{10}\}$	25	$\{s_5, s_6, s_9\}$	24	$\{s_8, s_{10}, s_{11}\}$	26
$\{s_1, s_5, s_{11}\}$	22	$\{s_2, s_5, s_9\}$	20	$\{s_3, s_7, s_{11}\}$	25	$\{s_5, s_6, s_{10}\}$	26	$\{s_9, s_{10}, s_{11}\}$	25
$\{s_1, s_6, s_7\}$	22	$\{s_2, s_5, s_{10}\}$	22	$\{s_3, s_8, s_9\}$	25	$\{s_5, s_6, s_{11}\}$	26	$\{s_1, s_2, s_5, s_7$	25
$\{s_1, s_6, s_8\}$	24	$\{s_2, s_5, s_{11}\}$	22	$\{s_3, s_8, s_{10}\}$	27	$\{s_5, s_7, s_8\}$	21	$\{s_1, s_2, s_7, s_9$	25
$\{s_1, s_6, s_9\}$	23	$\{s_2, s_6, s_7\}$	22	$\{s_3, s_8, s_{11}\}$	27	$\{s_5, s_7, s_9\}$	20		
$\{s_1, s_6, s_{10}\}$	25	$\{s_2, s_6, s_8\}$	24	$\{s_3, s_9, s_{10}\}$	26	$\{s_5, s_7, s_{10}\}$	22		
$\{s_1, s_6, s_{11}\}$	25	$\{s_2, s_6, s_9\}$	23	$\{s_3, s_9, s_{11}\}$	26	$\{s_5, s_7, s_{11}\}$	22		

Table 2.1, the combined flow-rates of sensor nodes s_3 and s_6 satisfy A_{req}, i.e. $C(\{s_3, s_6\})$ = 20 = A_{req}. Thus, the second order combination $\{s_3, s_6\}$ of nodes s_3 and s_6 forms a valid node-set T_m. Therefore, the combination $\{s_3, s_6\}$ is refrained from being a part of any higher order combinations.

The next step involves the enumeration of Shortest Minimal Paths (*SMPs*) in order of their lengths from the nodes in T_m to the sink node.

2.5.2 Enumeration of Shortest Minimal Paths for a WSN

An approach is presented in this section to enumerate *SMPs* in order of their *lengths* from all nodes in each T_m to the sink node v. It starts by enumerating all minimal paths from every node $s_i \in N$ to the sink node by implementing the *backtracking* method. It then enumerates *SMPs* in order of their *lengths* from the nodes in T_m to the sink node. The proposed approach can be used for both bidirectional and directional networks. The proposed *SMPs* enumeration is a two-step process: 1) Enumeration of all possible minimal paths from each $s_i \in N$ to v, and 2) Enumeration of *SMPs* in order of their lengths from each $T_m \in T$ to v. Section 2.5.2.1 and Section 2.5.2.2 describe the *SMPs* enumeration for the proposed approach with proper illustrations.

2.5.2.1 Enumeration of Minimal Paths (*MPs*)

To enumerate the minimal paths from $s_i \in N$ to v, a depth-first-search technique is applied to construct a spanning tree rooted at the sink, and the results of each level of the constructed tree are stored in a specific format. This format helps to create a database of the entire network. The table constructed is then backtracked for each s_i to find the minimal paths from each $s_i \in N$ to v. The process of enumerating all minimal paths consists of three steps:

a) Tracing all the paths from the sink node to every node s_i in the network and storing them in a specific format (see **Algorithm 1.1** presented in Section 2.6).
b) Retrieving all the paths from the stored format by backtracking (see **Algorithm 1.2** presented in Section 2.6).
c) Storing the minimal paths from all paths information.

Step a) starts by tracing the nodes attached to the sink node. The neighbor node information is gathered by tracking the v^{th} column of the connection matrix, *CM* (see Table 2.2). For reference, the *currently* traced columns of Table 2.2 are indicated by an arrow (↓), and the elements are bold-faced. The elements of *CM* indicate whether the pair of vertices of the WSN is connected or not. This tracing process may lead to traversing the same nodes, again and again, leading to loop formation. To avoid this, each element in the v^{th} row of *CM* is set to "0" after tracing the connected nodes. The next level is formed by traversing the columns of the j^{th} node attached to the sink node. At each stage, the j^{th} row of the connection matrix is set to "0" in order to avoid traversing the same node again. When the last node of the search tree is met, or it does not find any alternative, the search stops at this stage and moves back to the previous level to find the next alternative and start the search process again. This process is continued until all the nodes are traversed, or all the entries in the *CM* are "0".

Each level's information is stored in terms of the variables *BranchNo*, *NodNo*, and *ParentBr*; see Table 2.3. *BranchNo* gives information of the corresponding number given to the branches as the method proceeds through the *CM*, *NodNo* gives information about

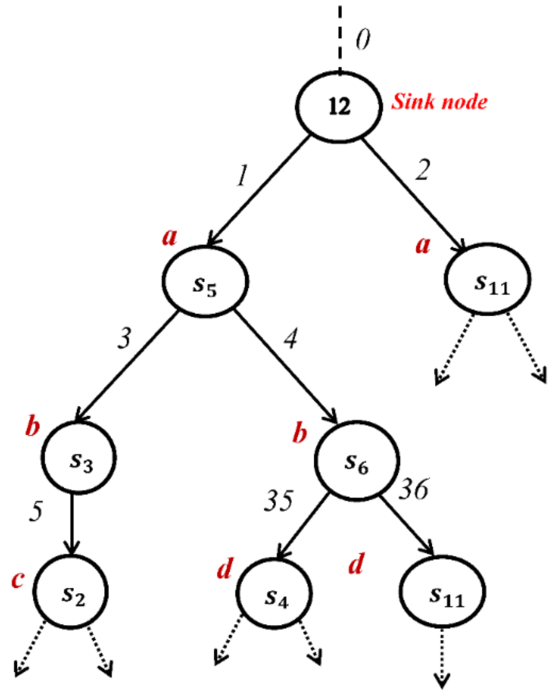

Figure 2.3 Tracing paths for WSN in Figure 2.2. *BranchNo* is indicated in italics. Input *CM* is shown above nodes in alphabets.

the current node, and *ParentBr* gives information about the *BranchNo* of the current node from which the current node has been branched. As an example, let us consider enumeration of the first three levels of the path search tree for the WSN in Figure 2.2. The *CM* for the WSN in Figure 2.2 is given in Table 2.2. The sink (v =12) forms the root of the search tree. The variable stored at the onset of the process is *BranchNo* = [*0*], *NodNo* = [12], and *ParentBr* = φ. *BranchNo* value '*0*' indicates the root of the search tree, i.e., the

Table 2.2 Intermediate CM for search tree in Figure 2.3

Connection Matrix for WSN in Figure 2.2

Currently traced node

#	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	**12**
1	0	1	0	0	0	0	1	0	0	0	0	**0**
2	1	0	1	1	0	0	0	0	0	0	0	**0**
3	0	1	0	0	1	0	0	0	0	0	0	**0**
4	0	1	0	0	0	1	1	0	0	0	0	**0**
5	0	0	1	0	0	1	0	0	0	0	0	**1**
6	0	0	0	1	1	0	0	0	0	0	1	**0**
7	1	0	0	1	0	0	0	1	0	0	0	**0**
8	0	0	0	0	0	0	1	0	1	0	0	**0**
9	0	0	0	0	0	0	0	1	0	1	0	**0**
10	0	0	0	0	0	0	0	0	1	0	1	**0**
11	0	0	0	0	0	1	0	0	0	1	0	**1**
12	0	0	0	0	1	0	0	0	0	0	1	**0**

CM(a)

#	s_1	s_2	s_3	s_4	$\mathbf{s_5}$	s_6	s_7	s_8	s_9	s_{10}	s_{11}	12
s_1	0	1	0	0	**0**	0	1	0	0	0	0	0
s_2	1	0	1	1	**0**	0	0	0	0	0	0	0
s_3	0	1	0	0	**1**	0	0	0	0	0	0	0
s_4	0	1	0	0	**0**	1	1	0	0	0	0	0
s_5	0	0	1	0	**0**	1	0	0	0	0	0	1
s_6	0	0	0	1	**1**	0	0	0	0	0	1	0
s_7	1	0	0	1	**0**	0	0	1	0	0	0	0
s_8	0	0	0	0	**0**	0	1	0	1	0	0	0
s_9	0	0	0	0	**0**	0	0	1	0	1	0	0
s_{10}	0	0	0	0	**0**	0	0	0	1	0	1	0
s_{11}	0	0	0	0	**0**	1	0	0	0	1	0	1
12	0	0	0	0	**0**	0	0	0	0	0	0	0

CM(b)

#	s_1	s_2	$\mathbf{s_3}$	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	12
s_1	0	1	**0**	0	0	0	1	0	0	0	0	0
s_2	1	0	**1**	1	0	0	0	0	0	0	0	0
s_3	0	1	**0**	0	1	0	0	0	0	0	0	0
s_4	0	1	**0**	0	0	1	1	0	0	0	0	0
s_5	0	0	**0**	0	0	0	0	0	0	0	0	0
s_6	0	0	**0**	1	1	0	0	0	0	0	1	0
s_7	1	0	**0**	1	0	0	0	1	0	0	0	0
s_8	0	0	**0**	0	0	0	1	0	1	0	0	0
s_9	0	0	**0**	0	0	0	0	1	0	1	0	0
s_{10}	0	0	**0**	0	0	0	0	0	1	0	1	0
s_{11}	0	0	**0**	0	0	1	0	0	0	1	0	1
12	0	0	**0**	0	0	0	0	0	0	0	0	0

CM(c)

#	s_1	$\mathbf{s_2}$	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	12
s_1	0	**1**	0	0	0	0	1	0	0	0	0	0
s_2	1	**0**	1	1	0	0	0	0	0	0	0	0
s_3	0	**0**	0	0	0	0	0	0	0	0	0	0
s_4	0	**1**	0	0	0	1	1	0	0	0	0	0
s_5	0	**0**	0	0	0	0	0	0	0	0	0	0
s_6	0	**0**	0	1	1	0	0	0	0	0	1	0
s_7	1	**0**	0	1	0	0	0	1	0	0	0	0
s_8	0	**0**	0	0	0	0	1	0	1	0	0	0
s_9	0	**0**	0	0	0	0	0	1	0	1	0	0
s_{10}	0	**0**	0	0	0	0	0	0	1	0	1	0
s_{11}	0	**0**	0	0	0	1	0	0	0	1	0	1
12	0	**0**	0	0	0	0	0	0	0	0	0	0

CM(d)

#	s_1	s_2	s_3	s_4	s_5	$\mathbf{s_6}$	s_7	s_8	s_9	s_{10}	s_{11}	12
s_1	0	1	0	0	0	**0**	1	0	0	0	0	0
s_2	1	0	1	1	0	**0**	0	0	0	0	0	0
s_3	0	1	0	0	1	**0**	0	0	0	0	0	0
s_4	0	1	0	0	0	**1**	1	0	0	0	0	0
s_5	0	0	0	0	0	**0**	0	0	0	0	0	0
s_6	0	0	0	0	0	**0**	0	0	0	0	0	0
s_7	1	0	0	1	0	**0**	0	1	0	0	0	0
s_8	0	0	0	0	0	**0**	1	0	1	0	0	0
s_9	0	0	0	0	0	**0**	0	1	0	1	0	0
s_{10}	0	0	0	0	0	**0**	0	0	1	0	1	0
s_{11}	0	0	0	0	0	**1**	0	0	0	1	0	1
12	0	0	0	0	0	**0**	0	0	0	0	0	0

sink node. By traversing the 12th column of the *CM* shown in Table 2.2(a), the neighbor node information gathered is s_5 and s_{11}. This then leads to the formation of the tree shown in Figure 2.3 with two branches: *1* and *2*. The currently stored variables are *BranchNo* = [*0, 1, 2*], *NodNo* = [12, s_5, s_{11}], and *ParentBr* = [φ, *0, 0*]. To avoid traversing node #12 again, the 12th row of the *CM* is set to “0”. The *CM* now takes the values as *CM*(a) as given in Table 2.2. The current node whose neighboring nodes information has to be gathered is s_5 as it is the child of the already traversed node #12.This leads to the formation of the second level of the search tree (see Figure 2.3) and *CM*(b) (see Table 2.2). The current values of stored variables are *BranchNo* = [*0, 1, 2, 3, 4*], *NodNo* = [12, s_5, s_{11}, s_3, s_6], and *ParentBr* = [φ, *0, 0, 1, 1*]. The next node to be traversed is node s_3 (column #3) in *CM*(b). This leads to the formation of the third level of the search tree (see Figure 2.3) and *CM*(c)

Table 2.3 Path Tracing for the WSN in Figure 2.2

BranchNo	***0***	*1*	***2***	*3*	*4*	*5*	*6*	*7*	*8*	*11*	*12*	*13*	*14*
NodNo	**12**	s_5	$\boldsymbol{s_{11}}$	s_3	s_6	s_2	s_1	s_4	s_7	s_6	s_{11}	s_{10}	s_9
ParentBr	**–**	*0*	***0***	*1*	*1*	*3*	*5*	*5*	*6*	*9*	*11*	*12*	*13*
BranchNo	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*26*	*27*	*28*	*29*
NodNo	s_8	s_9	s_{10}	s_{11}	s_6	s_4	s_6	s_7	s_{11}	s_8	s_7	s_1	s_1
ParentBr	*14*	*10*	*16*	*17*	*18*	*19*	*7*	*7*	*21*	*25*	*26*	*27*	*22*
BranchNo	*30*	*31*	*32*	*33*	*34*	*35*	*36*	*37*	*38*	*41*	*42*	*43*	*44*
NodNo	s_8	s_9	s_{10}	s_{11}	s_6	s_4	s_{11}	s_2	s_7	s_7	s_8	s_9	s_{10}
ParentBr	*22*	*30*	*31*	*32*	*33*	*4*	*4*	*35*	*35*	*39*	*41*	*42*	*43*
BranchNo	*45*	*46*	*47*	*48*	*49*	*50*	*51*	*52*	*53*	*56*	*57*	*58*	*59*
NodNo	s_{11}	s_1	s_8	s_2	s_3	s_9	s_{10}	s_{11}	s_{10}	s_7	s_1	s_4	s_2
ParentBr	*44*	*38*	*38*	*46*	*48*	*47*	*50*	*51*	*36*	*55*	*56*	*56*	*57*
BranchNo	*60*	*61*	*62*	*63*	*64*	*65*	***66***	*67*	*68*	*71*	*72*	*73*	*74*
NodNo	s_3	s_4	s_2	s_1	s_3	s_6	$\boldsymbol{s_{10}}$	s_4	s_5	s_1	s_3	s_7	s_8
ParentBr	*59*	*59*	*58*	*62*	*62*	*2*	***2***	*65*	*65*	*69*	*69*	*71*	*73*
BranchNo	*75*	*76*	*77*	*78*	*79*	*80*	*81*	*82*	*83*	*86*	*87*	*88*	*89*
NodNo	s_9	s_{10}	s_5	s_1	s_8	s_2	s_3	s_5	s_9	s_2	s_1	s_4	s_7
ParentBr	*74*	*75*	*72*	*70*	*70*	*78*	*80*	*81*	*79*	*85*	*86*	*86*	*87*
BranchNo	*90*	*91*	*92*	*93*	*94*	*95*	*96*	*97*	*98*	***101***	*102*	***103***	*104*
NodNo	s_4	8	s_9	s_{10}	s_7	s_1	s_8	s_9	s_{10}	$\boldsymbol{s_7}$	s_1	$\boldsymbol{s_4}$	s_2
ParentBr	*89*	*89*	*91*	*92*	*88*	*94*	*94*	*96*	*97*	***100***	*101*	***101***	*102*
BranchNo	*105*	*106*	*107*	*108*	*109*	*110*	*111*	***113***	*114*	*115*			
NodNo	s_3	s_4	s_5	s_6	s_4	s_6	s_5	$\boldsymbol{s_2}$	s_6	$\boldsymbol{s_1}$			
ParentBr	*104*	*104*	*105*	*107*	*108*	*106*	*110*	***103***	*103*	***113***			
BranchNo	*116*	*117*	*118*	*119*	*120*	*121*	*122*						
NodNo	s_3	s_5	s_6	s_5	s_3	s_2	s_1						
ParentBr	*113*	*116*	*117*	*114*	*119*	*120*	*121*						

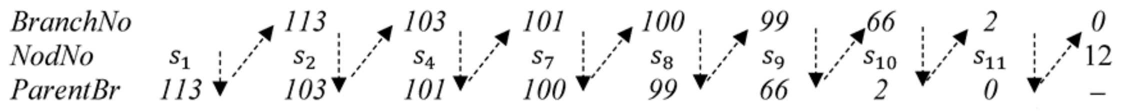

Figure 2.4 Backtracking for path enumeration

(see Table 2.2). The current values of stored variables are *BranchNo* = [*0, 1, 2, 3, 4, 5*], *NodNo* = $[12, s_5, s_{11}, s_3, s_6, s_2]$, and *ParentBr* = [φ, *0, 0, 1, 1, 3*]. This *CM*(c) is now used to find the neighbors connected to node s_2. The process continues until all entries of the *CM*(c) become zero. Then, it returns to *CM*(d) to find the nodes attached to node s_6. Continuing this way, the path search tree is constructed and the entire level information of the WSN in Figure 2.2 is stored in the specific format, as shown in Table 2.3.

In Step b), all paths from each node v to the sink node are retrieved by *backtracking* the stored format from each sensor s_i to the sink node. As an example, consider the listing of a possible path from sensor s_1 to the sink node for the WSN in Figure 2.2 from the stored format in Table 2.3. From Table 2.3, start tracking back from any column (say 115th column of *BranchNo*) of *NodNo* having the value 's_1' as we are interested in enumerating paths from sensor s_1 to the sink node. The sequence of backtracking is *NodNo* → *ParentBr* → *BranchNo* → *NodNo* → *ParentBr* → *BranchNo* → … → until *ParentBr* value is '0'. '0' indicates the root of the search tree, i.e., the sink node. The path thus formed is $\{s_1, s_2, s_4, s_7, s_8, s_9, s_{10}, s_{11}, 12\}$, and its formation is depicted in Figure 2.4. For reference, the columns of Table 2.3 corresponding to Figure 2.4 are turned boldface. Following the same process, the set of all possible paths from sensor s_1 to the sink for the WSN in Figure 2.2 is given in the second column of Table 2.4. Once all the paths are found, minimal paths are enumerated by removing the supersets. These minimal paths are stored in increasing order of the number of hops required to reach the sink node in Step c). All *MPs* from node s_1 to the sink node is given in the third column of Table 2.4. The set of minimal paths from all nodes $s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}, s_{11}$ to the sink node, arranged in increasing order of the number of hops required to reach the sink node, is given in Table 2.5. Once Step 1) completes generating all path information of the WSN, the next task, Step 2), is to enumerate the *SMPs* in order of their lengths from each T_m to the sink node.

Table 2.4 List of paths from node s_1 to sink node $v = 12$

No.	All Paths	Minimal Paths
1	$\{s_1, s_2, s_3, s_5, s_6, s_4, s_7, s_8, s_9, s_{10}, s_{11}, 12\}$	$\{s_1, s_2, s_3, s_5, 12\}$
2	$\{s_1, s_2, s_4, s_7, s_8, s_9, s_{10}, s_{11}, 12\}$	$\{s_1, s_7, s_4, s_6, s_{11}, 12\}$
3	$\{s_1, s_7, s_8, s_9, s_{10}, s_{11}, 12\}$	$\{s_1, s_2, s_4, s_6, s_{11}, 12\}$
4	$\{s_1, s_7, s_4, s_2, s_3, s_5, s_6, s_{11}, 12\}$	$\{s_1, s_7, s_4, s_6, s_5, 12\}$
5	$\{s_1, s_2, s_3, s_5, s_6, s_{11}, 12\}$	$\{s_1, s_2, s_4, s_6, s_5, 12\}$
6	$\{s_1, s_7, s_4, s_6, s_{11}, 12\}$	$\{s_1, s_7, s_8, s_9, s_{10}, s_{11}, 12\}$
7	$\{s_1, s_2, s_4, s_6, s_{11}, 12\}$	
8	$\{s_1, s_2, s_4, s_7, s_8, s_9, s_{10}, s_{11}, s_6, s_5, 12\}$	
9	$\{s_1, s_7, s_8, s_9, s_{10}, s_{11}, s_6, s_5, 12\}$	
10	$\{s_1, s_7, s_4, s_6, s_5, 12\}$	
11	$\{s_1, s_2, s_4, s_6, s_5, 12\}$	
12	$\{s_1, s_7, s_4, s_2, s_3, s_5, 12\}$	
13	$\{s_1, s_7, s_8, s_9, s_{10}, s_{11}, s_6, s_4, s_2, s_3, s_5, 12\}$	
14	$\{s_1, s_2, s_3, s_5, 12\}$	

Table 2.5 MPs from every sensor $s_i \in N$ to $v = 12$ for WSN in Figure 2.2

From Node #	Minimal Paths	Hop Count	From Node #	Minimal Paths	Hop Count
$\boldsymbol{s_1}$	$\{s_1, s_2, s_3, s_5, 12\}$	4	s_7	$\{s_7, s_4, s_6, s_{11}, 12\}$	4
	$\{s_1, s_4, s_6, s_7, s_{11}, 12\}$	5		$\{s_7, s_4, s_6, s_5, 12\}$	4
	$\{s_1, s_2, s_4, s_6, s_{11}, 12\}$	5		$\{s_7, s_8, s_9, s_{10}, s_{11}, 12\}$	5
	$\{s_1, s_4, s_5, s_6, s_7, 12\}$	5		$\{s_7, s_4, s_2, s_3, s_5, 12\}$	5
	$\{s_1, s_2, s_4, s_5, s_6, 12\}$	5		$\{s_7, s_1, s_2, s_3, s_5, 12\}$	5
	$\{s_1, s_7, s_8, s_9, s_{10}, s_{11}, 12\}$	6	s_8	$\{s_8, s_9, s_{10}, s_{11}, 12\}$	4
$\boldsymbol{s_2}$	$\{s_2, s_3, s_5, 12\}$	3		$\{s_8, s_7, s_4, s_6, s_{11}, 12\}$	5
	$\{s_2, s_4, s_6, s_{11}, 12\}$	4		$\{s_8, s_7, s_4, s_6, s_5, 12\}$	5
	$\{s_2, s_4, s_6, s_5, 12\}$	4		$\{s_8, s_7, s_4, s_2, s_3, s_5, 12\}$	6
	$\{s_2, s_4, s_7, s_8, s_9, s_{10}, s_{11}, 12\}$	7		$\{s_8, s_7, s_1, s_2, s_3, s_5, 12\}$	6
	$\{s_2, s_1, s_7, s_8, s_9, s_{10}, s_{11}, 12\}$	7	s_9	$\{s_9, s_{10}, s_{11}, 12\}$	3
$\boldsymbol{s_3}$	$\{s_3, s_5, 12\}$	2		$\{s_9, s_8, s_7, s_4, s_6, s_{11}, 12\}$	6
	$\{s_3, s_2, s_4, s_6, s_{11}, 12\}$	5		$\{s_9, s_8, s_7, s_4, s_6, s_5, 12\}$	6
	$\{s_3, s_2, s_4, s_7, s_8, s_9, s_{10}, s_{11}, 12\}$	8		$\{s_9, s_8, s_7, s_4, s_2, s_3, s_5, 12\}$	7
	$\{s_3, s_2, s_1, s_7, s_8, s_9, s_{10}, s_{11}, 12\}$	8		$\{s_9, s_8, s_7, s_1, s_2, s_3, s_5, 12\}$	7
$\boldsymbol{s_4}$	$\{s_4, s_6, s_{11}, 12\}$	3	s_{10}	$\{s_{10}, s_{11}, 12\}$	2
	$\{s_4, s_5, s_6, 12\}$	3		$\{s_{10}, s_9, s_8, s_7, s_4, s_6, s_5, 12\}$	7
	$\{s_4, s_2, s_3, s_5, 12\}$	4		$\{s_{10}, s_9, s_8, s_7, s_4, s_2, s_3, s_5, 12\}$	8
	$\{s_4, s_7, s_8, s_9, s_{10}, s_{11}, 12\}$	6		$\{s_{10}, s_9, s_8, s_7, s_1, s_2, s_3, s_5, 12\}$	8
$\boldsymbol{s_5}$	$\{s_5, 12\}$	1	s_{11}	$\{s_{11}, 12\}$	1
$\boldsymbol{s_6}$	$\{s_6, s_{11}, 12\}$	2			
	$\{s_6, s_5, 12\}$	2			

2.5.2.2 Enumeration of *SMP*s

Each set $SMP(T_m)$ to the sink node is enumerated by *ORing* the *MPs* obtained from each node $s_i \in T_m$ to v. While doing so, a single $SMP_n \in SMP(T_m)$ to v is enumerated. Each SMP_n consists of sensors in ACTIVE and RELAY state, i.e., the nodes in T_m should work in the ACTIVE state, and the intermediate nodes should work in RELAY state to form a path between T_m and sink. At any point of time, due to energy depletion or hardware failure of any of the intermediate nodes, if this shortest path is unavailable, the WSN self-configures and transfers data through the next available *SMP*. Hence, it is important to enumerate the next available *SMPs* from T_m to sink. The next *SMPs* are enumerated by considering the failure of each RELAY node in the already enumerated *SMP*. This leads to the selection of only those minimal paths (from the all path information of the WSN) which do not contain one or more of these RELAY nodes. Then the set of *SMPs* in order of their lengths are enumerated by sequentially *ORing* these *MPs*. *Example* 1 explains this.

Table 2.6 Shortest Minimal Path for $T_4 = \{s_1, s_2, s_6\}$

	Node #			Shortest Minimal path for
Shortest Minimal Path	s_1	s_2	s_6	$T_4 = \{s_1, s_2, s_6\}$
	$\{s_1, s_2, s_3, s_5, 12\}$	$\{s_2, s_3, s_5, 12\}$	$\{s_6, s_{11}, 12\}$	$\{s_1, s_2, s_3, s_5, s_6, s_{11}, 12\}$
			$\{s_6, s_5, 12\}$	$\{s_1, s_2, s_3, s_5, s_6, 12\}$

Table 2.7 Enumeration of *SMP*s from node $T_4 = \{s_1, s_2, s_6\}$ to sink

Off Relay node s_3^R	
MPs from s_1	$\{s_1, s_4, s_6, s_7, s_{11}, 12\}$
	$\{s_1, s_2, s_4, s_6, s_{11}, 12\}$
	$\{s_1, s_4, s_5, s_6, s_7, 12\}$
	$\{s_1, s_2, s_4, s_5, s_6, 12\}$
	$\{s_1, s_7, s_8, s_9, s_{10}, s_{11}, 12\}$
MPs from s_2	$\{s_2, s_4, s_6, s_{11}, 12\}$
	$\{s_2, s_4, s_6, s_5, 12\}$
	$\{s_2, s_4, s_7, s_8, s_9, s_{10}, s_{11}, 12\}$
	$\{s_2, s_1, s_7, s_8, s_9, s_{10}, s_{11}, 12\}$
MPs from s_6	$\{s_6, s_{11}, 12\}$
	$\{s_6, s_5, 12\}$
Shortest Minimal Pathset	$\{s_1, s_2, s_4, s_6, s_{11}, 12\}$
	$\{s_1, s_2, s_4, s_6, s_5, 12\}$
	$\{s_1, s_2, s_6, s_7, s_8, s_9, s_{10}, s_{11}, 12\}$

Off Relay node s_5^R	
MPs from s_1	$\{s_1, s_4, s_6, s_7, s_{11}, 12\}$
	$\{s_1, s_2, s_4, s_6, s_{11}, 12\}$
	$\{s_1, s_7, s_8, s_9, s_{10}, s_{11}, 12\}$
MPs from s_2	$\{s_2, s_4, s_6, s_{11}, 12\}$
	$\{s_2, s_4, s_7, s_8, s_9, s_{10}, s_{11}, 12\}$
	$\{s_2, s_1, s_7, s_8, s_9, s_{10}, s_{11}, 12\}$
MPs from s_6	$\{s_6, s_{11}, 12\}$
Shortest Minimal Pathset	$\{s_1, s_2, s_4, s_6, s_{11}, 12\}$
	$\{s_1, s_2, s_6, s_7, s_8, s_9, s_{10}, s_{11}, 12$

Example 1: Let us consider the valid node-set $T_4 = \{s_1, s_2, s_6\}$ of the WSN shown in Figure 2.2 for A_{req}= 20. The second, third, and fourth columns of Table 2.6 show the *SMP*s from source nodes s_1, s_2, and s_6, respectively. The last column shows the enumerated *SMP*s for T_4. As there are two shortest paths from node s_6 to the sink node, two *SMP*s are enumerated. However, the minimal path $\{s_1^A, s_2^A, s_3^R, s_5^R, s_6^A, s_{11}^R, 12\}$ turns out to be redundant as it is a superset of $\{s_1^A, s_2^A, s_3^R, s_5^R, s_6^A, 12\}$. After removing redundant paths, the only *SMP* from T_4 to the sink node is $\{s_1^A, s_2^A, s_3^R, s_5^R, s_6^A, 12\}$. The superscripts 'A' and 'R' indicate the ACTIVE and RELAY states of the sensor nodes in the *SMP*s.

To enumerate the set of all *SMP*s in order of their lengths, only those minimal paths from the source nodes s_1, s_2, and s_6 are considered in which any of the RELAY nodes s_3^R and s_5^R, is not present. Recall that this leads to the enumeration of the next available *SMP*s, through which $T_4 = \{s_1, s_2, s_6\}$ transfers its gathered data to the sink on the failure of the RELAY nodes s_3^R and s_5^R. Table 2.7 gives the minimal paths from the source nodes s_1, s_2, and s_6, which do not contain s_3^R and s_5^R. When node s_3^R is not operating, the SMPs enumerated are $\{s_1^A, s_2^A, s_6^A, s_7^R, s_8^R, s_9^R, s_{10}^R, s_{11}^R, 12\}$, $\{s_1^A, s_2^A, s_4^R, s_6^A, s_{11}^R, 12\}$ and $\{s_1^A, s_2^A, s_4^R, s_5^R, s_6^A, 12\}$. Similarly, if s_5^R does not work then the *SMP*s enumerated are $\{s_1^A, s_2^A, s_6^A, s_7^R, s_8^R, s_9^R, s_{10}^R, s_{11}^R, 12\}$, and $\{s_1^A, s_2^A, s_4^R, s_6^A, s_{11}^R, 12\}$. After removing redundancies, the SMP enumerated from $T_4 = \{s_1, s_2, s_6\}$ to the sink node $v = 12$ in order of increasing lengths are $SMP(T_4) = \{s_1^A, s_2^A, s_3^R, s_5^R, s_6^A, 12\}, \{s_1^A, s_2^A, s_4^R, s_5^R, s_6^A, 12\}$, $\{s_1^A, s_2^A, s_4^R, s_6^A, s_{11}^R, 12\}$, and $\{s_1^A, s_2^A, s_6^A, s_7^R, s_8^R, s_9^R, s_{10}^R, s_{11}^R, 12\}$. ■

These *SMPs*' information as well as the node-state information in each of these enumerated *SMPs* are stored in a node-state matrix, denoted by *NdState*. This *NdState* matrix is used for enumerating the disjoint sets discussed in the next section.

2.5.3 Enumeration of Disjoint-Sets

The proposed disjointing process is a single-variable inversion sum-of-disjoint products process in which variables are inverted sequentially one at a time. The node-state matrix, *NdState* is used to find the disjointed terms for the symbolic reliability expression of the WSN. The rows of *NdState* matrix correspond to the nodes contained in the *SMPs*, and the columns show the nodes present in that corresponding *SMP*. Each non-zero entry in *NdState* gives the node-states.

For ease of computer operation, the 'ACTIVE' state of any sensor s_i is denoted by '***3***', the 'RELAY' state by '***2***', the 'FAIL' state by '***1***'; in bold italic. Note that '***-1***' denotes the node is not present in the path.

For example, $SMP(T_4) = \{s_1^A, s_2^A, s_3^R, s_5^R, s_6^A, 12\}, \{s_1^A, s_2^A, s_4^R, s_5^R, s_6^A, 12\}$, $\{s_1^A, s_2^A, s_4^R, s_6^A, s_{11}^R, 12\}$, and $\{s_1^A, s_2^A, s_6^A, s_7^R, s_8^R, s_9^R, s_{10}^R, s_{11}^R, 12\}$ enumerated in Example 1 is stored in *NdState* as {***3 3 2 -1 2 3 -1 -1 -1 -1 -1***}, {***3 3 -1 2 2 3 -1 -1 -1 -1 -1***}, {***3 3 -1 2 -1 3 -1 -1 -1 -1 2***}, and {***3 3 -1 -1 -1 3 2 2 2 2 2***}, respectively. Note that the sink has unlimited resources and is always reliable. Hence, *NdState* is devoid of the state of the sink node.

At any step *k* of the disjointing process, the product $NdState_k$ is made disjoint with each of the products $NdState_1$, $NdState_2$,…, $NdState_{k-1}$. In the disjointing process, $NdState_k$ is expanded to a set of disjointed terms PD_k. Each product in PD_k is disjoint to $NdState_1$, $NdState_2$,…, $NdState_{k-1}$. The disjointing process is accomplished using Theorems 3 and 4. While enumerating the disjoint set PD_k, variables of each term of $P_i \in PD_k$, and $NdState_j$ ($1 \le j \le k$-1) are compared. If P_i and $NdState_j$ are disjoint, then P_i is retained in PD_k, and P_{i+1} is compared with $NdState_j$. If P_i is a superset of $NdState_j$, then it is deleted from PD_k. If P_i and $NdState_j$ are to be made disjoint, then P_i is replaced by terms enumerated by *Theorem* 3 and *Theorem* 4.

It can be proved that the ACTIVE, RELAY, and FAIL states of any node are mutually disjoint, i.e., if a node is in one state it cannot be in any other state at the same time. *Theorem* 1 and *Theorem* 2, and few disjointing rules, which are used in the proposed disjointing process, are presented next. Table 2.8 provides a node-state decision truth table where the transceiver or the communication unit, and sensing unit of a sensor node are denoted by *C* and *S*, respectively.

Theorem 1: ACTIVE *state of any node s_i, is disjoint to its* RELAY *state and* FAIL *state*.
Proof: From the third column of Table 2.8, it can be seen that the ACTIVE state is denoted by CS, and its disjointed form is represented by $\overline{CS}$. To prove the theorem, it is sufficient to prove the following:

Table 2.8 Node State Decision Truth Table

C	S	Node-State	Denoted by	
			Boolean	Code
0	0	FAIL	$\overline{C}$	***1***
0	1	FAIL		
1	0	RELAY	$C\overline{S}$	***2***
1	1	ACTIVE	CS	***3***

$$\overline{CS} = C\overline{S} + \overline{C}. \tag{2.8}$$

We start from l.h.s. of Equation (2.8). From Boolean algebra we have,

$$\begin{aligned} \overline{CS} &= 1 - CS \\ &= C + \overline{C} - CS \\ &= C(1-S) + \overline{C} \\ &= C\overline{S} + \overline{C} \\ &= r.h.s. \end{aligned}$$

■

***Theorem* 2**: RELAY *state of any node* s_i *is disjoint to its* ACTIVE *and* FAIL *state*.

Proof: From the third column of Table 2.8, it can be seen that RELAY state is denoted by $C\overline{S}$. Its disjointed form is represented by $\overline{C\overline{S}}$. To prove the theorem it is sufficient to prove the following:

$$\overline{C\overline{S}} = CS + \overline{C}. \tag{2.9}$$

We start from l.h.s. of Equation (2.9). From Boolean algebra we have,

$$\begin{aligned} \overline{C\overline{S}} &= 1 - C\overline{S} \\ &= C + \overline{C} - C\overline{S} \\ &= C(1-\overline{S}) + \overline{C} \\ &= CS + \overline{C} \\ &= r.h.s \end{aligned}$$

■

From *Theorem* 1 and *Theorem* 2, and Table 2.8, the following rules can be formulated, which governs the disjointing process.

Rule 1: $\overline{\boldsymbol{3}} = \boldsymbol{2} \bigcup \boldsymbol{1}$. ■

Rule 2: $\overline{\boldsymbol{2}} = \boldsymbol{3} \bigcup \boldsymbol{1}$. ■

Rule 3: ***1*** is disjoint to both ***2*** and ***3***. ■

Rule 4: ***2*** is disjoint to both ***1*** and ***3***. ■

Rule 5: ***3*** is disjoint to both ***1*** and ***2***. ■

Rule 6: If $P_i \supseteq NdState_j$, then drop P_i from PD_k. ■

Theorem 3: *If $NdState_j$ and P_i are not disjoint, and $X \equiv \{x_u, x_v, \ldots, x_w\}$ be the set of variables present in the* ACTIVE *state in S_j and not present in P_i, then $NdState_j \cup P_i = NdState_j \cup \boldsymbol{1}_u P_i \cup \boldsymbol{2}_u P_i \cup \boldsymbol{3}_u \boldsymbol{1}_v P_i \cup \boldsymbol{3}_u \boldsymbol{2}_v P_i \cup \boldsymbol{3}_u \boldsymbol{3}_v \boldsymbol{1}_w P_i \cup \boldsymbol{3}_u \boldsymbol{3}_v \boldsymbol{2}_w P_i$, and all the products in the r.h.s. are mutually disjoint*. *Example* 2 explains *Theorem* 3.

Proof: The equality condition holds if $NdState_j$ and P_i are both FALSE or if $NdState_j$ is TRUE regardless of P_i. Now, we need to prove the equality when $NdState_j$ is FALSE, and P_i is TRUE. This happens only when one or more of the variables x_u, x_v, …, x_w is FALSE. If x_u is FALSE, then $\overline{x_u}$ is TRUE, the r.h.s. is TRUE. If x_u is TRUE, then if x_v is FALSE, the r.h.s. is again TRUE. Continuing in this way, we find that the r.h.s. has to be TRUE. ■

Theorem 4: *If $NdState_j$ and P_i are not disjoint, and $X \equiv \{x_u, x_v, \ldots, x_w\}$ be the set of variables present in the* RELAY *state in S_j and not present in P_i, then $NdState_j \cup P_i = NdState_j \cup \boldsymbol{1}_u P_i \cup \boldsymbol{3}_u P_i \cup \boldsymbol{2}_u 1_v P_i \cup \boldsymbol{2}_u \boldsymbol{3}_v P_i \cup \boldsymbol{2}_u \boldsymbol{2}_v \boldsymbol{1}_w P_i \cup \boldsymbol{2}_u \boldsymbol{2}_v \boldsymbol{3}_w P_i$ and all the products in the r.h.s. are mutually disjoint*.

Proof: The equality holds if $NdState_j$ and P_i are both FALSE or if $NdState_j$ is TRUE regardless of P_i. Now, we need to prove the equality when $NdState_j$ is FALSE and P_i is TRUE. This happens only when one or more of the variables x_u, x_v, …, x_w are FALSE. If x_u is FALSE, then $\overline{x_u}$ is TRUE, the r.h.s. is TRUE. If x_u is TRUE, then if x_v is FALSE, the r.h.s. is again TRUE. Continuing in this way, we find that the r.h.s. has to be TRUE. ■

Table 2.9 Disjointing process of *NdState*3

Comparing with	X	Disjointed Terms
		s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9 s_{10} s_{11}
		1. ***-1 1 1 3 3 -1 -1 -1 -1 -1 -1***
*NdState*1	$\{s_2, s_3\}$	2. ***-1 2 1 3 3 -1 -1 -1 -1 -1 -1***
		3. ***-1 3 1 3 3 -1 -1 -1 -1 -1 -1***
		4. ***-1 3 2 3 3 -1 -1 -1 -1 -1 -1***
		1. ***-1 1 1 3 3 3 -1 -1 -1 -1 -1***
		2. ***-1 1 2 3 3 3 -1 -1 -1 -1 -1***
	1. $\{s_3, s_{11}\}$	3. ***-1 1 3 3 3 3 -1 -1 -1 -1 1***
		4. ***-1 1 3 3 3 3 -1 -1 -1 -1 2***
		5. ***-1 2 1 3 3 3 -1 -1 -1 -1 -1***
*NdState*2		6. ***-1 2 2 3 3 3 -1 -1 -1 -1 -1***
	2. $\{s_3, s_{11}\}$	7. ***-1 2 3 3 3 3 -1 -1 -1 -1 1***
		8. ***-1 2 3 3 3 3 -1 -1 -1 -1 2***
	3. disjoint	9. ***-1 3 1 3 3 -1 -1 -1 -1 -1-1***
	4. disjoint	10. ***-1 3 2 3 3 -1 -1 -1 -1 -1-1***

Example 2: Consider the disjointing process of *NdState*3 = {***-1 -1 -1 3 3 3 -1 -1 -1 -1 -1***} for the WSN given in Figure 2.2. While comparing *NdState*3 with *NdState*1 = {***-1 3 3 -1 3 -1 -1 -1 -1 -1 -1***}, the resulting set of nodes (present in *NdState*1 but not in *NdState*3) are X = {2, 3}. The product, *NdState*3, is then replaced by four products (*PD*3). Each of these four terms is mutually disjoint and disjoint with *NdState*1. The enumerated *PD*3 is then disjointed with *NdState*2 = {***-1 -1 3 -1 3 -1 -1 -1 -1 -1 3***}. The product *NdState*3 is then replaced by ten products (*PD*3). All these ten terms are mutually disjoint and disjoint with *NdState*1 and *NdState*2. The disjointing process and the disjointed terms are shown in Table 2.9.

2.6 Algorithms

This section provides the pseudocode of our proposed algorithm, i.e., **Multi Node-State Reliability Evaluator** (**MNRE**). **MNRE** uses two algorithms, **Algorithm 1** and

Algorithm 2, to evaluate $WSNRel$. **Algorithm 1** shows the process of enumerating all *SMPs* in order of their lengths. **Algorithm 2** shows the disjointing process for evaluating $WSNRel$ from the *SMPs* enumerated in **Algorithm 1**. In the following, we explain the details of MNRE and its time complexity analysis.

Lines 2-22 of **Algorithm 1** generate each possible valid node-set T_i to ensure all possible *SMPs* are generated as required by (7). More specifically, *Line 2* generates first-order combination of nodes in N. *Lines 4-13* checks if any of the generated node-combination satisfies A_{req}, and forms the set T. *Lines 14-21* generates higher-order node combinations from the non-valid *NComb*s. Then, *Line 23* uses a recursive function **Enum_MP_a()**, shown in **Algorithm 1.1**, to generate all possible *MPs*, and store them in arrays *BranchNo, NodNo,* and *ParentBr*, i.e., the tabular format discussed in Section 2.5.2.1. Note that **Enum_MP_a()** implements Step a) of Step 1).

Lines 25-28 then use the obtained *BranchNo, NodNo,* and *ParentBr* to generate all *MPs* for each node s_i and the number of hops from each node s_i to sink for each enumerated *MP*. More specifically, for each node s_i, *Line 25* uses function **Enum_MP_b(),** shown in **Algorithm 1.2**, to generate two variables, MP_{s_i} and Hop_{s_i}, that store the *MPs* and *#Hops* of node s_i respectively. Then, *Lines 26* and *27* stores the *MPs* and their #hops in variables *MPs*[s_i, 1] and *MPs*[s_i, 2], respectively. Note that the first column of the variable *MPs*, i.e., *MPs*[s_i, 1] stores a set of *MPs* from node s_i to v, and the second column of the variable *MPs*, i.e., *MPs*[s_i, 2] stores hop count for each corresponding *MP* stored in *MPs*[s_i, 1]. To mention here, **Enum_MP_b()** implements Step b) of Step 1) discussed in Section 2.5.2.1.

Finally, *Lines 29-44* generate each set $SMP(T_m)$ for each valid node-set T_m. Specifically, *Line 30* checks if T_m is directly connected to v. If T_m is directly connected to v, then T_m itself becomes an *SMP* and its node-state information, i.e., *State*[m, s_i] is appended in matrix *NdState*, as given in *Lines 31-32*. Otherwise, *Lines 33-43* generate set $SMP(T_m)$ for the T_m's that are not directly connected to v.

Specifically, *Line 34* of **Algorithm 1** calls function **Enum_SMP()**, to enumerate the *SMPs* for each T_m that is not directly connected to v. As shown in **Algorithm 1.3**, *Lines 1-2* find the shortest *MPs* from each $s_i \in T_m$ to v from the *MP* information stored in the *Lines 26-27* of **Algorithm 1**. *Line 3* then enumerates the first *SMP* from T_m to v by *OR*ing the shortest *MPs*. Note that the output of this line is a single *SMP*, as shown in

Algorithm 1 (Enum_SMP_WSN): Enumerates SMPs for WSN.

Input: $G = ((v) \cup N, L)$, represented as a Connection matrix - CM, and Sink Node - v.
Output: Shortest Minimal Paths - SMP and Node State matrix - $NdState$ - for WSN $G = ((v) \cup N, L)$.

```
1.  Initialize $SMP \leftarrow \varphi$, $T \leftarrow \varphi$, $NdState \leftarrow \varphi$, PrevNd $\leftarrow$ 0, CurCol $\leftarrow s$, and count $\leftarrow$ 1,

    // Step 1 of MNRE
2.  $Y \leftarrow$ Generate first-order combination from nodes in $N$
3.  While ($Y \neq \varphi$ )
4.       For each $NComb \in Y$
5.           If $C(NComb) \geq A_{req}$
6.             $Y \leftarrow Y - NComb$
7.             $T_m \leftarrow NComb$
8.             Append $T_m$ to $T$
9.             For each node $s_i \in T_m$
10.               $State[m, s_i] \leftarrow$ '3'
11.            End For
12.          End If
13.      End For
14.      $NY \leftarrow \varphi$
15.      For each pair of $NComb_a$ and $NComb_b$ in $Y$
16.          $NewNComb \leftarrow NComb_a \cup NComb_b$
17.          If $NewNComb \not\supset$ any $NComb$ in $NY$
18.             Append $NewNComb$ to $NY$
19.          End If
20.      End For
21.      $Y \leftarrow NY$
22. End while

    // Step 2 of MNRE
23. $[BranchNo, NodNo, ParentBr] \leftarrow$ Enum_MP_a (0, $v$, $CM$, $count$)
24. For each node $s_i \in N$
25.    $[MP_{s_i}, Hop_{s_i}] \leftarrow$ Enum_MP_b ($s_i$, BranchNo, NodNo, ParentBr)
26.    $MPs[s_i, 1] \leftarrow MP_v$
27.    $MPs[s_i, 2] \leftarrow Hop_v$
28. End For
29. For $T_m \in T$
30.    If each node $s_i \in T_m$ is directly connected to $v$
31.      Append $T_m$ to $SMP$
32.      Append $State[m, s_i]$ to $NdState$
33.    Else
34.      $SMP(T_m) \leftarrow$ Enum_SMP ($MPs$, $T_m$, $CM$)
35.      For each $SMP_n \in SMP(T_m)$
36.         $X \leftarrow SMP - T_m$
37.         For each node $v \in X$
38.            $State[m, s_i] \leftarrow$ '2'
39.         End For
40.         Append $State[m, s_i]$ to $NdState$
41.      End For
42.      Append $SMP(T_m)$ to $SMP$
43.    End If
44. End For
```

Algorithm 1.1 (Enum_MP_a): Stores all minimal path information in a specific format.

Input: Previously traced node - *PrevNd*, current tracing node - *CurCol*, Connection matrix – *CM*.
Output: An array of generated sequential nodes - *NodNo*, an array of parent branches – *ParentBr*, and Number of branches – *BranchNo*.

```
1.  PrevCount ← $count$
2.  For each node $s_i \in N$
3.      If $CM[s_i, CurCol] = 1$
4.          BranchNo[$count$] ← $count$
5.          ParentBr [$count$] ← PrevNd
6.          NodNo [$count$] ← $s_i$
7.          $count \leftarrow count + 1$
8.      End If
9.  End for
10. EndCount ← $count - 1$
11. Set each element at row ‘$CurCol$’ in $CM$ to 0
12. For $j$ = PrevCount to Endcount
13.     If $CM \neq 0$
14.         PrevNd ← BranchNo[$j$]
15.         $CurCol$ ← NodNo[$j$]
16.         Enum_MP_a ($PrevNd, CurCol, CM, count$)
17.     End If
18. End For
```

Algorithm 1.2 (Enum_MP_b): Enumerates all minimal paths by retrieving path information from the specific format.

Input: Any node ($s_i \in N$), an array of generated sequential nodes - *NodNo*, an array of parent branches – *ParentBr*, and Number of branches – *BranchNo*.
Output: A set of all paths of a WSN - *MP*, Length of each path –*Hop*.

```
1.  Initialize $Nodes \leftarrow \varphi$
2.  For each $s_i \in N$
3.      Initialize $k = 1$
4.      For each $NodNo_z \in NodNo$
5.          If $NodNo_z = s_i$
6.              $indx$ = z
7.              break
8.          End If
9.      End For
10.     Append element in ‘$indx$’ column of $NodNo$ to $MP_k$
11.     Set each element of $NodNo$ corresponding to $indx$ to 0
12.     X1 ← $ParentBr$ corresponding to $NodNo[indx]$
13.     X2 ← $NodNo$ corresponding to $BranchNo$[X1]
14.     If $NodNo[indx] \neq 0$
15.         Append X2 to $MP_k$
16.     End If
17.     If $ParentBr[indx] = 0$
18.         Append $MP_k$ to $MP_v$
19.         Append $|MP_k|$ to $Hop_v$
20.         $k = k + 1$
21.         break
22.     Else
23.         $indx = ParentBr[indx]$
24.         Repeat Lines 10 to 24
25.     End If
26. End For
```

Example 1. *Line 4* finds the nodes that work in RELAY state in the *SMP* enumerated in *Line 3*. This RELAY node information is used to enumerate the next available set of *SMPs* in order of their lengths in *Lines 6-10*. Once, all the *SMPs* are enumerated, the node state information in each *SMP* is generated (*Line 38*) and stored (*Line 40*) in *Lines 34-41* of **Algorithm 1**.

Algorithm MNRE then uses function **Enum_DisjointSets()**, shown in **Algorithm 2**, to evaluate $WSNRel$ as per (2.7) following the steps discussed in Section 2.5.3. Since the disjointing process is carried out sequentially, *Line 1* of **Algorithm 2** initializes the set *DS* with the first *SMP's* node-state information, i.e., *NdState*. The disjointing process starts by disjointing the second term in matrix *NdState* with the first one, and so on until the last element in *NdState* is disjointed with all its previous elements. This is accomplished by finding out the nodes that work in different states. **Algorithm 2** uses function **COMP()**, shown in **Algorithm 2.1**, to accomplish this. *Line 2* of **Algorithm 2.1** checks if PD is already disjoint with *NdState*. *Lines 5-9* finds out the elements in PD that do not match the elements in current *NdState*. *Lines 10-25* checks for the elements that need to be disjointed. Further, *Lines 26-28* and *Lines 29-31* implement *Theorem* 1 and *Theorem* 2, respectively, to enumerate the disjoint terms of the symbolic reliability expression. *Line 7* of **Algorithm 2** stores the enumerated disjoint terms.

The *correctness* of the **MNRE algorithm** requires 1) enumeration of all possible *SMP*s for any WSN and 2) enumeration of correct symbolic reliability expression ($WSNRel$) in terms of sum-of-disjoint products of the enumerated *SMP*s. For 1), **Algorithm 1** will always enumerate all possible paths for any WSN. Firstly, *Lines 2-22* ensures the generation of all possible flow-satisfying node combinations. Then **Algorithm 1.1** sequentially constructs the path-search spanning tree rooted at the sink and stores this information in a tabular format through *Lines 2-18*. **Algorithm 1** then uses **Algorithm 1.2** to enumerate all possible MPs from each node $s_i \in N$ to the sink node. After enumeration of all possible *MPs*, **Algorithm 1.3** through *Line 3* and *Lines 5-11* ensures enumeration of all possible *SMP*s for the WSN with multi-state nodes. For 2), *correctness* of the modified sum-of-disjoint products approach has been proved through Theorems 1-4.

Algorithm 1.3 (Enum_SMP): Enumerates the set of shortest minimal paths – $SMP(T_i)$ for each valid node-set - T_m.

Input: A set of minimal paths from all nodes along with their hop-count - MPs, valid node-set - T_m, and Connection matrix – CM.
Output: A set of minimal shortest paths – $SMPs$, in order of their lengths.

```
1. For each $s_i \in T_m$
      SPath[$m$] ← $MPs[s_i, 1]$
2.End For
3.$SMP$ ← Paths enumerated by $ORing\ SPath[m]$
4.          RN ← Find nodes working in RELAY state in $SMP$
5.          For each RN$_j$ ∈ RN
6.              For each $s_i \in T_m$
7.                  X ← $MPs[s_i, 1]$ from $v$ to $s$ that do not have RN$_j$
8.                  Enumerate $SPaths1$ by $OR$ing elements in X
9.                  Append $SPath1$ to $SMP$
10.             End For
11.  End For
```

Algorithm 2 (Enum_DisjointSets): Enumerates disjoint sets of the WSN

Input: Node State matrix – $NdState$.
Output: Disjoint Set of the network – DS.

```
     // Step 3 of MNRE
1.   Initialize $DS$ ← $NdState[1, ACol]$ //$ACol$ denotes all columns of $NdState$
2.For $i$ = 2 to |$NdState$| // |.| indicates number of rows in the $NdState$ matrix
3.      PD[$i$] ← $NdState[i, ACol]$
4.      For $j$ = 1 to $i - 1$
5.          PD ← COMP (PD, $NdState[j, ACol]$)
6.      End For
7.      Append PD to DS
8.   End For
```

The time complexity of **MNRE algorithm** is analyzed as follows. *Lines 2-22* of **Algorithm 1** generate up to $2^{|N|}$ combinations to generate $|T|$ number of T_ms. The total combinations will depend on the value of the ratio λ_i/A_{req}. A lower value of the ratio will lead to more number of combination formation and vice-versa. *Line 5* requires $O(|N|+1)$ for each combination. The time complexity of *Lines 9-11* depend on $|T|$, i.e., $O((|N|+1)\times|T|)$. Note that *Lines 14-21* do not generate any higher order combinations for each obtained T_m. However, in the worst case the checking at *Line 17* in *total* is done in $O(2^N)$ times. Therefore, *Line 2-22* in the worst case requires $O((|N|+1)\times 2^N + (|N|+1)\times |T| + 2^N) = O((|N|+1)\times 2^{|N|})$. The complexity of **Enum_MP_a()**, called in *Line 23*, depends on the final value of variable *count*, i.e., the size of array *BranchNo*, *ParentBr*, or *NodNo*. In the worst case, i.e., for a fully connected network, the value of *count* reaches

Algorithm 2.1 (**COMP**): Compares each term and gives a disjoint-set.

Input: A matrix containing the paths to be disjointed - X, an array containing the node-states – Y.
Output: a set of disjoint products – PD.

```
1.  For each P[$i$] $\in$ X
2.      If P[$i$] $\supseteq$ Y
3.          Drop P[$i$] and go to next P[$i$]
4.      Else
5.          For $k$ =1 to |P[$i$]|
6.              If P[$i$, $k$] $\neq$ Y [$k$]
7.                  $NMatch \leftarrow k$
8.              End If
9.          End For
10.         For each $NMatch_m \in NMatch$
11.             If P[$i$, $NMatch_m$] = 1 and Y[$NMatch_m$] = 3
12.                 PD1 $\leftarrow$ P[$i$]
13.                 break
14.             ElseIf
15.                 P[$i$, $NMatch_m$] = 1 and Y[$NMatch_m$] = 2
16.                 PD1 $\leftarrow$ P[$i$]
17.                 break
18.             ElseIf
19.                 P[$i$, $NMatch_m$] = -1 and Y[$NMatch_m$] = 3
20.                 $u \leftarrow NMatch_m$
21.             ElseIf
22.                 P[$i$, $NMatch_m$] = -1 and Y[$NMatch_m$] = 2
23.                 $t \leftarrow NMatch_m$
24.             End If
25.         End For
26.         For each value of $z$ =1 to |$u$| // Theorem 1
27.             PD2 $\leftarrow$ Replace P[$i$, $u_z$] by 1 and 2
28.         End For
29.         For each value of $w$ =1 to |$t$| // Theorem 2
30.             PD3 $\leftarrow$ Replace P[$i$, $t_w$] by 1 and 3
31.         End For
32.         PD = PD1 $\cup$ PD2 $\cup$ PD3
33.     End If
34. End For
```

$2^{|N|}$. Except for *Line 26* that takes in total $O((|N|+1)^2)$, the remaining *Lines* in **Algorithm 1.1** can be implemented in $O(1)$. Thus the time complexity of **Algorithm 1.1** is $O(2^N + (|N|+1)^2) = O(2^N)$. *Lines 24-28* of **Algorithm 1** in total takes $O(|N| \times Z)$, where Z is the total number of *MPs* from each node $s_i \in N$ to the sink node. In the worst case, i.e., for a fully connected network, Z reaches $O(2^{|E|-(|N|+1)+2})$ (Aggarwal *et al.* 1982). The time complexity of *Lines 29-44* depends on the total number of T_m's, i.e., $|T|$, and the time complexity of **Enum_SMP()**. The number of *SMPs* generated for each T_m is bounded by the maximum node degree of nodes in the T_m. Thus, there are at maximum $|T| \times |N|$ *SMPs*, since the maximum node degree is bounded by $|N|$. Thus, the time complexity of *Lines 29-44* is $O(|T| \times (|N| + 1))$ and the time complexity of **Algorithm 1** is $O((|N| + 1) \times 2^{|N|} + (|N| + 1) \times 2^{|E|-(|N|+1)+2} + (|N| + 1) \times |T|) = O((|N| + 1) \times 2^{|E|-(|N|+1)+2})$ because $2^{|E|-(|N|+1)+2} \geq 2^{|N|} \geq |T|$.

The time complexity of *Line 2* of **Algorithm 2** depend on i) the total number of elements in *NdState*, which in turn, depends on the number of *SMPs*, and ii) the total number of generated disjoint terms in *Line 5*. The complexity of function **COMP(),** called in *Line 5*, depends on the value of variable *X*, i.e., the size of matrix "PD". For each non-matching element in *X*, the **Algorithm 2.1** generates at most $2^{|N|}$ disjoint terms. In the worst case, the value of |PD| can reach up to $|N| \times 2^{|N|}$, as *Line 7* of **Algorithm 2.1** can reach at most $|N|$. Therefore, *Line 5* of **Algorithm 2** has time complexity of $O((|N| + 1) \times 2^{|N|})$. Thus the time complexity of *Lines 2-8*, in the worst case is $O(|T| \times |N|^2 \times 2^{|N|})$ because |*NdState*| = $|T| \times |N|$. Therefore, the time complexity of MNRE is $O((|N| + 1) \times 2^{|E|-(|N|+1)+2}$ + $O(|T| \times |N|^2 \times 2^{|N|})$.

2.7 Illustrative Example

This section illustrates the proposed approach for the WSN given in Figure 2.5 for A_{req} = 9. Following *Lines 2-22* of **Algorithm 1** and by sequentially combining the nodes to search for flow-satisfying combination, node-sets that satisfy the flow requirement of A_{req} = 9 are $T = \{\{s_1, s_2, s_4\}, \{s_1, s_2, s_5\}, \{s_1, s_3, s_4\}, \{s_1, s_4, s_5\}, \{s_2, s_4, s_5\}\}$ with corresponding flows of 11, 10, 9, 11 and 10. Since, third order combination of nodes satisfies the required flow, formation of higher order is refrained. The nodes in each T_m have to work in ACTIVE state to satisfy the flow. The node-states thus enumerated are *NdState* = {***3, 3, -1, 3, -1***}, {***3, 3, -1, -1, 3***}, {***3, -1, 3, 3, -1***}, {***3, -1, -1, 3, 3***} and {***-1, 3, -1, 3, 3***}.

Lines 23-44 of **Algorithm 1** involves enumeration of *SMPs*, which is explained with the help of Figure 2.6 – Figure 2.7, and Table 2.10 –Table 2.11. The search for paths starts by tracing the nodes connected to the sink node by traversing the column of *CM* corresponding to the sink node. After tracing the nodes connected to the sink node, rows of the *CM* corresponding to the sink node is made zero in order to avoid infinite loop formation.

This results in Table 2.10(a) which is now the input *CM* for node s_1. Nodes connected to node s_1 is found by tracing the first column of Table 2.10(a) which results in node s_2 as the neighbor. At this stage, Table 2.10(a) is modified to Table 2.10(b) to avoid infinite looping. Table 2.10(b) is now the input for node s_2 to search for its neighbors. This search process is continued until the last node is met, or it does not find any alternative. Once all the entries of the *CM* are zero, indicating the condition that the last node is met, it moves to the previous level and starts the search process again. This explains **Algorithm 1.1** in *Line 23* of **Algorithm 1**. Figure 2.6 visually illustrates the concept of tracing all paths from sink node 6 to nodes s_1, s_2, s_3, s_4 and s_5 for the WSN in Figure 2.5 and Table 2.11 stores each level's information for the search tree formed in Figure 2.6. Table 2.10 shows the intermediate connection matrices formed for the left branch of the search tree formed in Figure 2.6.

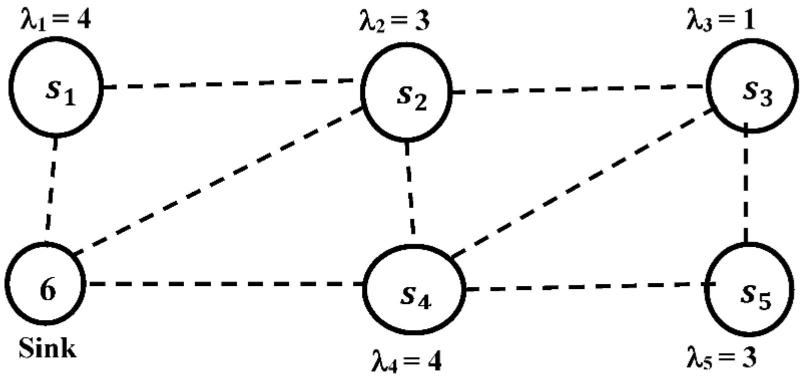

Figure 2.5 A WSN of six nodes. The traffic flow rate of each node is indicated by λ_i.

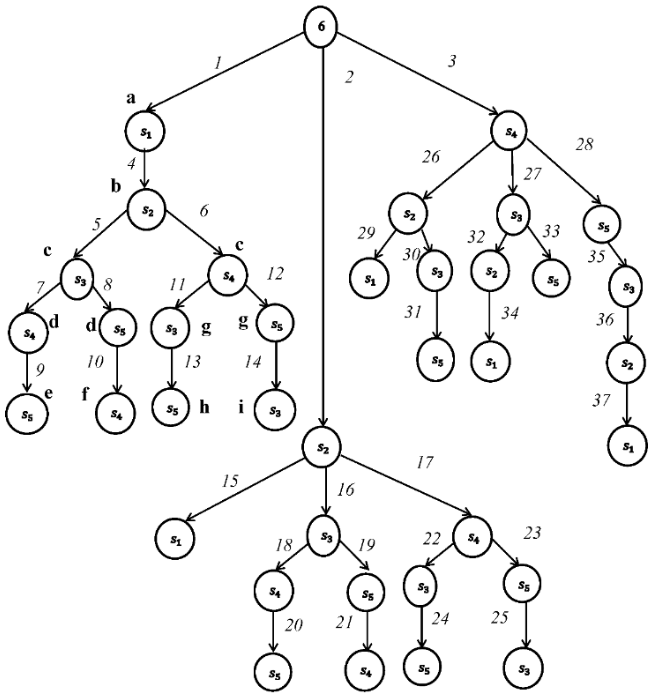

Figure 2.6 Tracing paths for the WSN in Figure 2.4. Branch numbers are indicated in italics. The input connection matrix is shown above nodes in alphabets.

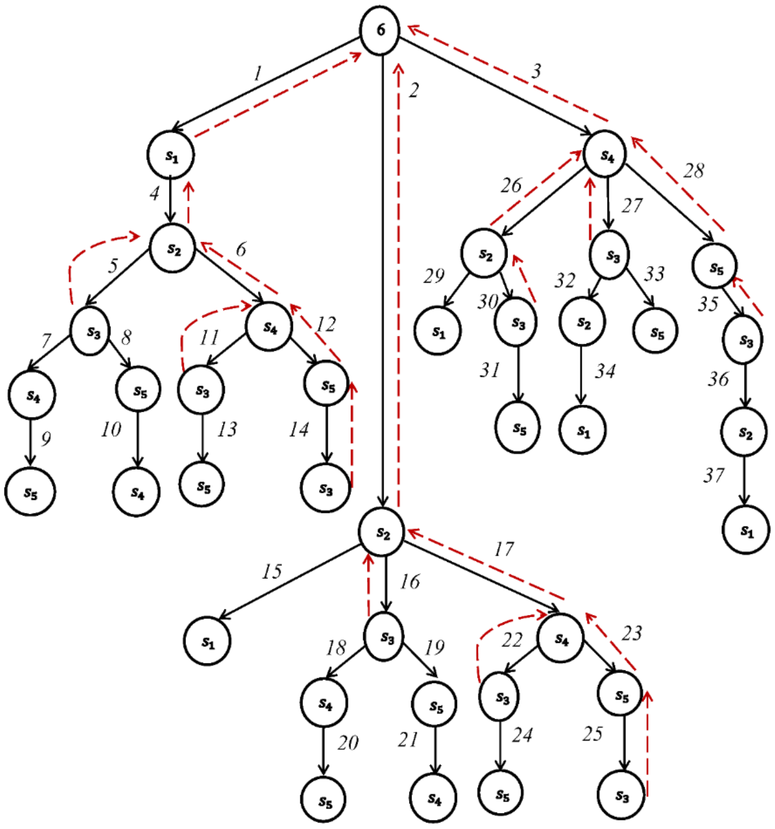

Figure 2.7 Retrieving paths for node s_3.

Table 2.10 Intermediate Connection Matrices for the left branch in Figure 2.5

Connection Matrix of WSN in Figure 2.5.

#	s_1	s_2	s_3	s_4	s_5	6
s_1	0	1	0	0	0	1
s_2	1	0	1	1	0	1
s_3	0	1	0	1	1	0
s_4	0	1	1	0	1	1
s_5	0	0	1	1	0	0
6	1	1	0	1	0	0

(a)

#	s_1	s_2	s_3	s_4	s_5	6
s_1	0	1	0	0	0	1
s_2	1	0	1	1	0	1
s_3	0	1	0	1	1	0
s_4	0	1	1	0	1	1
s_5	0	0	1	1	0	0
6	0	0	0	0	0	0

(b)

#	s_1	s_2	s_3	s_4	s_5	6
s_1	0	0	0	0	0	0
s_2	1	0	1	1	0	1
s_3	0	1	0	1	1	0
s_4	0	1	1	0	1	1
s_5	0	0	1	1	0	0
6	0	0	0	0	0	0

(c)

#	s_1	s_2	s_3	s_4	s_5	6
s_1	0	0	0	0	0	0
s_2	0	0	0	0	0	0
s_3	0	1	0	1	1	0
s_4	0	1	1	0	1	1
s_5	0	0	1	1	0	0
6	0	0	0	0	0	0

(d)

#	s_1	s_2	s_3	s_4	s_5	6
s_1	0	0	0	0	0	0
s_2	0	0	0	0	0	0
s_3	0	0	0	0	0	0
s_4	0	1	1	0	1	1
s_5	0	0	1	1	0	0
6	0	0	0	0	0	0

(e)

#	s_1	s_2	s_3	s_4	s_5	6
s_1	0	0	0	0	0	0
s_2	0	0	0	0	0	0
s_3	0	0	0	0	0	0
s_4	0	0	0	0	0	0
s_5	0	0	1	1	0	0
6	0	0	0	0	0	0

(f)

#	s_1	s_2	s_3	s_4	s_5	6
s_1	0	0	0	0	0	0
s_2	0	0	0	0	0	0
s_3	0	0	0	0	0	0
s_4	0	1	1	0	1	1
s_5	0	0	0	0	0	0
6	0	0	0	0	0	0

(g)

#	s_1	s_2	s_3	s_4	s_5	6
s_1	0	0	0	0	0	0
s_2	0	0	0	0	0	0
s_3	0	1	0	1	1	0
s_4	0	0	0	0	0	0
s_5	0	0	1	1	0	0
6	0	0	0	0	0	0

(h)

#	s_1	s_2	s_3	s_4	s_5	6
s_1	0	0	0	0	0	0
s_2	0	0	0	0	0	0
s_3	0	0	0	0	0	0
s_4	0	0	0	0	0	0
s_5	0	0	1	1	0	0
6	0	0	0	0	0	0

(i)

#	s_1	s_2	s_3	s_4	s_5	6
s_1	0	0	0	0	0	0
s_2	0	0	0	0	0	0
s_3	0	1	0	1	1	0
s_4	0	0	0	0	0	0
s_5	0	0	0	0	0	0
6	0	0	0	0	0	0

Figure 2.7 visually illustrates retrieving all the paths from node s_3 to sink node $v = 6$. Following the same process, all paths from all nodes are retrieved with the help of Table 2.11. After removing redundancies, *SMPs* enumerated from node s_1 is $\{s_1, s_6\}$, node s_2 is $\{s_2, s_6\}$, node s_3 is $\{s_3, s_4, s_6\}$, $\{s_3, s_2, s_6\}$, node s_4 is $\{s_4, s_6\}$, and node s_5 is $\{s_5, s_4, s_6\}$ and $\{s_5, s_3, s_2, s_6\}$. This explains **Algorithm 1.2** and *Lines 24-28* of **Algorithm 1**.

Algorithm 1.3 and *Lines 29-44* of **Algorithm 1** involves enumeration of the set of *SMPs* from the multisource nodes $T_1 = \{s_1, s_2, s_4\}$, $T_2 = \{s_1, s_2, s_5\}$, $T_3 = \{s_1, s_3, s_4\}$, $T_4 =$

$\{s_1, s_4, s_5\}$ and $T_5 = \{s_2, s_4, s_5\}$ to the sink node. The multi-source sets T_1, T_{3-5} can directly communicate with the sink node. Hence, these sets themselves form the shortest minimal paths and the corresponding *NdState*1, *NdState*3-5 remains unchanged. Thus, we are left with the task of enumerating *SMPs* from $T_2 = \{s_1, s_2, s_5\}$ to the sink node. Following *Line 10* of **Algorithm 1.3** and *OR*ing the shortest paths from all source nodes in the set $T_2 = \{s_1, s_2, s_5\}$ gives the *SMP* as $\{s_1^A, s_2^A, s_5^A, s_4^R, 6\}$. If node s_4 fails, then the next *SMP* available is $\{s_1^A, s_2^A, s_5^A, s_3^R, 6\}$. This explain *Lines 4-11* of Algorithm 1.3. The *SMPs* for the network in Figure 2.5 satisfying $A_{req} = 9$ are $\{s_1^A, s_2^A, s_4^A, 6\}$, $\{s_1^A, s_3^A, s_4^A, 6\}$, $\{s_1^A, s_4^A, s_5^A, 6\}$, $\{s_2^A, s_4^A, s_5^A, 6\}$, $\{s_1^A, s_2^A, s_5^A, s_4^R, 6\}$, and $\{s_1^A, s_2^A, s_5^A, s_3^R, 6\}$ and the corresponding *NdState* = {***3, 3, -1, 3, -1***}, {***3, -1, 3, 3, -1***}, {***3, -1, -1, 3, 3***} and {***-1, 3, -1, 3, 3***}, {***3, 3, -1, 2, 3***}, and {***3, 3, 2, -1, 3***}.

Algorithm 2 involves enumeration of disjoint sets from *NdState*. The disjointing process is explained in Table 2.12. First column shows the *NdState* to be disjointed, second column shows the *NdState* compared with, third column gives the elements to be disjointed and the last column shows the disjointed terms. The complete set of disjoint products obtained are {***3, 3, -1, 3, -1***}, {***3, 1, 3, 3, -1***}, {***3, 2, 3, 3, -1***}, {***3, 1, 1, 3, 3***}, {***3, 1, 2, 3, 3***}, {***3, 2, 1, 3, 3***}, {***3, 2, 2, 3, 3***},{***1, 3, -1, 3, 3***}, {***2, 3, -1, 3, 3***}, {***3, 3, -1, 2, 3***} and {***3, 3, 2, 1, 3***}.

After the disjointing process the symbolic reliability expression is

$$\begin{aligned} WSNRel = \; & p_A(s_1) \times p_A(s_2) \times p_A(s_4) + p_A(s_1) \times p_A(s_2) \times p_A(s_3) \times p_A(s_4) + p_A(s_1) \times p_R(s_2) \\ & \times p_A(s_3) \times p_A(s_4) + p_A(s_1) \times p_F(s_2) \times p_F(s_3) \times p_A(s_4) \times p_A(s_5) + p_A(s_1) \\ & \times p_F(s_2) \times p_R(s_3) \times p_A(s_4) \times p_A(s_5) + p_A(s_1) \times p_R(s_2) \times p_F(s_3) \times p_A(s_4) \\ & \times p_A(s_5) + p_A(s_1) \times p_R(s_2) \times p_R(s_3) \times p_A(s_4) \times p_A(s_5) + p_F(s_1) \times p_A(s_2) \\ & \times p_A(s_4) \times p_A(s_5) + p_R(s_1) \times p_A(s_2) \times p_A(s_4) \times p_A(s_5) + p_A(s_1) \times p_A(s_2) \\ & \times p_R(s_4) \times p_A(s_5) + p_A(s_1) \times p_A(s_2) \times p_R(s_3) \times p_F(s_4) \times p_A(s_5). \end{aligned}$$

Assuming $p_{cu}(s_i)$ = 0.9 and $p_{su}(s_i)$ = 0.8, we have $p_A(s_i)$ = 0.72, $p_R(s_i)$ = 0.18 and $p_F(s_i)$ = 0.1. Supplying these values to the reliability expression, $WSNRel$ for the WSN in Figure 2.5 is evaluated to be 0.6854326272.

Table 2.11 Path tracing and retrieval database table for WSN in Figure 2.5

BranchNo	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
NodNo	6	s_1	s_2	s_4	s_2	s_3	s_4	s_4	s_5	s_5	s_4	s_3	s_5	s_5	s_3	s_1
ParentBr	-	*0*	*0*	*0*	*1*	*4*	*4*	*5*	*5*	*7*	*8*	*6*	*6*	*11*	*12*	*2*
BranchNo	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*
NodNo	s_3	s_4	s_4	s_5	s_5	s_4	s_3	s_5	s_5	s_3	s_2	s_3	s_5	s_1	s_3	s_5
ParentBr	*2*	*2*	*16*	*16*	*18*	*19*	*17*	*17*	*22*	*23*	*3*	*3*	*3*	*26*	*26*	*30*
BranchNo	*32*	*33*	*34*													
NodNo	s_2	s_5	s_1													
ParentBr	*27*	*27*	*32*													

Table 2.12 Disjointing process for the WSN in Figure 2.5

NdState_i	Comparing with	X	Disjointed Terms
			s_1 s_2 s_3 s_4 s_5
{**3, 3, -1, 3, -1**}	–	–	1. {**3, 3, -1, 3, -1**}
{**3, -1, 3, 3, -1**}	*NdState*$_1$	{s_2}	1. {**3, 1, 3, 3, -1**} 2. {**3, 2, 3, 3, -1**}
{**3, -1, -1, 3, 3**}	*NdState*$_1$	{s_2}	1. {**3, 1, -1, 3, 3**} 2. {**3, 2, -1, 3, 3**}
	NdState$_2$	1. {s_3} 2. {s_3}	1. {**3, 1, 1, 3, 3**} 2. {**3, 1, 2, 3, 3**} 3. {**3, 2, 1, 3, 3**} 4. {**3, 2, 2, 3, 3**}
{**-1, 3, -1, 3, 3**}	*NdState*$_1$	{s_1}	1. {**1, 3, -1, 3, 3**} 2. {**2, 3, -1, 3, 3**}
	NdState$_2$	1. disjoint 2. disjoint	1. {**1, 3, -1, 3, 3**} 2. {**2, 3, -1, 3, 3**}
	NdState$_3$	1. disjoint 2. disjoint	1. {**1, 3, -1, 3, 3**} 2. {**2, 3, -1, 3, 3**}
{**3, 3, -1, 2, 3**}	*NdState*$_1$	disjoint	{**3, 3, -1, 2, 3**}
	NdState$_2$	disjoint	{**3, 3, -1, 2, 3**}
	NdState$_3$	disjoint	{**3, 3, -1, 2, 3**}
	NdState$_4$	disjoint	{**3, 3, -1, 2, 3**}
{**3, 3, 2, -1, 3**}	*NdState*$_1$	{s_5}	1. {**3, 3, 2, 1, 3**} 2. {**3, 3, 2, 2, 3**}
	NdState$_2$	1. disjoint 2. disjoint	1. {**3, 3, 2, 1, 3**} 2. {**3, 3, 2, 2, 3**}
	NdState$_3$	1. disjoint 2. disjoint	1. {**3, 3, 2, 1, 3**} 2. {**3, 3, 2, 2, 3**}
	NdState$_4$	1. disjoint 2. disjoint	1. {**3, 3, 2, 1, 3**} 2. {**3, 3, 2, 2, 3**}
	NdState$_5$	1. disjoint 2. drop (superset)	1. {**3, 3, 2, 1, 3**}

2.8 Results, Comparison, and Discussion

A Brute Force method consisting of complete state enumeration is developed to validate the proposed approach. In the Brute Force approach, all $3^{|N|}$ network states are enumerated and checked for its validity by checking two conditions: 1) if the network state satisfies the application-specific flow requirement and 2) whether the flow satisfying network states are connected to the sink node. The network states that satisfy both the conditions become a valid network state and contribute to WSN reliability. Assuming $p_{cu}(s_i)$ = 0.9 and $p_{su}(s_i)$ = 0.8 for the network in Figure 2.2, we have $p_A(s_i)$= 0.72, $p_R(s_i)$ = 0.18 and $p_F(s_i))$ = 0.1. Applying the proposed approach and the Brute Force approach on the network given in Figure 2.2 gives $WSNRel$ = 0.9713559257. This **experimentally validates** the proposed approach. This implies that the proposed approach correctly enumerates 1) all T of the WSN, 2) all *SMP*(*T*) of the WSN, and 3) the disjoint sets, that evaluates $WSNRel$. The proposed approach generates only 152 network states (*T*) and 750 valid-states (*SMP*(*T*)) as compared to the 3^{11} (= 177147) network states and 60424 valid states generated by the Brute Force approach. This shows the computational efficacy of the proposed approach. The proposed approach and the Brute Force approach are applied to different benchmark networks shown in Figure 2.2, Figure 2.8 – Figure 2.20, and the comparative results are shown in Table 2.13. To mention here, the $WSNRel$ values for Figure 2.2, Figure 2.8 – Figure 2.20 enumerated by the proposed approach exactly matches the $WSNRel$ values enumerated by the Brute Force approach. This illustrates the correctness and fidelity of the proposed approach.

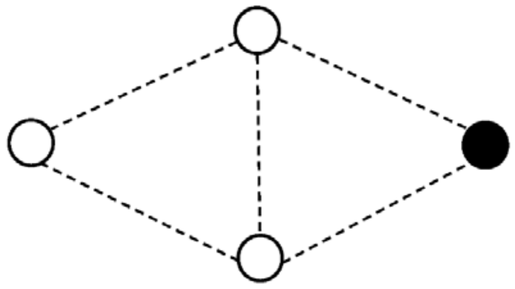
Figure 2.8 A WSN of 4 nodes (Xiao *et al.* 2008)

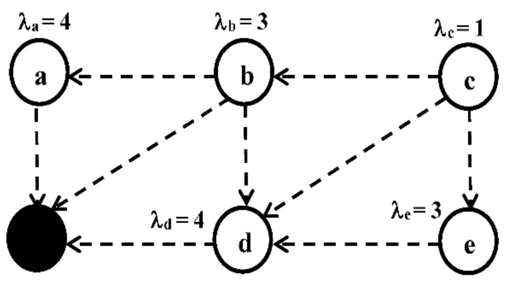

Figure 2.9 A WSN of 6 nodes (Shazly *et al.* 2010)

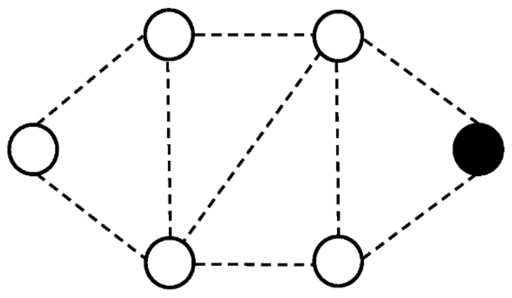
Figure 2.10 A WSN of 6 nodes.

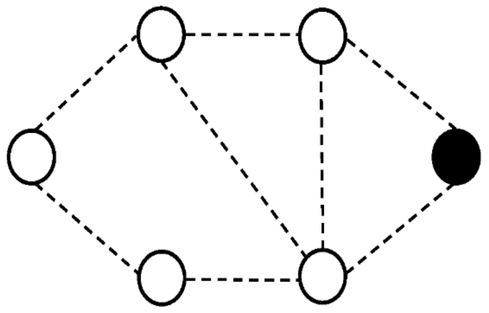
Figure 2.11 A WSN of 6 nodes (Xiao *et al.* 2009)

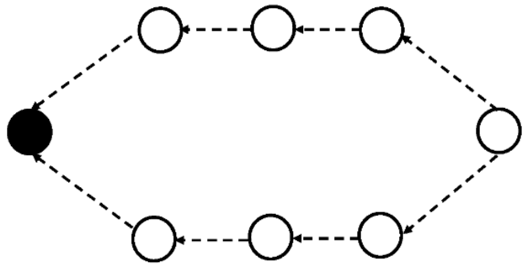
Figure 2.12 A WSN of 8 nodes.

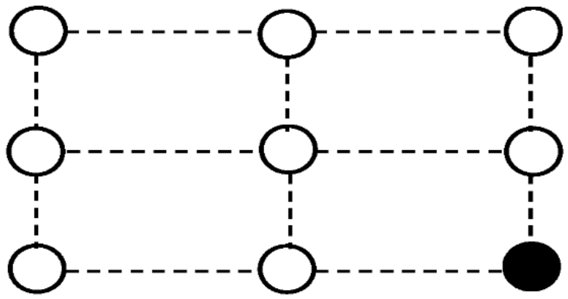
Figure 2.13 A WSN of 9 nodes

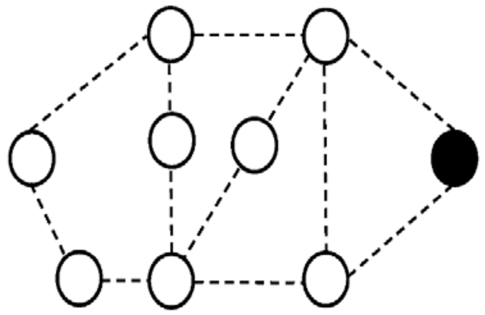
Figure 2.14 A WSN of 9 nodes (Xiao *et al.* 2009)

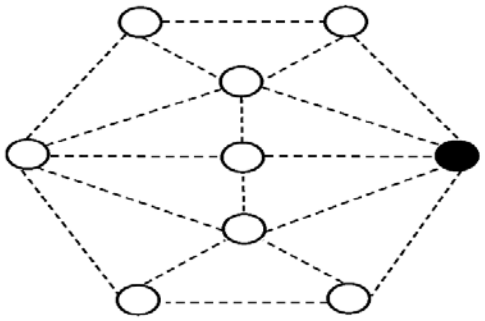
Figure 2.15 A WSN of 9 nodes

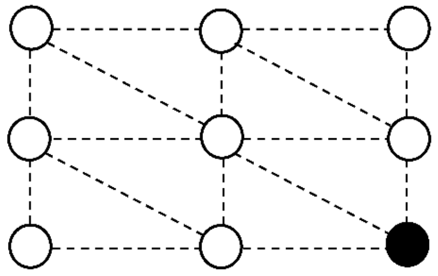
Figure 2.16 A WSN of 9 nodes

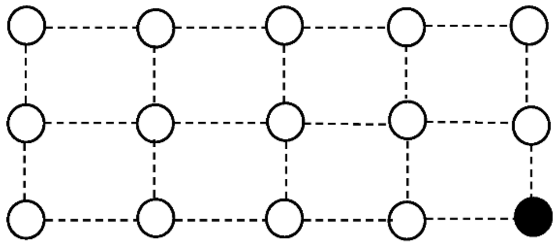
Figure 2.17 A WSN of 15 nodes (Xiao *et al.* 2008)

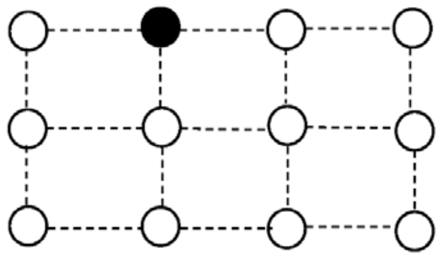
Figure 2.18 A WSN of 16 nodes (Xiao *et al.* 2009)

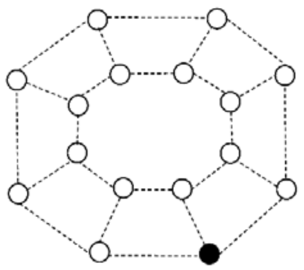
Figure 2.19 A WSN of 16 nodes (Xiao *et al.* 2009)

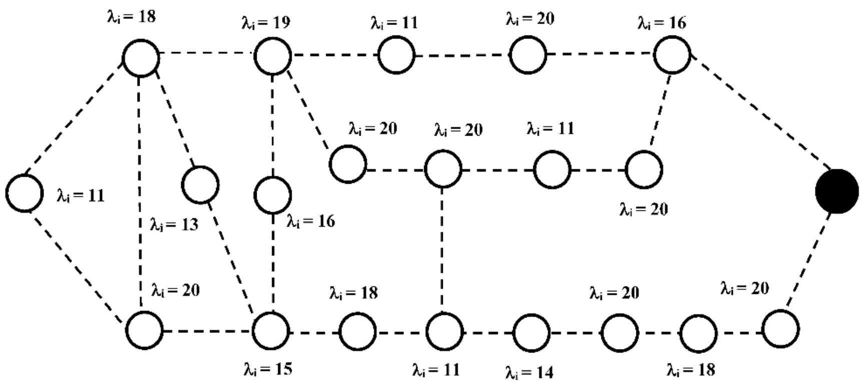

Figure 2.20 A WSN of 21 nodes (Xiao *et al.* 2009)

As evident from Table 2.13, the Brute Force approach fails to provide output within reasonable time for WSN given in Figure 2.17 – Figure 2.19, whereas the results enumerated by applying the proposed approach on WSN shown in Figure 2.17 – Figure 2.19 is given in sixth and eight columns of Table 2.13. To account for a moderately large-sized network, a WSN of 21 nodes (Figure 2.20) is considered, and the proposed approach, as well as the Brute Force approach, is applied to the network. The proposed approach enumerates only 227 network states and 2981 valid states, whereas it went out of computer memory for the Brute Force approach. The results thus obtained in Table 2.13 highlight the efficacy of the proposed approach.

Table 2.14 quantitatively analyzes the effect of the number of nodes and number of links on the actual memory and computational time for the proposed approach. Four D-grid networks with $|N| + 1$ = 6 to 15 ($|L|$ = 9 to 30) were considered for the analysis. The A_{req}, and λ_i are assumed to be 24 and 7 units, respectively. The first and second columns of Table 2.14 show the number of nodes and the number of links of the four networks, respectively, while the third and last columns give the computational time and memory required by the proposed approach to evaluate $WSNRel$. As expected, due to the NP-*hardness* of the problem to compute $WSNRel$, the computational time in column three of Table 2.14 increases in exponential order when the number nodes (links) increases from 6 (9) to 15 (30). However, the memory requirement is only slightly affected when the network sizes increase.

Table 2.13 Comparative Results for Figure 2.2, Figure 2.8 – Figure 2.20

Figure #	$\|N\|+1$	$\|L\|$	λ_i	A_{req}	$p_{cu}(s_i)$, $p_{su}(s_i)$	Network States Examined		Valid States Enumerated		$WSNRel$
						Proposed Approach	Brute Force Approach ($3^{\|N\|}$)	Proposed Approach	Brute Force Approach	
2.2	12	15	6	20	0.9, 0.8	152	177147	**750**	**60424**	0.971356
2.8	4	5	16	15	0.9, 0.8	3	27	**3**	**7**	0.970848
2.9	6	9	*****	9	0.9, 0.8	5	243	**5**	**28**	0.678714
2.10	6	9	7	20	0.99, 0.8	10	243	**13**	**48**	0.935602
2.11	6	8	6	20	0.99, 0.8	5	243	**5**	**11**	0.720818
2.12	8	8	6	20	0.99, 0.8	35	2187	**39**	**205**	0.948588
2.13	9	10	7	24	0.99, 0.8	70	6561	**162**	**1450**	0.987489
2.14	9	12	11	30	0.95, 0.8	56	6561	**149**	**2761**	0.904256
2.15	9	18	12	45	0.95, 0.8	70	6561	**86**	**1681**	0.976965
2.16	9	16	8	25	0.95, 0.8	70	6561	**136**	**1569**	0.975669
2.17	15	22	11	40	0.95, 0.8	1001	♦	**10400**	♦	0.996304
2.18	16	24	14	40	0.9, 0.8	455	♦	**4821**	♦	0.999245
2.19	16	24	8	15	0.9, 0.8	105	♦	**1688**	♦	0.986892
2.20	21	26	*****	40	0.95, 0.8	227	**!**	**2981**	**!**	0.987816

♦ Failed to provide results within a reasonable time
! Out of memory
***** depicted in figure

Table 2.14 Effect of $|N|+1$ and $|L|$ on computer memory and time

$\|N\|+1$	$\|L\|$	CPU Time (seconds)	Computer Memory (Bytes)
6	9	0.08872	1.59E+09
9	16	0.89828	1.78E+09
12	23	264.55209	1.79E+09
15	30	12.966**	1.89E+09

** in hours

2.9 Chapter Summary

In this chapter, we investigate the problem of evaluating the reliability of multi-source single-sink WSNs with multistate nodes. We have presented exact algorithms that work on arbitrary networks. Our algorithm is computationally less intensive as fewer network states are generated as compared to the Brute Force approach that requires generating all $3^{|N|}$ states. The proposed approach is advantageous in the sense that it paves the way towards a more realistic aspect of wireless sensor network reliability evaluation by considering the RELAY state of each sensor node (except sink). Analyzing $WSNRel$ with this method will allow a network designer to consider this critical metric as a part of sensor network design and determine the reliability quickly. The versatility of the proposed approach is that it can be applied to WSNs with flat, mesh, or grid topology, and planned as well as randomly deployed WSN.

Chapter 3 Area Coverage Reliability Evaluation of Mobile Wireless Sensor Networks with Multistate Nodes under Link Reliability Constraints

Keywords
Area-coverage,
Monte Carlo simulation,
multistate nodes,
network reliability,
wireless sensor
networks.

*This chapter proposes a quantitative measure Area Coverage Reliability (ACR) for WSN*s. *ACR brings together WSN reliability, area coverage, energy efficiency, mobility of sink, random duty cycle of nodes, and multistate nature of sensor nodes under a common umbrella. This chapter proposes a Monte Carlo simulation approach that utilizes an energy matrix to evaluate the effect of energy-depleted nodes and energy-oriented data transfer capability on ACR. The energy matrix reflects the residual energy of sensors, the energy required to transmit data to the neighboring nodes, connectivity, link reliability, and the multistate nature of the sensors. The proposed approach is illustrated through a series of random examples.*

3.1 Introduction

Recent development in sensor devices has tremendously increased human dependency on WSNs to achieve their targets of tracking things, monitor the environment, perform industrial and military operations, and so on. To achieve that, a large number of inexpensive sensor nodes, each capable of sensing, processing, and transmitting environmental information, are deployed in the region of interest. Deployment of sensor nodes can be preplanned (Tolle *et al.* 2005) or random (Zou and Chakrabarty 2004; Li *et al.* 2010), depending on the types of application environment. In remote monitoring applications, WSNs are randomly distributed, i.e., geographically deployed in a random manner, in the field of interest and remain unattended after deployment. These sensor nodes are vulnerable to different kinds of failure due to external causes such as variability in environmental conditions, including rainfall, humidity, foliage, and internal reasons like noise, lack of battery power, hardware failure, etc. As such, due to the hardware components working or failure states, and due to sensor nodes' residual energy, a sensor can exhibit multistate operational nature during its operational period. Under such a scenario, it becomes extremely essential to assess how well a WSN can monitor the region of interest. This assessment depends on quantifying the performance measures: coverage and reliability under one frame. Specifically, for successful operation, a WSN is required to provide sensing coverage of the monitored area that satisfies a given application-specific coverage-area requirement (Shazly *et al.* 2011), and successful transfer of collective sensor data to the sink.

To conserve the limited resources of the battery-powered sensor nodes, mobile sink nodes that could travel in and around the monitoring region were implemented (see Section 1.4). From Section 1.3, it can be concluded that though the literature studies coverage and reliability of mWSNs individually, the combinatorial analysis of (i) multistate nature of nodes, (ii) coverage-area requirement, (iii) Euclidean distance between two sensors, (iv) node's transmission range, (v) energy required to transmit data, (vi) node's residual energy, and (vii) node and link reliability on WSN reliability still needs to be studied and quantified. Aiming to quantify WSN reliability considering aspects (i)-(vii), this chapter proposes an approach that quantifies the application-specific coverage-oriented multi-source single-sink mWSN reliability with multistate nodes and a mobile sink. The quantifying measure, coined as Area Coverage Reliability (ACR), unifies the aspects (i)-(vii) stated above.

The proposed Monte Carlo simulation approach to evaluate ACR randomly generates states for all nodes considering random duty-cycle, hardware components' failure, and node energy states while the sink moves along the periphery of the monitoring region. These generated node-states are random and independent of each other. As sensors are changing states and not always operating; the area-covered by the mWSN varies with time depending on node-states. The non-overlapping area sensed by sensors is evaluated and aggregated to calculate the total area sensed by the mWSN at each instance of data collection. Finally, the capability of the mWSN to transfer the application-specific coverage-oriented data to the mobile sink is checked through a connectivity matrix that accounts for both residual energies of all nodes and link reliability. This connectivity matrix evaluates the availability of link $\mathbb{L}_{i,j}$ between sensors s_i and s_j, including not only the communication range but also the energy availability of each node to transmit data to its neighbor as well as node and link reliability. The effectiveness of our proposed approach is shown through several performance comparisons on mWSNs with various sizes.

The layout of the chapter is as follows. Section 3.2 describes the basic building blocks for the proposed approach. Section 3.3 enlightens the proposed metric ACR. Section 3.4 discusses the proposed approach and methodology. Section 3.5 provides simulation results, and finally, Section 3.6 summarizes the chapter.

3.2 Basic Building Blocks

This section explains the modeling process of a typical WSN system. It first explains the characteristics of the mWSNs with modeling assumptions. Then, it discusses the modeling of a network followed by the definition and evaluation of area-coverage.

3.2.1 Assumptions

Most WSN applications use battery-powered sensor nodes. These batteries can be chargeable or non-rechargeable. In this study, we have considered only non-rechargeable sensor nodes. Apart from this, the assumptions considered for this study are:

- All states of a sensor node are statistically independent of each other.
- Once a sensor node fails, it stays failed for the remaining period of the mission time.
- All links are bidirectional without any constraint on their load-carrying capacity.
- Sensor nodes are resource-constrained, and the sink node has sufficient resources.

3.2.2 Mobile Wireless Sensor Networks (mWSNs) Model

An mWSN is composed of a set of N randomly deployed sensor nodes and a mobile sink v that moves along the periphery of a simulation region with a speed of u m/s. The mWSN is modeled as an undirected graph $G = ((v) \cup N,\ L)$ where L is the set of communication links. After deployment, the sensors remain geographically static throughout their lifetime. Figure 3.1 shows a generic mWSN with 21 sensors deployed randomly in the monitoring field, and a mobile sink that travels along the periphery of the simulation region.

As depicted in Figure 2.1, each sensor node is comprised of four major components: a sensing unit, a power device, a processing unit, and a transceiver unit. Owing to a hardware component's failure, energy shortage and/or random duty-cycle, at any point of time, a sensor node may be in one of the following four states: ACTIVE, RELAY, SLEEP, or FAIL. An ACTIVE state is a state in which a node can sense, transmit, and receive data as its sensing, powering, and transceiver units are operating reliably. Failure of the processing unit leads to complete node failure. Therefore, in our study, the processing unit is assumed to be perfectly reliable. A sensor node in a RELAY state can transmit and receive data; however, it cannot sense due to the failure of its sensing unit. In SLEEP state, a node can neither sense nor transmit or receive data (*temporarily*) as all its circuitry is temporarily *turned off*, depicting the effect of random duty-cycle. In contrast, a sensor in the FAIL state can neither sense nor transmit or receive data (*permanently*) due to the failure of either the transceiver unit or battery depletion or both. Evaluation of probabilities of a node being in any of the above states is discussed in Section 3.4.1.

At any point of time, the network's connectivity is determined by the graph $G = \{\{v\} \cup N,\ L\}$. A communication link exists between any two sensor nodes s_i and s_j with reliability γ_{ij}, for $0 \leq \gamma_{ij} \leq 1$, if and only if, i) nodes s_i and s_j are in either ACTIVE or RELAY state, ii) nodes s_i and s_j are within communication range (r_{s_i}) of each other, and iii) node s_i has sufficient amount of energy required to transmit its collected data to node s_j. A link that complies with these three requirements is called an *available* link. Such link formation model is proposed owing to the fact that even if the Euclidian distance $d(i,j)$ between sensors s_i and s_j, is less than the communication range, i.e., $d(i,j) \leq r_{s_i}$, a link between nodes s_i and s_j will not be formed if sensor node s_i has insufficient amount of energy to establish a link and send data to sensor node s_j. Moreover, in a wireless scenario, weather, hills, high-rise buildings, noise, interference, etc. also affect a link's quality, which in turn affects a link's reliability. Thus, while determining a link's existence, it does not suffice only to consider node-energy.

Formally, a link between sensors s_i and s_j is *available*, if and only if, the following conditions are satisfied:

Condition (i): nodes s_i and s_j are in either ACTIVE or RELAY state, *Condition* (*ii*): $d(i,j) \leq \min(d_{En,s_i}, r_{s_i})$, and the link between two nodes s_i and s_j is operating with a reliability of γ_{ij},

where d_{En,s_i} denotes the maximum distance afforded by the battery-power of sensor s_i, up to which it can transmit data. This distance, i.e., d_{En,s_i}, is termed as the *energy-oriented transmission range* of sensor s_i. The evaluation of energy-oriented transmission range will be discussed later in this section. The incorporation of link's reliability while determining its availability is discussed in Section 3.4.3, while the node-state determination for *condition* (*i*) is discussed in Section 3.4.1.

We follow the energy model as in (Heinzelman *et al.* 2002; Younis and Fahmy 2004) to evaluate the energy consumed by a sensor node s_i in sensing (E_{sense,s_i}), transmitting (E_{Tx,s_i}), and receiving (E_{Rx,s_i}) a data bit. Equations (3.1) and (3.2) respectively give expressions for calculating the energy required to transmit and receive data of size λ bits to a distance d:

$$E_{Tx,s_i}(\lambda, d) = \begin{cases} \lambda E_{elec} + \lambda \varepsilon_{fs} d^2, & d < d_0 \\ \lambda E_{elec} + \lambda \varepsilon_{mp} d^4, & d \geq d_0 \end{cases} \tag{3.1}$$

$$E_{Rx,s_i}(\lambda) = \lambda E_{elec} \tag{3.2}$$

where E_{elec} is the energy consumed by the transmitter circuitry per data bit, ε_{fs} and ε_{mp} are respectively the energy consumed by the power amplifier per data bit for the free-space (*fs*) and multi-path (*mp*) fading channel model, and $d_0 = \sqrt{\varepsilon_{fs}/\varepsilon_{mp}}$ is the threshold distance that determines the channel model.

At any point of time, residual energy E_{Res,s_i} of a sensor node s_i determines its communication capability, i.e., the maximum distance up to which a node can communicate. Therefore, assuming E_{Res,s_i} to be the currently available energy that the sensor node s_i can spend on transmitting data, from (3.1), we have,

$$E_{Res,s_i} = \lambda E_{elec} + \lambda \varepsilon_{fs} d^2_{En,s_i}. \tag{3.3}$$

Solving (3.3) for d_{En,s_i}, we get the energy-oriented transmission range of sensor s_i as

$$d_{En,s_i} = \sqrt{\frac{(\frac{E_{Res}}{\lambda} - E_{elec})}{\varepsilon_{fs}}}. \tag{3.4}$$

3.2.3 Area-Coverage

Area-Coverage refers to the region of interest under surveillance by a sensor network. In most applications, the network's overall capability to monitor the region of interest determines its performance. Given a region to be monitored, a network's area-coverage is the total area sensed by the network through its multistate sensor nodes. The total area covered by an mWSN, $A(N_A)$, is given as

$$A(N_A) = \bigcup_{i=1}^{|N_A|} A(s_i) \tag{3.5}$$

where N_A denotes the set of nodes in the ACTIVE state, $|*|$ represents the number of elements in "$*$", and $A(s_i)$ is the area sensed by sensor s_i. Note that $A(s_i)$ refers to the disjoint-area sensed by sensor s_i; Section 3.4.2 describes the evaluation of $A(s_i)$.

3.3 Area-Coverage Reliability (ACR) of an mWSN

To model an mWSN's ability to satisfy the application-specific area-coverage requirement in the presence of multistate sensor nodes, a new performance index, Area-Coverage Reliability (ACR) is introduced in this chapter. ACR quantifies mWSN's (with multistate nodes) capability of:

(i) satisfying the application-specific area-coverage requirement A_{req}, i.e., $A(N_A) \geq A_{req}$ and,

(ii) successfully transmitting the sensed coverage-oriented data to the mobile sink node.

Note that any mWSN is considered to be working reliably iff it satisfies both the conditions. Therefore, based on the above discussion, the ACR of mWSN is defined as follows.

Definition 1: (ACR): Given an application-specific coverage-area monitoring requirement A_{req} of an mWSN, ACR is defined as the probability that the mWSN can transmit at least A_{req} to the mobile sink node for a given period of time under given environmental conditions.

Definition 2: (ACR Problem): Given a graph $G = \{\{v\} \cup N, L\}$ of an mWSN with mobile sink node v, a set of ACTIVE and RELAY nodes in N, and a set of communication links L, compute the probability that there exists an operational path from all source (ACTIVE) nodes to the mobile sink node, iff the total non-overlapping area sensed by the ACTIVE nodes, is at least A_{req}.

Figure 3.1 gives a typical example of the ACR *Problem*. It shows an mWSN of 21 sensor nodes randomly deployed in the region of interest with an A_{req} = 50% of the total monitoring region. As depicted in Figure 3.1, the sensor nodes can be in ACTIVE, SLEEP, RELAY, or FAIL states depending on hardware failure or working states, random duty-cycle, and energy states of the communicating node. Figure 3.1a shows an mWSN that satisfies its area-coverage requirement of 50% of the total monitoring region, i.e., $A(N_A) > A_{req}$. However, the mWSN in Figure 3.1a is unreliable as it is not connected to the mobile sink, and thus fails to transfer the collected data to the sink. Figure 3.1b shows another instance of the mWSN, where the network is unreliable because it fails to satisfy A_{req}, i.e. ($A(N_A)$ = 45% of the total monitoring region) < (A_{req} = 50% of the total monitoring region). Thus, merely satisfying A_{req} or being connected to the mobile sink does not make any mWSN a reliable network. In other words, an mWSN is reliable iff it satisfies A_{req} and is capable of transferring $A(N_A) \geq A_{req}$ to the mobile sink. Figure 3.1c accounts for a reliable network as (i) $A(N_A) > A_{req}$, and (ii) the network is connected to the mobile sink. The mWSN in Figure 3.1c, thus satisfies the application-specific coverage-area requirement and can successfully transfer the required amount of sensor data to the mobile sink.

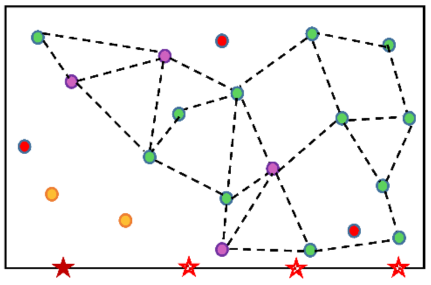

Figure 3.1a Unreliable Network (A_{req}= 50% of total monitoring region, $A(N_A)$ = 60% of total monitoring region)

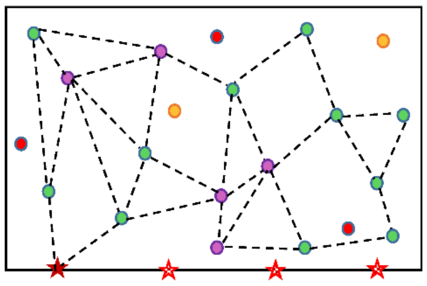

Figure 3.1b. Unreliable Network (A_{req} = 50% of total monitoring region, $A(N_A)$ = 45% of total monitoring region)

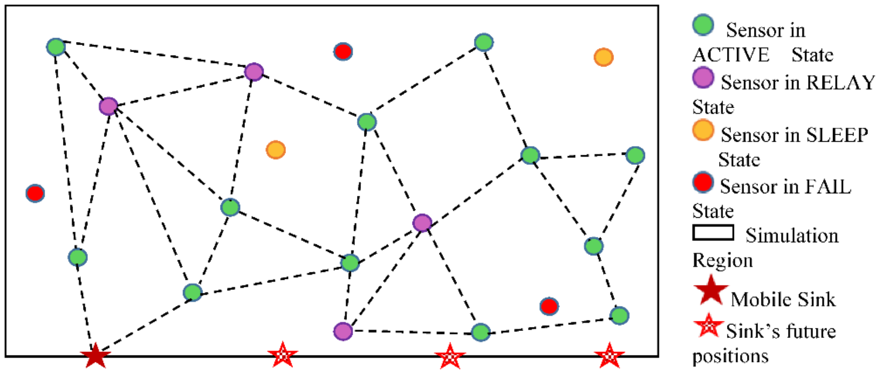

Figure 3.1c Reliable Network (A_{req}= 50% of total monitoring region, $A(N_A)$ = 60% of total monitoring region)

Figure 3.1 A generic example of mWSN for the area-coverage problem.

3.4 Proposed Methodology and Approach

This section discusses the proposed approach, shown in **Algorithm 1**, for ACR evaluation of an mWSN. The proposed approach addresses quantitative evaluation of the area-coverage oriented reliability, ACR, of mWSNs. **Algorithm 1** starts by deploying the sensor nodes randomly in the field to be monitored (*Line 1*) with area $\mathcal{A}$ m^2. Once deployed, the sensors, except the sink, remain geographically static and free from human intervention throughout their lifetime. Thus, randomness in the *network configuration* of an mWSN is an effect of the random node-states, energy-availability of nodes, and link reliability. The mobile sink moves along the simulation boundary and collects data after every *t* seconds each time from a newly generated *network configuration*. For each iteration $q = 1, 2, 3, \ldots$, Q, *Line* 3 generates $M = \sqrt{\mathcal{A}}/_{t}$ number of sink positions. For each sink position, *Lines 4-*

20 simulate one network configuration. Thus, there are $Q \times M$ network configurations, and **Algorithm 1** in total performs $Q \times M$ number of simulation runs. To be more specific, *Lines 5-6* enumerate the *current state* of each node. The *current state* of each node is affected by its random duty-cycle, hardware components' failure or working states, and battery energy state of each sensor node. Recall that at any instant of time, a sensor node can be in ACTIVE, RELAY, SLEEP, or FAIL state. Section 3.4.1 elaborately discusses the random node-states generation of a sensor node and the *current state* determination of any sensor. *Line 7* stores the sensors in the ACTIVE state. In *Line 9* the proposed approach calls **Algorithm 1.1** to evaluate the disjoint-area sensed by each node in the ACTIVE state, and in *Line 10*, the total area sensed by the random network configuration is evaluated (see Section 3.4.2). Each random network configuration is checked for being a successful network state. A network configuration is in a successful state iff it satisfies A_{req} and successfully transmits the sensed data to the mobile sink node. If any network configuration satisfies A_{req}, *Line 12* enumerates the energy-available link matrix $\mathbb{L}$ (see Section 3.4.3). This matrix is then used to check the connectivity, i.e., the existence of a path from each of the ACTIVE nodes to the sink for the current network configuration in *Line 13*. To mention here, *Line 13* uses the *node-fusion* method (Deo 1974) to check the existence of a path between each ACTIVE node and the mobile sink. As discussed in Section 3.4.4, a Boolean variable $\chi_{q,m}$, whose value is determined in *Lines 14-16*, keeps track of the reliability/unreliability of the m^{th} $(m \in M)$ network configuration in the q^{th} iteration. After every simulation run, the residual energy of each node is updated in *Line 17*. The residual energy is updated by subtracting the energy spent in the current simulation run from the energy the node had before starting the current simulation run. Finally, in *Line 21*, after $Q \times M$ number of simulation runs, ACR of the mWSN is evaluated (see Section 3.4.4).

The major steps of **Algorithm 1** are discussed in the following sub-sections.

3.4.1 Random Node States Generation

This section discusses the process of random node-states generation at each sink position, i.e., after every *t* seconds. The sensor nodes are random in nature and exhibit multiple states due to the random duty-cycling approach followed by the sensors, random failure of the sensor node's hardware components, and limited battery life of the sensors. It is assumed that the sensor nodes follow a random duty-cycling approach to conserve energy where sensors *turn on* and *turn off* in a random fashion independent of each other.

Algorithm 1: *ACR* Evaluation

Inputs: N, E_{in}, A_{req}, $\mathcal{L}$, K, Q, t, u, α_{s_i}, r_{s_i}
N: A set of nodes.
E_{in}: Initial energy of each node.
A_{req}: Application-specific coverage area requirement.
$\mathcal{L}$: Co-ordinates of the simulation region
K: Co-ordinates of N_A nodes
Q: number of iterations.
t: data collection time interval.
u: speed of the mobile sink.
α_{s_i}: duty-cycle of each sensor node.
r_{s_i}: transmission range of a sensor s_i
Output: *ACR*
ACR: Area-Coverage Reliability.

```
1.  Generate random node positions within the simulation region
    // generate $Q \times M$ network configurations
2.  For each simulation $q = 1, 2, 3, \dots, Q$
3.      Find sink position $m = 1, 2, \dots, M$ // generated after every $t$ seconds
4.      For each sink position $m$
    // node-state generation
5.          For each node $s_i$ of $N$
6.              Enumerate the current state of the sensor node $s_i$ // refer Section 3.4.1
7.              $N_A \leftarrow$ sensors in ACTIVE state
8.          End For
    // disjoint-area evaluation
9.          $A(N_A) \leftarrow \mathbf{EvalDisjArea}(r_{s_i}, N_A, K, \mathcal{L})$ // refer Section 3.4.2
10.         Evaluate the total area sensed $A(N_A)$ by the network // by following (3.5)
11.         If $A(N_A) \geq A_{req}$
    // Link-State matrix enumeration
12.             Find the link-state matrix ($\mathbb{L}$) of the network // refer Section 3.4.3
13.             Check if the network is connected to the mobile sink
14.             If connected
15.                 $\chi_{q,m} = 1$ // Network configuration $(q,m)$ is reliable
16.             End If
17.             Update residual energy of each node and go to #Line 3.
18.         End If
19.     End For
20. End For
    // ACR evaluation from $Q \times M$ network configurations
21. Evaluate ACR using (3.12) // refer Section 3.4.4.
```

Algorithm 1.1: ***EvalDisjArea***(r_{s_i}, N_A, $\mathcal{L}$, K): Evaluate the disjoint sensing area of a sensor s_i

Input:

r_{s_i}: Transmission range

N_A: a set of sensors in ACTIVE state

$\mathcal{L}$*:* Co-ordinates of the simulation region

K*:* Co-ordinates of N_A nodes

Output*:*

$\boldsymbol{A}[\boldsymbol{S}]$*:* An array containing the sensing area of each node $s_i \in N_A$

```
1.  $A[\boldsymbol{S}] \leftarrow \Phi, V'(s_i) \leftarrow \Phi$
2.  Construct $Voronoi$ diagram for $|N_A|$ sensor nodes
3.  **For** each $s_i \in N_A$
4.      $V(s_i) \leftarrow$ corner points of the $Voronoi$ polygon of sensor $s_i$
5.      $D(s_i) \leftarrow$ co-ordinates of the points in $V(s_i)$
6.  **End For**
7.  **For** each sensor $s_i \in N_A$
8.  $C(s_i) \leftarrow \boldsymbol{EnumCP}(r_{s_i}, K(s_i))$
9.     **If** $A_{sense}(s_i) \subset V(s_i)$                  // Case 1
10.        $A(s_i) = \Delta A_{sense}(s_i)$
11.    **ElseIf** $A_{sense}(s_i) \supset V(s_i)$              // Case 2
12.        $A(s_i) = \Delta V(s_i)$
13.    **ElseIf** $A_{sense}(s_i)$ *intersects* $V(s_i)$     // Case 3
14.        **For** $k$ = 1, …, $|D(s_i)|$
15.            **For** $m$ = 1, …, $|C(s_i)|$
16.                **If** $D_k(s_i)$ matches $C_m(s_i)$
17.                    Append $D_k(s_i)$ to $V'(s_i)$
18.                **Else**
19.                    $[P_x, P_y]$ = co-ordinates of points closest to $K(s_i)$ between $D_k(s_i)$ and $C_m(s_i)$
20.                    Append $[P_x, P_y]$ to $V'(s_i)$
21.                **End If**
22.            **End For**
23.        **End For**
24.        $A(s_i) = \Delta V'(s_i)$
25.    **Else**                                            // Case 4
26.        $[P_x, P_y]$ = co-ordinates of points closest to $K(s_i)$ between $D(s_i)$ and $C(s_i)$ and $\mathcal{L}$
27.        $V''(s_i) \leftarrow [P_x, P_y]$
28.        $A(s_i) = \Delta V('' s_i)$
29.    **End If**
30. $A[N_A] \leftarrow$ Append $A(s_i)$
31. **End For**
```

The duty-cycle (α_{s_i}) of each sensor node s_i is the ratio of the time period for which the radio of the sensor is turned *on* to the total time for which the radio is *on* and *off* (Hsin and Liu 2006). Thus, the probability that any sensor node remains in *SLEEP* state is

$$p_S(s_i) = \left(1 - \alpha_{s_i}\right). \tag{3.6}$$

Any node in working state can be in ACTIVE or RELAY state depending on the success or failure states of the sensing and transceiver unit. Equation (3.7) and Equation (3.8), as shown below, respectively give the reliability of a node s_i in ACTIVE and RELAY state:

$$p_A(s_i) = p_{cu}(s_i) \times p_{su}(s_i) \times \alpha_{s_i} \tag{3.7}$$

$$p_R(s_i) = p_{cu}(s_i) \times (1 - p_{su}(s_i)) \times \alpha_{s_i}. \tag{3.8}$$

A node can be in the FAIL state due to hardware failure and/or energy depletion. Each node starts operating with an initial energy E_{in} and gradually depletes its battery due to sensing and/or transmission of data. Thus, at the onset of operation, the probability that any sensor s_i is in FAIL state is

$$p_F(s_i) = \left(1 - p_{cu}(s_i)\right) \times \alpha_{s_i}. \tag{3.9}$$

In due course, if any node runs out of energy, it remains in the FAIL state for the entire network lifetime.

To determine the *current* state of any sensor node s_i, a random value, $test_{s_i}$, following a uniform distribution ($0 < test_{s_i} < 1$) is generated. This randomly generated $test_{s_i}$ value is then compared against the following conditions to determine the current state of any sensor node s_i:

***Condition* 1**: $0 < test_{s_i} \leq p_A(s_i)$

If the generated $test_{s_i}$value falls in this category, then a sensor node s_i is in ACTIVE state.

Condition 2: $p_A(s_i) < test_{s_i} \leq p_A(s_i) + p_R(s_i)$

If the generated $test_{s_i}$value falls in this category, then a sensor node s_i is in RELAY state.

Condition 3: $p_A(s_i) + p_R(s_i) < test_{s_i} \leq p_A(s_i) + p_R(s_i) + p_F(s_i)$

If the generated $test_{s_i}$value falls in this category, then a sensor node s_i is in FAIL state.

Condition 4: $test_{s_i} > p_A(s_i) + p_R(s_i) + p_F(s_i)$

If the generated $test_{s_i}$value falls in this category, then a sensor node s_i is in SLEEP state.

3.4.2 Disjoint Area Sensing Scheme (*DASS*)

The Disjoint Area Sensing Scheme (*DASS*) proposes an approach to evaluate the area covered by a sensor node under the network configuration discussed in Section 3.4.1. As given in (3.5), the total area covered by an mWSN is evaluated by summing (union of) the areas sensed by each of the energy-sufficient ACTIVE nodes present within the monitoring region.

A very simple and facile approach to calculate the area sensed by a sensor is to assume a unit disk sensing model and calculate the area sensed, $A(s_i)$, by each sensor s_i, as $A(s_i) = \pi r_{s_i}^2$ where r_{s_i} is the sensing range of a sensor. However, it would lead to erroneous estimation of the area sensed as two or more sensors, as shown in Figure 3.2, might sense parts of the same circular area. Hence, to avoid sensing of such overlapped areas by two or more sensors, it is important to evaluate the disjoint-area sensed by each sensor. In the proposed *DASS* approach, by following **Algorithm 1.1**, each sensor is allocated a disjoint-area to be sensed by examining a suitable relationship between the circle $A_{sense}(s_i)$ and the *Voronoi* polygon $V(s_i)$, where $A_{sense}(s_i)$ denotes the circular region sensed by the sensor s_i and $V(s_i)$ denotes the *Voronoi* polygon containing sensor s_i. Note that $A(N_A)$ can be evaluated by the union of the areas covered by $A_{sense}(s_i)$ for all $s_i \in N_A$ and then

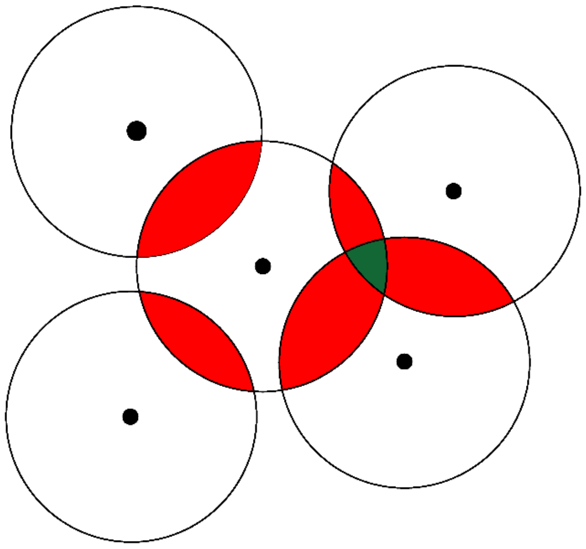
Figure 3.2 Overlapping area sensed by sensors.

exclude the overlapping areas. However, this inclusion-exclusion process becomes tedious and cumbersome to evaluate $A(N_A)$ with increasing number of nodes. Therefore, the *DASS* is adopted to evaluate (3.5).

The *DASS* starts by dividing the monitored region into *Voronoi* cells, wherein each cell contains an ACTIVE sensor node $s_i \in N_A$ (see *Line 2* of **Algorithm 1.1**). For each sensor $s_i \in N_A$, *Line 4* obtains the corner points of the *Voronoi* polygon of s_i, and *Line 5* uses variable $D(s_i)$ to store the co-ordinates of each corner point in $V(s_i)$. Let $K(s_i)$ denote the spatial information of each sensor node s_i in the simulation region. Thus, $A_{sense}(s_i)$ represents the circular region with $K(s_i)$ as the center. *Lines 7-31* evaluate the disjoint area sensed by each sensor $s_i \in N_A$. To mention here, the circumference of each circular region with $K(s_i)$ as center is divided into z points such that the distance between each pair of consecutive points is same. *Line 8* obtains the Cartesian co-ordinates of each point and stores the points in $C(s_i)$. Without loss of generality, we set z = 101 points, and thus $|C(s_i)|$ = 101. While determining the disjoint areas, *DASS* considers four *cases*, as discussed below. Note that *Case 1* and *Case 2* occur when $A_{sense}(s_i)$ and $V(s_i)$ do not intersect each other, while *Case 3* and *Case 4* occur when $A_{sense}(s_i)$ and $V(s_i)$ intersect each other at various points.

- <u>*Case 1*</u>: $A_{sense}(s_i) \subset V(s_i)$

This case arises when the *Voronoi* polygon encircles the circular area sensed by a sensor node. In such a case, the disjoint area assigned to the sensor node is $A(s_i) = \Delta A_{sense}(s_i)$, as given in *Lines 9-10* of **Algorithm 1.1**, where $\Delta A_{sense}(s_i)$ denotes the area of the circular region $A_{sense}(s_i)$.

- <u>*Case 2*: </u>$A_{sense}(s_i) \supset V(s_i)$

This case arises when the circular area sensed by a sensor node encircles the *Voronoi* polygon. In this case, the disjoint area assigned to the sensor node is $A(s_i) = \Delta V(s_i)$, where $\Delta V(s_i)$ denotes the area of the polygon $V(s_i)$. This is shown in *Lines 11-12* of **Algorithm 2**.

- <u>*Case 3:*</u> $A_{sense}(s_i)\ intersects\ V(s_i)$

This case arises when the circular area and polygon intersect at various points. In such a case, to avoid sensing of overlapping area and evaluate the disjoint area sensed by the sensor s_i, it becomes essential to enumerate those intersecting points between $V(s_i)$ and $A_{sense}(s_i)$. This leads to the formation of an irregular region $V'(\ s_i)$. $V'(\ s_i)$ consists of i) the common points between $D(s_i)$ and $C(s_i)$ (see *Lines 16-17*), and ii) the points nearer to $K(s_i)$ between $D(s_i)$ and $C(s_i)$ (as shown in *Lines 18-19*). Then, the disjoint area assigned to the sensor node is $A(s_i) = \Delta V'(\ s_i)$, where $\Delta V'(\ s_i)$ denotes the area of the polygon $V'(\ s_i)$; see *Line 24*. Note that one can use the in-built MATLAB function *polyarea*() that requires the Cartesian co-ordinates of $V'(\ s_i)$ to compute $\Delta V'(\ s_i)$.

Figure 3.3 shows a randomly deployed WSN with 40 sensors in the ACTIVE state in a 100*100 m^2 simulation region with r_{s_i} = 10 *m.* As depicted in Figure 3.3, for sensor node s_3, if the circular area centered at node s_3 is assigned to be its sensed coverage area, then it partially covers the areas sensed by s_7, s_{11}, s_{25}, and s_{33}. Again, following the same process (as followed for s_3) for s_7, s_{11}, s_{25}, and s_{33}, leads to over-sensing of the same areas by different sensors. Hence, to avoid such type of sensing, *DASS* advises assigning each node a disjoint area. As an example, in this case, the disjoint area assigned to s_3 is evaluated by calculating the area of the polygon $V'(\ s_3)$ (shown by dotted points on the polygon $V(s_3)$ and the circle centered at s_3 in Figure 3.3) formed by calculating the intersecting points between $A_{sense}(s_3)$ and $V(s_3)$. A similar case is also shown for sensor s_{29}.

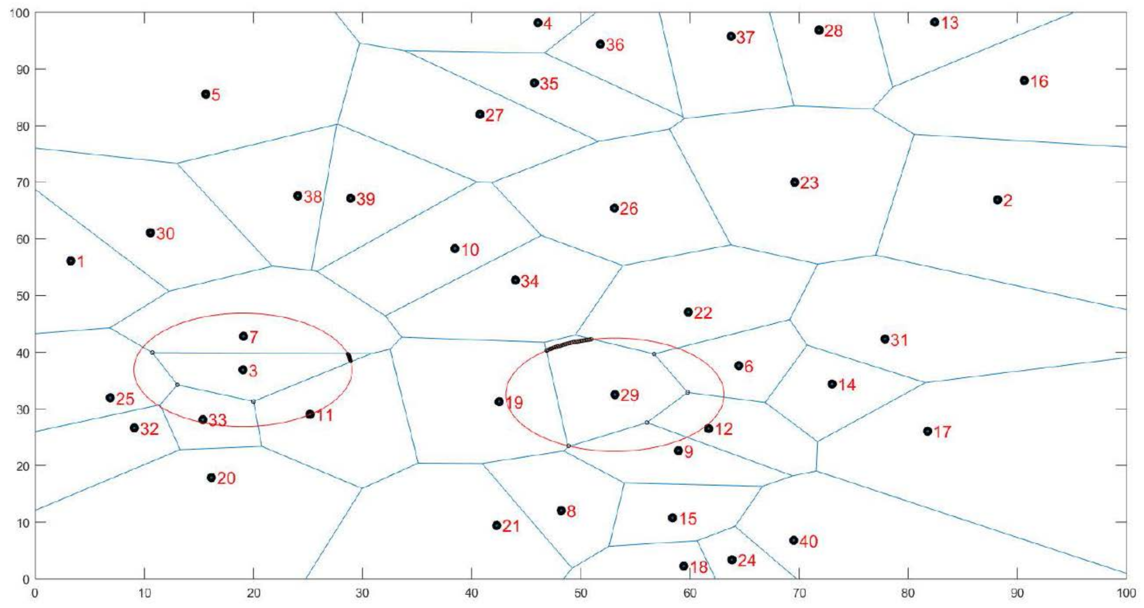

Figure 3.3 Disjoint-area sensed by a sensor.

- *Case 4*: The sensor node s_i is a border node.

This case arises when a sensor s_i is located near the boundary of the monitoring region, and the *Voronoi* polygons formed are not closed polygons. In such a case, to evaluate the disjoint area sensed by the border node (say s_{18}, or s_{24}), it is crucial to enumerate the intersecting points between $V(s_i)$, $A_{sense}(s_i)$, and the simulation region boundary. This generates a non-polygonal region $V''(s_i)$, consisting of i) the common points among $D(s_i)$ and $C(s_i)$ and $\boldsymbol{\mathcal{L}}$, and, ii) the points nearer to $K(s_i)$ among $D(s_i)$, $C(s_i)$ and $\boldsymbol{\mathcal{L}}$. Then the disjoint area assigned to the sensor node is $A(s_i) = \Delta V''(s_i)$, where $\Delta V''(s_i)$ denotes the area of the polygon $V''(s_i)$. This is achieved by following *Lines 25-29* of **Algorithm 1.1**.

3.4.3 Enumeration of Link Matrix

This section shows the enumeration of the *available* links in set L of an mWSN $G = \{\{v\} \cup N, L\}$. We use an adjacency matrix $\mathbb{L}$ of size $|N| \times (|N| + 1)$ to represent the availability of each link in $\mathbb{L}$. Let $\mathbb{L}_{i,j}$ denote the entry in $\mathbb{L}$ at row i and column j. $\mathbb{L}_{i,j}$ is set to '1' if the link is *available*; otherwise $\mathbb{L}_{i,j}$ is set to '0'. More specifically, we have $\mathbb{L}_{i,j} = 1$ when the link satisfies *condition* (*i*) and (*ii*) of Section 3.2.2. In other words, each entry $\mathbb{L}_{i,j} \in \mathbb{L}$

signifies the capability of sensor s_i to transfer the sensed data to its neighbor node s_j. It also signifies that despite node s_i being in ACTIVE or RELAY state, it may not have sufficient energy to transfer data to its next node s_j. Further, while the energy is adequate, the link is operational with reliability γ_{ij}. To account for a link's functioning with reliability γ_{ij}, while constructing the matrix $\mathbb{L}$, a random value $test_{L_{ij}}$, following a uniform distribution ($0 < test_{L_{ij}} \leq 1$), is generated. This $test_{L_{ij}}$ is generated when a link's end nodes, s_i, and s_j, are either in ACTIVE or RELAY state, and they have sufficient energy for data transmission. Formally, we have

$$\mathbb{L}_{i,j} = \begin{cases} 1 & test_{L_{ij}} \leq \gamma_{ij} \cap (n_i = 1) \cap (n_j = 1) \\ 0 & otherwise \end{cases}. \tag{3.10}$$

where $n_i = 1$ and $n_j = 1$ symbolize that the sensor nodes s_i and s_j are either in ACTIVE or RELAY state, and they have sufficient energy for data transmission.

Note that the last column of matrix $\mathbb{L}$ signifies the connectivity of any node s_i in ACTIVE / RELAY state with the mobile sink node v.

3.4.4 ACR Evaluation

The enumerated matrix $\mathbb{L}$ is now analyzed to determine the coverage-oriented data flow capacity of the mWSN. A randomly generated network configuration is said to satisfy the application-specific area-coverage requirement if the area sensed by the network configuration, $A(N_A)$, is at least A_{req}. If the ACTIVE sensors of any network configuration at qth iteration and mth sink position satisfy the area-coverage requirement and have a path to the mobile sink, then we set $\chi_{q,m} = 1$. On the other hand, if the network configuration fails to meet the required area-coverage, then it is dropped and not checked for its connectivity with the sink node. In this case, we set $\chi_{q,m} = 0$. Finally, ACR is evaluated by the Monte Carlo simulation approach as

$$ACR = \frac{\sum_{q=1}^{Q} \sum_{m=1}^{M} \chi_{q,m}}{Q \times M}. \tag{3.11}$$

3.5 Simulation Results

In this section, the simulation results are presented. All simulations use the parameters given in Table 3.1, unless otherwise specified. We assume that for a square meter of the sensed area, a sensor transmits 1 bit of data. For example, a sensor with a sensed area of 100 m^2 will transmit data in a packet of size 100 bits. Failing to find a set of benchmark networks in the literature, we generate various random mWSNs (called *problems*) and use our approach to evaluate their ACR. Each *problem* refers to one spatial location for the sensors. Since we could not find any comparable approach in the literature to gauge the performance of our approach, in Section 3.5.1 we compare its fidelity and performance against a *Complete State enumeration* (*CS*) approach. As described in Section 3.5.1, a *CS* approach enumerates all possible network states for each mWSN. Thus, the *CS* approach is applicable only for use in small-sized networks because it generates an exponential number of network states in terms of the number of nodes. Section 3.5.2 presents the applicability of the proposed approach on 24 comparatively large-sized networks. Then, we study the effect of *duty-cycle* and A_{req} on ACR in Section 3.5.3. Finally, in Section 3.5.4, the effect of link reliability on ACR is studied.

Table 3.1 Parameter Settings

Parameter	Value	Parameter	Value
Number of Nodes ($\|N\|$)	50-100	Area to be covered (θ)	50% to 95%
Simulation Area ($\mathcal{A}$)	100 ×100 m^2		
r_{s_i}	80 m		
Initial Energy (E_{in})	2 Joule	Coverage Area requirement (A_{req})	$\theta \times \mathcal{A}$
E_{elec}	0.937×10^{-6} Joules per bit		
$\varepsilon_{fs}(\varepsilon_{mp})$	0.10 (0.0013) $\times10^{-12}$ Joules per bit		
E_{sense}	50×10^{-9} Joules per bit		
Duty-cycle (α_{s_i})	Random values between 0.4 to 0.7		
p_{comm} (p_{sense})	0.95 (0.8)		
γ_{ij}	0.9		

3.5.1 Comparison and Fidelity Analysis of the Proposed Approach

In this section, we explore the performance of the proposed approach against a *CS* approach for small-sized networks with $|N|$ = 5, 6, 7, and 8. The *CS* approach generates all $4^{|N|}$ possible network states that the mWSN may reside in its entire life cycle. It then analyzes each of these $4^{|N|}$ states for being a successful network state. More specifically, the *CS* approach comprises of four steps:

1) Generate a set NS containing all possible $4^{|N|}$ network states. Note that each state $NS_m \in NS$ consists of nodes in ACTIVE, RELAY, SLEEP, and FAIL states. Therefore, $NS_m = \{N_A \cup N_R \cup N_S \cup N_F\}$, where N_A, N_R, N_S, and N_F denote the set of sensors in ACTIVE, RELAY, SLEEP, and FAIL states, respectively.
2) For each state $NS_m \in NS$, check if NS_m satisfies condition (i) of Section 3.3. If so, store it in $\mathbb{T}$.
3) For each $\mathbb{T}_n \in \mathbb{T}$
    a) Construct a connectivity matrix CM. Each entry $CM_{ij} \in CM$ is set to "1" when any sensor s_j lies within the transmission range of a sensor s_i, otherwise, we have CM_{ij} = 0. The connectivity of each ACTIVE node in $\mathbb{T}_n$ with the sink is then checked by performing the *node-fusion* (Deo 1974) on CM. Note that this step addresses *condition* (*ii*) of Section 3.3.
    b) If all nodes in $\mathbb{T}_n$ are connected to the sink node, then the mWSN is in a successful state, and the probability that $\mathbb{T}_n$ arises is

$$Prob(\mathbb{T}_n) = \prod_{i\in N_A} p_A(s_i) \prod_{i\in N_R} p_R(s_i) \prod_{i\in N_S} p_S(s_i) \prod_{i\in N_F} p_F(s_i). \qquad (3.12)$$

4) The reliability of the mWSN is then calculated as

$$ACR' = \sum_{n=1}^{|\mathbb{T}_n|} Prob(\mathbb{T}_n). \qquad (3.13)$$

It is important to mention here that ACR' in (3.13) is the ACR when links are perfectly reliable, and each node always has sufficient energy to transmit data to its next node. In other words, in the *CS* approach, the existence of a link between sensors s_i and s_j is always dominated by the transmission ranger_{s_i} of the sensor s_i. More specifically, for the *CS* approach, we set $d(i,j) \leq r_{s_i}$, and $\gamma_{ij} = 1$ for *condition* (*ii*) in Section 3.2.2. For a fair comparison of the proposed approach against the *CS* approach, **Algorithm 1** determines the link-state matrix in *Line 13* with $d(i,j) \leq r_{s_i}$, and $\gamma_{ij} = 1$.

For both the *CS* approach and the proposed approach, the monitoring area is set to 10×10 m^2, duty-cycle is 0.8, and the minimum required fraction of the total area (θ) is 50%. In both approaches, the mobile sink collects data every t = 5 seconds, and it moves along the x-axis of the monitoring field until it reaches the simulation boundary. The assumed speed of the mobile sink is 1 *m*/*sec*. In **Algorithm 1**, for each *problem*, we set $Q = 10000$, and thus it has 20000 simulation runs per *problem*.

The compared results of the proposed approach and the *CS* approach for $|N|$ = 5, 6, 7, and 8 are given in Table 3.2. The second column of Table 3.2 shows the number of random problems generated corresponding to each N. A total of 10 random problems were developed. The third and fourth columns of Table 3.2 depict ACR values evaluated by the proposed approach and *CS* approach, denoted by ACR'. Recall that ACR' considers each node to always have sufficiently high energy for data transmission, and each link to be reliable.

To show the effect of node-energy constraint on ACR, we use the proposed approach to generate ACR values for each problem in Table 3.2 with each node having initial energy $E_{in} = 2$ J; note that we still consider reliable links. The results are shown in Table 3.3; for convenience, Table 3.3 also shows the ACR' values of Table 3.2. As can be seen from Table 3.3, energy constraint in each node reduces the mWSN's reliability, i.e., the *reliability* values when nodes have energy constraints are less than the *reliability* values for nodes without energy constraints. More specifically, across all problems, i.e., for $|N|$= 5 to $|N|$ = 8, the ACR value is reduced by up to 43.67%; note its ACR' and ACR values. This depicts a realistic aspect of the fact that despite sensors being in ACTIVE or RELAY state, after few rounds of communication, the sensors might not have a sufficient amount of energy to transmit the sensed data to their neighbors, thus resulting in unsuccessful communication. This leads to a reduction in the reliability value. Table 3.3 also shows the required CPU time to generate each ACR value. As shown, generating ACR is faster as compared to generating ACR'; see the sixth columns in Table 3.2 and Table 3.3. This is because the link-state matrix, generated while evaluating the ACR'**,** has more number of connected nodes as compared to the link-state matrix generated while evaluating the ACR. The *node-fusion* method for checking the connectivity with the sink node takes a longer time when the matrix contains a larger number of connected nodes.

Table 3.2 Network simulation results

Number of Nodes	Problem No.	ACR'		Simulation Error (%)	Time (sec)	
		Proposed Approach	CS Approach		Proposed Approach	CS Approach
5	1	0.938909	0.928675	0.001	1.3233	2.2026
	2	0.847067	0.846336	0.086	1.2921	2.0631
6	3	0.817767	0.817623	0.017	1.4988	27.6558
	4	0.718950	0.716285	0.370	1.6301	21.4357
	5	0.846550	0.846228	0.038	1.7688	32.4486
7	6	0.860617	0.859103	0.175	3.1210	521.4552
	7	0.922850	0.921986	0.093	2.0075	530.8450
	8	0.939917	0.939764	0.016	1.8149	518.2398
8	9	0.901737	0.900067	0.185	2.6754	8161.0605
	10	0.866516	0.865200	0.152	2.7714	8226.4729

Table 3.3 ACR results on small networks

Number of Nodes	Problem No.	ACR'		Proposed Approach	
		Proposed Approach	CS Approach	ACR	Simulation time (sec)
5	1	0.938908	0.928675	0.894675	0.000246
	2	0.847067	0.846336	0.607350	0.000213
6	3	0.817767	0.817623	0.538217	0.000264
	4	0.718950	0.716285	0.404950	0.000241
	5	0.846550	0.846228	0.583580	0.000245
7	6	0.860617	0.859103	0.628433	0.000526
	7	0.922850	0.921986	0.682650	0.000388
	8	0.939917	0.939764	0.722467	0.000379
8	9	0.901737	0.900067	0.630750	0.000312
	10	0.866516	0.865200	0.651650	0.000277

3.5.2 Application of ACR to Larger Networks

To show its scalability, the proposed approach is applied to networks of sizes $|N|$ = 50, 60, 70, 80, 90, and 100. For each N, the proposed approach is run four times, giving 24 randomly generated networks/*problems*. For these simulations, we set $\theta = 50\%$; see Table 3.1 for the remaining parameters.

Table 3.4 shows the resulting ACR value and the average energy consumed to transmit sensed data to the mobile sink per simulation run (shown in a bracket, and given in milliJoule). One would expect a higher value of ACR when the number of nodes within the same simulation region is increased. However, the results in Table 3.4 are against the intuition, i.e., more nodes do not always increase ACR value. More specifically, while increasing $|N|$ = 70 to $|N|$ = 80 in *problem* 4 improves ACR from 0.69190 to 0.90018, increasing $|N|$ = 60 to $|N|$ = 70 in *problem* 3, decreases the ACR value from 0.99952 to 0.73776. This is because, with more nodes, the number of successful communication increases. In some cases, this causes the sensor nodes to die out earlier, leading to unsuccessful communication in later simulation runs. This causes a reduction in ACR value.

By analyzing the energy consumption values in Table 3.4, one cannot conclude that larger ACR values imply larger energy consumption. Rather, stochastic behavior is observed. For example, for $|N|$= 90, the energy consumed by the mWSN of *problem* 2 with

Table 3.4 ACR results on large networks

	Prob. No.	Number of Nodes					
		50	60	70	80	90	100
ACR (*E_Trans*)	1	0.75910 (*1.03524*)	0.76612 (*0.55907*)	0.96387 (*0.65320*)	0.99975 (*0.56167*)	0.98192 (*2.36098*)	0.93013 (*3.23094*)
	2	0.67603 (*3.95523*)	0.79136 (*2.99145*)	0.99953 (*0.37600*)	0.99818 (*1.52657*)	0.91456 (*2.20738*)	0.99633 (*0.56113*)
	3	0.68028 (*2.91286*)	0.99952 (*0.90649*)	0.73776 (*3.11549*)	0.92839 (*2.16130*)	0.99967 (*0.85014*)	0.99989 (*0.90939*)
	4	0.62073 (*1.6684*)	0.99963 (*0.80993*)	0.69190 (*2.99985*)	0.90018 (*2.08195*)	0.99935 (*0.77504*)	0.98977 (*0.44087*)

ACR = 0.9145 is 2.20738 mJ, whilst the energy used by the mWSN in *problem* 3 with ACR = 0.99967 is 0.85014 mJ. This shows the combined effect of random duty-cycle, random node-states, node energy, mWSN's topology, and/or link reliability.

3.5.3 Effect of Duty-cycle and A_{req} on ACR

Table 3.5 and Table 3.6 show the impact of duty cycle and A_{req} on ACR. While performing this analysis, the links constructed are considered to be always reliable, i.e., $\gamma_{ij} = 1$. The proposed approach is applied on six large arbitrary networks with $|N|$ = 200, 300, 400, and 500 sensor nodes. We set the duty-cycle α_{s_i}= 0.8 for Table 3.5 and α_{s_i} = 0.9 for Table 3.6, and θ varies from 50% to 90% for both tables; note that θ = 50% is equivalent to A_{req} = 100 × 100 × 50% = 5000 m^2 of the sensed area. All other parameters are given in Table 3.1. As can be seen from Table 3.5 and Table 3.6, increasing the duty-cycle from 0.8 to 0.9 enables more number of nodes to stay awake, and thus the ACR values in Table 3.6 is greater than the ACR values in Table 3.5. For example, let us consider the ACR values of Table 3.5 and Table 3.6 for *problem* 1 with $|N|$ = 300, and θ = 50%. An increase in the duty-cycle from 0.8 to 0.9 leads to an increase of ACR value by 11%.

Table 3.5 and Table 3.6 also show the effect of the values of θ on ACR. As shown in Table 3.5, increasing θ from 50% to 90% for each *problem* decreases the ACR values. As an example, for *problem* 1 with $|N|$ = 200, increasing θ from 50% to 90% leads to a decrease in ACR value by 2.1%. This is because the same mWSN now has to satisfy a larger area-coverage requirement. Analyzing the results in Table 3.6, it can be concluded that Table 3.6 follows the trend of Table 3.5, in terms of the effect of θ.

3.5.4 Effect of Link Reliability on ACR

This section depicts the effect of link reliability on ACR. A total of 18 random *problems* are generated for $|N|$ = 50, 60, 70, 80, 90, and 100. In this section, we set θ = 50%, while all other parameters obey Table 3.1. As evident from Table 3.7, increasing the link reliability from 0.5 to 1, consistently improves the ACR value. This is because, with higher link reliability, more links are formed amongst the sensors as well as between the sensors and the mobile sink. As a result, the possibility of each ACTIVE node being connected to the mobile sink increases; this eventually increases the ACR value.

Table 3.5 ACR values for WSN with duty-cycle (α_{s_i}) = 0.8

Number Of Nodes	Problem Number	θ (%)	ACR	Number Of Nodes	Problem Number	θ (%)	ACR
200	1	50	0.76744	300	1	50	0.76211
		60	0.76388			60	0.76033
		70	0.76211			70	0.75889
		80	0.75544			80	0.75867
		90	0.75146			90	0.75211
	2	50	0.76122		2	50	0.76122
		60	0.75767			60	0.75744
		70	0.75689			70	0.75711
		80	0.75578			80	0.75700
		90	0.75300			90	0.75689
400	1	50	0.76478	500	1	50	0.76611
		60	0.76422			60	0.76467
		70	0.76189			70	0.76056
		80	0.75844			80	0.75789
		90	0.75744			90	0.75400

Table 3.6 ACR values for WSN with duty-cycle (α_{s_i}) = 0.9

Number Of Nodes	Problem Number	θ (%)	ACR	Number Of Nodes	Problem Number	θ (%)	ACR
200	1	50	0.86367	300	1	50	0.85744
		60	0.86011			60	0.85333
		70	0.85678			70	0.85311
		80	0.85444			80	0.85122
		90	0.85311			90	0.84833
	2	50	0.85856		2	50	0.85522
		60	0.85656			60	0.85411
		70	0.85478			70	0.85200
		80	0.85289			80	0.85033
		90	0.85000			90	0.84744
400	1	50	0.86089	500	1	50	0.86167
		60	0.85911			60	0.85933
		70	0.85900			70	0.85711
		80	0.85589			80	0.85522
		90	0.85211			90	0.85411

Table 3.7 ACR values with increasing link reliability

$\|N\|$	Prob. No.	Link Reliability 0.5	0.7	0.8	0.9	0.99	1
50	1	0.37949	0.38145	0.38202	0.38208	0.38210	0.3821
	2	0.23812	0.24408	0.24751	0.25049	0.25270	0.25277
	3	0.65167	0.65463	0.65511	0.65528	0.65532	0.65535
60	1	0.95554	0.96766	0.97144	0.97324	0.97393	0.97398
	2	0.57819	0.61946	0.63841	0.64728	0.64925	0.64930
	3	0.93673	0.94682	0.94889	0.95036	0.95112	0.95115
70	1	0.59373	0.59863	0.60252	0.60581	0.60797	0.60828
	2	0.79428	0.79631	0.79648	0.79653	0.79654	0.79654
	3	0.81585	0.87247	0.89610	0.90700	0.90985	0.91004
80	1	0.30191	0.30248	0.30293	0.3034	0.30356	0.30356
	2	0.54929	0.64035	0.65495	0.65869	0.65975	0.65976
	3	0.87010	0.87712	0.87874	0.87944	0.87963	0.87966
90	1	0.97932	0.98441	0.98571	0.98607	0.98621	0.98623
	2	0.54929	0.64035	0.65495	0.65869	0.65975	0.65976
	3	0.59193	0.62922	0.64209	0.64645	0.64757	0.64764
100	1	0.98837	0.99113	0.99169	0.99229	0.99249	0.99254
	2	0.64524	0.69221	0.70102	0.70397	0.70503	0.70515
	3	0.96030	0.97550	0.97804	0.97902	0.97915	0.97915

3.6 Chapter Summary

Among numerous challenges, when designing a new area-monitoring scheme, maintaining connectivity and maximizing network reliability stand out as critical challenges. In this chapter, a simulation scheme to approximate the WSN coverage-oriented reliability under multistate nodes is proposed. The proposed approach accounts for node-states, node and link reliability, and nodes performance on network's capability of fulfilling an application-specific area-coverage requirement.

Chapter 4

A Monte-Carlo Markov Chain Approach for Coverage-area Reliability of Mobile Wireless Sensor Networks with Multistate Nodes

Keywords
Coverage area reliability,
Markov chain,
Monte -Carlo,
Node energy,
Network reliability,
Random duty-cycle,
WSN.

This chapter presents a new coverage-reliability index, coined as CORE that unifies i) application-specific coverage area requirement, ii) multistate nodes, iii) nodes' residual energy, and iv) reliability, in one frame. A Monte-Carlo Markov Chain simulation approach is proposed for evaluating $CORE$. The computational experiments are carried on WSNs of various sizes to demonstrate the versatility of the proposed approach.

4.1 Introduction

Wireless Sensor Networks have achieved the glory of practically implementing 'automation' in the real-world, covering almost all areas, viz., healthcare, habitat monitoring, precision agriculture (Gutierrez *et al.* 2014), animal tracking (Dominguez-Morales *et al.* 2016), forest fire detection (Garcia-Jimenez *et al.* 2017), early landslide (Giri *et al.* 2019), earthquake detection (Rahman *et al.* 2016) and so on. A mobile Wireless Sensor Network (mWSN) is composed of a large number of tiny, inexpensive resource-constrained sensors scattered in the field of interest, with the sink node moving around the field. One fundamental concern of an mWSN is to provide application-specific coverage of the area under surveillance. The reliability of an mWSN depends on sensing area coverage, network connectivity, and data handling capacity of the mWSN in the presence of multistate sensors. To mention here, each sensor node during its lifecycle may exist in ACTIVE, SLEEP, RELAY, IDLE, or FAIL states due to its hardware components success/failure, random duty cycle, and/or energy limitations. To assess the performance of an mWSN under such a scenario, this chapter proposes a new coverage-reliability index, $CORE$. $CORE$ gives a measure of the ability of a sensor network with multistate nodes to satisfy the application-specific coverage area requirement with reliable data delivery to the mobile sink. A Monte-Carlo Markov Chain simulation approach is proposed for evaluating $CORE$. Further, a Discrete-Time Markov Chain model is proposed to study the multistate behavior of a sensor node. $CORE$ is expected to help the network designers to investigate the network performance in the presence of sensor node dynamics.

This chapter is organized as follows. Section 4.2 describes the basic building blocks for the proposed approach. Section 4.3 discusses the proposed approach. Section 4.4 shows the simulation results, and Section 4.5 summarizes the chapter.

4.2 Preliminaries

The proposed mWSN model with multistate nodes is introduced here before formalizing the problem. This work considers the following assumptions:

- The sensor nodes are statistically independent.
- The sensor nodes are battery-powered.
- There is no obstacle between any two sensor nodes.
- Sensor nodes are resource-constrained, and the sink node has sufficient resources.
- All states of a node are mutually exclusive.
- Links are perfectly reliable.

4.2.1 Network Model

We model an mWSN as a graph $G = ((v) \cup N,\ L)$ with a mobile sink v travelling along a pre-defined path in the sensor field at a constant speed u. A communication link exists between any two sensor nodes s_i and s_j iff i) the Euclidean distance between the two nodes is less than r_{s_i} where r_{s_i} is the transmission range of the sensor s_i and ii) node s_i has sufficient amount of energy to transmit the data to its neighbor node s_j. At any point of time, the network's connectivity is determined by the location of the sensor nodes, the state of the sensor nodes and residual energy of the sensor nodes. We have used the energy model as discussed in Section 3.2.2.

4.2.2 Sensor Node-States

Depending on the hardware architecture (see Figure 2.1), random duty-cycle, and/or node energy, a sensor node can take on any of the states discussed below during the network operation.

a) ACTIVE: a state in which the sensor node can sense its environment, transmit the sensed data to its neighbors, and receive the forwarded data from its neighbors. The energy spent by a node s_i in this state is $E_{spent,s_i^A} = E_{Tx,s_i} + E_{Rx,s_i} + E_{sense,s_i}$.

   In due time, due to lack of residual energy, a sensor node in an ACTIVE state may not have sufficient energy to transmit data to its neighbor. In such cases, it enters into the IDLE state where it waits to send/receive the data. Thus, in the IDLE state, the sensor has data to send/receive, the transceiver is ready and active but cannot send/receive due to insufficient residual energy. Any node in IDLE state in the current round will remain in that state for this round but may become ACTIVE in the next round. The energy spent by a node in the IDLE state is obtained with a best-fit correlation of $E_{spent,s_i^I} = 0.58823 \times E_{spent,s_i^A}$, where the correlation co-efficient is found to be around 0.99. The estimation process is well established in the literature (Xu *et al.* 2001).

b) RELAY: a state in which a node can transmit and receive data but cannot sense due to the failure of the sensing unit. The energy spent by a node in this state is $E_{spent,s_i^R} = E_{Tx,s_i} + E_{Rx,s_i}$.

   A sensor node currently working in RELAY state can enter the IDLEr state if it does not have sufficient energy to transmit the data to the next node. The energy spent by a node in IDLEr state is $E_{spent,s_i^{I_R}} = 0.58823 \times E_{spent,s_i^R}$.

c) SLEEP: a state in which it can neither sense nor transmit or receive data (temporarily). After a random time period, $(1-\alpha_{s_i})$, any sensor (ACTIVE) node can move to this state. The energy spent by a node in the SLEEP state is $E_{spent,s_i^S} = 0.70588 \times E_{spent,s_i^A}$ (Xu *et al.* 2001).

d) SLEEPr: a state in which it can neither sense nor transmit or receive data (temporarily). After a random time period, (1−α_{s_i}), any sensor (RELAY) node can move to this state. The energy spent by a node in SLEEPr state is $E_{spent,s_i^{S_R}} = 0.70588 \times E_{spent,s_i^R}$.

e) FAIL: a state in which it can neither sense nor transmit or receive data (permanently) due to failure of either sensing unit or communication unit or exhaustion of the battery. Six different failure states (see Figure 4.1 – Figure 4.2) are identified based on the combinatorial effects of component failure and duty-cycle.

4.2.3 Coverage-area Reliability ($\boldsymbol{CORE}$)

The proposed approach aims at providing a method that quantifies the ability of mWSN with multistate nodes to provide an application-specific sensing coverage of the region under surveillance. Thus, mWSN coverage-oriented reliability is defined as follows:

$CORE$: Given an application-specific coverage-area requirement, A_{req}, to monitor a certain region of interest, $CORE$ is defined to be the probability that the ACTIVE nodes can transmit the application-specific required amount of coverage-area oriented data to the mobile sink node for a given period of time.

4.3 Proposed Approach and Methodology

The proposed approach starts by enumerating the random states of nodes due to the random duty-cycle, node energy, and/or hardware failure. The random network state is enumerated each time the sink changes its location while moving through a pre-defined path of the monitoring region (Zhang *et al.* 2017). Given an mWSN with its sensors located randomly in the monitoring field and the sink moving along a pre-defined path, the proposed $CORE$ evaluation process is a two-step approach:

Step 1. Generate Network Configuration for each sink position.

Step 1.1. Form a new network configuration owing to generated random node-states, which depicts the effect of current (residual) node energy, hardware failure, and random duty cycle for each sensor node (see Section 4.3.1).

Step 1.2. Evaluate the area covered under such network configuration (see Section 4.3.2).

Step 1.3. Enumerate the energy-available link-state matrix considering the energy required for transmitting data to the next node and residual energy of each node (see Section 4.3.3).

Step 1.4. Utilize the energy-available matrix to find if a path exists from the sensor nodes satisfying the application-specific coverage area requirement to the sink node (see Section 4.3.4).

Step 2. Evaluate the $CORE$ of the mWSN (see Section 4.3.5).

4.3.1 Random Generation of Sensor Node States

Sensor node-states are modeled using a Discrete-Time Markov Chain (DTMC) model. This model characterizes: i) the behavior of a single sensor as well as the dynamics of the entire network, ii) the application-specific coverage area requirement, and iii) random duty-cycle. At any point of time, due to hardware failure, random duty-cycle, and/or node energy, any sensor node can be in any of the states mentioned in subsection 4.2.4. Our complete sensor node states model is shown in Figure 4.1, and the graphical model is shown in Figure 4.2, where the links show the transitional probabilities from one state to another. In Figure 4.1, for any node-state, 'SU', 'CU', and 'B' denote the sensing unit, the communication unit, and the battery; and q_{su}, q_{cu} and q_b denote the failure probabilities of the sensing unit, the communication unit, and the battery respectively. Failure of the processing unit leads to entire node failure. Hence, in our study, the processing unit is assumed to be perfectly reliable. In Figure 4.2, each link $P_{\mathcal{M}_i,\mathcal{M}_j}$ denotes the probability of transition from state '$\mathcal{M}_i$' to state '$\mathcal{M}_j$', where, $\mathcal{M} \in [A, S, R, S_R, F_1, F_2, F_3, F_4, F_5, F_6]$. The symbols $A, S, R, S_R, F_1, F_2, F_3, F_4, F_5, F_6$ denote the ACTIVE, SLEEP, RELAY, SLEEPr and FAIL ($F_1 - F_6$) states respectively. Noticeably, in the DTMC model, the IDLE state being a low-energy state of a sensor is hidden. To mention here, the energy deficiency of a sensor (say s_i) to transmit data to its neighbor varies from any other sensor (say s_j). In such a scenario, a suitable solution would be to determine a threshold energy level for all the sensors above which the sensor would act as an ACTIVE node and below which it would act as a non-ACTIVE node; and incorporate it in the DTMC model with the IDLE state as a separate state. However, this would not have depicted the actual effect of residual energy on the connection making capability of each sensor. Hence, it is not shown in the DTMC model. However, to account for the actual effect of non-identical residual energy of each sensor on $CORE$, in the proposed approach, an Energy Available Matrix (discussed in subsections 4.3.3 and 4.3.4) that takes care of the energy-oriented link formation capability of any sensor node, is constructed.

The DTMC approach for determining the state of a sensor node can be described as follows:

Let *P* be a transitional probability matrix, where each element $P_{\mathcal{M}_i,\mathcal{M}_j}$ denotes the probability that in one time slot (t seconds), the chain moves from state $\mathcal{M}_i$ to $\mathcal{M}_j$. The matrix *P* is shown in (4.1). At the onset of the operation, it is assumed that all the sensor nodes are in the ACTIVE state; hence the initial condition for any sensor is given by (4.3). Note that in (4.1) and (4.3), the rows and columns are ordered as ACTIVE, SLEEP, RELAY, SLEEPr, and FAIL (F_1 – F_6) states. As an example, $P_{1,2}$ represents the probability that in one time slot (t seconds) the chain moves from ACTIVE to SLEEP state. After every t seconds, a random number Z ($0 < Z < 1$) is generated for each sensor node, determining the *current* state of the sensor node. Current probabilities of a sensor node s_i of being in any of the *ten* states after n time intervals is represented by a probability vector $P_i(n)$; Theorem 1, given next, shows the method of enumerating $P_i(n)$. The generated number Z is then compared with $P_i(n)$ to determine the vector X_i^n, representing the *current* state of the sensor node s_i.

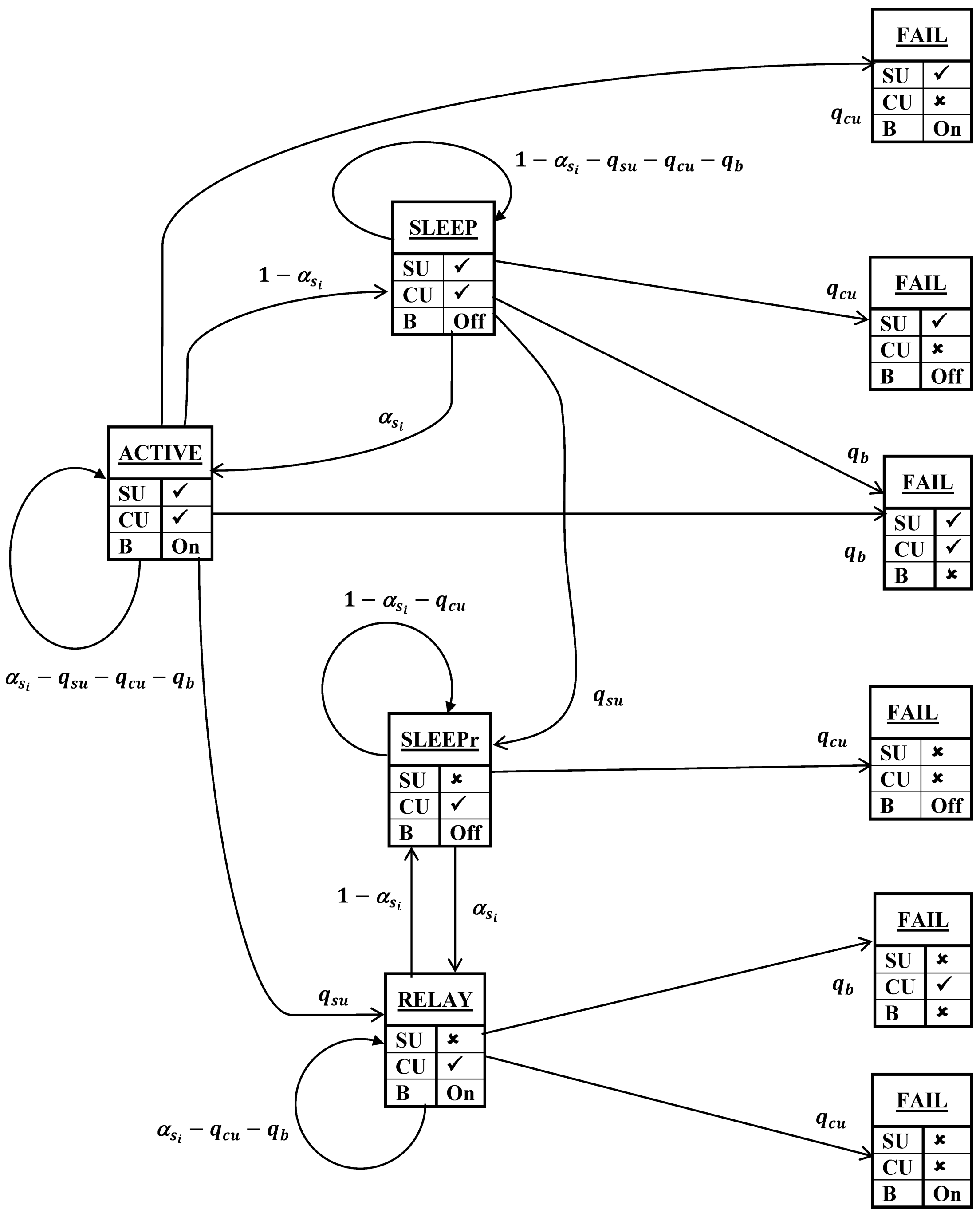

Figure 4.1 Markov Model for sensor node behavior

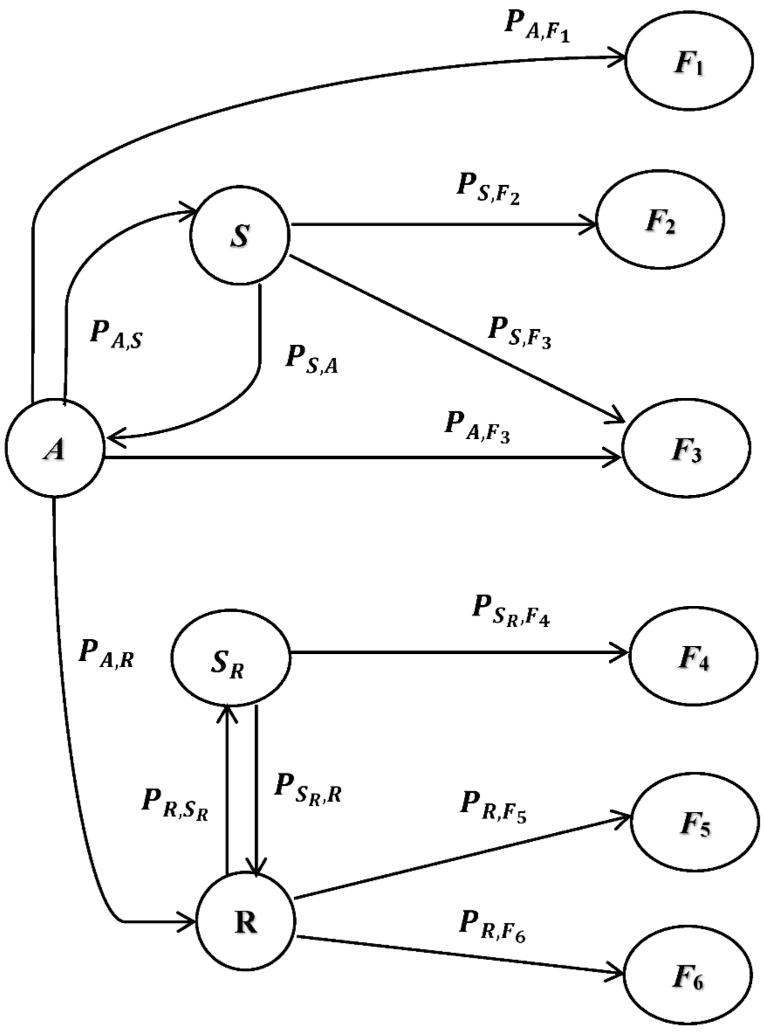

Figure 4.2 Graphical representation for the Markov model

$$P = \begin{bmatrix} (\alpha - q_{cu} - q_b - q_{su}) & (1-\alpha) & q_{su} & 0 & q_{cu} & 0 & q_b & 0 & 0 & 0 \\ \alpha & (1-\alpha-q_{cu}-q_b-q_{su}) & 0 & q_{su} & 0 & q_{cu} & q_b & 0 & 0 & 0 \\ 0 & 0 & (\alpha - q_{cu} - q_b) & (1-\alpha) & 0 & 0 & 0 & 0 & q_b & q_{cu} \\ 0 & 0 & \alpha & (1-q_{cu}-\alpha) & 0 & 0 & 0 & q_{cu} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

Theorem 1: *The probability vector representing the state probabilities after n time intervals of any sensor node s_i is*

$$P_i(n) = P_i(n-1)P \tag{4.2}$$

where $P_i(n-1)$ is the probability vector representing the state of a sensor s_i after the (n–1) *time interval.*

Proof: The Markovian property of any component states that *the probability that the component will undergo a transition from one state to another depends on the current state of the component and not on any previous states the component might have experienced* (Ebeling 2000; Billinton and Allan 1994). Following the same property, in the proposed technique, the *current state* of the sensor node depends on its immediate preceding state and not on any other previous states that the sensor node might have undergone. As the sensor nodes are assumed to start operating in an ACTIVE state, the initial probability vector for any sensor, s_i, is given by (4.3),

$$P_i(0) = [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]. \tag{4.3}$$

Equation (4.3) portrays that the probability of the sensor node being at ACTIVE state at time zero (onset of the operation) is unity, and any other state is zero.

Assuming the sink is at the position, (x_1, y_1) at the end of the first time interval, the probability vector representing the state probability at this time is

$$P_i(1) = P_i(0)P \tag{4.4}$$

At this stage, a random number Z is generated and compared with the *current state* probability $P_i(1)$ of a sensor s_i, to determine the *current state* of the sensor s_i. Depending on the values of Z and $P_i(1)$, the *current state* of the sensor s_i after the first time interval is enumerated by following **Algorithm 1**, and the corresponding column (representing *states*) entry in the vector X_i^1 is set to ‘1’.

```
Algorithm 1: EnumNodeStates($P$, $N$, $n$): Simulate node-state of each node $s_i$
Input:
$P$: Transitional probability matrix
$N$: A set of sensor nodes
$n$: time interval
Output:
$X_i^n$: current state of sensor $s_i$
1.  For each sensor $s_i$ in $N$
2.      Generate a random number $Z$ ($0 < Z < 1$) from a uniform distribution
3.      Enumerate $P_i(n)$                    // Theorem 1
4.      Compare $Z$ and $P_i(n)$ to find the current state of the sensor node
5.      If $Z \in (0, \sum_{j=0}^{j=1} P_{i,j}(n)]$ // $P_{i,j}(n)$ denotes the $j^{th}$ column of the vector $P_i(n)$
6.          $X_i^n$ = [1 0 0 0 0 0 0 0 0 0]          // represents ACTIVE state
7.      ElseIf $Z \in ( \sum_{j=0}^{j=1} P_{i,j}(n), \sum_{j=0}^{j=2} P_{i,j}(n)]$
8.          $X_i^n$ = [0 1 0 0 0 0 0 0 0 0]          // represents SLEEP state
9.      ElseIf $Z \in ( \sum_{j=0}^{j=2} P_{i,j}(n), \sum_{j=0}^{j=3} P_{i,j}(n)]$
10.         $X_i^n$ = [0 0 1 0 0 0 0 0 0 0]          // represents RELAY state
11.     ElseIf $Z \in ( \sum_{j=0}^{j=3} P_{i,j}(n), \sum_{j=0}^{j=4} P_{i,j}(n)]$
12.         $X_i^n$ = [0 0 0 1 0 0 0 0 0 0]          // represents SLEEPr state
13.     ElseIf $Z \in ( \sum_{j=0}^{j=4} P_{i,j}(n), \sum_{j=0}^{j=5} P_{i,j}(n)]$
14.         $X_i^n$ = [0 0 0 0 1 0 0 0 0 0]          // represents $F_1$ state
15.     ElseIf $Z \in ( \sum_{j=0}^{j=5} P_{i,j}(n), \sum_{j=0}^{j=6} P_{i,j}(n)]$
16.         $X_i^n$ = [0 0 0 0 0 1 0 0 0 0]          // represents $F_2$ state
17.     ElseIf $Z \in ( \sum_{j=0}^{j=6} P_{i,j}(n), \sum_{j=0}^{j=7} P_{i,j}(n)]$
18.         $X_i^n$ = [0 0 0 0 0 0 1 0 0 0]          // represents $F_3$ state
19.     ElseIf $Z \in ( \sum_{j=0}^{j=7} P_{i,j}(n), \sum_{j=0}^{j=8} P_{i,j}(n)]$
20.         $X_i^n$ = [0 0 0 0 0 0 0 1 0 0]          // represents $F_4$ state
21.     ElseIf $Z \in ( \sum_{j=0}^{j=8} P_{i,j}(n), \sum_{j=0}^{j=9} P_{i,j}(n)]$
22.         $X_i^n$ = [0 0 0 0 0 0 0 0 1 0]          // represents $F_5$ state
23.     ElseIf $Z \in ( \sum_{j=0}^{j=9} P_{i,j}(n), \sum_{j=0}^{j=10} P_{i,j}(n)]$
24.         $X_i^n$ = [0 0 0 0 0 0 0 0 0 1]          // represents $F_6$ state
25.     End If
26. End For
```

After determining the *current state* of all the sensor nodes, Step 1.2 to Step 1.4 of the proposed approach (given in Section 4.3) are followed to evaluate $CORE$ of the current network configuration. The mobile sink then moves to the next position (say (x_2, y_2)). Now, at this point, the *current state* of a sensor in the first time interval becomes the *previous state* of the corresponding sensor in the second time interval.

Thus, the probability vector representing the state probability after the two time intervals is given in (4.5).

$$\underbrace{P_i(2)}_{\textit{Current state probability after two time intervals}} = \underbrace{P_i(1)}_{\textit{Previous state vector}}\underbrace{P}_{\textit{Transitional probability matrix}}. \tag{4.5}$$

At this stage, a random number, Z, is again generated and compared with the *current state* probability $P_i(2)$ of a sensor s_i to determine the *current state* of the sensor s_i after two time intervals. Depending on the values of Z and $P_i(2)$, the *current state* of the sensor s_i after the second time interval is enumerated, and the corresponding column (representing *states*) entry in the vector X_i^2 is set to '1'. Following the same process, after determining the *current state* of all the sensor nodes, Step 1.2 to Step 1.4 of the proposed approach (given in Section 4.3) are followed to evaluate the $CORE$ of the current network configuration. The mobile sink then moves to the next position (say (x_3, y_3)). At this point, the *current state* probability vector, $P_i(2)$, of the second time interval is set to X_i^2, which now becomes the previous state probability vector for the third time interval.

Consequently, the principle can be extended to give (4.2). ■

4.3.2 Evaluation of Disjoint-Area

This section uses the disjoint area sensing scheme, *DASS* approach proposed in Section 3.4.2 to evaluate the disjoint-area sensed by each sensor $s_i \in N_A$.

4.3.3 Energy Available Matrix (EAM) Enumeration

This section involves enumeration of the Energy Available Matrix (EAM) depicting the status of the *energy available links* between sensors. Any sensor, s_j, might be within the transmission range of a sensor s_i, but the energy available in the sensor s_i may not be sufficient enough to transmit the sensed data to its neighbor s_j. In such cases, the sensor s_i (in ACTIVE or RELAY state) enters into IDLE/IDLE$_r$ state to conserve energy and remains in that state for the current round. If the node has a sufficient amount of energy required to transmit data to its neighbors, it establishes *energy available links* with its neighbors. This energy-available network topology information is stored in EAM and is used to evaluate the $CORE$ of mWSNs. Each element in this matrix stores energy-available connectivity information with its neighbors and with the mobile sink node. The mobile sink travels along a special path (such as road, pipeline, etc. which cannot be changed (Zhang *et al.* 2017)) at a constant speed u to collect data. Data is collected after

Algorithm 2: *EnumEnergyAvalMatrix*(N, E_{Tx}, r_{s_i}): *Simulate energy available link status of the WSN*

Input:
N: A set of sensor nodes
E_{Tx}: Energy required for data transmission
r_{s_i}: is the transmission range of a sensor s_i
Output:
EAM: Energy Available Matrix

```
1.  For $i = 1, 2, \dots, |N|$
2.      If any $s_i$ is in ACTIVE or RELAY state
3.          For $j = 1, 2, \dots, |N| + 1$
4.              If any $s_j$ is in ACTIVE or RELAY state
5.                  $d_{i,j} \leftarrow$ Euclidian distance between $s_i$ and $s_j$
6.                  If $d_{i,j} \leq r_{s_i}$
7.                      Evaluate $E_{Tx,s_i}$
8.                      If ($E_{Tx,s_i} \leq E_{Res,s_i}$) // by following (4.7)
9.                          $EAM_{i,j} = 1$
10.                     Else
11.                         $EAM_{i,j} = 0$
12.                     End If
13.                 End If
14.             End If
15.         End For
16.         $EAM \leftarrow$ load $EAM_{i,j}$ into the matrix $EAM$   // by following (4.8)
17.     Else
18.         break
19.     End If
20. End For
```

every t seconds. It signifies that sensors change their states randomly after every t seconds.

As shown in (4.6), EAM is $|N| \times (|N| + 1)$ matrix, where the last column denotes the energy-available connection of any sensor s_i with the sink node v. Each entry in EAM signifies the capability of a sensor s_i to transfer the sensed data to its neighbor node, and each entry in the matrix $EAM_{i,j}$ can be enumerated by following in (4.7). The energy spent by each sensor, and the residual energy of each sensor is updated after every t seconds. The residual energy of each sensor s_i is calculated by following (4.8). The detailed process for EAM enumeration is presented in **Algorithm 2**.

$$EAM = \begin{bmatrix} EAM_{1,1} & EAM_{1,2} \cdots & EAM_{1,|N|} & EAM_{1,v} \\ EAM_{2,1} & EAM_{2,2} \dots & EAM_{2,|N|} & EAM_{2,v} \\ \vdots & \ddots & \ddots & \ddots \\ EAM_{|N|,1} & EAM_{|N|,2} \dots & EAM_{|N|,|N|} & EAM_{|N|,v} \end{bmatrix} \tag{4.6}$$

$$EAM_{i,j} = \begin{cases} 1 \ if \ s_i \ \& \ s_j \text{ are ACTIVE or RELAY and } E_{Tx,s_i} \le E_{Res,s_i} \\ 0 \ otherwise \end{cases} \tag{4.7}$$

$$E_{Res,s_i} = E_{cur,s_i} - E_{spent,s_i^{\mathcal{M}}} \tag{4.8}$$

where, E_{cur,s_i} is the sensor node energy before starting a round, E_{Res,s_i} is the residual energy of a sensor after completing a round, $E_{spent,s_i^{\mathcal{M}}}$ is the energy spent in the round by a sensor s_i in state $\mathcal{M} \in [A, S, R, S_R, F_1, F_2, F_3, F_4, F_5, F_6]$.

4.3.4 Utilization of EAM for $CORE$

At this stage, for the network configurations that satisfy the application-specific coverage-area requirement, the EAM enumerated in Section 4.3.3 is used to evaluate a connectivity path between the participating sensors and the mobile sink. Any network configuration that fails to satisfy the required coverage-area (A_{req}) is dropped and is not checked further for its connectivity with the mobile sink. The randomly generated network configuration is said to satisfy the application-specific required coverage-area, if the area sensed by the current network configuration, $A(N_{\boldsymbol{A}})$ is greater than or equal to A_{req}. EAM is now utilized to check if the mWSN configuration that satisfies A_{req} is connected to the mobile sink node. Therefore, for each simulation run, let $path = 1$ iff the mWSN satisfies the following conditions:

Condition (i) $A(N_A) \ge A_{req}$ and

Condition (ii) Each sensor $s_i \in N_A$ is connected to the mobile sink node.

If any $s_i \in N_{\boldsymbol{A}}$ is not connected to the sink, then $path = 0$, indicating the situation where the mWSN is not able to deliver A_{req} despite the network is successful in sensing a minimum area of A_{req}.

Algorithm 3: $\boldsymbol{EvalConnectivity}(EAM, A(N_A), A_{req}, N_A)$: *Evaluates connectivity of mWSN*

Input:
EAM: Energy Available Matrix
$A(N_A)$: Area sensed by the WSN
A_{req}: Application-specific required coverage area
N_A: a set of sensors in ACTIVE state
Output:
$path$: network connectivity

```
1.  If $A(N_A) < A_{req}$            //condition ($i$)
2.      $path \leftarrow 0$
3.  Else
4.      $i \leftarrow 1$
5.      While ($i \leq |N_A|$)
6.          If $EAM_{i,|N|+1} = 1$  // condition ($ii$)
7.              $path \leftarrow 1$
8.              Remove $s_i$ from $\boldsymbol{S}$
9.          Else
10.             $EAM1 = EAM$
11.             $[path_{s_i}, N_A] \leftarrow \boldsymbol{isConnected}(EAM1, N_A)$
12.         End If
13.         If $path_{s_i} \neq 1$
14.             $path \leftarrow 0$
15.             break
16.         End If
17.     End While
18. End If
```

Algorithm 3 gives the pseudocode for evaluating the connectivity of the mWSN. *Line 1-3* checks *condition* (i); specifically, *Line 2* sets the variable *path* to '0' if the network configuration does not satisfy *condition* (i). If any network configuration satisfies *condition* (i), then *Lines 6-12* check *condition* (ii). More specifically, *Lines 6-8* check if a direct connection exists between any $s_i \in \boldsymbol{N_A}$ and the sink. For a direct connection, if it exists, *Line 7* assigns a value '1' to the variable *path*. If any source node, s_i is not directly connected to the sink node, then **Algorithm 3** calls **Algorithm 3.1**, in *Line 11* to check if the source node s_i is connected to the sink node via intermediate nodes. **Algorithm 3.1** uses the *node-fusion* (see *Lines 22-25*) method to check if all nodes in N_A are connected to the sink node via intermediate nodes. These intermediate nodes might be in ACTIVE or RELAY state. In other words, it is quite possible that for any source node s_i, source node s_j, where $i \neq j$ can act as intermediate nodes towards the sink. In such cases, as a result of *node fusion*, if sensor s_i is connected/not connected to the

Algorithm 3.1: $\boldsymbol{isConnected}(EAM, N_A)$: checks connectivity with the sink node

```
Input:
$EAM$: Energy Available Matrix
$N_A$: A set of sensors in ACTIVE state
Output:
$\Psi$: A connectivity vector
$N_A$: A set of sensors in ACTIVE state (updated)
1.  Initialize $u \leftarrow \Phi$
2.  $Indx \leftarrow$ Find $EAM_{1,ACol}$ containing '1' // $ACol\ denotes\ All\ columns$
3.  while (1)
4.          For $u1 = 1, 2, ..., |N_A|$
5.              For $v = 1, 2, ..., |Indx|$
6.                  If $Indx_{1,v}$ matches $N_{A_{1,u1}}$
7.                      Append $u1$ to $u$
8.                  End If
9.              End For
10.         End For
11.         If $u \neq \Phi$
12.             Delete $N_{A_{\boldsymbol{u}}}$
13.         End If
14.         If $Indx \neq \Phi$
15.             If $EAM_{rows} \geq 1$
16.                 $\Psi \leftarrow 0$
17.             Else
18.                 $\Psi \leftarrow 1$
19.             End If
20.             break
21.         End If
22.         For $i = Indx$
23.             Replace $EAM_{1,ACol}$ by $ORing$ elements in $EAM_{1,ACol}$ with $EAM_{i,ACol}$
24.             Replace $EAM_{ACol,1}$ by $ORing$ elements in $EAM_{ACol,1}$ with $EAM_{ACol,i}$
25.         End For
26.         Delete $EAM_{Indx,ACol}$ and $EAM_{ACol,Indx}$
27.         If $EAM$ has only $one$ row
28.             $\Psi \leftarrow 1$
29.             break
30.         End If
31. End while
```

sink, then the intermediate nodes are also connected/not connected to the sink. Specifically, *Lines 6-8* keep a record of such intermediate nodes that are also source nodes, and *Lines 11-13* delete such nodes from the variable N_A, if s_i is connected to sink, to save the computational time. Again, any node $s_i \in N_A$ in isolation results in network disconnectivity. Thus, *Line 14* checks for isolation of any sensor s_i. EAM represents a connected graph only when the number of rows equals one. If EAM has only one row, it indicates that the network has been successfully fused to a single node; hence the network containing $N_{\boldsymbol{A}}$ nodes is connected, and the connectivity vector is assigned '1' (*Line 28*). When the $path_{s_i}$ value enumerated in *Line 11* of **Algorithm 1** is '0', it indicates the disconnectivity of one or more source nodes with the sink. This violates *condition* (*ii*). Therefore, *Line 14* sets the variable *path* to '0', indicating the disconnectivity of the mWSN.

4.3.5 $CORE$ Evaluation

The number of network states increases exponentially with the number of nodes. To evaluate exact reliabilities of such type of mWSNs is both time consuming and computationally expensive, hence infeasible. Thus, in the proposed work, a Monte-Carlo (MC) simulation method is designed to find the mWSN coverage-area reliability. The MC method is a straightforward simulation method for large complex networks. Assuming Q as the number of simulations for each sink position, and W is the number of sink-stop positions, $CORE$ is evaluated by the Monte Carlo simulation approach as given in (4.9).

$$CORE = \frac{\sum_{pos=1}^{W}\left(\sum_{q=1}^{Q} path_q\right)}{W \times Q} \tag{4.9}$$

4.4 Results and Discussion

In this analysis, the mobile sink node collects data at every *t* = 5 second while it moves along the x-axis ($y = 0, x \in (0,100)$) of the monitoring field until it reaches the boundary. Note that the sink mobility model used in the proposed approach follows (Zhang *et al.* 2017). However, the proposed approach is not constrained only to such type of sink movement. The speed of the mobile sink is assumed to be 1 *m*/*sec*. The simulation parameters are set as shown in Table 4.1 unless otherwise specified. To mention here, without loss of generality, the sensing radius and the communication radius are assumed to be the same in our simulation. Section 4.5.1 shows the application of the proposed approach on random networks of various sizes when the application-specific coverage-area requirement is set to 70% and 90% of the total monitoring region. The effects of various scenario metrics, viz., varying duty cycle, varying network size, varying coverage-area, varying transmission range on $CORE$ is studied in Section 4.5.2 to Section 4.5.4. Section 4.5.5 discusses the efficiency of the proposed approach.

4.4.1 Application of $CORE$ on Random Networks

Random networks are generated with $|N|$ = 50, 60, 70, 80, 90, 100, 150, and 200. For each value of N, four random networks are generated, giving a total of 32 random networks to be analyzed. Table 4.2 includes the $CORE$ values for these 32 random networks when A_{req} is set to 70% of the total monitoring region.

It can be observed from the $CORE$ values in Table 4.2 that, despite increasing the number of nodes within the same simulation region, the $CORE$ value does not always increase. Rather, randomness is observed in the $CORE$ values. This is due to more number of nodes in a non-ACTIVE state, signifying the effect of random duty-cycle, residual energy of nodes, and spatial position of nodes on $CORE$.

Table 4.1 Parameter Settings

Parameter	Value
Number of Nodes ($\|N\|$)	50-200
Area ($\mathcal{A}$)	100×100 m^2
r_{s_i}	65 m
Initial Energy (E_{in})	2 Joule
E_{elec}	0.937×10^{-6} Joules per bit
$\varepsilon_{fs}(\varepsilon_{mp})$	0.10 (0.0013) $\times10^{-12}$ J per bit
E_{sense}	50×10^{-9} Joules per bit
Duty-cycle (α_{s_i})	Random values between 0.1 to 0.7
p_{cu} (p_{su})	0.99999 (0.99998)
p_b	0.9999
Area to be covered (θ)	70% to 90%
Data collection interval (t)	5 seconds
Coverage Area requirement (A_{req})	$\theta \times \mathcal{A}$
Number of Simulations (Q)	(86400 * t) / $\mathcal{A}$

Table 4.2 $CORE$ results on mWSNs with $A_{req} = 0.7 \times \mathcal{A}$

Prob. No.	Number of Nodes ($\lvert N \rvert$)							
	50	60	70	80	90	100	150	200
1	0.47006	0.53351	0.99996	0.85428	0.48573	0.57297	0.93048	0.99974
2	0.30822	0.56257	0.96026	0.66536	0.52400	0.88266	0.95358	0.99969
3	0.41498	0.41466	0.99872	0.84895	0.99887	0.84363	0.51589	0.97851
4	0.27708	0.44241	0.77801	0.99872	0.99781	0.99961	0.93529	0.99995

Table 4.3 $CORE$ results on mWSNs with $A_{req} = 0.9 \times \mathcal{A}$

Prob. No.	Number of Nodes ($\lvert N \rvert$)			
	90	100	150	200
1	0.659072	0.994323	0.890101	0.615454
2	0.340850	0.982076	0.999977	0.994686

Table 4.3 shows the $CORE$ values for mWSN with 90, 100, 150, and 200 sensors distributed over an area of 100 × 100 m^2 with 90% area-coverage requirement. As depicted in Table 4.3, the results are consistent with Table 4.2. This confirms the effects of random duty-cycle and random node-positions (spatial location) on $CORE$. It can also be concluded that increasing node-density does not much impact on $CORE$ for mWSNs with a random duty cycle.

A relationship between $CORE$, node-density, and transmission range has been explored through Figure 4.3 and Figure 4.4 to study the impact of sink movement on $CORE$. As depicted in Figure 4.3, for a particular N (say $|N|$ = 40), increasing transmission range increases the reliability. Moreover, it can be observed in Figure 4.3 that as the sink approaches the middle of the field, $CORE$ increases. This is due to an increase in the number of communicating paths and the possibility of remaining connected with the sink node.

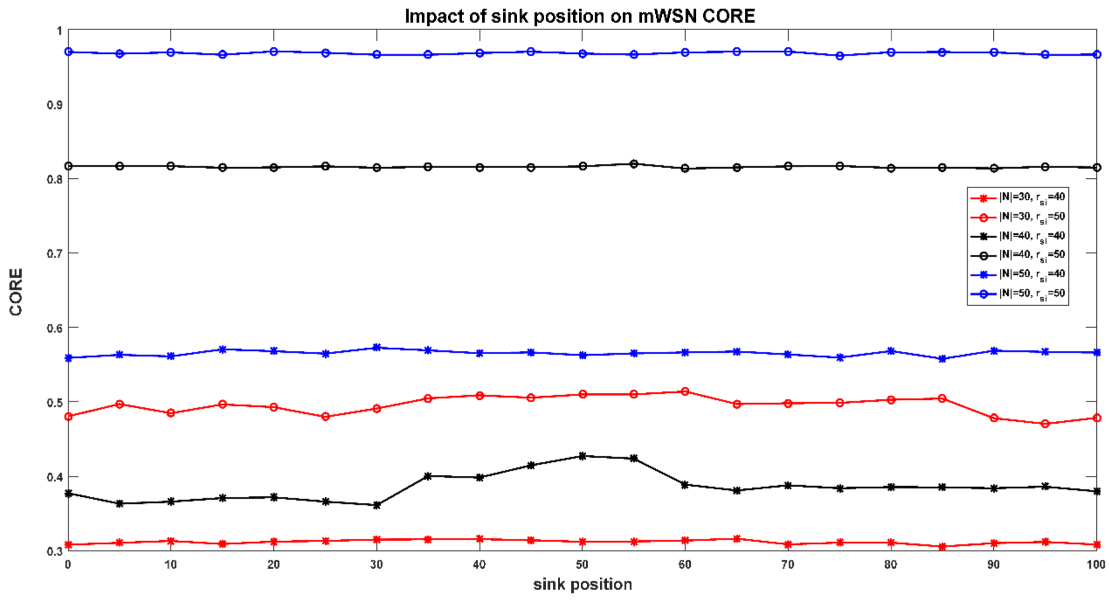

Figure 4.3 Impact of sink movement on $CORE$

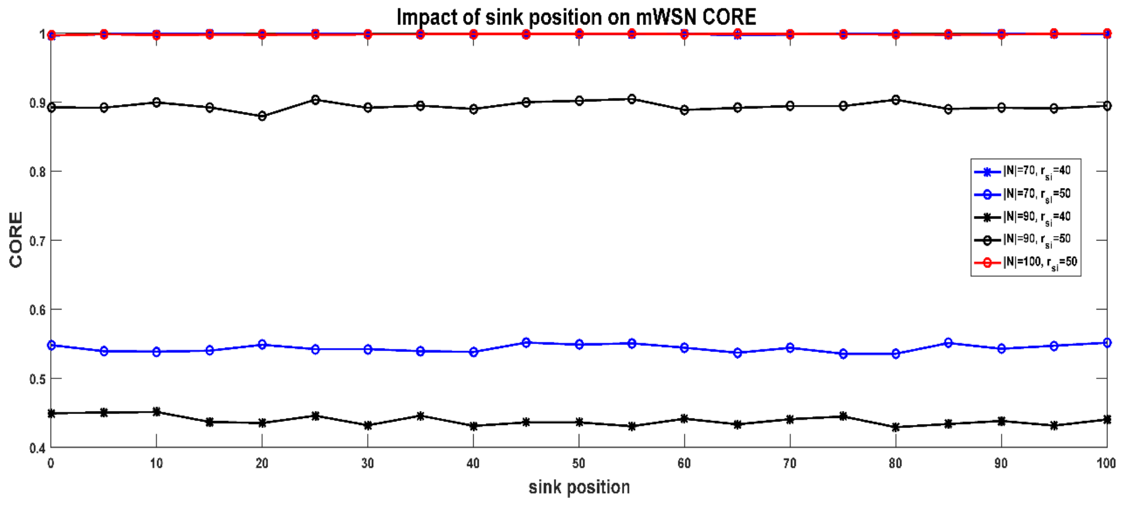

Figure 4.4 Impact of sink movement and transmission range on $CORE$

Figure 4.4 depicts the relationship between sink movement and $CORE$ values for large networks. The number of sensors chosen was $|N|$ = 70, 90, and 100, with transmission ranges of 40 m and 50 m. As the number of nodes in the simulation region is increased, it is intuitive that the $CORE$ will increase with the increase in the number of nodes. This shows the effect of node density on the $CORE$. The results in Figure 4.4 confirms the discussion.

For the same number of nodes (say for $|N|$ = 90) within the same simulation area but with random duty-cycled sensors, increasing the transmission range increases the network reliability. As depicted in Figure 4.4, it is intuitive that the network reliability will be proportional to the ratio of transmission range to the coverage area.

4.4.2 Effect of Duty-cycle on $CORE$

Table 4.4 shows the effect of varying duty-cycle on $CORE$. The duty-cycle is varied from 0.5 to 0.9. Twelve random mWSNs with $|N|$ = 50, 60, 70, 80, 90, and 100 for $A_{req} = 0.7 \times \mathcal{A}$ are analyzed. Table 4.4 shows that for any N, increasing the duty-cycle improves $CORE$. This is because a larger duty-cycle leads to more number of ACTIVE nodes in the monitoring area resulting in more number of successful communication. In other words, increasing the duty-cycle results in more number of nodes monitor the same area at any point of time, and thus each node has a lesser region to sense and transmit to small distances. Thus, each sensor spends less amount of energy. Each sensor can, therefore, remain working for a longer number of simulations, thus contributing to the larger value of $CORE$.

Table 4.4 Effect of varying duty cycle on $CORE$

Problem No.	$\|N\|$	*Duty-cycle* (α_{s_i})				
		0.5	0.6	0.7	0.8	0.9
1	50	0.3959028	0.4492593	0.4781134	0.9999999	0.9999999
2		0.3318056	0.9029861	0.9999768	0.9999999	0.9999999
3	60	0.5819792	0.5820602	0.6729398	0.7700116	0.7868634
4		0.4119676	0.4119792	0.4849306	0.5765625	0.7163194
5	70	0.6517245	0.8315509	0.9999653	0.9999884	0.9999999
6		0.4374189	0.5921646	0.9716204	0.9999999	0.9999999
7	80	0.9532755	0.9999421	0.9999653	0.9999999	0.9999999
8		0.9140625	0.9999421	0.9999653	0.9999999	0.9999999
9	90	0.9821296	0.9999768	0.9999884	0.9999999	0.9999999
10		0.7148148	0.8966087	0.9999999	0.9999999	0.9999999

4.4.3 Effect of Coverage-area Requirement on $CORE$

Table 4.5 depicts the effects of increasing the required coverage-area, A_{req} on $CORE$. Ten random mWSNs with α_{s_i} = 0.6 were generated, and the effects of increasing the required coverage-area, A_{req} on $CORE$, are studied. In most cases, irrespective of N, the $CORE$ values decrease with increasing A_{req}, while in few cases, some contrasting results (say $|N|$ = 70) can be seen, where the $CORE$ value initially increases and then decreases with increasing A_{req}. This shows the effect of the randomness in the node-states due to hardware failure and energy depletion of the participating nodes.

4.4.4 Effect of Transmission Range on $CORE$

A general intuition follows that increasing the transmission range of the sensors would lead to an increase in network reliability. However, a contradictory observation has been investigated by varying the transmission range of the sensors and analyzing the effect of increasing transmission range on $CORE$. The results of increasing r_{s_i} from 40 to 80 m on mWSNs with $|N|$ = 40, 50, 60, 70, 80, 90, 100 and 200 with the duty-cycle α_{s_i} = 0.6 on $CORE$ are shown in Table 4.6. For any mWSN, from Table 4.6, it is observed that an increase in the transmission range leads to a decrease in $CORE$ values. This is because

Table 4.5 Effect of varying $A_{req} = \theta \times \mathcal{A}$ on $CORE$

Problem No.	$\|N\|$	*Area to be covered* (θ)			
		0.6	0.7	0.8	0.9
1	50	0.70086	0.79495	0.93133	0.74041
2		0.29962	0.38244	0.35133	0.35100
3	60	0.74118	0.73140	0.73781	0.73433
4		0.99944	0.99880	0.99464	0.98459
5	70	0.54765	0.54953	0.54154	0.53728
6		0.98323	0.93971	0.99161	0.98932
7	80	0.73929	0.47234	0.56325	0.51446
8		0.99995	0.99998	0.99998	0.99993
9	90	0.44078	0.44865	0.43912	0.43016
10		0.99998	0.99997	0.99995	0.99988

Table 4.6 Effect of varying transmission range on $CORE$

Problem No.	$\lvert N \rvert$	*Transmission Range* (r_{s_i})				
		40	50	60	70	80
1	50	0.3473032	0.1929745	0.1507639	0.1309606	0.1286227
2		0.8140509	0.7682755	0.7087616	0.7080440	0.7184259
3	60	0.6446181	0.4291435	0.3016782	0.2077894	0.1992477
4		0.6046296	0.5670718	0.3900116	0.3644444	0.3268171
5	70	0.9798495	0.7787500	0.7505903	0.7470255	0.7269097
6		0.9996991	0.9998958	0.9998843	0.9996644	0.7250579
7	80	0.9995139	0.9998611	0.9999653	0.9999421	0.9999537
8		0.9996991	0.8353009	0.8188079	0.7196644	0.5147106
9	90	0.9998958	0.9999884	0.9999769	0.9999685	0.9999537
10		0.9444792	0.8082153	0.7311806	0.2049884	0.1896181

Table 4.7 $CORE$ values with varied Q

Network Density $\lvert N \rvert / \mathcal{A}$	Problem No.	Runs (Q)	$CORE$	Variance ($CORE(1-CORE))/Q$	CPU time (s)
0.4	1	10000	0.82792	0.000014246	845.0907
		15000	0.83200	0.000009318	1286.0112
		20000	0.83279	0.000006963	1738.0421
		40000	0.83255	0.000003485	3711.8612
0.5	2	10000	0.99354	0.000000642	1047.1760
		15000	0.99442	0.000000369	1562.8718
		20000	0.99417	0.000000289	2054.6964
		40000	0.99299	0.000000173	4043.6092
	3	10000	0.96364	0.000003503	992.8758
		15000	0.96318	0.000002363	1528.6551
		20000	0.96079	0.000001883	2009.2231
		40000	0.96064	0.000000945	3975.0179
0.7	4	10000	0.99836	0.000000163	1107.4408
		15000	0.99808	0.000000127	1514.6092
		20000	0.99786	0.000000106	2086.9749
		40000	0.99751	0.000000061	4043.6227
	5	10000	0.95696	0.000004124	946.6014
		15000	0.99888	0.000000074	2046.1232
		20000	0.99918	0.000000040	2795.3451
		40000	0.99926	0.000000018	5405.3553
0.8	6	10000	0.99988	0.000000011	1185.2282
		15000	0.99980	0.000000013	1728.7215
		20000	0.99983	0.000000008	2316.9446
		40000	0.99982	0.000000004	4733.5752
	7	10000	0.99976	0.000000023	1342.6843
		15000	0.99984	0.000000010	1938.3324
		20000	0.99986	0.000000006	2581.2461
		40000	0.99981	0.000000047	5146.9014

increasing r_{s_i} increases the number of communication links. This increases the number of successful communication in the initial rounds of the simulation. Eventually, within a few rounds of simulation, the sensors run out of energy and enter the absorbing state (FAIL). This leads to unsuccessful communication in the later rounds of communication, resulting in a lower value of $CORE$. Thus, low $CORE$ values with higher transmission range are the result of the early death of sensors due to energy depletion.

4.4.5 Efficiency of the Proposed Approach

Table 4.7 lists the $CORE$ values and the running time required by the proposed approach to evaluate $CORE$ for an mWSN with network densities ranging from 0.4 to 0.8. For each network density, various random networks were generated, giving rise to seven random network configurations. The number of simulation runs Q, is varied to capture the efficiency and accuracy of the proposed approach. Table 4.7 shows that for large values of Q, the variation in the $CORE$ value is small. Clearly, the simulation time increases with an increase in Q. However, a non-linearity in CPU time is observed with increasing $|N|$ as in the worst case for a fully connected network, the number of communication links to be simulated increases exponentially, i.e., $(|N| \times (|N| - 1)) / 2$. It is worth mentioning that the measured CPU time is an approximation of the time required to evaluate $CORE$, as it also depends on other background processes and application demands.

4.5 Chapter Summary

In this chapter, the problem of successful delivery of application-specific coverage area by a set of sensors randomly distributed in the monitoring field has been addressed. A new reliability metric, called $CORE$, has been proposed in this chapter. The proposed Monte Carlo DTMC is a simple and computationally efficient method for computing $CORE$. The combinatorial, as well as individual effects of random duty cycle, residual energy, hardware failure on the $CORE$ of mWSNs also have been investigated. Extensive simulations have been performed to demonstrate the efficiency of the proposed algorithm.

Chapter 5

Maximizing Coverage-area Reliability of Mobile Wireless Sensor Networks with Multistate Nodes

Keywords
flow allocation,
multi-path,
multistate sensors,
multi-source WSNs,
routing,
split flow.

A new heuristic optimization approach is proposed in this chapter to improve the coverage area reliability of WSNs with multistate nodes and a mobile sink. We model the problem as an energy minimization problem and propose a smart multi-path split-flow routing approach to preserve energy. We have simulated the proposed approach on some randomly generated networks. The obtained results show improvements in the reliability and effectiveness of the proposed approach.

To conserve the limited resources of WSNs, the concept of mobile sink was introduced (Mohamed *et al.* 2017), wherein the onfield sensors are static, and the sink moves along a predefined path. One fundamental concern of an mWSN is to provide application-specific coverage of the area under surveillance. The reliability of an mWSN depends on sensing area coverage, network connectivity, and data handling capacity of the mWSN in the presence of multistate sensors. To mention here, each sensor node during its lifecycle may exist in ACTIVE, SLEEP, SLEEPr, RELAY, IDLE, IDLEr or FAIL states (see Figure 4.1) as a combined effect of its hardware components working/failure states, random duty-cycle, and/or energy limitations. The metric, $CORE$, discussed in Chapter 4, quantifies the capability of an mWSN to satisfy the application-specific coverage of an area under surveillance in the presence of multistate nodes.

As discussed in Section 1.4, to efficiently use the limited resources of sensors, various techniques that consider duty-cycle optimization, use of multiple sink nodes, optimal placement of multiple sink nodes, optimizing trajectory movement of a mobile sink, optimal deployment pattern of the sensors, network coding techniques, routing, etc., can be found in contemporary literature. However, as mWSNs are essentially multi-source single-sink networks with multistate nodes, these approaches are inapplicable. Further, although the single-path reliability maximization problem has been studied (Zonouz *et al.*

2014), multi-path reliability maximization problem for mWSNs with multistate nodes has not yet been investigated. Particularly, the effect of splitting data through the multiple paths simultaneously while transmitting the sensed data through the multistate nodes is yet to be analyzed. Therefore, in this chapter, a multi-path split-flow energy-efficient routing scheme that aims to maximize the $CORE$ of mWSNs with multistate sensor nodes is presented.

This chapter is organized as follows. Section 5.1 formulates the problem. Section 5.2 presents the proposed approach and methodology used to maximize $CORE$. Section 5.3 presents and discusses the results, and finally, Section 5.4 summarizes the findings presented in this chapter.

5.1 Problem Formulation

We aim to maximize the $CORE$ of an mWSN. Referring (4.9), it can be analyzed that the metric $CORE$ will be maximized if the successful data flow possibility of the network is increased, i.e., the mWSN remains connected for a longer time period. In other words, we aim to increase the number of times the ACTIVE nodes can successfully transmit A_{req} to the mobile sink node during Q number of simulations for W sink stop positions. Conclusively, we intend to increase the $path = 1$ (connectivity) possibilities during Q. Recall that $path$ is a binary variable that accounts for connectivity/disconnectivity of the mWSN at any qth simulation round and posth sink position where $q \in Q, pos \in W$. This connectivity possibility can be increased by minimizing the energy consumed by the mWSN in each round of communication; thereby keeping sensors in ACTIVE state for more number of communications rounds. This will eventually increase the plausibility that mWSN remains connected for more time. Note that by connected we mean that the mWSN satisfies both *condition* (i) and *condition* (ii) of Section 4.3.4.

As stated in (4.8), $E_{spent,s_i^{\mathcal{M}}}$ is the energy spent in a round by a sensor s_i in state $\mathcal{M} \in [A, S,\ R, S_R, F_1, F_2, F_3, F_4, F_5, F_6]$. Therefore, the amount of energy consumed by a random network configuration at the qth simulation round and posth sink position is equal to the sum of the energy spent by each sensor s_i at the qth simulation round and posth sink position. Mathematically, the energy spent by a random network configuration at the qth simulation round and posth sink position is shown in (5.1).

$$E_{cons}^{q,pos} = \sum_{i\in|N|} E_{spent,s_i^{\mathcal{M}}} \tag{5.1}$$

It is expected that by reducing $E_{cons}^{q,pos}$, most of the sensors will have sufficient amount of energy in later communication rounds, contributing to a higher $CORE$. Therefore, we aim to minimize the energy consumed per simulation round per sink position. This will ensure that the nodes retain their energy for long, refraining them from dying out early, and remain connected for longer duration. This heuristic approach is expected to improve $CORE$ performance.

5.2 Proposed Approach and Methodology

With an aim to minimize $E_{cons}^{q,pos}$, this section proposes a multi-path split-flow approach that minimizes the energy spent by each sensor node in a data communication round. We now explore the routes followed by each ACTIVE sensor node to deliver its sensed data to the mobile sink, and propose a smart routing scheme that helps to maximize $CORE$ by minimizing $E_{cons}^{q,pos}$. The proposed routing approach is applied at the qth simulation round and posth sink position. It is a three-step approach:

Step 1. Arrange sensors in order of number of hops required to reach the sink node and their distance from the sink node,

Step 2. Enumerate multi-source single-sink minimal paths of the network,

Step 3. Split the flow from each source node to its neighbors through the enumerated minimal paths.

All the above steps are discussed in the following subsections.

5.2.1 Arranging the ACTIVE Sensors

As a first step, the minimal paths for nodes in ACTIVE and RELAY state in the mWSN graph at the q^{th} simulation round and pos^{th} sink position are enumerated using the *MP* enumeration method of the MNRE approach presented in Chapter 2 (see Section 2.5.2.1). By following the approach in Section 2.5.2.1, we get the information about the hop-count between each of the nodes and the sink. With the help of this hop-count information, the sensor nodes are arranged in decreasing order of the minimum number of hops required to reach the sink node. Nodes having the same minimum hop-count are arranged in order of decreasing distance to the sink node. This enables the farthest source node to communicate before the nearer ones; the nearer ones can thereby receive and forward the received and their own sensed data to their neighbors towards the sink node.

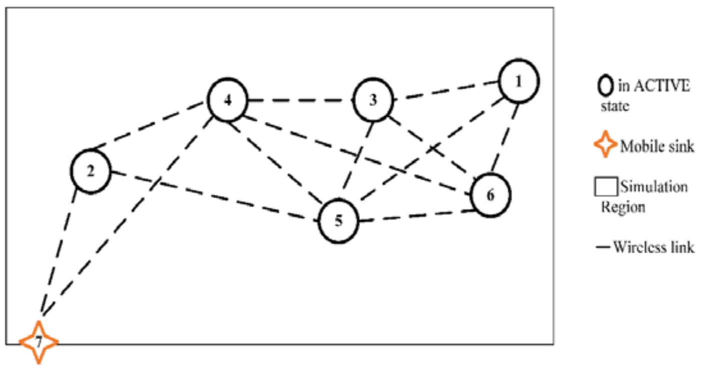

Figure 5.1 A typical mWSN

Table 5.1 List of MPs from $s_i \in N_A$ to sink for mWSN in Figure 5.1

From sensor #	Minimal Paths	Hop_i
s_1	$\{s_1, s_6, s_4, s_7\}$	3
	$\{s_1, s_5, s_4, s_7\}$	3
	$\{s_1, s_3, s_4, s_7\}$	3
	$\{s_1, s_5, s_2, s_7\}$	3
s_2	$\{s_2, s_7\}$	1
s_3	$\{s_3, s_4, s_7\}$	2
	$\{s_3, s_5, s_2, s_7\}$	3
s_4	$\{s_4, s_7\}$	1
s_5	$\{s_5, s_4, s_7\}$	2
	$\{s_5, s_2, s_7\}$	2
s_6	$\{s_6, s_4, s_7\}$	2
	$\{s_6, s_5, s_2, s_7\}$	3

For example, let us consider the mWSN in Figure 5.1 with six nodes and a mobile sink at any qth simulation round and posth sink position. The distance between the sensors $N_A = \{s_1, s_2, s_3, s_4, s_5, s_6\}$, and the sink node v is $d_{i,v} = (9.1542, 1.7947, 8.6321, 3.7375, 6.3694, 7.7734)$, where $d_{i,v}$ denotes the distance between the sensor $s_i \in N_A$ and the sink node v, for $i = 1, 2, \ldots, |N_A|$. By following the approach in Section 2.5.2.1, the set of minimal paths from each node $s_i \in N_A$ to the sink node, along with the number of hops required to reach the sink node, denoted by Hop_i, is enumerated and shown in Table 5.1. Let C_i denote the minimum number of hops required by a sensor $s_i \in N_A$ to reach the sink node, and C is an array that contains all C_i, i.e., $C = (C_i | s_i \in N_A)$. Therefore, by observing the third column of Table 5.1, we have $C =$ (3, 1, 2, 1, 2, 2), wherein C_1 contains the minimum Hop_1 = 3, C_2 contains the minimum Hop_2 = 1, and so on. Sorting C in decreasing order we have $C = (3, 2, 2, 2, 1, 1)$, and after arranging N_A in accordance to C, we have $N_A = \{s_1, s_3, s_5, s_6, s_2, s_4\}$. This indicates that sensor s_1 with a minimum hop count of 3 is the farthest node from the sink. Sensors s_3, s_5, and s_6, each is at least 2 hops away from the sink. Now, as $d_{3,v} = 8.6321, d_{5,v} = 6.3694$, and $d_{6,v} = 7.7734$, after arranging the three sensors s_3, s_5, and s_6 in decreasing order of their Euclidian distance from the sink node, we have $N_A = \{s_1, s_3, s_6, s_5, s_4, s_2\}$. This ensures that the sensor s_1 will start transmitting its data first. Sensor s_3 will then forward the received data (from s_1) along with its sensed data to its neighbors s_4 and s_5.

Algorithm 1 depicts the arrangement of the sensor nodes in the ACTIVE state. For each sensor $s_i \in N_A$, *Lines 2–5* account for the minimum hop count to the sink node. More specifically, *Line 3* accounts for the minimum hop-count to the sink, and *Line 4* stores it in an array C. *Line 6* sorts the array C in decreasing order, and *Line 7* arranges the nodes in N_A in accordance to C. If two or more sensors are equally distant from the sink node in terms of its minimum hop count, then *Lines 8–17* sort the sensors in accordance to the Euclidian distance to the sink node. More specifically, *Line 9* stores the nodes that are at same hop-count distance from the sink node. For the nodes stored in *Line 9*, *Lines 10–13* evaluate their Euclidian distance to the sink node and stores them in an array $\mathcal{D}$. *Line 14* sorts the array $\mathcal{D}$ in decreasing order and *Line 15* arranges the equi-hop-distant nodes in accordance to $\mathcal{D}$ and stores them in $Temp2$. Finally, *Line 16* updates the arranged sensors

Algorithm 1: **SortSrCNodes**(MPs, N_A):Arranges the source nodes

Input:

MPs: Minimal Pathsets for the mWSN containing minimal paths for each sensor s_i and its respective hop-count

N_A: nodes in ACTIVE state / source nodes

Output:

N_A: Ordered sensors nodes

```
1.  $C \leftarrow \varphi, \mathcal{D} \leftarrow \varphi$
2.  For each $s_i \in N_A$
3.      $Z_i = \min(Hop_i)$
4.      Append $Z_i$ to $C$
5.  End For
6.  Sort $C$ in decreasing order
7.  Arrange $N_A$ in accordance to $C$
8.  If any $C$ contains same value
9.      $Temp1 \leftarrow$ Nodes in $N_A$ having same hop-count in $C$
10.     For each $s_j \in Temp1$
11.         $d_{j,sink} \leftarrow$ Find Euclidian distance between sensor $s_j$ and sink
12.         Append $d_{j,sink}$ to $\mathcal{D}$
13.     End For
14.     Sort $\mathcal{D}$ in decreasing order
15.     $Temp2 \leftarrow$ Nodes in $N_A$ in accordance to $\mathcal{D}$
16.     $N_A \leftarrow (N_A - Temp1) \cup Temp2$
17. End If
```

in terms of their hop-count and distance to the sink node.

5.2.2 Enumerating the Network Multi-Source Minimal Paths

This section discusses about enumerating the Network multi-source Minimal Paths (NMPs) for the random network configuration at the q^{th} simulation round and pos^{th} sink position. Note that this step requires all the minimal paths from each node $s_i \in N_A$ to be known a priori. The enumeration of NMPs is a two-step approach:

1. Take each source node $s_i \in N_A$, and enumerate Multi-Source Minimal Paths (MSMP) for each source node. Note that while enumerating the MSMPs for source node s_i towards sink node v, it is possible that another source node $s_k \in N_A$ serves

as an intermediate node for s_i towards the sink node v, for $i \neq k$. Eventually, while enumerating the paths for s_i to v, the paths for s_k to v are also enumerated.

2. Check if all nodes in N_A are encountered. If not, then enumerate paths from the source nodes in N_A that are not encountered or present in the already enumerated MSMPs as intermediate nodes.

For each sensor node $s_i \in N_A$ that is not directly connected to the sink node, MSMPs are evaluated. Any *j*th minimal path, $MP_{i,j}$ of s_i, is selected, and the intermediate nodes in $MP_{i,j}$, i.e., the nodes connecting the sensor node s_i and the sink v, are sequentially explored to enumerate $MSMP_i$. The basic idea is to track the paths sequentially from each intermediate node s_k in $MP_{i,j}$ and construct $MSMP_i$ by appending $MP_{k,j}$ to $MP_{i,j} - \{s_k \cup v\}$. While enumerating $MSMP_i$, it is possible that many source nodes will be connected to the mobile sink node v through the enumerated MSMPs. If any sensor node is not connected through the already enumerated MSMPs, then the MSMP enumeration process is repeated, and the MSMPs for that node is enumerated. This completes the enumeration of NMPs. The process of MSMP enumeration for any sensor is discussed through an example presented next.

As an example, from Table 5.1 let us consider an MP, $MP_{1,1} = \{s_1, s_6, s_4, s_7\}$ from sensor s_1 to sink node v. The intermediate nodes in $MP_{1,1}$, i.e., the nodes connecting s_1 and the sink $v = 7$, are $X = \{s_6, s_4\}$. The next task is to sequentially track the MPs from

Table 5.2 List of multi-source minimal paths from sensor s_1 to sink

Node	MP	Find paths from	$MSMP_1$
s_1	$\{s_1, s_6, s_4, s_7\}$	s_4	$\{s_1, s_6, s_4, s_7\}$
		s_6	$\{s_1, s_6, s_4, s_7\}$
			$\{s_1, s_6, s_5, s_2, s_7\}$
	$\{s_1, s_5, s_4, s_7\}$	s_4	$\{s_1, s_5, s_4, s_7\}$
		s_5	$\{s_1, s_5, s_2, s_7\}$
	$\{s_1, s_3, s_4, s_7\}$	s_4	$\{s_1, s_3, s_4, s_7\}$
		s_3	$\{s_1, s_3, s_4, s_7\}$
			$\{s_1, s_3, s_5, s_2, s_7\}$
	$\{s_1, s_5, s_2, s_7\}$	s_2	$\{s_1, s_5, s_2, s_7\}$
		s_5	$\{s_1, s_5, s_4, s_7\}$
			$\{s_1, s_5, s_2, s_7\}$

each of the nodes in X to the sink node v, and append it to $\{MP_{i,j} - MP_{i,j,Indx:end}\}$ to construct $MSMP_1$. Considering $X_1 = s_6$, and appending $MP_{6,1} = \{s_6, s_4, s_7\}$, and $MP_{6,2} = \{s_6, s_5, s_2, s_7\}$ to $\{MP_{1,1} - MP_{1,1,2:end}\} = \{s_1\}$, the enumerated MSMP are $\{s_1, s_6, s_4, s_7\}$ and $\{s_1, s_6, s_5, s_2, s_7\}$. Again, considering $X_2 = s_6$, and appending $MP_{4,1} = \{s_4, s_7\}$ to $\{MP_{1,1} - MP_{1,1,3:end}\} = \{s_1, s_6\}$, the enumerated MSMP is $\{s_1, s_6, s_4, s_7\}$. Therefore, considering $MP_{1,1}$, the set of multi-source MPs enumerated are $\{s_1, s_6, s_4, s_7\}$, $\{s_1, s_6, s_5, s_2, s_7\}$ and $\{s_1, s_6, s_4, s_7\}$. Following the same process, Table 5.2 shows the enumerated MSMPs for sensor node s_1 by accounting each minimal path from sensor s_1 to sink node v. From Table 5.2, after removing redundancies, the MSMPs followed by sensor s_1 are $MSMP_1 = \{s_1, s_6, s_4, s_7\}, \{s_1, s_6, s_5, s_2, s_7\}, \{s_1, s_5, s_4, s_7\}, \{s_1, s_5, s_2, s_7\}$, $\{s_1, s_3, s_4, s_7\}$, $\{s_1, s_3, s_5, s_2, s_7\}$.

As observed from the enumerated MSMPs, all sensors in N_A are connected to the mobile sink. Hence, the already enumerated MSMPs form the NMPs. Therefore, the Network Minimal Paths for the mWSN in Figure 5.1 is $NMP = \{s_1, s_6, s_4, s_7\}, \{s_1, s_6, s_5, s_2, s_7\}, \{s_1, s_5, s_4, s_7\}, \{s_1, s_5, s_2, s_7\}, \{s_1, s_3, s_4, s_7\}, \{s_1, s_3, s_5, s_2, s_7\}$. These are the paths which are used simultaneously by each $s_i \in N_A$ to transmit their data to the mobile sink. To be more specific, the multi-paths followed by sensor s_1 is $MSMP_1 = \{s_1, s_6, s_4, s_7\}, \{s_1, s_6, s_5, s_2, s_7\}, \{s_1, s_5, s_4, s_7\}, \{s_1, s_5, s_2, s_7\}, \{s_1, s_3, s_4, s_7\}, \{s_1, s_3, s_5, s_2, s_7\}$, by s_2 is $MSMP_2 = \{s_2, s_7\}$, by s_3 is $MSMP_3 = \{s_3, s_4, s_7\}, \{s_3, s_5, s_2, s_7\}$, by s_4 is $MSMP_4 = \{s_4, s_7\}$, by s_5 is $MSMP_5 = \{s_5, s_4, s_7\}, \{s_5, s_2, s_7\}$, and by s_6 is $MSMP_6 = \{s_6, s_4, s_7\}, \{s_6, s_5, s_2, s_7\}$.

Algorithm 2 gives the pseudocode for NMPs enumeration at any qth simulation round and posth sink position. For each sensor s_i, *Lines 3–18* enumerate the MSMPs. More specifically, for each sensor, *Line 3* takes each minimal path connecting s_i and v, and *Line 4* finds the intermediate nodes between s_i and v. For each intermediate node, *Lines 7–9* record the indexes of the intermediate sensor nodes present in $MP_{i,j}$. *Lines 10–13* enumerate the multi-source minimal paths from each of the nodes in X, and stores them in $MSMP_i$. After removing redundant elements in $MSMP_i$ in *Line 16*, *Line 17* appends the irredundant $MSMP_i$s to NMP. *Line 19* finds the source nodes that are yet to be encountered, and *Line 20* updates the set of ACTIVE nodes based on the output generated in *Line 19.*

Algorithm 2: **EnumNMP**(MPs, N_A): Enumerates Network Multi-Paths

Input:

MPs: Minimal Pathsets containing minimal paths for each sensor s_i and its respective hop-count

N_A: ordered set of nodes in ACTIVE state / source nodes

Output:

NMPs: Network Minimal Paths

```
1.  $i \leftarrow 1, NMP \leftarrow \varphi$
2.  While ($i \leq |N_A|$)
3.      For $j = 1,2,\ldots,|MP_i|$
4.          $X = MP_{i,j} - \{s_i \cup v\}$
5.          For $k = 1,2,\ldots,|MP_{i,j}|$
6.              For $u = 1,2,\ldots,|X|$
7.                  If $MP_{i,j,k}$ matches $X_u$
8.                      $Indx = k$
9.                  End If
10.                 For $indx1 = 1,2,\ldots,|MP_{X_u}|$
11.                     $Temp = \{MP_{i,j} - MP_{i,j,Indx:end}\} \cup \{MP_{X_u,indx1}\}$
12.                     Append $Temp$ to $MSMP_i$
13.                 End For
14.             End For
15.         End For
16.         Remove redundancy from $MSMP_i$
17.         Append $MSMP_i$ to $NMP$
18.     End For
19.     $Y \leftarrow$ find $s_i$ not present in $NMP$
20.     $N_A = N_A - Y$
21. End While
```

5.2.3 Splitting Flow through NMPs

Aiming to save $E_{cons}^{q,pos}$, the next task is to decide the amount of flow to be transferred through each of the multi-paths enumerated in Section 5.2.2. More specifically, we aim to split the flow from each source node to its neighbor through the enumerated multi-paths. This section enlightens the proposed flow allocation scheme. As stated by Razzaque and Dobson (2014) and Tony *et al.* (2019), a sensor node consumes more energy while transmitting data to its neighbors compared to the energy spent by the sensor in receiving data from its neighbors or sensing its own area, i.e., $E_{Tx,s_i} > E_{Rx,s_i} > E_{sense,s_i}$. Hence, for each sensor node s_i, firstly, we aim to reduce the energy spent during transmission. This aim is majorly fulfilled with the usage of multi-paths while transferring data from each source node to its neighbors. Next, we look to reduce the energy consumed during reception. This is achieved by splitting the data by a sensor based on residual energy of the receiving nodes as well as the distance between the sender and the receiving nodes. Few rules are formulated that governs the flow splitting through the multi-paths.

Let $Nbr(i)$ denote the neighbors of sensor s_i. Consider sensor s_i wants to transmit data of size λ_i to all sensors in $Nbr(i)$. Let $d_{i,u}$ denote the distance from s_i to neighbor $s_u \in Nbr(i)$ that has residual energy E_{Res,s_u}, D_i be the sum of distances from s_i to each node $s_u \in Nbr(i)$, i.e., $D_i = \sum_{s_u \in Nbr(i)} d_{i,u}$, and E_i be the sum of the residual energies of the neighbor nodes, i.e., $E_i = \sum_{s_u \in Nbr(i)} E_{Res,s_u}$. Let $\lambda_{i,u}$ denote the amount of data transmitted by s_i to each neighbor $s_u \in Nbr(i)$. Thus, we have $\lambda_i = \sum_{s_u \in Nbr(i)} \lambda_{i,u}$. For ease of expression, in the following rules, consider sensor s_i has three neighbors s_u, s_v, and s_w. Thus, the three neighbors are located at a distance of $d_{i,u}$, $d_{i,v}$, and $d_{i,w}$ from s_i, and have residual energies of E_{Res,s_u}, E_{Res,s_v}, and E_{Res,s_w} respectively. We now present few rules that aid in saving $E_{cons}^{q,pos}$.

Rule 1. If $E_{Res,s_u} = E_{Res,s_v} = E_{Res,w}$, then, $\lambda_{i,u} = \frac{d_{i,u}}{D_i} \times \lambda_i$, $\lambda_{i,v} = \frac{d_{i,v}}{D_i} \times \lambda_i$ and $\lambda_{i,w} = \frac{d_{i,w}}{D_i} \times \lambda_i$. ■

Rule 2. If $d_{i,u} > d_{i,v} > d_{i,w}$, and $E_{Res,s_u} < E_{Res,s_v} < E_{Res,w}$, then $\lambda_{i,u} = \lambda_i - \left(\frac{\frac{d_{i,u}}{D_i} \times \lambda_i}{|Nbr(i)|-1}\right)$, $\lambda_{i,v} = \lambda_i - \left(\frac{\frac{d_{i,v}}{D_i} \times \lambda_i}{|Nbr(i)|-1}\right)$ and $\lambda_{i,w} = \lambda_i - \left(\frac{\frac{d_{i,w}}{D_i} \times \lambda_i}{|Nbr(i)|-1}\right)$ ■

Rule 3. If $d_{i,u} < d_{i,v} < d_{i,w}$, and $E_{Res,s_u} > E_{Res,s_v} > E_{Res,w}$, then $\lambda_{i,u} = \lambda_i - \left(\frac{\frac{d_{i,u}}{D_i} \times \lambda_i}{|Nbr(i)|-1}\right)$, $\lambda_{i,v} = \lambda_i - \left(\frac{\frac{d_{i,v}}{D_i} \times \lambda_i}{|Nbr(i)|-1}\right)$ and $\lambda_{i,w} = \lambda_i - \left(\frac{\frac{d_{i,w}}{D_i} \times \lambda_i}{|Nbr(i)|-1}\right)$. ■

Rule 4. If $d_{i,u} = d_{i,v} = d_{i,w}$, and

i. $E_{Res,s_u} > E_{Res,s_v} > E_{Res,w}$, or

ii. $E_{Res,s_u} < E_{Res,s_v} < E_{Res,w}$,

then $\lambda_{i,u} = \frac{E_{Res,s_u}}{E_i} \times \lambda_i$, $\lambda_{i,v} = \frac{E_{Res,s_v}}{E_i} \times \lambda_i$ and $\lambda_{i,w} = \frac{E_{Res,s_w}}{E_i} \times \lambda_i$ ■

Rule 5. If $d_{i,u} > d_{i,v} > d_{i,w}$, and $E_{Res,s_u} > E_{Res,s_v} > E_{Res,w}$, then

$$\lambda_{i,u} = \frac{(D_i - d_{i,u})\, E_{Res,s_u}}{(D_i - d_{i,u})\, E_{Res,s_u} + (D_i - d_{i,v})\, E_{Res,s_v} + (D_i - d_{i,w})\, E_{Res,s_w}} \times \lambda_i,$$

$$\lambda_{i,v} = \frac{(D_i - d_{i,v})\, E_{Res,s_v}}{(D_i - d_{i,u})\, E_{Res,s_u} + (D_i - d_{i,v})\, E_{Res,s_v} + (D_i - d_{i,w})\, E_{Res,s_w}} \times \lambda_i,$$ and

$$\lambda_{i,w} = \frac{(D_i - d_{i,w})\, E_{Res,s_w}}{(D_i - d_{i,u})\, E_{Res,s_u} + (D_i - d_{i,v})\, E_{Res,s_v} + (D_i - d_{i,w})\, E_{Res,s_w}} \times \lambda_i.$$ ■

Rule 6. If $d_{i,u} < d_{i,v} < d_{i,w}$, and $E_{Res,s_u} < E_{Res,s_v} < E_{Res,w}$, then $\lambda_{i,u} =$

$$\frac{(D_i - d_{i,u})\, E_{Res,s_u}}{(D_i - d_{i,u})\, E_{Res,s_u} + (D_i - d_{i,v})\, E_{Res,s_v} + (D_i - d_{i,w})\, E_{Res,s_w}} \times \lambda_i,$$

$$\lambda_{i,v} = \frac{(D_i - d_{i,v})\, E_{Res,s_v}}{(D_i - d_{i,u})\, E_{Res,s_u} + (D_i - d_{i,v})\, E_{Res,s_v} + (D_i - d_{i,w})\, E_{Res,s_w}} \times \lambda_i,$$ and

$$\lambda_{i,w} = \frac{(D_i - d_{i,w})\, E_{Res,s_w}}{(D_i - d_{i,u})\, E_{Res,s_u} + (D_i - d_{i,v})\, E_{Res,s_v} + (D_i - d_{i,w})\, E_{Res,s_w}} \times \lambda_i.$$ ■

5.3 Results and Discussion

In this section, the simulation results are presented. All simulations use the parameters given in Table 5.3, unless otherwise specified. Arbitrary networks are generated with $|N|$ = 6, 7, 8, 9, 10, and 15. For each value of $|N|$, two random networks are generated, giving a total of 12 random networks to be analyzed. Table 5.4 includes the $CORE$ values for these 11 random networks when A_{req} is set to 50% of the total monitoring region. To show its effectiveness, the proposed approach is compared with the approach in (Chakraborty *et al.* 2020). The third and fourth columns of Table 5.4 show the effectiveness of the proposed scheme. As shown in the third and fourth columns of Table 5.4, the proposed scheme increases the coverage-oriented reliability. This is because, by splitting the flow through multi-paths, the proposed approach is capable of saving energy that enables the sensors to remain in the operational state for a longer period of time, leading to a larger value of $CORE$. This comparison is limited to small network sizes of 6–15 nodes due to the computational time involved in enumerating paths of a WSN for large or moderately large-sized networks.

Table 5.3 Simulation parameters

Parameters	Value	Parameters	Value
Area ($\mathcal{A}$)	10×10 m^2	$\|N\|$	6–15
r_{s_i}	5 m	p_{comm}	0.99999
Initial Energy	2 Joule	p_{sense}	0.99998
E_{elec}	0.937×10^{-6} J per bit	$P_{battery}$	0.9999
ε_{fs}	0.10×10^{-12} J per bit	Q	10,000
ε_{mp}	0.0013×10^{-12} J per bit	A_{req}	50% of $\mathcal{A}$
α_{s_i}	Random values between 0.8 to 1		

Table 5.4 Simulation results

Number of Nodes ($\|N\|$)	Problem No.	$CORE$	
		Approach in (Chakraborty *et al.* 2020)	Proposed Approach
6	1	0.486666	0.902000
	2	0.482159	0.885761
7	3	0.538152	0.907630
	4	0.723967	0.928571
8	5	0.677084	0.865077
	6	0.796825	0.882539
9	7	0.685425	0.893217
	8	0.541269	0.807392
10	9	0.721067	0.850333
	10	0.71100	0.827667
15	11	0.713133	0.982567
	12	0.795302	0.989235

Table 5.5 $CORE$ results on different splitting schemes

Number of Nodes ($\|N\|$)	Problem No.	$CORE$	
		Equal Split	Proposed Split
6	1	0.69800	0.90200
	2	0.69039	0.88576
7	3	0.89959	0.90763
	4	0.90476	0.92857
8	5	0.78968	0.86507
	6	0.80000	0.88253
9	7	0.88177	0.89322
	8	0.76190	0.80739
10	9	0.78826	0.85033
	10	0.74810	0.82766
15	11	0.84666	0.98256
	12	0.85699	0.989235

Table 5.5 presents the comparative results of splitting the flow from each sensor equally among its neighbors, and the proposed scheme. Previously analyzed networks of sizes $|N|$ = 6 to $|N|$ = 15 are used for this comparison. As can be seen from Table 5.5, the $CORE$ values with equal splitting are less than the $CORE$ values with proposed splitting. This is because of the fact that equal splitting despite a node having less residual energy leads to the early death of such nodes, resulting in lower $CORE$ as compared to the proposed approach.

5.4 Chapter Summary

In this chapter, we studied the maximization of coverage-oriented reliability ($CORE$) through smart routing of WSNs. We have modeled the reliable routing problem as an energy consumption minimization problem. We have proposed a multi-path split-flow routing algorithm to improve reliability for packet routing in WSNs. In terms of energy efficiency, our method saves energy of sensor nodes by enabling the farthest node to communicate first and split the flow through concurrent multi-paths. This allows them to stay in the ACTIVE/RELAY state for a longer duration.

Chapter 6

Conclusions and Scope for Future Work

Keywords
Conclusions,
Future scope.

In this chapter, the contributions of the thesis are highlighted. The areas that need further investigations as the future scope of this research are also conveyed.

6.1 Conclusions

This dissertation deals with the development of some new approaches to quantify the performance of WSNs with multistate nodes.

The hardware design of a sensor node and the working/failure states of its components put a sensor node to be in either of ACTIVE/ RELAY/ FAIL states during its lifecycle. To evaluate the flow-oriented reliability of arbitrary topology WSNs under such multistate behavior of sensor nodes, a simple and computationally efficient Multi Node-State Reliability Evaluator (MNRE) approach is presented in Chapter 2. MNRE computes the exact reliability of arbitrary WSNs with multistate nodes. The proposed approach enumerates the shortest minimal paths from application-specific flow satisfying sensor nodes (source nodes) to the sink node. It then proposes a new sum-of-disjoint products approach, considering multiple states of nodes, to evaluate WSN reliability from the enumerated shortest minimal paths. Simulations are performed on WSNs of various sizes to show the applicability of the proposed approach on arbitrary WSNs. The results indicate that the proposed algorithm is computationally less intensive as fewer network states are generated than the Brute Force approach that requires generating all ($3^{|N|}$) states while it produces the same results.

Besides quantifying the data handling capability of a WSN, it is also important to quantify the ability of a WSN to provide application-specific coverage of a region of interest. The combinatorial effect of hardware components' working/failure, node energy, and/or random duty-cycle results in a sensor node to exist in either ACTIVE/ RELAY/

SLEEP/ FAIL state. Chapter 3 proposes a new performance metric, ACR, that quantifies the capability of a WSN to provide application-specific coverage of a region of interest under such multistate behavior of sensors. Generally, it is assumed that nodes within the transmission range can successfully communicate with each other. However, a node’s capability of communicating with its neighbor is also dependent on its energy. This aspect has been taken care of in Chapter 3 by considering a link’s existence only when a node is within the transmission range and has sufficient energy. Further, this chapter proposes a disjoint-area sensing scheme to preserve the energy wasted due to sensing overlapping regions by two or more sensors. The ACR information allows the network designers to achieve a better understanding of the impact of random duty cycle, node energy, node/link reliability, and randomly deployed sensors on reliability.

Chapter 4 deals with developing a DTMC approach that models the multistate behavior of each sensor node. A new metric, $CORE$, is presented, which accounts for node-states and node’s performance in terms of a network’s capability of fulfilling an application-specific area-coverage requirement. The proposed Monte Carlo DTMC is a computationally efficient method for computing $CORE$. The combinatorial, as well as individual effects of random duty-cycle, residual energy, hardware failure on the $CORE$ of mWSNs are studied. The impact of various scenarios, viz., varying duty cycle, varying network size, varying coverage-area, and varying transmission range on $CORE$ also have been investigated.

Finally, Chapter 5 aims to maximize the $CORE$. The chapter presents a heuristic split-flow multi-path approach to preserve the energy of nodes. This scheme improves the $CORE$. Each sensor node splits its data based on the distance and residual energy of neighboring nodes. The results depict that the proposed multi-path split-flow approach is able to effectively increase $CORE$. This increase implies that the probability of failure of the network due to lack of energy or unavailability of connection is reduced, and the network can survive longer.

6.2 Scope for Future Work

The present dissertation is an attempt towards the development of some new approaches for the evaluation of flow-oriented and coverage-oriented WSN reliability in the presence of multistate nodes. The following plausible recommendations can be made for future extension of the present research work:

- The main focus of the present work is on the evaluation of WSN reliability for battery-constrained sensors with non- rechargeable nodes. Future research interests in this area include the evaluation of $CORE$ with rechargeable sensor nodes and multistate links.
- The present work possesses binary (failure/success) behavior of links and binary capacity states of sensors. Due to various energy levels, an ACTIVE node may exist in different capacity states. These multiple capacity states of components have different performance levels with various effects on the performance of the entire network. The proposed methods can be extended to include such stochastic behavior of the nodes.
- Multivalued Decision Diagrams (MDDs), an extended version of the traditional BDD, is used to quantify the performance level of multistate networks. However, MDDs focusses on two-terminal reliability evaluation. Thus, in future, it would be interesting to apply MDDs for reliability analysis of multisource single sink WSNs with multistate nodes.
- Investigations can be carried out to optimally design an mWSN that maximizes $CORE$ under design constraints such as the number of sensor nodes, the transmission range, and the sensing range.
- The Markov model presented in Chapter 4 represents the multistate behavior of battery-powered sensor nodes. This could be extended to investigate the multistate nature of rechargeable sensors.
- The present work proposes approaches that are theoretically applied to various benchmark and randomly generated arbitrary networks. The implementation of the proposed methods in the real-world with appropriate statistical information is still a challenging task.
- It would be interesting to extend the present work by considering dependent failures in WSNs, e.g., common cause failures when a large number of sensors fail due to hostile weather.

- The present work focusses on evaluation of application communication reliability. Prospects to include infrastructure communication while evaluating reliability of WSNs with multistate nodes needs further exploration.
- With the growing technological development and human dependency on WSNs, these networks are frequently modified. The present work can be used for planning and designing such networks that work efficiently and effectively with a desired reliability.

REFERENCES

AboElFotoh, H. M. F., ElMallah, E. S. and Hassanein, H. S. (2006), On The Reliability of Wireless Sensor Networks, IEEE International Conference on Communication, pp. 3455–3460.

AboElFotoh, H. M. F., ElMallah, E. S. and Hassanein, H. S. (2007), A Flow-based Reliability Measure for Wireless Sensor Networks, *International Journal of Sensor Networks*, vol. 2, no. 5/6, pp. 311–320.

AboElFotoh, H. M. F., Iyengar, S. S. and Chakrabarty, K. (2005), Computing reliability and message delay for cooperative wireless distributed sensor networks subject to random failures, *IEEE Transactions on Reliability*, vol. 54, no. 1, pp. 145–155.

Alemdar, H. and Ersoy, C. (2010), Wireless sensor networks for healthcare: A survey, *Computer Networks*, vol. 54, no. 15, pp. 2688–2710.

Baali, H., Djelouat, H., Amira, A., Member, S., Bensaali, F. and Member, S. (2018), Empowering Technology Enabled Care Using IoT and Smart Devices : A Review, *IEEE Sensors Journal*, vol. 18, no. 5, pp. 1790–1809.

Billinton, R. and Allan, R. N. (1994), Reliability Evaluation of Engineering Systems: Concepts and Techniques, New York: Plenum Press.

Butenko, V. M., Nazarenko, A., Sarian, V. and Sushchenko, N. (2014), SERIES Y. 2000 : Next Generation Networks Applications of Wireless Sensor Networks in Next Generation Networks, Telecommunication Standardization Sector of ITU, pp. 1–94.

Chakraborty, S. and Goyal, N. K. (2015a), Irredundant Subset Cut Enumeration for Reliability Evaluation of Flow Networks, *IEEE Transactions on Reliability*, vol. 64, no. 4, pp. 1194–1202.

Chakraborty, S. and Goyal, N. K. (2015b), Subset Cut Enumeration of Flow Networks with Imperfect Nodes, *International Journal of Performability Engineering*, vol. 11, no. 1, pp. 81–90.

Chakraborty, S. and Goyal, N. K. (2017), An Efficient Reliability Evaluation Approach for Networks with Simultaneous Multiple-Node-Pair Flow Requirements, *Quality and Reliability Engineering International*, vol. 33, no. 5, pp. 1067–1082.

Chakraborty, S., Chaturvedi, S. K. and Goyal, N. K. (2014), Comments on an efficient method based on self-generating disjoint minimal cut-sets for evaluating reliability measures of interconnection networks, *International Journal of Performability Engineering*, vol. 10, no. 7, pp. 771–774.

Chakraborty, S., Goyal, N. K., Mahapatra, S. and Soh, S. (2020), A Monte-Carlo Markov chain approach for coverage-area reliability of mobile wireless sensor networks with multistate nodes, Reliability Engineering and System Safety, vol. 193,106662.

Chaturvedi, S. K. and Misra, K. B. (2002), A hybrid method to evaluate reliability of complex networks, *International Journal of Quality & Reliability Management*, vol. 19, no. 8/9, pp. 1098–1112.

Chen, J., Li, J., He, S., Sun, Y. and Chen, H. H. (2010), Energy-efficient coverage based on probabilistic sensing model in wireless sensor networks, *IEEE Communications Letters*, vol. 14, no. 9, pp. 833–835.

Chinrungrueng, J., Sununtachaikul, U. and Triamlumlerd, S. (2006), A Vehicular Monitoring System with Power-Efficient Wireless Sensor Networks, *International Conference on ITS Telecommunications*, pp. 951–954.

Choi, S., Kim, J. and Lee, J. (2010), REDM : Robust and Energy efficient Dynamic routing for a Mobile sink in a Multi Hop Sensor Network, *International Conference on Communication Software and Networks*, pp. 1–4.

Cook, J. L. and Ramirez-Marquez, J. E. (2007), Two-terminal reliability analyses for a mobile ad hoc wireless network, *Reliability Engineering & System Safety*, vol. 92, no. 6, pp. 821–829.

Cook, J. L. and Ramirez-Marquez, J. E. (2008), Reliability analysis of cluster-based ad-hoc networks, *Reliability Engineering and System Safety*, vol. 93, no. 10, pp. 1512–1522.

Cook, J. L. and Ramirez-Marquez, J. E. (2009), Mobility and reliability modeling for a mobile ad hoc network, *IIE Transactions*, vol. 41, no. 1, pp. 23–31.

Cristescu, R. and Beferull-Lozano, B. (2006), Lossy network correlated data gathering with high-resolution coding, *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2817–2824.

Deif, D. S. and Gadallah, Y. (2017), An Ant Colony Optimization Approach for the Deployment of Reliable Wireless Sensor Networks, *IEEE Access*, vol. 5, pp. 10744–10756.

Deo, N. (1974), Graph Theory with Applications to Engineering and Computer Science. Prentice-Hall of India.

Di Francesco, M., Das, S. K. and Anastasi, G. (2011), Data Collection in Wireless Sensor Networks with Mobile Elements, *ACM Transactions on Sensor Networks*, vol. 8, no. 1, pp. 1–31.

Dominguez-Morales, J. P., Rios-Navarro, A., Dominguez-Morales, M., Tapiador-Morales, R., Gutierrez-Galan, D., Cascado-Caballero, D., Jimenez-Fernandez, A. and Linares-Barranco, A. (2016), Wireless Sensor Network for Wildlife Tracking and Behavior Classification of Animals in Doñana, *IEEE Communications Letters*, vol. 20, no. 12, pp. 2534–2537.

Dong, Z., Shang, C., Chang, C. Y. and Roy, D. S. (2020), Barrier Coverage Mechanism Using Adaptive Sensing Range for Renewable WSNs, *IEEE Access*, vol. 8, pp. 86065–86080.

Ebeling, C. E. (2000), An Introduction to Reliability and Maintainability Engineering. Tata McGraw-Hill.

Egeland, G. and Engelstad, P. E. (2009), The availability and reliability of wireless multi-hop networks with stochastic link failures, *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 7, pp. 1132–1146.

Fan, G. and Jin, S. (2010), Coverage Problem in Wireless Sensor Network: A Survey, *Journal of Networks*, vol. 5, no. 9, pp. 1033–1040.

Garcia-Jimenez, S., Jurio, A., Pagola, M., De Miguel, L., Barrenechea, E. and Bustince, H. (2017), Forest fire detection: A fuzzy system approach based on overlap indices, *Applied Soft Computing Journal*, vol. 52, pp. 834–842.

Giri, P., Ng, K. and Phillips, W. (2019), Wireless Sensor Network System for Landslide Monitoring and Warning, *IEEE Transactions on Instrumentation and Measurement*, vol. 68, no. 4, pp. 1210–1220.

Gutierrez, J., Villa-Medina, J. F., Nieto-Garibay, A. and Porta-Gandara, M. A. (2014), Automated irrigation system using a wireless sensor network and GPRS module, *IEEE Transactions on Instrumentation and Measurement*, vol. 63, no. 1, pp. 166–176.

Habib, M. A., Saha, S., Razzaque, M. A., Mamun-or-Rashid, M., Fortino, G. and Hassan, M. M. (2018), Starfish routing for sensor networks with mobile sink, *Journal of Network and Computer Applications*, vol. 123, pp. 11–22.

Hardy, G., Lucet, C. and Limnios, N. (2007), K-Terminal Network Reliability Measures With Binary Decision Diagrams, *IEEE Transactions on Reliability*, vol. 56, no. 3, pp. 506–515.

Heinzelman, W., Chandrakasan, A. and Balakrishnan, H. (2002), An Application-Specific Protocol Architecture for Wireless Microsensor Networks, *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670.

Hsin, C. F. and Liu, M. (2006), Randomly duty-cycled wireless sensor networks: Dynamics of coverage, *IEEE Transactions on Wireless Communications*, vol. 5, no. 11, pp. 3182–3192.

Huang, C. and Tseng, Y. (2005), The Coverage Problem in a Wireless Sensor Network, *Mobile Networks and Applications*, vol. 10, no. 4, pp. 519–528.

Huang, X., Zhai, H. and Fang, Y. (2008), Robust Cooperative Routing Protocol in Mobile Wireless Sensor Networks, *IEEE Transactions On Wireless Communications*, vol. 7, no. 12, pp. 5278–5285.

In, O. and Hoc, D. (2006), MAC Protocols for Wireless Sensor Networks : A Survey, *IEEE Communications Magazine*, vol. 4, no. 4, pp. 115–121.

Jelicic, V., Razov, T., Oletic, D., Kuri, M. and Bilas, V. (2011), MasliNET: A Wireless Sensor Network based environmental monitoring system, *Proceedings of the 34th International Convention MIPRO*, pp. 150–155.

Jurdak, R., Ruzzelli, A. G. and O'Hare, G. M. P. (2010), Radio sleep mode optimization in wireless sensor networks, *IEEE Transactions on Mobile Computing*, vol. 9, no. 7, pp. 955–968.

Kabadurmus, O. and Smith, A. E. (2018), Evaluating Reliability/Survivability of Capacitated Wireless Networks, *IEEE Transactions on Reliability*, vol. 67, no. 1, pp. 26–40.

Kharbash, S. and Wang, W. (2007), Computing two-terminal reliability in mobile ad hoc networks, *IEEE Wireless Communications and Networking Conference*, pp. 2833–2838.

Kim, K., Yun, Jeongbae, Yun, Jangkyu, Lee, B. and Han, K. (2009), A Location Based Routing Protocol in Mobile Sensor Networks, *Interrnational Conference on Advanced Communication Technology*, pp. 1342–1345.

Ko, J., Lu, C., Srivastava, M. B., Stankovic, J. A., Terzis, A. and Welsh, M. (2010), Wireless sensor networks for healthcare, Proceedings of the IEEE, vol. 98, no. 11, pp. 1947–1960.

Kumar, S., Kuila, P. and Jana, P. K. (2016), Genetic algorithm approach for k -coverage and m -connected node placement in target based wireless sensor networks, *Computers and Electrical Engineering*, vol. 56, pp. 544–556.

Lee, E., Park, S., Lee, J., Oh, S. and Kim, S. H. (2011), Novel service protocol for supporting remote and mobile users in wireless sensor networks with multiple static sinks, *Wireless Networks*, vol. 17, no. 4, pp. 861–875.

Lee, E., Park, S., Yu, F. and Kim, S. H. (2010), Communication model and protocol based on multiple static sinks for supporting mobile users in wireless sensor networks, *IEEE Transactions on Consumer Electronics*, vol. 56, no. 3, pp. 1652–1660.

Lee, S. J. and Gerla, M. (2001), Split multipath routing with maximally disjoint paths in adhoc networks, *IEEE International Conference on Communications*, vol. 10, pp. 3201–3205.

Leung, R., Liu, J., Poon, E., Chan, A. L. C. and Li, B. (2001), MP-DSR: A QoS-aware multi-path dynamic source routing protocol for wireless ad-hoc networks, *IEEE Conference on Local Computer Networks*, pp. 132–141.

Li, D. and Liu, H. (2009), Sensor Coverage in Wireless Sensor Networks, Wireless Networks: Research Technology and Applications, Commack, NY, USA: Nova, pp. 1–30.

Li, J., Cui, L. and Zhang, B. (2010), Self-deployment by distance and orientation control for mobile sensor networks, *International Conference on Networking Sensing and Control*, pp. 549–553.

Liu, X. (2015), A Deployment Strategy for Multiple Types of Requirements in Wireless Sensor Networks, *IEEE Transactions on Cybernetics*, vol. 45, no. 10, pp. 2364–2376.

Liu, Y. and Liang, W. (2005), Approximate coverage in wireless sensor networks, *IEEE Conference on Local Computer Networks*, pp. 68–75.

Luo, J., Panchard, J., Piorkowski, M., Grossglauser, M. and Hubaux, J.-P. (2006), MobiRoute: Routing Towards a Mobile Sink for Improving Lifetime in Sensor Networks, *Distributed Computing in Sensor Systems*, pp. 480–497.

Ma, M. and Yang, Y. (2007), Adaptive triangular deployment algorithm for unattended mobile sensor networks, *IEEE Transactions on Computers*, vol. 56, no. 7, pp. 946–958.

Marina, M. K. and Das, S. R. (2001), Ad hoc on-demand multipath distance vector routing, *International Journal of Wireless Communications and Mobile Computing*, vol. 6, no. 7, pp. 969–988.

Men, M. and Chen, L. (2016), Node Energy and State Probabilities in Linear Wireless Sensor Networks, *International Journal of Performability Engineering*, vol. 12, no. 1, pp. 95–99.

Mohamed, S. M., Hamza, H. S. and Saroit, I. A. (2017), Coverage in mobile wireless sensor networks (M-WSN): A survey, *Computer Communications*, vol. 110, pp. 133–150.

Morino, H., Kawamura, H., Inoue, M. and Sanefuji, T. (2009), Load-balanced multipath routing for wireless mesh networks: A step by step rate control approach, *International Symposium on Autonomous Decentralized Systems*, pp. 281–286.

Mostafaei, H. and Meybodi, M. R. (2013), Maximizing Lifetime of Target Coverage in Wireless, *Wireless Personal Communications*, vol. 71, pp. 1461–1477.

Mostafaei, H., Chowdhury, M. U. and Obaidat, M. S. (2018), Border Surveillance with WSN Systems in a Distributed Manner, IEEE Systems Journal, vol. 12, no. 4.

Mostafaei, H., Esnaashari, M. and Meybodi, M. R. (2015), A Coverage Monitoring algorithm based on Learning Automata for Wireless Sensor Networks, *Applied Math Inf Sci*, vol. 9, no. 3, pp. 1317–1325.

Mostafaei, H., Montieri, A., Persico, V. and Pescapé, A. (2016), An efficient partial coverage algorithm for wireless sensor networks, *IEEE Symposium on Computers and Communications*, pp. 1–6.

Mostafaei, H., Montieri, A., Persico, V. and Pescapé, A. (2017), A sleep scheduling approach based on learning automata for WSN partial coverage, *Journal of Network and Computer Applications*, vol. 80, pp. 67–78.

Pantazis, N. A., Nikolidakis, S. A. and Vergados, D. D. (2013), Energy-efficient routing protocols in wireless sensor networks: A survey, IEEE Communications Surveys and Tutorials, vol. 15, no. 2, pp. 551–591.

Park, F. Y. S. and Kim, E. L. S. (2010), Elastic routing : a novel geographic routing for mobile sinks in wireless sensor networks, *IET Communications*, vol. 4, no. 6, pp. 716–727.

Parra, L., Lloret, G., Lloret, J., Member, S. and Rodilla, M. (2018), Physical Sensors for Precision Aquaculture : A Review, *IEEE Sensors Journal*, vol. 18, no. 10, pp. 3915–3923.

Periyasamy, P. and Karthikeyan, E. (2017), End-to-End Link Reliable Energy Efficient Multipath Routing for Mobile Ad Hoc Networks, *Wireless Personal Communications*, vol. 92, no. 3, pp. 825–841.

Perkins, C. E. and Royer, E. M. (1999), Ad-hoc on-demand distance vector routing, *IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90–100.

Provan, J. S. and Ball, M. O. (1984), Computing Network Reliability in Time Polynomial in the Number of Cuts., *Operations Research*, vol. 32, no. 3, pp. 516–526.

Rahman, M. U., Rahman, S., Mansoor, S., Deep, V. and Aashkaar, M. (2016), Implementation of ICT and Wireless Sensor Networks for Earthquake Alert and Disaster Management in Earthquake Prone Areas, *Procedia Computer Science*, vol. 85, pp. 92–99.

Ramirez-Marquez, J. E., Coit, D. W. and Tortorella, M. (2006), A generalized multistate-based path vector approach to multistate two-terminal reliability, *IIE Transactions*, vol. 38, no. 6, pp. 477–488.

Razzaque, M. A. and Dobson, S. (2014), Energy-efficient sensing in wireless sensor networks using compressed sensing, *Sensors*, vol. 14, no. 2, pp. 2822–2859.

Ren, Q., Li, J. and Cheng, S. (2011), Target Tracking under Uncertainty in Wireless Sensor Networks, *International Conference on Mobile Ad-Hoc and Sensor Systems*, pp. 430–439.

Rout, R. R. and Ghosh, S. K. (2013), Enhancement of lifetime using duty cycle and network coding in wireless sensor networks, *IEEE Transactions on Wireless Communications*, vol. 12, no. 2, pp. 656–667.

Sara, G. S. and Sridharan, D. (2014), Routing in mobile wireless sensor network: A survey, *Telecommunication Systems*, vol. 57, no. 1, pp. 51–79.

Sha, K., Gehlot, J. and Greve, R. (2013), Multipath routing techniques in wireless sensor networks: A survey, *Wireless Personal Communications*, vol. 70, no. 2, pp. 807–829.

Shazly, M. H., Elmallah, E. S. and AboElFotoh H.M.F. (2010), A Three-State Node Reliability Model for Sensor Networks, *IEEE Global Telecommunications Conference*.

Shazly, M., Elmallah, E. S., Harms, J. and AboElFotoh, H. M. F. (2011), On area coverage reliability of wireless sensor networks, *Conference on Local Computer Networks*, pp. 580–588.

Shrestha, A. and Xing, L. (2008), Quantifying application communication reliability of wireless sensor networks, *International Journal of Performability Engineering*, vol. 4, no. 1, pp. 43–56.

Shrestha, A., Xing, L. and Coit, D. W. (2010), An efficient multistate multivalued decision diagram-based approach for multistate system sensitivity analysis, *IEEE Transactions on Reliability*, vol. 59, no. 3, pp. 581–592.

Shrestha, A., Xing, L. and Liu, H. (2007), Modeling and evaluating the reliability of wireless sensor networks, *Reliability and Maintainability Symposium*, pp. 186–191.

Shrestha, A., Xing, L., Sun, Y. and Vokkarane, V. M. (2012), Infrastructure communication reliability of wireless sensor networks considering common-cause failures, *International Journal of Performability Engineering*, vol. 8, No. 2, pp. 141–150.

Silva, I., Guedes, L. A., Portugal, P. and Vasques, F. (2012), Reliability and availability evaluation of wireless sensor networks for industrial applications, *Sensors*, vol. 12, no. 1, pp. 806–838.

Stoianov, I., Nachman, L., Madden, S. and Tokmouline, T. (2007), PIPENET: A Wireless Sensor Network for Pipeline Monitoring, *International Symposium on Information Processing in Sensor Networks*, pp. 264–273.

Sweidan, H. I. and Havens, T. C. (2018), Sensor relocation for improved target tracking, *IET Wireless Sensor Systems*, vol. 8, no. 2, pp. 76–86.

Tolle, G., Polastre, J., Szewczyk, R., Culler, D., Turner, N., Tu, K., Burgess, S., Dawson, T., Buonadonna, P., Gay, D. and Hong, W. (2005), A Macroscope in the Redwoods, *SenSys'05*, pp. 51–63.

Tony, T., Soh, S., Chin, K. and Lazarescu, M. (2019), Link Scheduling in Rechargeable Wireless Sensor Networks With Imperfect Batteries, *IEEE Access*, vol. 7, pp. 104721–104736.

Tsai, Y. R. (2008), Sensing coverage for randomly distributed wireless sensor networks in shadowed environments, *IEEE Transactions on Vehicular Technology*, vol. 57, no. 1, pp. 556–564.

US National Intelligence Council (2008), National Intelligence Council, Disruptive Technologies Global Trends 2025: Six Technologies with Potential Impacts on US Interests Out to 2025.

Waharte, S. and Boutaba, R. (2006), Totally disjoint multipath routing in multihop wireless networks, *IEEE International Conference on Communications*, pp. 1–6.

Wang, C., Xing, L., Vokkarane, V. M. and Sun, Y. (2012), Reliability analysis of wireless sensor networks using different network topology characteristics, *International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering*, pp. 12–16.

Wang, C., Xing, L., Zonouz, A. E., Vokkarane, V. M. and Sun, Y. L. (2017), Communication Reliability Analysis of Wireless Sensor Networks Using Phased-Mission Model, *Quality and Reliability Engineering International*, vol. 33, no. 4, pp. 823–837.

Xiao, Y., Chen, S., Li, X. and Li, Y. (2008), An Enhanced Factoring Algorithm for Reliability Evaluation of Wireless Sensor Networks, *The 9th International Conference for Young Computer Scientists*, pp. 2175–2179.

Xiao, Y., Li, X., Li, Y. and Chen, S. (2009), Evaluate reliability of wireless sensor networks with OBDD, *IEEE International Conference on Communications*, pp. 1–5.

Xing, L. and Dai, Y. S. (2009), A new decision-diagram-based method for efficient analysis on multistate systems, *IEEE Transactions on Dependable and Secure Computing*, vol. 6, no. 3, pp. 161–174.

Xu, Y., Heidemann, J. and Estrin, D. (2001), Geography-informed energy conservation for Ad Hoc routing, *International Conference on Mobile Computing and Networking*, Rome, Italy, pp. 70–84.

Yang, Q., He, S., Li, J., Chen, J. and Sun, Y. (2015), Energy-Efficient Probabilistic Area Coverage in Wireless Sensor Networks, *IEEE Transactions on Vehicular Technology*, vol. 64, no. 1, pp. 367–377.

Yang, T., Mu, D., Hu, W. and Zhang, H. (2014), Energy-efficient border intrusion detection using wireless sensors network, *EURASIP Journal on Wireless Communications and Networking*, pp. 1–12.

Younis, O. and Fahmy, S. (2004), HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad Hoc Sensor Networks, *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366–379.

Zhang, Y., He, S. and Chen, J. (2017), Near Optimal Data Gathering in Rechargeable Sensor Networks with a Mobile Sink, *IEEE Transactions on Mobile Computing*, vol. 16, no. 6, pp. 1718–1729.

Zhao, M., Yang, Y. and Wang, C. (2015), Mobile data gathering with load balanced clustering and dual data uploading in wireless sensor networks, *IEEE Transactions on Mobile Computing*, vol. 14, no. 4, pp. 770–785.

Zhu, X., Lu, Y., Han, J. and Shi, L. (2016), Transmission Reliability Evaluation for Wireless Sensor Networks, *International Journal of Distributed Sensor Networks*, vol. 12, no. 2.

Zonouz, A. E., Xing, L., Vokkarane, V. M. and Sun, Y. (2013), Application communication reliability of wireless sensor networks supporting K-coverage, *IEEE International Conference on Distributed Computing in Sensor Systems*, DCoSS 2013, pp. 430–435.

Zonouz, A. E., Xing, L., Vokkarane, V. M. and Sun, Y. L. (2014), Reliability-Oriented Single-Path Routing Protocols in Wireless Sensor Networks, *IEEE Sensors Journal*, vol. 14, no. 11, pp. 4059–4068.

Zou, Y. and Chakrabarty, K. (2004), Sensor Deployment and Target Localization in Distributed Sensor Networks, *ACM Transactions on Embedded Computing Systems*, vol. 3, no. 1, pp. 61–91.

Zuo, M. J., Tian, Z. and Huang, H. Z. (2007), An efficient method for reliability evaluation of multistate networks given all minimal path vectors, *IIE Transactions*, vol. 39, no. 8, pp. 811–817.