Node Replica Detection in Wireless Sensor Networks

Alekha Kumar Mishra

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela, Odisha 769008, India

Node Replica Detection in Wireless Sensor Networks

Thesis submitted in partial fulfillment
of the requirements for the degree
of

Doctor of Philosophy

by

Alekha Kumar Mishra
(509CS103)

under the guidance of

Dr. Ashok Kumar Turuk

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela, Odisha 769008, India

Department of Computer Science and Engineering
National Institute of Technology Rourkela,
Rourkela - 769008, India.

Certificate

This is to certify that the thesis entitled **Node Replica Detection in Wireless Sensor Networks**, submitted by **Alekha Kumar Mishra**, a Research Scholar, in the *Department of Computer Science and Engineering*, *National Institute of Technology Rourkela*, *Rourkela*, *India*, for the award of the degree of **Doctor of Philosophy**, is a record of an original research work carried out by him under my supervision and guidance. The thesis fulfills all requirements as per the regulations of this Institute and in my opinion has reached the standard needed for submission. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Ashok Kumar Turuk

Acknowledgments

First, and foremost I would like to thank my supervisor Dr. Ashok Kumar Turuk for giving me the guidance, encouragement, counsel throughout my research and painstakingly reading my reports. Without his invaluable advice and assistance it would not have been possible for me to complete this thesis.

I would like to express my gratitude to Prof. Bibhudatta Sahoo and Prof. Ratnakar Dash, who was constant source of encouragement to me and helping me with his insightful comments.

I take this opportunity to express my sincere thanks to Prof. G. K. Panda, HOD Dept. of Mathematics for providing mathematical support at the initial stage of my research.

I also thank Prof. B. Majhi, Prof. P. M. Khilar, Prof. S. K. Patra, and Prof. K. C. Pati for serving on my Doctoral Scrutiny Committee.

I wish to thank all Faculty and staff of the CSE Department for their sympathetic cooperation.

I thank my fellow researchers Ashis, Arunanshu, Rahul, and Sambit who made my stay at NIT Rourkela a memorable one.

Finally, I would like to thank all of them whose names are not mentioned here but have helped me in any way to accomplish the work.

Alekha Kumar Mishra

Abstract

In various applications of wireless sensor network, nodes are mostly deployed unattended and unsupervised in hostile environment. They are exposed to various kinds of security threat, and node replication attack is one among them. In this attack, an adversary captures a legitimate node from the network. Then, she creates a number of clones of the original node, and deploys them back into the network. The adversary can gain control of various network activities and launch other insider attacks using these replicas. Most of the replica detection schemes reported in the literature are centralized and location dependent. Centralized schemes are vulnerable to a single point of failure. Forwarding location information incurs additional overhead in location dependent schemes.

Most replica detection schemes require exchange of membership information among nodes. To reduce communication overhead we propose two techniques called *transpose bit-pair coding* (TBC), and *sub-mat coding* (SMC) for efficient exchange of group membership information among the nodes in wireless sensor network. These schemes are lossless and do not generate false positive. Next, we propose two replica detection schemes for static wireless sensor networks called *zone-based node replica detection* (ZBNRD), and *node coloring based replica detection* (NCBRD). In ZBNRD, nodes are divided into a number of zones. Each zone has a zone-leader, who is responsible for detecting replica. ZBNRD is compared with a few existing schemes such as LSM, P-MPC, SET and RED. It is observed that ZBNRD has higher detection probability and lower communication cost. In NCBRD, each node is assigned with a color (value), which is unique within its neighborhood. A color conflict within the neighborhood of a node is detected as a replica. The performance of NCBRD is compared with LSM, SET, and RED. It is found that NCBRD has higher detection probability than the above schemes and lower communication overhead than LSM and RED. The techniques for replica detection in static wireless sensor networks cannot be applied to mobile wireless sensor networks (MWSN) because of nodes mobility. We propose a technique called *energy based replica detection* (EBRD) for MWSN. In EBRD, the residual energy of a node is used to detect replicas. Each node in the network monitors and is monitored by a set of nodes. Conflict in the timestamp-residual energy pair of a node is detected as replica. EBRD is compared with two existing schemes EDD, and MTLSD. It is found that EBRD has excellent detection probability in comparison to EDD and MTLSD, and the communication overhead of EBRD is higher than EDD and lower than MTLSD. Simulations were performed using Castalia simulator.

Dissemination of Work

1. Alekha Kumar Mishra and Ashok Kumar Turuk, A Comparative Analysis of Node Replica Detection Schemes in Wireless Sensor Networks, Communicated to *Journal of Networks and Computer Applications*. [Chapter # 2]

2. Alekha Kumar Mishra and Ashok Kumar Turuk, Efficient Mechanism To Exchange Group Membership Identities Among Nodes In Wireless Sensor Networks, *Wireless Sensor Systems Journal, IET*, Volume 3, Issue 4, December 2013 Page(s) 289 - 297. [Chapter # 3]

3. Alekha Kumar Mishra and Ashok Kumar Turuk, A Zone-Based Replica Detection Scheme for Wireless Sensor Networks, *Wireless Personal Communications Journal, Springer* , Volume 69, Issue 2, March 2013, Page(s) 601 - 621. [Chapter #4]

4. Alekha Kumar Mishra and Ashok Kumar Turuk, Node Coloring Based Replica Detection Technique in Wireless Sensor Networks, *Wireless Networks, Springer*, June 2014(Online), DOI: 10.1007/s11276-014-0751-9. [Chapter # 5]

5. Alekha Kumar Mishra and Ashok Kumar Turuk, Energy Based Replica Detection Scheme for Mobile Wireless Sensor Network, *Security and Communication Networks Journal, Wiley*, April 2014 (Online), DOI: 10.1002/sec.1012. [Chapter # 6]

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Wireless sensor network (WSN) [1, 2] is an emerging technology that provides a new paradigm for computation and communication. It consists of large number of autonomous sensing devices that are responsible for monitoring the physical and/or environmental conditions such as temperature, humidity, sound, pressure, motion, vibration, pollution etc. of a target area. Data collected by these sensing devices are transmitted to the destination called sink or base station. Usually, the base station have higher computation and communication capability. Sensing devices co-ordinate among themselves to carry out a given task. An example of a WSN is shown in Figure 1.1.

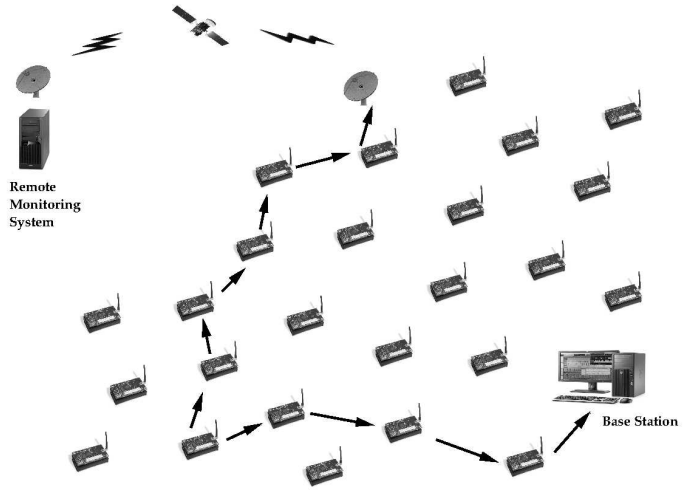


Figure 1.1: An example of wireless sensor network.

1.1 Characteristics of WSN

Sensor networks are usually designed and deployed for a specific application. They are scalable with a minimal effort. Network topology changes frequently in WSN due to energy depletion, channel fading, node failure and damage. Sensor nodes are self configurable and they are densely deployed in the target area. Battery is the only source of energy for most of the sensing devices. Most of the applications of WSN are data centric and the data-flows within the network obey many-to-one traffic pattern. Due to higher node density, data redundancy may exist in the network.

1.2 WSN Architecture

1.2.1 Node Architecture

The architecture of a sensor node is shown in Figure 1.2. A sensor node consists of four major components: *i)* Sensing unit, *ii)* Processing unit, *iii)* Transceiver unit, and *iv)* Power unit [3]. A sensor may also have a global positioning system (GPS) [4, 5] and a mobilizer for localization and mobility respectively.

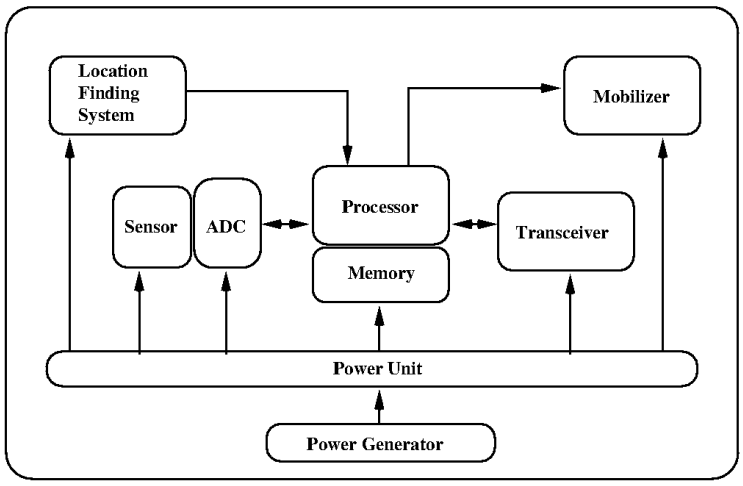


Figure 1.2: Sensor node architecture.

Sensor generates analog signal of sensed data, which is converted to digital signal by the analog-to-digital converter (ADC), and is transmitted to the processing unit. The processing unit has an embedded micro-controller that performs the computing job. Transceiver unit is responsible for data transmission. Power unit manages the power supply to all other components.

1.2.2 Network Architecture

WSN architecture can be broadly divided into two types [6]: *Flat* and *Hierarchical*. In flat architecture, sensor nodes are assigned with the same task, and all nodes are equally responsible to perform various network activities. In hierarchical architecture, a group of sensors form a cluster. Each cluster have a cluster-head, who is responsible for sending data from the cluster members to the sink node. Usually, a node with lower energy performs the sensing task, whereas a high-powered node becomes a cluster-head.

1.2.3 WSN Protocol Stack

Figure 1.3 shows the protocol stack of a WSN. It consists of five layers: *i)* Physical, *ii)* Data link, *iii)* Network, *iv)* Transport, and *v)* Application. The functionality of each layer are similar to the functionality of wireless ad hoc network protocol layers except the application layer, which includes various protocols to support higher level operations such as in-network applications, application processing, data aggregation, and external query processing. Protocol stack is also divided into various

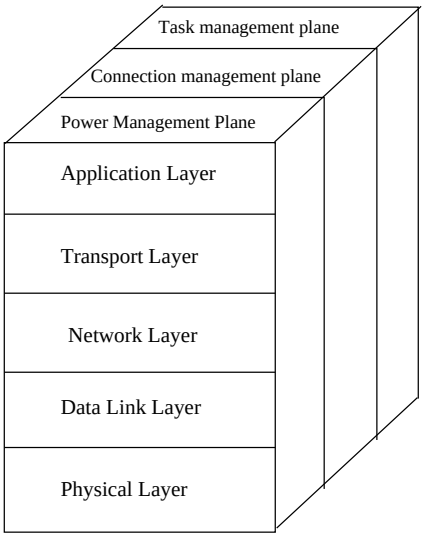


Figure 1.3: WSN protocol stack.

management planes. Power management plane is responsible for managing the power level to be used for various operations in a sensor. Connection management plane is responsible for managing the configuration of a node to establish and maintain connectivity. The task management plane distributes a task among the sensor nodes to improve energy efficiency.

Various communication standards has been developed for WSN. The commonly used standards are *IEEE 802.15.4* developed by IEEE 802.15 task group 4 [7, 8], *The ZigBee standard* [9] which is built on the top of the IEEE 802.15.4, and *IEEE 1451* which is the family of smart transducer interfaces [10].

1.3 Applications

WSN supports a wide range of applications [1, 11]. A few of these applications are described below: *i) Habitat monitoring*: This includes monitoring of birds and animals, observing the environmental aspects such as marine, soil, atmosphere, forest fire, and many more; *ii) Health-care application*: WSN supports patient monitoring and diagnosis such as monitoring of heart-beat, blood pressure, medicine supply and doses to a patient, etc; *iii) Household application*: Sensors can be used in appliances such as microwave, refrigerator, television etc. to enable remote access; *iv) Traffic monitoring*: WSN can be used to facilitate smooth traffic in a city; *v) Agricultural application*: This involves deployment of sensors in an agricultural field to monitor soil moisture level, temperature, influence of pest on plant growth; *vi) Battlefield*: WSN is used for monitoring enemy activities such as movement of vehicles and troops; They can be also deployed in buildings, bridges, ships, and aeroplanes to monitor the effect of load distribution.

1.4 Security Issues in WSN

Wireless sensor networks inherit the security threats of wired network such as Denial of Service (DoS) attack, packet dropping, false routing etc. Most of the security threats of WSNs such as jamming, collision, spoofing, hello flooding etc. [12] are similar to that of Ad hoc networks. However, security mechanisms devised for Ad hoc networks are not applicable to WSN due to the application-dependent architecture of sensor networks.

In applications, such as military, wild-life monitoring, and traffic monitoring, it is possible to secure the base station. However, the major challenge is to secure and protect the tiny sensors that are deployed either in the enemy territory, open space, or in hazardous areas [11]. As a result, they are more prone to physical attacks and other possible security threats. A secured WSN must satisfy the following security requirements [13, 14]:

i) Information confidentiality and privacy,

ii) Data integrity,

iii) Entity authentication,

iv) Key distribution and management,

v) Secure routing,

vi) Secure data aggregation, and

vii) Intrusion detection.

An adversary can launch various types of attack on WSN depending on its ability and objective [12, 15]. These attacks can be classified into two categories [16]: *i)* layer-dependent, and *ii)* layer-independent. Layer-dependent attacks are specific to communication protocol layers. They mostly target a node's functionality such as routing, node localization, time synchronization, and data aggregation. The list of possible layer-dependent attacks on WSN, and their countermeasures are summarized in Table 1.1. Layer-independent attacks are not restricted to any communication protocol layers. These attacks can be launched independent of the communication protocol stack. Two common attacks in this category are the Sybil attack and node capture/replication attack. The attacks under later category are more severe than their counter parts.

1.5 Node Replication Attack

The low-cost sensor nodes lack protective shield that would allow unauthorized access to their functional units such as memory, processing and communication. It is infeasible to manufacture tamper-resistant sensors due to cost-constraint. Deploying such vulnerable sensors in unsupervised environment encourages an adversary to launch physical attack with a little effort. In node replication attack, an adversary physically captures a node from its deployed location. She then accesses the memory, processing, and communication unit of the captured node, and steals relevant information such as identity, secret keys, intrusion detection characteristics etc. Using the stolen information, she then generates a number of replicas of the captured node by incorporating useful stolen information, and deploys them back into the network. These replicas operate under the control of the adversary. They behave like a legitimate node, and participate in the communication using the stolen keying materials. The aim of an adversary in node replication attack is to control

Table 1.1: Security attacks at various layers of WSN.

Layer	**Attack**	**Countermeasures**
Physical	Jamming	Spread-spectrum, Low duty cycle
	Tampering	Self-destruction, Tamper-proof
Data link/MAC	Collision	Error correction code
	Exhaustion	Limited retransmission
	Interrogation	Accepting limited connections
	Unfairness	Small frames
Network	Misdirection	Sleep mode
	Spoofing	Authentication, Encryption of important fields
	Sinkhole	Authentication, Monitoring, and Redundancy
	Wormhole	Location-based routing, Authentication, Checking bi-directional links
	Selective Forwarding	Redundancy, Probing
	Hello flood attack	Authentication, Checking bi-directional links
Transport	Flooding	Limiting number of connections
	De-synchronization	Authentication

the network activities using replicas. An adversary may either extract useful data from the network or hinder the network operations. With the help of replicas, an adversary can launch insider attacks such as wormhole, selective forwarding, hello flooding, false data injection etc. An adversary can perform all of the above mentioned activities only by compromising a single node in the network. Therefore, node replication is considered as one of the most serious threats in WSN [16, 17].

One of the important characteristics that make a node replication attack more challenging is to differentiate between a legitimate node and its clone[1]. Since, replicas execute the same network protocols and have the same keying materials as that of a original node, they pass all authentication and verification process. Most of the solutions reported in the literature recommends for the detection of existence of replicas in the network [18]. These schemes mostly use the parameters such as position, a unique set of neighboring nodes etc. that can differentiate a replica from its original node.

1.6 Motivation of the Work

A sensor node deployed unattended in an insecure environment such as in enemy territory or in a dense forest, is prone to serious threats like node replication attack. In a node replication attack, original node and its replica behave exactly the same way. They use the same keying materials and follow the same protocol. Therefore, it is difficult to distinguish between a clone and its original node. Replica detection mechanisms rely on the abnormal behavior that would distinguish a replica from its original node. It is also a challenging task to identify the abnormal characteristics that will certainly detect a replica in the network.

Replica detection mechanisms must monitor the nodes on a regular basis to identify the replicas. This requires additional power and radio resources. Since, the nodes in sensor networks are resource constrained, the detection mechanisms should minimize resource consumption.

Longer the duration, replicas stay in the network, more damage they can do to the network. Therefore, a faster replica detection mechanism is desirable.

Deciding the number of nodes responsible to detect replica affects the performance of detection mechanisms. A single node such as BS leads to a single point of failure. Whereas, if all nodes are equally responsible for detecting a replica, then the communication overhead of the network is increased. Therefore, replica detec-

[1] In this thesis, the term replica and clone have been used interchangeably.

tion mechanisms must judiciously select the required number of nodes to detect a replica. Moreover, all detection mechanism should have low false positive.

This motivates to design an efficient node replica detection mechanism having higher detection probability, lower detection time, lower false positive, and lower communication and storage overhead.

1.7 Objective of the Work

Replica detection is a widely accepted approach to handle node replication attack in sensor networks. An efficient node replica detection mechanism should not only detect a replica but also optimize the overall network performance. Further, the replica detection mechanism should emphasize not only on higher detection probability but also on lower communication and storage overhead. In this thesis, we have studied the behavior of node replication attack and identified the possible ways an original node can be distinguished from its replica.

Accordingly, we identified the objective of the thesis as following: *i)* Propose a mechanism for efficient communication among nodes, *ii)* Propose a distributed node replica detection mechanism for static as well as mobile WSN, and *iii)* Compare the performance of the proposed replica detection mechanism with existing ones.

The parameters considered for comparison are: *i)* Detection probability, *ii)* Communication overhead, *iii)* Storage overhead, *iv)* Energy consumption, and *v)* Detection time. We have simulated the proposed mechanisms using Castalia simulator under Omnet++ environment.

1.8 Organization of the Thesis

The thesis is organized into the following chapters:

In **Chapter 2**, we have classified the existing clone detection schemes and highlighted the contribution of each scheme. We have also made an analysis of the existing schemes based on their communication cost, message overhead, and storage requirement. A few of the schemes are simulated using Castalia simulator and their performance is compared.

Chapter 3 proposes two schemes called Transpose Bit-Pair Coding (TBC), and Sub-Mat Coding (SMC) for exchanging group membership information in WSN.

The proposed schemes do not generate false positive, and have a lower communication and storage overhead. We have compared TBC and SMC with Bloom filter. Parameters considered for the comparison are: *i)* communication overhead in terms of number of bits required to exchange the group membership information, and *ii)* false positive.

In **Chapter 4**, we propose a location independent zone-based node replica detection technique (ZBNRD). In ZBNRD, the network is logically divided into a number of zones. Each zone has a zone-leader, and they share their zone membership information among themselves. It is the responsibility of the zone-leader to detect the clone. The proposed technique is a deterministic one, and found to have a higher clone detection probability and a lower communication cost in comparison to existing scheme such as LSM, RED, and P-MPC.

In **Chapter 5**, we propose a technique called node coloring based replica detection (NCBRD). In the proposed scheme, each node is assigned with a color (value), which is unique within its neighborhood. A color conflict within the neighborhood of a node is detected as a replica. We made a comparison of the proposed scheme with LSM, SET, and RED. Parameters considered for comparison are detection probability, communication and storage overhead. We observed that the proposed scheme has a higher detection probability, and lower communication and storage overhead.

Chapter 6 proposes a distributed replica detection mechanism called energy based replica detection (EBRD) for mobile wireless sensor networks. In the proposed scheme, the residual energy of a node is used to detect replica. Each node in the network acts as a monitoring node to a set of nodes in the network. A conflict in the time-residual energy pair of a node is detected as clone by its monitoring node. We have simulated and compared the proposed scheme with SEDD and MTLSD. It is observed that the proposed scheme has higher detection probability, and lower communication and storage overhead.

In **Chapter 7** we have summarized the work done, highlighted the contribution and suggest the directions for possible future work.

Chapter 2

Literature Survey

In this chapter, we made a survey of node replica detection schemes reported in the literature. We have also classified the existing schemes and analyzed their communication, storage, and message overhead. A few of the replica detection schemes are simulated and their performance is compared. The metrics considered for comparison are: detection probability, detection time, average number of packets sent/received, and energy consumption.

2.1 Classification

The existing mechanisms for replica detection can be broadly classified into the following categories:

a) Based on the detection mechanism, we classify into the following types:

   i) *Centralized*: In a centralized scheme, there exists a centralized trusted entity such as BS, which is responsible for replica detection.

   ii) *Partially distributed*: In this scheme, replica detection mechanism is distributed in nature. However, the involvement of BS is necessary for certain activities such as broadcasting a random number, revocation of message and global keys, etc.

   iii) *Fully Distributed*: This scheme does not require the involvement of any centralized entity. Nodes cooperate among themselves to detect replica.

b) Based on the need of geographical information, we classify into the following types:

i) Location dependent: This scheme uses node's location information to detect replica. In this scheme, the location-claim of a node is forwarded to a set of randomly generated witness nodes/location, or cells, by its neighboring nodes. Two or more different location-claims for the same node result in a location conflict, and a replica is detected. Location dependent replica detection is depicted in Figure 2.1. The nodes denoted as **W** in the figure are witness nodes. On receiving the location claim for node A from two different locations, the witness node **W** detects it as a replica. A node can compute its location information either using GPS [4] or using the mechanisms mentioned in [5, 19].

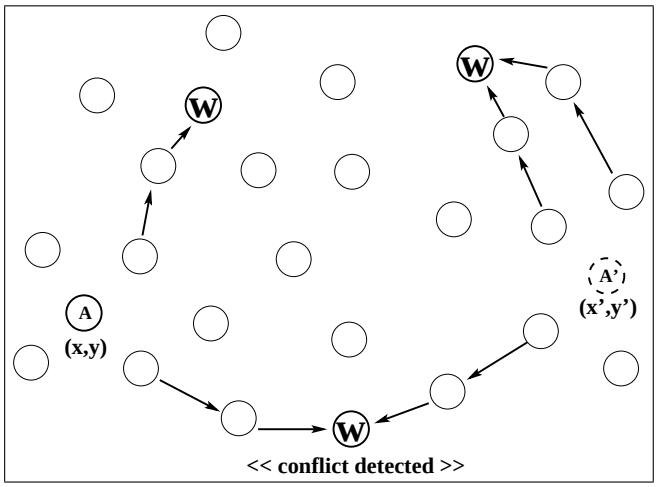


Figure 2.1: Location based replica detection.

ii) Location independent: A location independent scheme does not use location information for replica detection. This scheme relies on various parameters such as group membership of a node, synchronization timer/counter value etc. to detect a replica. Figure 2.2 depicts the group membership based replica detection. In this figure, node is detected as replica, since witness node **W** receives different group membership list for the same node A.

c) Based on the claim forwarding strategy, we classify the node replica detection schemes into the following types:

i) Deterministic forwarding: In this scheme, a claim received by a node is always forwarded to its destination.

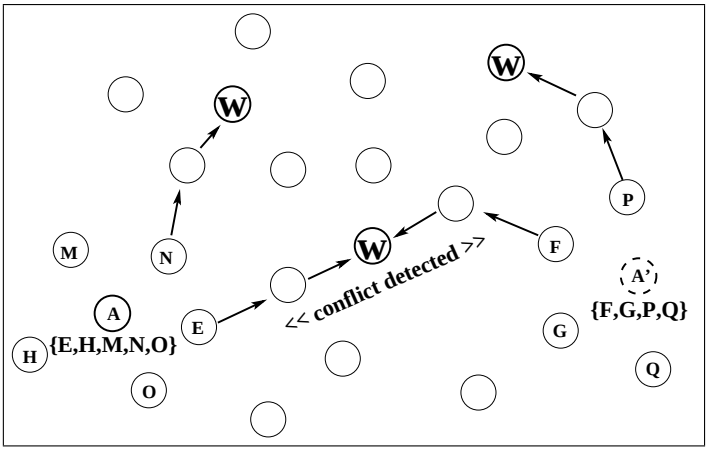


Figure 2.2: Group membership based replica detection.

ii) Probabilistic forwarding: In this scheme, a claim received by a node is forwarded with a certain probability depending on the claim forwarding strategy.

d) Based on message routing, we classify the node replica detection schemes into the following types:

i) Link-based routing: This scheme uses Link-based routing protocols [20] for routing messages in the network.

ii) Geographical routing: This scheme uses Geographical Routing protocols [21] for routing messages in the network.

Figure 2.3 shows the proposed classification of replica detection schemes.

2.2 Replica Detection Schemes

In this section, we have briefly discussed the existing schemes for replica detection, with their relative merits and demerits. Notations used in this chapter are enlisted in the Table 2.1.

2.2.1 Centralized Schemes

SET

Choi *et al.* [25] proposed a scheme called SET, in which the network is divided into a number of non-overlapping sub-regions. Nodes in each sub-region are one-hop

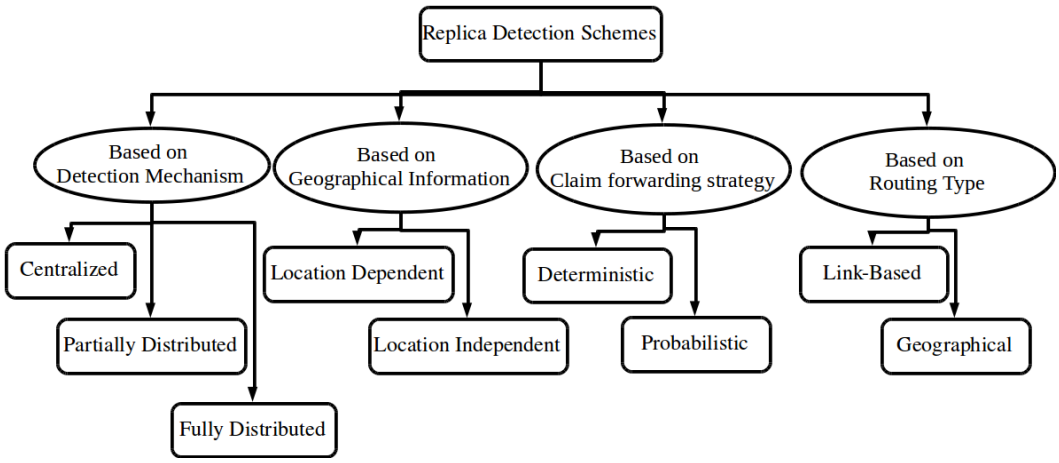


Figure 2.3: Classification of replica detection schemes in WSN.

Table 2.1: Notations

Symbol	**Meaning**
N	Network Size
d	Average Degree of a node
p	Probability of forwarding a message
w	Number of nodes (witnesses) that store a location-claim
s	Number of sensor node in a cell [22]
t	Number of clusters in [23]
g	Number of witnesses generated by a neighboring node
vp	Verification point used by [24]

neighbor of each other. A leader is elected within each sub-region. Then, a tree is constructed with the BS as root, and the leaders at different levels in the tree. Each sub-region leader sends their members identity (ID) to its parent node. A parent node performs intersection operation between all child sub-regions before forwarding them to its parent. A non-empty intersection at any level of the tree leads to a conflict and is reported to the BS for further action. SET has lower communication overhead in replica detection. However, SET is a centralized detection mechanism, which is vulnerable to single-point of failure. Size of the member-ID message grows rapidly at higher levels of the tree close to BS.

An Area-Based Approach

An area based scheme is proposed by Naruephiphat *et al.* [26]. In their scheme, a node having maximum number of neighbors is selected as the *central node*. Then, the network is divided into sub-areas. Each sub-area has equal angle around the *central node*. A witness node is selected in each sub-area using the method similar to the one used to select the *central node*. A node sends its location-claim to the witness node of its area. If a witness node detects a location conflict, then it broadcasts a conflict notification to all nodes in the network. On reception of claims without conflict, the witness node sends these claims to the *central node* for further detection of replica at network level. In this scheme, sub-areas can be scaled easily by dividing their angles. However, this scheme is subject to single-point of failure. Moreover, the communication overhead is quite high in inter sub-area replica detection.

2.2.2 Partially Distributed Schemes

Real-time Detection Scheme

A fingerprint based scheme is proposed by Xing *et al.* [27], where the fingerprint of a node is computed using the neighborhood information. In this scheme, each node is preloaded with a codeword generated from a superimposed s-disjunct code [28] prior to their deployment. A node computes its unique fingerprint as well as the fingerprints of its neighboring nodes using their codeword. The fingerprint of a node is included in every message that is sent to the BS. Neighboring nodes verify the genuineness of a node by comparing the fingerprint in the message with their own record. A conflict in the fingerprint of any node is reported to the BS, and is detected as a replica. In this scheme, a replica is detected either by the neighboring nodes or the BS. The generation of codeword from superimposed s-disjunct code is computationally expensive.

Neighbor-Based Detection Scheme (NBDS)

A neighbor-based detection scheme was proposed by Ko *et al.* [29]. According to their scheme, a node moving to a new location must broadcast a rejoining claim to its new neighbors. The rejoining claim consists of the list of its old neighbors. A new neighbor on receiving the rejoining claim forwards it to a randomly selected node from the list of old neighbors. An old neighbor on receiving the rejoining claim checks for the node ID in its neighbor table. If the node ID is present, then the old neighbor reports to the BS, otherwise it concludes that the node has moved to

another location. The old neighbor then sends a challenge to the node to verify its existence in its neighborhood. If the node is present in the old position, then it is detected as a replica. On the other hand, if the new neighbors do not receive any revocation message within a stipulated time period, then, they accept the newly joined node as their neighbor. This scheme allows occasional mobility of nodes in the network. However, an intelligent adversary may bypass the replica detection process. The verification process of the scheme increases the communication overhead.

Localized Multicast

Zhu *et al.* [22] proposed a method called Localized Multicast where the witness nodes are randomly selected and localized to a restricted geographical region. There are two variations of Localized Multicast: *i) Single Deterministic Cell* (SDC), and *ii) Parallel Multiple Probabilistic Cells* (P-MPC). In their scheme, Zhu *et al.* have considered a geographical grid network and divided it into a number of rectangular cells. In SDC, neighboring nodes forward the location claim of a node to its target cell with a probability p. A geographic hash function [30] is used to map the node's ID to a target cell. The authenticity of location claim is verified by all nodes within this target cell. On successful verification, each node in the cell caches the location claim with a certain probability. The claimer of two conflicting location claims is detected as a replica by a node within the target cell. P-MPC is an extension of SDC. In P-MPC, the location claim is mapped to a number of destination cells instead of one. This improves the detection probability in P-MPC. This scheme is simple and its communication overhead is low. However, the strength of SDC and P-MPC in replica detection depends on the probability of forwarding the location-claim to a cell and storing the claim by the cell members.

A Note-Based Randomized and Distributive Protocol

Meng *et al.* [31] proposed a mechanism that uses a note containing the subset of neighbors as a claim. The claim-forwarding process starts with the selection of a reporter node among the neighbors. Once a reporter node is selected, the claimer requests for a signature note from the reporter node. Upon receiving the signature note, claimer first verifies and then forwards the claim. The reporter node on receiving a claim, generates a pre-defined number of witness nodes and forwards the claim to those nodes. A conflict in the claim of a node at any of its witness

node is detected as a replica. This scheme has an additional overhead of selecting reporter nodes. This is because, the trust worthiness of a claim-forwarding node can easily be verified by listening to its broadcast message. Moreover, the replica detection depends on the uniqueness of the subset member-list which may not be ensured in all cases.

Randomized, Efficient and Distributed Scheme (RED)

Conti *et al.* [32] proposed a scheme called randomized, efficient and distributed protocol (RED) in order to improve the performance of location-claim based schemes. In their scheme, a random value is broadcasted to all nodes. Then the nodes digitally sign their location-claim, and broadcast to their neighboring nodes. Each neighbor generates a number of pseudo-random witness locations using the random value, their own ID, and the number of witnesses. Then they send the location-claim to the witness locations with a probability p. Nodes located at a distance less than a pre-defined value from these witness locations store the location-claim. A conflict in the location claim is detected as a replica. RED is secured as witness locations are generated randomly. However, it has limitations in sparse as well as dense sensor networks. In a sparsely distributed sensor network, a case may arise where there will be no or limited witness nodes within the predefined range of a randomly generated location. In such a scenario, the claim may or may not reach a witness node. This will limit detection probability. In a highly dense network, the number of witness nodes can be significantly higher. In this scenario, the average storage overhead of a network increases.

Hierarchical Node Replication Detection Scheme

Znaidi *et al.* [23] proposed a mechanism for three-tier hierarchical network structure. Their mechanism is based on the use of Bloom filter [33, 34]. The replica detection process is divided into the following three phases:

i) Pre-distribution Phase: In this phase, nodes are equipped with cryptographic keying materials and other parameters required for Bloom filter operation.

ii) Election Phase: This phase is performed periodically to select cluster-head using Local Negotiated Clustering Algorithm (LNCA) protocol [35].

iii) Detection Phase: Elected cluster-heads exchange their member IDs among themselves using Bloom filter. A node ID that is found to be a member of

more than *one* cluster is detected as a replica.

The memory overhead of this scheme is significantly lower. However, it has an additional communication overhead in cluster formation. For a small number of hash functions the false detection probability is higher and requires relatively larger number of bits in Bloom filter.

2.2.3 Fully Distributed Schemes

Distributed Detection Scheme

Parno *et al.* [36] proposed two mechanisms which they called: *i) Randomized Multicast* (RM), and *ii) Line-Selected Multicast* (LSM). In RM, witness nodes are randomly chosen. The location claim broadcast of a node is forwarded to a set of randomly selected witness nodes by its neighbors. If a clone is present in the network, then the location claim of the clone is also forwarded to a set of witness nodes. According to birthday paradox [37], if each location claim is forwarded to $\sqrt{N}$ number of witnesses, then at least one witness node will receive two different location claims, one from the original node and the other from its replica with a higher probability. This leads to a location conflict and the node is detected as a replica. LSM has lower communication cost than RM. In LSM, the location claim of a node is cached at each intermediate node before forwarding to the next-hop node on the path to the witness node. This creates a line across the cached intermediate nodes. When the line of a clone's location-claim crosses the line of its legitimate node, the intermediate node at the crossing point detects it as a replica. In RM, the broadcasting of location claim of a node to all of its witness nodes is expensive in terms of communication overhead. In LSM, nodes that are common to multiple location-claim lines suffer from higher storage overhead.

Symmetric pair-wise key establishment scheme

Bekara *et al.* [38] proposed a scheme based on symmetric pair-wise key establishment. In their scheme, each node is associated with a unique generation or group. The generation of a node can be computed using its ID and a symmetric polynomial. According to their protocol, only a newly deployed node that belongs to a newly deployed generation is able to establish a pair-wise key with their neighbors. Therefore, when a clone that belongs to an old generation tries to establish a pair-wise key with its new neighbors, it is detected as a replica. This scheme is simple and

incurs less communication overhead. The duration of a nodes' generation is a critical parameter. An inaccurate duration may detect the genuine nodes of previous generation as replicas.

Distributed Detection Scheme Resilient to Many Compromised Nodes

Sei *et al.* [39] have proposed a resilient replica detection scheme. This scheme does not require any trusted entity, and is resilient to a number of compromised nodes in the network. Each node is pre-loaded with detection process start time. A node in its turn sends a one-time seed, its ID, and location with a signature to all other nodes. If a node fails to start the detection process in its turn within a pre-defined interval of time, then the next node starts its process. To improve resiliency against node capture, nodes are divided into groups, and each node starts its detection process using a role ID assigned to it. The neighboring nodes responsible for forwarding the location claim of a node to its witness nodes are called *reporter nodes*. A reporter node forwards the location claim to a number of witness nodes that are responsible for detecting replica. This scheme incurs significantly higher communication overhead because, the detection process is based on message broadcast over the network. Moreover, the scheme does not explain the procedure to ensure that at least one neighbor would voluntarily become a reporter node.

Memory Efficient Protocols

Zhang *et al.* [40] have identified two problems associated with LSM protocol; which they called *crowded-center* problem, and *cross-over* problem. To overcome the above two problems, they proposed the following four protocols: *i)* B-MEM, *ii)* BC-MEM, *iii)* C-MEM, and *iv)* CC-MEM. In B-MEM, the location claim of a node is sent to a random location with a probability of p. A node located closer to this location, stores the claim. An intermediate node on the path, also known as *watcher node* stores the ID and location of the claimer using Bloom filter. Any location conflict with a stored ID is detected as a replica by the *watcher node*. In BC-MEM protocol, the deployment area is divided into a number of virtual cells. Each node in the cell is associated with an anchor point and an anchor node. The anchor point is determined using the node ID, whereas the anchor node of a node is the node, which is closer to its anchor point. In BC-MEM, a location claim is forwarded from an anchor point of one cell to another cell, which are intersected by the line segments until it reaches the last cell. The intermediate anchor nodes act

as a watcher to detect clone. This overcomes the *cross-over* problem, and reduces the storage overhead. C-MEM protocol overcomes the *crowded-center* problem by forwarding the location claim to a random point called *cross point*. The claim is again forwarded along the horizontal and vertical lines from the *cross point*. Nodes on these lines act as watcher, and those closer to the cross point are witness node. CC-MEM uses the concept of both cross-forwarding and cell-forwarding used in C-MEM and BC-MEM respectively. The detection probability is higher in CC-MEM. Above memory efficient schemes are able to achieve lower storage overhead using Bloom filter. However, their communication overhead increases significantly with improvement in detection probability in C-MEM and CC-MEM.

Distributed detection with group deployment knowledge

Group deployment knowledge is used in the schemes proposed by Ho *et al.* [41]. They have proposed three schemes. In their first scheme, a node is preloaded with its pre-determined group deployment point. The nodes within the same group are expected to be deployed closer to each other. Nodes closer to their group deployment point are considered to be trusted, whereas the nodes that are far away from the group deployment point are considered to be untrusted. A node ignores the message received from the untrusted nodes. Nodes belonging to different groups can communicate with each other, only if the distance between their group deployment point is less than a threshold value. In their second scheme, they have relaxed the criteria to communicate with untrusted neighbor. In this scheme, a node can communicate with an untrusted node, only if the untrusted node provides sufficient evidence in terms of location claim that it is not a replica. In their third scheme, a node forwards the location claim of an untrusted neighbor to multiple groups instead of the untrusted neighbor's home group. Since, the above schemes use deployment knowledge, their memory requirements are significantly lower. Verification of untrusted nodes may require large number of message exchanges. This will increase the communication overhead of the network. Moreover, the above schemes may not be suitable for the applications, where it is difficult to determine or compute the group deployment point well in advance.

Randomly Directed Exploration

Li *et al.* [42] proposed a claim broadcast based technique called randomly directed exploration. In this scheme, nodes generate a number of claim messages and each

message containing a Time-to-leave (TTL) value is forwarded to a randomly selected neighbor. An intermediate node computes its angle with the witness location. To forward the location claim, an intermediate node selects a node that is closer to the computed angle plus π. When two or more conflicting location claims of a node is received by a witness node, it is detected as a replica. This scheme is similar to RM, but differs from RM in the claim forwarding mechanism. In RDE, a TTL field is used to prevent indefinite routing. This scheme suffers from higher communication overhead.

Distributive, Deterministic and Resilient Scheme (DDR)

Kim *et al.* [24] proposed a distributive, deterministic, and resilient (DDR) scheme. In this scheme, nodes are associated with a verification point in the network. Verification point is generated by the BS prior to the node deployment. The verification point is the destination for the location-claim of a node. A node generates the location-claim, estimates the expected hop-count to the verification point, and then sends the location claim to the verification point through a randomly chosen neighboring node. An intermediate node on receiving the location claim decreases the hop-count by one, and caches the location claim based on some probability, before forwarding it to the verification point. A node that receives two different location-claims for the same ID detects it as a replica. DDR is similar to LSM. Both schemes cache the location-before claim forwarding to next node. Common intermediate nodes for a large number of forwarding paths are overloaded with the task of storing and forwarding location claim.

Early and Lightweight Distributed Detection Protocol

Tran *et al.* [43] proposed two light-weight protocols which they called: *i) LANCE*, and *ii) SACRED*. In LANCE protocol, a network-wide counter is maintained which is incremented by one, after a pre-defined time interval. A clone will have the counter value different from others. When a clone broadcasts its counter value, it is considered as a replayed counter value and is detected as a replica. The SACRED protocol is a secured version of LANCE. In this protocol, nodes take the help of location claim and authentication mechanism similar to LSM [36]. LANCE assumes that nodes are time-synchronized. It is difficult to ensure time-synchronization in a sensor network. Moreover, a powerful adversary can forge the original counter value.

Table 2.2: Categorization of different replica detection schemes.

Scheme	**Detection Mechanism**			**Geographical Information**		**Claim forwarding**		**Routing type**	
	Centralized	**Partially Distributed**	**Fully Distributed**	**Location Dependent**	**Location Independent**	**Deterministic**	**Probabilistic**	**Geographical**	**Link-based**
RM & LSM [36]	-	-	√	√	-	-	√	√	-
SET [25]	√	-	-	-	√	√	-	-	√
Bekara *et al.* [38]	-	√	-	-	-	√	-	-	√
Xing *et al.* [27]	-	√	-	-	√	√	-	-	√
Sei *et al.* [39]	-	-	√	√	-	√	-	-	√
NBDS [29]	-	√	-	-	√	-	√	-	√
B-MEM, BC-MEM C-MEM, CC-MEM [40]	-	-	√	√	-	√	-	√	-
Ho *et al.* [41]	-	-	√	√	-	-	√	√	-
RDE [42]	-	-	√	√	-	√	-	√	-
DDR [24]	-	-	√	√	-	√	-	-	√
LANCE [43]	-	-	√	-	√	-	-	-	-
SACRED [43]	-	-	√	√	-	-	√	√	-
SDC, P-MPC [22]	-	√	-	√	-	-	√	√	-
RAWL, TRAWL [44]	-	-	√	√	-	-	√	√	-
Meng *et al.* [31]	-	√	-	-	√	-	√	-	√
RED [32]	-	√	-	√	-	-	√	√	-
Naruephiphat *et al.* [26]	√	-	-	√	-	√	-	-	√
Znaidi *et al.* [23]	-	√	-	-	√	√	-	-	-

Random-Walk Based Scheme

Two non-deterministic and fully distributed protocols called: *i) Random-Walk Based Detection* (RAWL), and *ii) Table Assisted Random-Walk Based Detection* (TRAWL) are proposed by Zeng *et al.* [44]. In RAWL, the location-claim of a node is forwarded by its neighbor to a selected number of nodes with probability p. These nodes initiate the random-walk process in the network. A node that passes the random-walk process acts as a witness node, and stores the location-claim. A location conflict is detected as a replica by the witness nodes. TRAWL is a variation of RAWL, where a node that passes the random-walk, stores the location-claim with a probability equal to $\frac{c}{\sqrt{N}\log N}$, where c is a constant. The location-claim is stored as a digest in the trace table of witness nodes. Since the location-claim is sent to a set of witness nodes, the communication and storage overhead is expensive.

Table 2.2 shows the categorization of node replica detection schemes. It is observed from the table that most of the fully distributed detection schemes are location-dependent. They use node's location-claim for replica detection. Therefore, geographical routing mechanisms are more suitable to minimize routing path-length for these schemes. Location-independent schemes mostly use link-based routing protocols for communication.

The content of claim message and the authentication technique used in each of the above schemes is shown in Table 2.3. It is observed from the table that the size of the claim message in Znaidi *et al.* [23] scheme is significantly smaller than rest of the schemes. This is because the Znaidi et al.'s scheme uses encrypted Bloom filter. The message overhead is higher for those schemes that contains list of node IDs in the message such as in SET [25], NBDS [29], RDE [42], and Meng *et al.* [31]. In subsequent sections, we have compared and analyzed a few detection schemes that are widely referred by the researchers of this field.

2.3 Analysis

In this section, we made a quantitative comparison of different replica detection schemes. The metrics used for comparison are communication and storage overhead associated in detecting a replica. Communication and storage overhead of different schemes are analyzed in sub-section 2.3.1. Number of nodes required for detecting a replica also affects replica detection ability of a scheme. In sub-section 2.3.2, we have analyzed the number of nodes required for replica detection.

Table 2.3: Authentication technique used and the content of claim message in different scheme.

Scheme	**Content of the claim message**	**Authentication Technique**
RM, LSM [36]	(ID, Location, Signature)	Merkle-Winternitz signature [45]
SET [25]	(ID, Sets of Node IDs, MAC)	HMAC [46]
Bekara *et al.* [38]	(Generation ID, MAC)	Symmetric Bivariate Polynomial [47]
Xing *et al.* [27]	(ID, Node's Fingerprint, data)	-
Sei *et al.* [39]	(ID, Location, Signature)	Identity-Based Signature Scheme (IBSS) [48, 49]
NBDS [29]	(ID, Neighbor ID List, Signature)	EIBSSBP [50]
B-MEM, BC-MEM C-MEM, CC-MEM [40]	(ID, Location, Signature)	IBSS
Ho *et al.* [41]	(ID, Location, MAC)	TinyECC [51]
RDE [42]	(TTL, ID, Location, Neighbor ID List, Signature)	IBSS
DDR [24]	(ID, Location, hop-count, verification point, MAC)	Symmetric Key [52]
SACRED [43]	(ID, Location, Signature)	Symmetric Key
SDC, P-MPC [22]	(ID, Location, Signature)	EIBSSBP [50]
RAWL, TRAWL [44]	(ID, Location, Signature)	TinyECC
Meng *et al.* [31]	(ID, Neighbor ID sub-list, time, Signature)	EIBSSBP [50]
RED [32]	(ID, Location, Signature)	IBSS
Naruephiphat *et al.* [26]	(ID, Location, Signature)	-
Znaidi *et al.* [23]	(Encrypted Bloom Filter, Signature)	Elliptive-Curve Cryptography (ECC) [53]

2.3.1 Communication and Storage Overhead

Communication cost is the number of messages exchanged in the network for replica detection. Storage overhead is the additional information stored per node in replica detection. The analysis of communication and storage overhead of different schemes is given below.

RM and LSM [36]

In RM [36], each node broadcasts all received location-claims. Therefore, communication overhead is $O(N^2)$. However, LSM forwards the location-claim in a line to the witness nodes. In a network of size N, the average path length is given by $O(\sqrt{N})$ [36]. Therefore, communication overhead of LSM is $O(N.\sqrt{N})$.

Parno *et al.* [36] using birthday paradox, have shown that replica may be detected with a higher probability if at least $\sqrt{N}$ number of nodes will store the location-claim of a node. Therefore, storage overhead of a node in both RM and LSM scheme is $O(\sqrt{N})$.

SET [25]

In SET [25], each node participates in the subset leader election process by sending a broadcast message. In addition to leader election, each subset leader has an additional task of forwarding membership list to the BS. The average communication overhead of each node is $O(1)$ and therefore, communication cost of SET is $O(N)$. In SET, there is no storage overhead because none of the nodes including the subset leaders need to store the membership information.

Symmetric pair-wise key establishment scheme [38]

In this scheme, each node establishes pair-wise key with its neighbors on deployment. Therefore, overall communication overhead is $O(N)$. There is no storage overhead associated with this scheme because the replica detection depends only on the pair-wise key establishment made by a node with its neighbors.

Real-time Detection Scheme [27]

In this scheme, each node needs to share its codeword with the neighboring node for the computation of fingerprint. This sharing requires communication overhead

of $O(N)$ in the network. Each node needs to store the codeword of the neighboring nodes. Therefore, storage overhead of this scheme $O(d)$.

Distributed Detection Scheme Resilient to Many Compromised Nodes [39]

In this scheme, the reporter node forwards the claim to a number of witness nodes similar to LSM scheme. Therefore, communication overhead of this scheme is $O(N.\sqrt{N})$. Each node becomes witness of g number of nodes and store their claim. Therefore, storage overhead is $O(g)$.

NBDS [29]

In NBDS, the neighboring nodes of a newly joined node forwards the existence verification message to old neighbor's of the node with a probability p. Therefore, communication overhead of this scheme is $O(d.p.\sqrt{N})$. On receiving the rejoining claim only the old neighbors store in their cache. Therefore, storage overhead of the scheme is $O(d.p)$.

Memory Efficient Protocols [40]

All variations of this scheme forward the location-claim of a node to a random witness location. Therefore, communication overhead is same as the overhead of LSM and Sei *et al.*, *i.e,* $O(N.\sqrt{N})$. The *watcher* and *anchor* nodes store the location-claim using Bloom filter. Therefore, no storage overhead is associated with this scheme.

Distributed detection with group deployment knowledge [41]

In *Ho et al.* [41], the location-claim of a node is forwarded to its detector with a probability p. Therefore, communication overhead of the scheme is $O(N.d.p.\sqrt{N})$. Storage overhead of this scheme is $O(w)$, where w is the number of witness nodes holding the location-claim of a node.

RDE [42]

In this scheme, selected neighbors forward the location-claim in a random direction. Therefore, communication overhead of $O(N.\sqrt{N})$. Since, the number of witness nodes for a claim is d, the storage overhead is $O(d)$.

DDR [24]

This scheme is similar to LSM. Therefore, communication and storage overhead of this scheme is $O(N.\sqrt{N})$ and $O(\sqrt{N})$ respectively.

LANCE & SACRED [43]

LANCE protocol includes the counter value in Hello messages. Therefore, it does not require additional communication to share the counter value of a node among its neighbors. SACRED mechanism is similar to LSM. Therefore, communication overhead of SACRED is $O(N.\sqrt{N})$. The counter variable in LANCE does not incur any additional storage overhead; whereas SACRED has storage overhead of $O(\sqrt{N})$.

SDC & P-MPC [22]

Communication overhead of SDC and P-MPC is the sum of overheads in sending a location-claim to the destination cell and forwarding the claim to all the nodes within the destination cell. The communication cost of sending location-claim to destination cell(s) depends on the number of neighboring nodes and the probability p. Therefore, communication cost of sending a location-claim to a destination cell is $O(N.d.p.\sqrt{N})$. Similarly, the communication cost for sending a claim to g number of destination witness cells is $O(N.g.d.p.\sqrt{N})$. If the number of nodes within a cell is given by s, then communication overhead for sending the claim to all nodes within a cell is given by $O(s)$. Therefore, the overall communication overhead of SDC and P-MPC is $O(N.d.p.\sqrt{N}) + O(s)$ and $O(N.g.d.p.\sqrt{N}) + O(s)$ respectively.

Let p_s be the probability of storing a location-claim by a node within the destination cell(s). The additional storage overhead of both the schemes to store the location-claim is $O(w)$, where w is the number of witness nodes, which is equal to $s.p_s$ and $s.g.p_s$ for SDC and P-MPC respectively.

RAWL & TRAWL [44]

In [44], communication overhead of a node is evaluated in terms of number of random-walks performed per each claim. It is given that $\sqrt{N}.\log N$ number of random-walks are performed for each location-claim and is sufficient to detect a replica. Therefore, communication and storage overhead of both the schemes are $O(N.\sqrt{N}.\log N)$ and $O(\sqrt{N}.\log N)$ respectively.

A Note-Based Randomized and Distributive Protocol [31]

In this scheme, a reporter node forwards the subset claim of a node to g number of witness nodes. Therefore, communication overhead of this scheme is $O(N.g.\sqrt{N})$. The storage overhead is $O(g)$.

RED [32]

In RED, each neighboring node generates a random number of witness locations for a location-claim and forwards with a probability p. For g number of witnesses, communication overhead of RED is $O(N.g.d.p.\sqrt{N})$. Nodes that are closer to witness locations receive and store the location-claim of a node. Therefore, storage overhead of RED is $O(g.d.p)$.

An Area-Based Approach [26]

In this approach, location-claim of a node is sent to the witness node of the respective area and then to the central node following a path with an average path length of $\sqrt{N}$. Therefore, communication overhead of this scheme is $O(N.\sqrt{N})$. The highest storage overhead is associated with the central and witness nodes. The number of claims held by a central and witness node is given by N and $\frac{N}{a}$ respectively, where a denotes the number of areas defined in the network. Therefore, the storage overhead of this scheme is given by $O(N)$.

Hierarchical Node Replication Detection Scheme [23]

The major communication overhead of this scheme is in exchanging the member IDs among the cluster-heads. In a network of t number of clusters, the communication overhead is $O(t^2)$. The cluster-heads keep the member IDs of its own and other clusters with the help of Bloom filter. Therefore, the storage overhead of this scheme is $O(t)$.

2.3.2 Comparison of Number of Nodes Responsible for Replica Detection.

The probability of replica detection also depends on the number of nodes responsible for detecting a replica. We call this as *Number of Replica Detectors Per Node*. The measure of detection probability and the detection time mostly depends on this

Table 2.4: A quantitative comparison of different schemes based on communication, storage overhead, and number of nodes responsible for replica detection per node.

Scheme	**Communication Overhead**	**Storage Overhead**	**Number of Replica Detectors Per Node**
RM [36]	$O(N^2)$	$O(\sqrt{N})$	$\sqrt{N}$
LSM [36]	$O(N.\sqrt{N})$	$O(\sqrt{N})$	$\sqrt{N}$
SET [25]	$O(N)$	-	$\sqrt{t}$
Bekara *et al.* [38]	$O(N)$	-	d
Xing *et al.* [27]	$O(N)$	$O(d)$	d
Sei *et al.* [39]	$O(N.\sqrt{N})$	$O(g)$	g
NBDS [29]	$O(d.p.\sqrt{N})$	$O(d.p)$	d
B-MEM [40]	$O(N.\sqrt{N})$	-	$\sqrt{N}$
BC-MEM [40]	$O(N.\sqrt{N})$	-	$\sqrt{s}$
C-MEM [40]	$O(N.\sqrt{N})$	-	$2.\sqrt{N}$
CC-MEM [40]	$O(N.\sqrt{N})$	-	$2.\sqrt{s}$
Ho *et al.* [41]	$O(N.d.p.\sqrt{N})$	$O(w)$	w
RDE [42]	$O(N.\sqrt{N})$	$O(d)$	d
DDR [24]	$O(N.\sqrt{N})$	$O(\sqrt{N})$	v_p
LANCE [43]	-	-	d
SACRED [43]	$O(N.\sqrt{N})$	$O(\sqrt{N})$	$\sqrt{N}$
SDC [22]	$O(N.d.p.\sqrt{N}) + O(s)$	$O(w)$	s
P-MPC [22]	$O(N.g.d.p.\sqrt{N}) + O(s)$	$O(w)$	$g.s$
RAWL [44]	$O(N.\sqrt{N}.\log N)$	$O(\sqrt{N}.\log N)$	g
TRAWL [44]	$O(N.\sqrt{N}.\log N)$	$O(\sqrt{N}.\log N)$	g
Meng *et al.* [31]	$O(N.g.\sqrt{N})$	$O(g)$	g
RED [32]	$O(N.g.d.p.\sqrt{N})$	$O(g.d.p)$	$g.d$
Naruephiphat *et al.* [26]	$O(N.\sqrt{N})$	$O(N)$	w
Znaidi *et al.* [23]	$O(t^2)$	$O(t)$	t

parameter. That is, a higher *Number of Replica Detectors Per Node* will have a higher detection probability and lower detection time.

In the detection schemes such as RM and LSM [36], Sei *et al.* [39], [40], Ho *et al.* [41], DDR [24], SACRED [43], SDC and P-MPC [22], RAWL and TRAWL [44], RED [32], and Naruephiphat *et al.* [26], *Number of Replica Detectors Per Node* is equal to number of witnesses per node. In SET [25], all subset leaders in the path from the node to the BS are responsible for replica detection. For t number of subset leaders, *Number of Replica Detectors Per Node* is given by $\sqrt{t}$. In the schemes such as Bekara *et al.* [38], Xing *et al.* [27], NBDS [29], RDE [42], and LANCE [43], the neighboring nodes are responsible for detecting replica. Therefore, *Number of Replica Detectors Per Node* for these schemes is equal to d. In hierarchical detection schemes such as Znaidi *et al.* [23], the cluster-heads are responsible for replica detection. Therefore, the *Number of Replica Detectors Per Node* for these schemes is equal to t. It can be observed from the Table that the *Number of Replica Detectors Per Node* is comparatively higher in location-based schemes.

Communication and storage overhead associated with each scheme, and the *Number of Replica Detectors Per Node* are summarized in Table 2.4.

Table 2.5: Simulation parameters.

Parameter	**Value**
Area	1000 x 1000 m^2
Network size	1000 – 10000
Deployment type	Uniformly random
Communication range	55 meter
Average number of neighbors	30
Number of clones deployed	10
Simulation Time	600 sec
Number of witness	10
Claim forwarding probability	0.5

2.4 Simulation

We have simulated a few replica detection schemes to analyze their effectiveness. Schemes considered for simulation are LSM [36], SET [25], Sei *et al.* [39], SDC [22],

and RED [32]. The above schemes are often referred by researchers in replica detection. Castalia-3.2 [54] is used for simulation that runs on the top of Omnet++ [55]. Parameters considered for simulation are summarized in Table 2.5. Metrics considered for comparison are: *i)* Detection probability, *ii)* Average number of packets sent/received per node, *iii)* Energy consumed per node, *iv)* First clone detection time. The above metrics are considered because it is desirable to have a replica detection scheme with higher detection probability, lower detection time, lesser number of packets sent/receive per node and consume less energy. A large number of packets sent/receive in the detection process will not only utilize higher bandwidth but also depletes the energy at the node. Higher energy consumption in the detection process will decrease the longevity of the node.

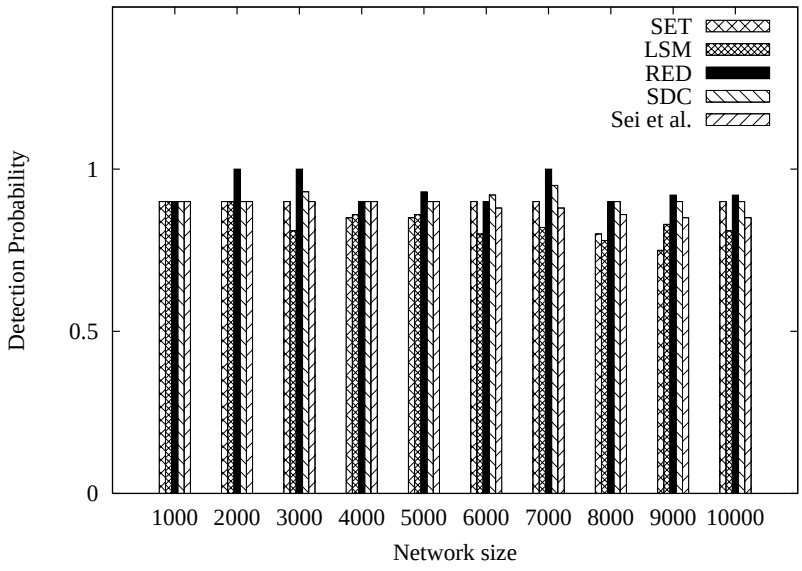


Figure 2.4: Detection probability *vs.* Network size.

The plot for Detection probability *vs.* Network size is shown in Figure 2.4. From the figure it is observed that RED has the highest detection probability among the schemes selected for comparison. This is because of the higher *Number of Replica Detectors Per Node*. From Table 2.4, it can be observed that RED have comparatively higher *Number of Replica Detectors Per Node* than LSM, SET, Sei *et al.*, and SDC. Moreover, RED was proposed to improve the performance of location-based schemes.

Figure 2.5 shows Detection probability *vs.* The number of witnesses for a network of 1000 nodes. It is observed that the detection probability increases with

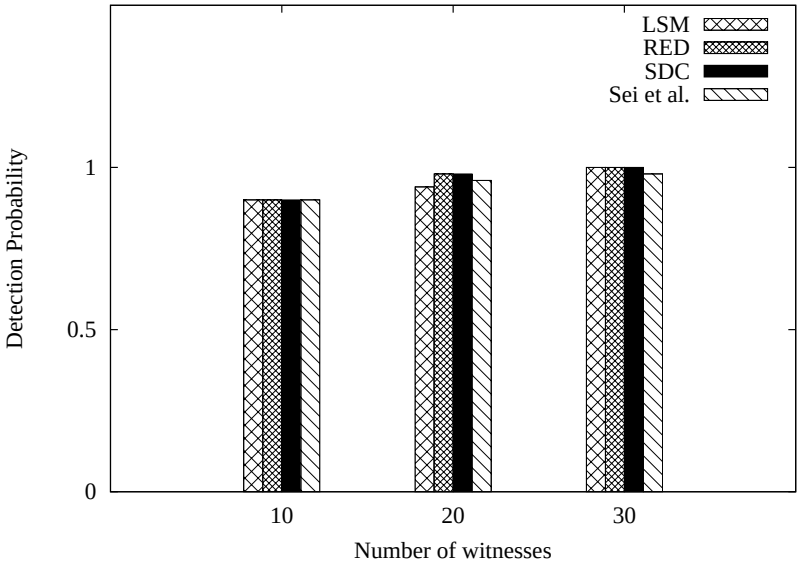


Figure 2.5: Detection probability *vs.* Number of witnesses for N = 1000.

increase in the number of witnesses for each scheme. Increase in the number of witnesses increases the probability that the conflicting location-claims would reach to at least one witness in the network. Therefore, this leads to improvement in detection probability.

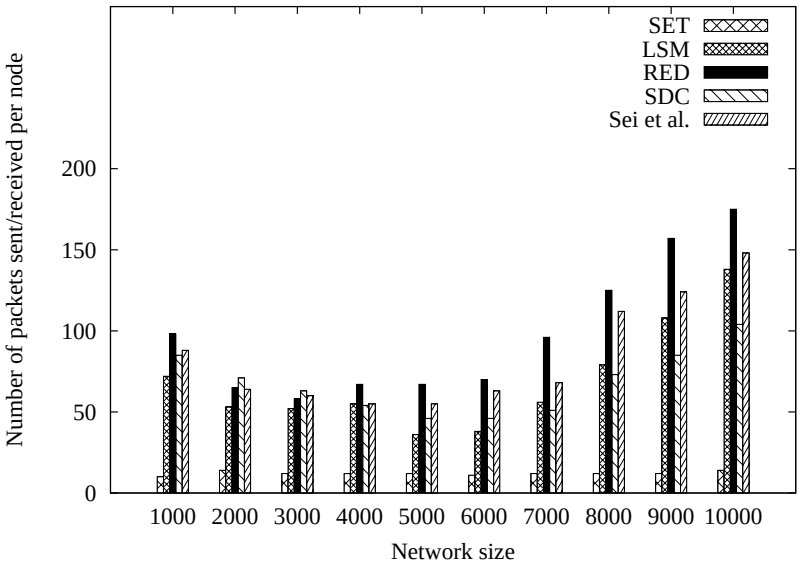


Figure 2.6: Average number of packets sent/received *vs.* Network size.

Average number of packets sent/received per node *vs.* Network size is shown in Figure 2.6. It is observed from the figure that the number of packets exchanged per node is higher in RED [32] and lower in SET [25]. This is because, RED have higher communication overhead and SET have lower communication overhead than other schemes. This is shown in Table 2.4.

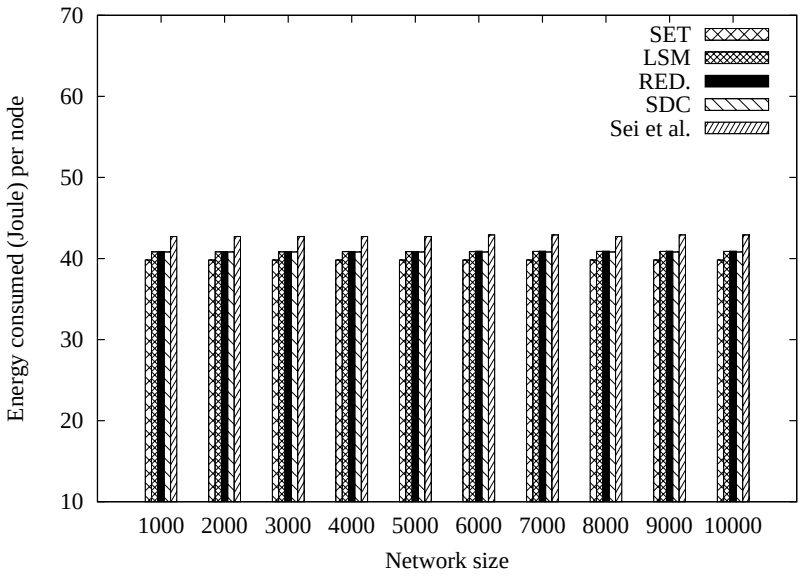


Figure 2.7: Energy consumed *vs.* Network size.

The energy consumed per node *vs.* Network size is shown in Figure 2.7. The energy consumption of a sensor node is the sum of the energy consumed by processor function, transceiver function, and sensor function. It is observed from the figure that the energy consumption in SET is marginally lower than SDC, LSM, and RED. This is because of lower communication overhead and longer message size in SET. A marginal higher energy consumption in Sei *et al.* [39] is attributed to the full broadcast mechanism it uses to forward the claim; which consumes more energy even for a smaller path than the scheme that adapts probabilistic forwarding mechanism.

The plot for Clone detection time *vs.* Network size is shown in Figure 2.8. Clone detection time is the time difference between the first clone deployed in the network and the first clone detected. From the figure, it is observed that LSM takes lesser time to detect the deployed clone in comparison to other schemes. This is because, in LSM each intermediate node between the location claimer and its witness nodes caches the location-claim. These intermediate nodes, in addition to

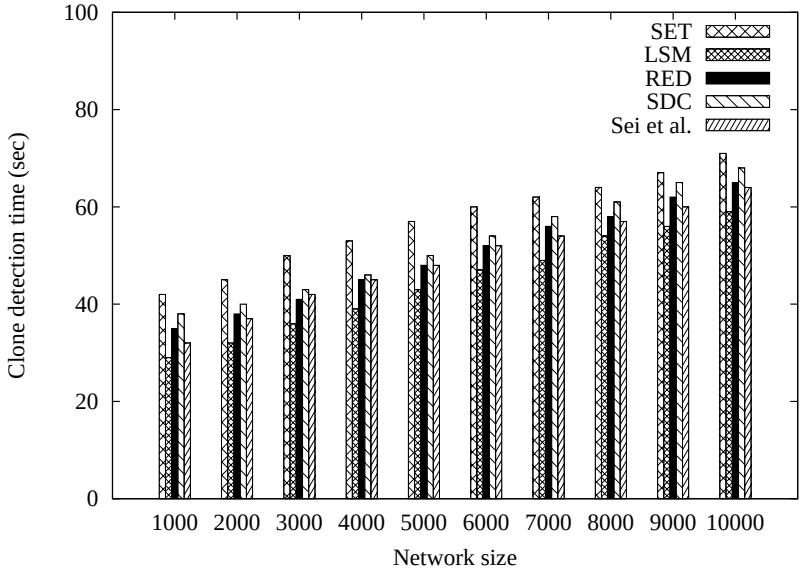


Figure 2.8: First clone detection time *vs.* Network size.

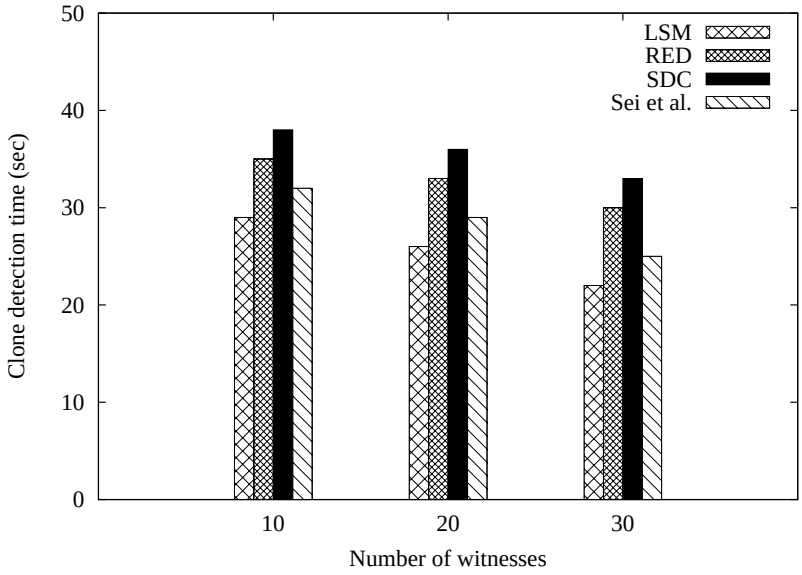


Figure 2.9: First clone detection time *vs.* Number of witnesses for N = 1000.

the witness nodes also detect a replica when a location conflict occurs; whereas in other schemes, replica is detected either by witness nodes or BS. It is also observed that SET have higher clone detection time. This is because, in SET, replica is mostly detected either by the BS or the nodes closer to BS. This requires more detection time compared to other schemes.

Finally, the detection time of LSM, RED, SDC, and Sei *et al.* is plotted varying the number of witnesses to 10, 20, and 30 for a network of 1000 nodes. This is shown in Figure 2.9. It is observed that detection time reduces with increase in the number of witnesses. This is because, a higher number of witnesses reduce the number of hops between a claimer and its witness nodes.

2.5 Summary

In this chapter, we have briefly discussed the various schemes for replica detection as reported in the literature. We have classified the existing schemes based on the detection mechanism, need of geographical information, claim forwarding strategy, and message routing type. A quantitative analysis of different schemes is presented. We have simulated a few popular replica detection schemes such as SET [25], LSM [36], SDC [22], Sei *et al.* [39], and RED [32] to compare their effectiveness. The metrics considered for comparison are detection probability, detection time, average number of packets sent/received, and energy consumption. We observed that the RED has higher detection probability, and relatively higher detection time. The SET incurs lower communication overhead than other schemes. We need a scheme that not only have higher detection probability but also lower detection time. Since, energy is an important issue in WSN, a replica detection scheme should also consume lesser energy as well. In a nutshell it is desirable to have a replica detection scheme that has higher detection probability, lower detection time, communication overhead and energy consumption.

Most of the replica detection mechanisms require the exchange of membership information among the nodes. Therefore, a lightweight mechanism for exchanging information among the nodes is required in replica detection. Next chapter, describes a mechanism for exchanging group information among the nodes.

Chapter 3

Mechanism for Exchanging Group Membership Information

Applications such as cluster formation, clone detection, and neighbor-list sharing require the exchange of group membership information among the nodes. Group membership information consists of the identity of all nodes in the group. Trivial method of exchanging the group membership information is to send the individual identity of all nodes in the group. Though, this method of information exchange is simpler, yet it leads to higher message overhead even for a group of smaller size. Another method for exchanging group membership information is to use bit-stream. The number of bits in the bit-stream is equal to the size of the network. The i^{th} bit in the bit-stream is set to *One*, if the node with identity, i, is a member of the group. In this scheme, the size of the bit-stream increases proportionately with the size of the network. For larger networks, the overhead associated with communication and storage is higher. Another mechanism proposed in the literature is to use Bloom filter [33, 34, 56]. This is an efficient technique for exchanging group membership information. However, it suffers from higher cases of false positive *i.e.*, a node may be reported as a member of a group when it is not.

For exchanging group membership information the overhead associated with a scheme should not only be lower, but also should not give rise to any case of false positive. In this chapter, we propose two schemes, namely, *Transpose Bit-Pair Coding* (TBC), and *Sub-Mat Coding* (SMC) to reduce the communication and storage overhead associated with exchanging group membership information among the nodes. The proposed schemes not only reduce the communication and storage overhead, but also do not generate any false positive.

3.1 Related Work

One of the simplest method for exchanging group membership information is to exchange the identity of each member of the group. This form of exchange was adopted by Choi *et al.* [25], Meng *et al.* [31], and Naruephiphat *et al.* [26] in their proposed schemes. The number of identities exchanged per message depend on the size of the message. With the increase in the network size, the number of nodes per group also increases. As a result, the number of messages exchanged to share the group membership information increases. Therefore, this method is not suitable for a dense network. To reduce the communication overhead, a subset of group membership identities is exchanged instead of the identities of entire group membership [31]. This scheme can reduce communication overhead to some extent, but cannot provide the complete group membership information of a node.

Znaidi *et al.* [23], Zhang *et al.* [40], and Deng *et al.* [57] have used Bloom filter [33, 34, 56] in their proposed schemes to share the group membership information. A Bloom filter consists of the following: *i) One* b-bit array B, to store and test the membership of a node, and *ii)* k number of hash functions, $H_i()$, $0 \leq i \leq k$ to map the identity of a group member to certain number of bits in B.

To add a member ID_x to the Bloom filter, all $H_i(ID_x)$ positions are set to *one* in the bit-array B. The membership of a node ID_y is tested by checking for a value *one* in every position, $H_i(ID_y)$ of B. Group membership sharing mechanisms that are based on Bloom filter have lower memory and communication cost. However, they suffer from higher probability of false positive. Given k, the probability of false positive p is given by

$$p = \left(1 - \left(1 - \frac{1}{b}\right)^{k.n}\right)^k \approx \left(1 - e^{-\frac{k.n}{b}}\right)^k \tag{3.1}$$

where, b is the length of the array B and n is the number of members in a group [33, 34]. For a given value of k and b, the probability of false positive p, increases with the increase in the size n and decreases as b increases. The optimal number of hash functions as computed in [34] is given by

$$k = \left(\frac{b}{n}\right) \ln 2 \tag{3.2}$$

For a given false positive probability p, the length of the Bloom filter b, is proportional to the number of elements being filtered n [34], which is expressed as follows:

$$b = -\frac{n \ln p}{(\ln 2)^2} \tag{3.3}$$

For an optimal value of k, Bloom filter with *one* percent of error requires 9.6 bits per element. Size of the array B, determines the communication and storage cost associated with Bloom filter. Each additional 4.8 bits per element, decreases the error rate by *one-tenth*. Further details on Bloom filter can be found in [33, 34, 56].

Another method for exchanging group membership information is the use of bit-stream [58, 59]. In this scheme, the number of bits in a bit-stream is equal to the size of the network. The i^{th} bit in the bit-stream is set to *one*, if the node with identity, i, is a member of the group. This method works well in a network of fixed size. However, the size of the bit-stream increases proportionately with the size of the network. In a dense network with relatively smaller group size, the overhead associated with communication and storage is higher.

Data compression techniques for WSNs are proposed in [60–62]. These techniques operate on sensed data and make extensive use of RAM. Therefore, they are not suitable for exchanging group membership in WSN.

The bit-stream based mechanisms do not generate false positives. However, they have higher storage and communication cost. Mechanisms based on Bloom filter have higher cases of false positives. The communication and storage cost associated with such mechanisms are lower. Therefore, we need a scheme that does not generate false positive, and the associated communication and storage cost is comparable with that of Bloom filter. In TBC and SMC, we have tried to balance between the cases of false positives and associated cost.

3.2 Proposed Mechanism

In this section, we propose two schemes: *i) Transpose Bit-Pair Coding* (TBC), and *ii) Sub-Mat Coding* (SMC) for exchanging group members' identity among the nodes. In the proposed scheme, group membership information is encoded into a bit-stream, which is exchanged among the nodes. The encoded bit-stream is stored at a node and is decoded to obtain the group membership information. The encoding technique is lossless and does not generate false positives.

The format used in representing the group membership information is described in sub-section 3.2.1. TBC and SMC are described in sub-section 3.2.2 and 3.2.3 respectively.

3.2.1 Representation of Group Membership Information

This sub-section describes the representation of group membership information that is used in the proposed schemes. Let N be the network size and n_G be the average size of a group where, $n_G < N$, and let S_G be an arbitrary group. Members of the group, S_G, are represented in the form of a matrix, $\boldsymbol{M}$, as shown below:

$$\boldsymbol{M} = \begin{bmatrix} m_{0,0} & m_{0,1} & \dots & m_{0,Dim-1} \\ m_{1,0} & m_{1,1} & \dots & m_{1,Dim-1} \\ \dots & \dots & \dots & \dots \\ m_{Dim-1,0} & m_{Dim-1,1} & \dots & m_{Dim-1,Dim-1} \end{bmatrix} \tag{3.4}$$

where, the element $m_{i,j}$ stores one-bit of information about the membership of a node whose identity is $(Dim * i + j)$. Given the N, n_G and S_G, the matrix $\boldsymbol{M}$ is constructed as follows:

$$m_{i,j} = \begin{cases} 1 & \textit{iff} \text{ node with an identity, } (Dim * i + j) \in S_G \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

We have assumed the network size, $N \leq Dim \times Dim$, and the identity of nodes are unique integral value in $[0, N-1]$.

3.2.2 Transpose Bit-Pair Coding (TBC)

TBC takes the group membership matrix $\boldsymbol{M}$, as input and produces a bit-stream C, where $|C| << N$. The bit-stream is generated by considering each pair of elements $m_{i,j}$ and $m_{j,i}$ of $\boldsymbol{M}$. An element is considered only once in the generation of a bit-stream. Let c be the partial bit-stream generated from a pair of elements $m_{i,j}$ and $m_{j,i}$. Then, c is generated using the following equation:

$$c = \begin{cases} m_{i,j} & i = j \\ 0 & i \neq j, m_{i,j} = m_{j,i} = 0 \\ 10 & i \neq j, m_{i,j} = 1, m_{j,i} = 0 \\ 110 & i \neq j, m_{i,j} = 0, m_{j,i} = 1 \\ 111 & i \neq j, m_{i,j} = 1, m_{j,i} = 1 \end{cases} \tag{3.6}$$

The encoding process in TBC is given in Algorithm 3.1, where only the upper triangular matrix of $\boldsymbol{M}$ is traversed. The bit-stream C, can also be generated by traversing only the lower triangular matrix of $\boldsymbol{M}$. In Step 3 of Algorithm 3.1 each element of the main diagonal is read, and written into the bit-stream. From Step 6 to 17 a pair of elements at index (i, j) and (j, i) of $\boldsymbol{M}$ is read and the corresponding

Algorithm 3.1: Transpose Bit-Pair Coding

Input: Matrix, $\boldsymbol{M}$
Output: Bit-stream, C

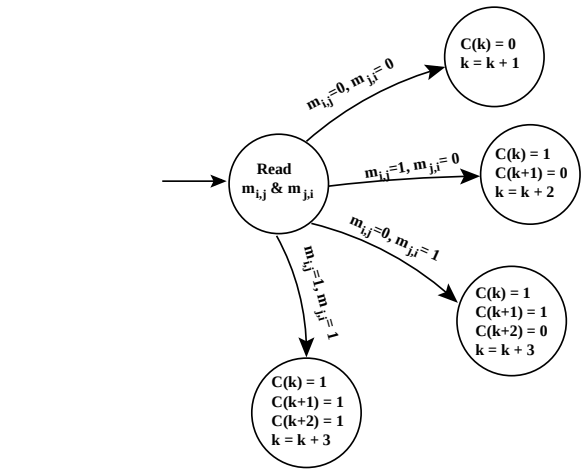
```
1  k ⟵ 0
2  for i ← 0 to Dim − 1 do
3      C(k) ⟵ m_{i,i}
4      k ⟵ k + 1
5      for j ← i + 1 to Dim − 1 do
6          if m_{i,j} = 0 & m_{j,i} = 0 then
7              C(k) ⟵ 0
8              k ⟵ k + 1
9          else if m_{i,j} = 1 & m_{j,i} = 0 then
10             C(k) ⟵ 1, C(k + 1) ⟵ 0
11             k ⟵ k + 2
12         else if m_{i,j} = 0 & m_{j,i} = 1 then
13             C(k) ⟵ 1, C(k + 1) ⟵ 1, C(k + 2) ⟵ 0
14             k ⟵ k + 3
15         else
16             C(k) ⟵ 1, C(k + 1) ⟵ 1, C(k + 2) ⟵ 1
17             k ⟵ k + 3
```

Algorithm 3.2: Transpose Bit-Pair Decoding

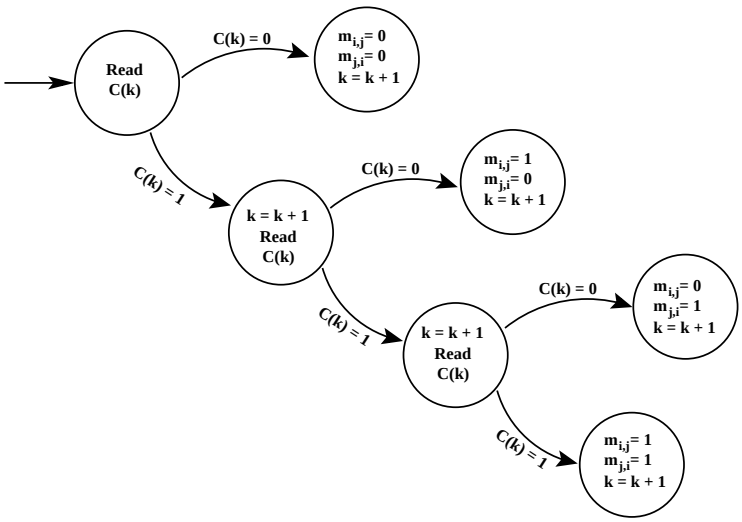
Input: Bit-stream, C
Output: Matrix, $\boldsymbol{M'}$

```
1  k ⟵ 0
2  for i ← 0 to Dim − 1 do
3      m'_{i,i} ⟵ C(k)
4      k ⟵ k + 1
5      for j ← i + 1 to Dim − 1 do
6          if C(k) = 0 then
7              m'_{i,j} ⟵ m'_{j,i} ⟵ 0
8          else
9              k ⟵ k + 1
10             if C(k) = 0 then
11                 m'_{i,j} ⟵ 1, m'_{j,i} ⟵ 0
12             else
13                 k ⟵ k + 1
14                 if C(k) = 0 then
15                     m'_{i,j} ⟵ 0, m'_{j,i} ⟵ 1
16                 else
17                     m'_{i,j} ⟵ 1, m'_{j,i} ⟵ 1
```

bit-stream is generated as per the transition diagram shown in Figure 3.1(a). The time complexity in generating the bit-stream is $O(Dim^2) = O(N)$ as $N \approx Dim^2$.



(a)



(b)

Figure 3.1: Transition diagrams showing: (a) Encoding in TBC, (b) Decoding in TBC.

The decoding process in TBC is given in Algorithm 3.2. Step 3 of the algorithm generates the element across the main diagonal. Step 5 to 17 generates a pair of elements $m_{i,j}$ and $m_{j,i}$ of matrix $\boldsymbol{M}$ as per the transition diagram shown in Figure 3.1(b). The time complexity of the decoding process in TBC is $O(N)$. We claim the following from the encoding and decoding process in TBC.

Claim 3.1. *The encoding and decoding process in TBC is unique.*

Proof. The encoding and decoding process in TBC is said to be unique, if the following two conditions hold true:

i) The mapping $f : \boldsymbol{M} \rightarrow C$ is one-to-one, and

ii) The decoding of bit-stream C is unambiguous

The mapping function $f : \boldsymbol{M} \rightarrow C$ is uniquely defined by the Equation 3.6. The state transition diagram shown in Figure 3.1(a), reads a pair of element and uniquely generates the corresponding bit-stream. Therefore, the mapping function f is one-to-one.

The decoding process follows the prefix codes, *i.e,* no code is a prefix of any other code. Transition diagram in Figure 3.1(b) shows the decoding process. From the transition diagram it is observed that the decoding process is unambiguous. □

We illustrate below the encoding and decoding process in TBC with a suitable example. Let us consider a network with the following parameters: $N = 100$, $n_G = 12$, and $S_G = \{ 6, 11, 19, 25, 28, 35, 38, 46, 59, 64, 87, 91\}$. The matrix $\boldsymbol{M}$ representing S_G is given below:

$$\boldsymbol{M} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Table 3.1 shows the encoding process of matrix, $\boldsymbol{M}$, to generate the bit-stream C, for the given S_G. The bit-stream generated by TBC for S_G in the above example is $\{0000001000010000000111000100010000100010000111000000010000001100000\}$. The generated bit-stream gives the membership information of a group, which is shared among the nodes.

The decoding algorithm generates a matrix, $\boldsymbol{M'}$, from which the membership information of the group is obtained. The elements of $\boldsymbol{M'}$ for the bit-stream, C,

Table 3.1: Encoding of matrix $\boldsymbol{M}$, in TBC.

Indices *(i,j)(j,i)*	**Bits** $m_{i,j}m_{j,i}$	**Code** c	**Code** *(i,j)(j,i)*	**Bits** $m_{i,j}m_{j,i}$	**Code** c	**Indices** *(i,j)(j,i)*	**Bits** $m_{i,j}m_{j,i}$	**Code** c
(0, 0)(0, 0)	00	0	(0, 1)(1, 0)	00	0	(0, 2)(2, 0)	00	0
(0, 3)(3, 0)	00	0	(0, 4)(4, 0)	00	0	(0, 5)(5, 0)	00	0
(0, 6)(6, 0)	10	10	(0, 7)(7, 0)	00	0	(0, 8)(8, 0)	00	0
(0, 9)(9, 0)	00	0	(1, 1)(1, 1)	11	1	(1, 2)(2, 1)	00	0
(1, 3)(3, 1)	00	0	(1, 4)(4, 1)	00	0	(1, 5)(5, 1)	00	0
(1, 6)(6, 1)	00	0	(1, 7)(7, 1)	00	0	(1, 8)(8, 1)	00	0
(1, 9)(9, 1)	11	111	(2, 2)(2, 2)	00	0	(2, 3)(3, 2)	00	0
(2, 4)(4, 2)	00	0	(2, 5)(5, 2)	10	10	(2, 6)(6, 2)	00	0
(2, 7)(7, 2)	00	0	(2, 8)(8, 2)	10	10	(2, 9)(9, 2)	00	0
(3, 3)(3, 3)	00	0	(3, 4)(4, 3)	00	0	(3, 5)(5, 3)	10	10
(3, 6)(6, 3)	00	0	(3, 7)(7, 3)	00	0	(3, 8)(8, 3)	10	10
(3, 9)(9, 3)	00	0	(4, 4)(4, 4)	00	0	(4, 5)(5, 4)	00	0
(4, 6)(6, 4)	11	111	(4, 7)(7, 4)	00	0	(4, 8)(8, 4)	00	0
(4, 9)(9, 4)	00	0	(5, 5)(5, 5)	00	0	(5, 6)(6, 5)	00	0
(5, 7)(7, 5)	00	0	(5, 8)(8, 5)	00	0	(5, 9)(9, 5)	10	10
(6, 6)(6, 6)	00	0	(6, 7)(7, 6)	00	0	(6, 8)(8, 6)	00	0
(6, 9)(9, 6)	00	0	(7, 7)(7, 7)	00	0	(7, 8)(8, 7)	01	110
(7, 9)(9, 7)	00	0	(8, 8)(8, 8)	00	0	(8, 9)(9, 8)	00	0
(9, 9)(9, 9)	00	0						

considered in the above example is shown in Table 3.2. From $\boldsymbol{M'}$, we obtain the membership information of the group as { 6, 11, 19, 25, 28, 35, 38, 46, 59, 64, 87, 91}, which is same as the one considered in the example.

3.2.3 Sub-Mat Coding (SMC)

In SMC scheme, the membership matrix, $\boldsymbol{M}$, is logically divided into 2 x 2 sub-matrices as shown below. Each sub-matrix of $\boldsymbol{M}$ is then used to generate a portion of the bit-stream C.

$$\boldsymbol{M} = \begin{bmatrix} \begin{bmatrix} m_{0,0} & m_{0,1} \\ m_{1,0} & m_{1,1} \end{bmatrix} & \cdots & \begin{bmatrix} m_{0,Dim-2} & m_{0,Dim-1} \\ m_{1,Dim-2} & m_{1,Dim-1} \end{bmatrix} \\ \cdots & \cdots & \cdots \\ \begin{bmatrix} m_{Dim-2,0} & m_{Dim-2,1} \\ m_{Dim-1,0} & m_{Dim-1,1} \end{bmatrix} & \cdots & \begin{bmatrix} m_{Dim-2,Dim-2} & m_{Dim-2,Dim-1} \\ m_{Dim-1,Dim-2} & m_{Dim-1,Dim-1} \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \boldsymbol{M}_{0,0} & \boldsymbol{M}_{0,2} & \dots & \boldsymbol{M}_{0,\frac{Dim}{2}-1} \\ \boldsymbol{M}_{2,0} & \boldsymbol{M}_{2,2} & \dots & \boldsymbol{M}_{2,\frac{Dim}{2}-1} \\ \dots & \dots & \dots & \dots \\ \boldsymbol{M}_{Dim-2,0} & \boldsymbol{M}_{Dim-2,2} & \dots & \boldsymbol{M}_{\frac{Dim}{2}-1,\frac{Dim}{2}-1} \end{bmatrix} \tag{3.7}$$

where,

$$\boldsymbol{M}_{i,j} = \begin{bmatrix} m_{2*i,2*j} & m_{2*i,2*j+1} \\ m_{2*i+1,2*j} & m_{2*i+1,2*j+1} \end{bmatrix} \tag{3.8}$$

. Let c be the partial bit-stream generated from a sub-matrix $\boldsymbol{M}_{i,j}$. Then, c is generated as shown below:

$$c = \begin{cases} 0 & \text{if } \boldsymbol{M}_{i,j} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \\ 1 \| m_{2*i,2*j} \| m_{2*i,2*j+1} \| m_{2*i+1,2*j} \| m_{2*i+1,2*j+1} & \text{otherwise} \end{cases} \tag{3.9}$$

where, $\|$ is used as concatenation operator.

Algorithm 3.3: Sub-MAT Coding

Input: Matrix, $\boldsymbol{M}$
Output: Bit-stream, C

1 $k \longleftarrow 0$
2 **for** $i \leftarrow 0$ **to** $Dim - 1$ *with increment 2* **do**
3 **for** $j \leftarrow 0$ **to** $Dim - 1$ *with increment 2* **do**
4 **if** $m_{i,j} = m_{i,j+1} = m_{i+1,j} = m_{i+1,j+1} = 0$ **then**
5 $C(k) \longleftarrow 0$
6 $k \longleftarrow k + 1$
7 **else**
8 $C(k) \longleftarrow 1$
9 $C(k+1) \longleftarrow m_{i,j}$
10 $C(k+2) \longleftarrow m_{i,j+1}$
11 $C(k+3) \longleftarrow m_{i+1,j}$
12 $C(k+4) \longleftarrow m_{i+1,j+1}$
13 $k \longleftarrow k + 5$

Algorithm 3.3 shows the encoding process in SMC. In Step 2 to 13 of the algorithm each element of the sub-matrix $\boldsymbol{M}_{i,j}$, is read and its corresponding bit-stream is generated using Equation 3.9. Figure 3.2(a) shows the generation of bit-stream from each sub-matrix $\boldsymbol{M}_{i,j}$. The time complexity of this algorithm is $O(Dim^2) = O(N)$.

Table 3.2: Decoding of bit-stream C, in TBC.

Index *(i, j)*	**Input Bit**	$m'_{i,j}$ & $m'_{j,i}$	**Index** *(i, j)*	**Input Bit**	$m'_{i,j}$ & $m'_{j,i}$
0, 0	0	$m'_{0,0} = 0$	0, 1	0	$m'_{0,1} = m'_{1,0} = 0$
0, 2	0	$m'_{0,2} = m'_{2,0} = 0$	0, 3	0	$m'_{0,3} = m'_{3,0} = 0$
0, 4	0	$m'_{0,4} = m'_{4,0} = 0$	0, 5	0	$m'_{0,5} = m'_{5,0} = 0$
0, 6	1	Read next bit	0, 6	0	$m'_{0,6} = 1, m'_{6,0} = 0$
0, 7	0	$m'_{0,7} = m'_{7,0} = 0$	0, 8	0	$m'_{0,8} = m'_{8,0} = 0$
0, 9	0	$m'_{0,9} = m'_{9,0} = 0$	1, 1	1	$m'_{1,1} = 1$
1, 2	0	$m'_{1,2} = m'_{2,1} = 0$	1, 3	0	$m'_{1,3} = m'_{3,1} = 0$
1, 4	0	$m'_{1,4} = m'_{4,1} = 0$	1, 5	0	$m'_{1,5} = m'_{5,1} = 0$
1, 6	0	$m'_{1,6} = m'_{6,1} = 0$	1, 7	0	$m'_{1,7} = m'_{7,1} = 0$
1, 8	0	$m'_{1,8} = m'_{8,1} = 0$	1, 9	1	Read next bit
1, 9	1	Read next bit	1, 9	1	$m'_{1,9} = 1, m'_{9,1} = 1$
2, 2	0	$m'_{2,2} = 0$	2, 3	0	$m'_{2,3} = m'_{3,2} = 0$
2, 4	0	$m'_{2,4} = m'_{4,2} = 0$	2, 5	1	Read next bit
2, 5	0	$m'_{2,5} = 1, m'_{5,2} = 0$	2, 6	0	$m'_{2,6} = m'_{6,2} = 0$
2, 7	0	$m'_{2,7} = m'_{7,2} = 0$	2, 8	1	Read next bit
2, 8	0	$m'_{2,8} = 1, m'_{8,2} = 0$	2, 9	0	$m'_{2,9} = m'_{9,2} = 0$
3, 3	0	$m'_{3,3} = 0$	3, 4	0	$m'_{3,4} = m'_{4,3} = 0$
3, 5	1	Read next bit	3, 5	0	$m'_{3,5} = 1, m'_{5,3} = 0$
3, 6	0	$m'_{3,6} = 1, m'_{6,3} = 0$	3, 7	0	$m'_{3,7} = 1, m'_{7,3} = 0$
3, 8	1	Read next bit	3, 8	0	$m'_{3,8} = 1, m'_{8,3} = 0$
3, 9	0	$m'_{3,9} = 1, m'_{9,3} = 0$	4, 4	0	$m'_{4,4} = 0$
4, 5	0	$m'_{4,5} = m'_{5,4} = 0$	4, 6	1	Read next bit
4, 6	1	Read next bit	4, 6	1	$m'_{4,6} = m'_{6,4} = 1$
4, 7	0	$m'_{4,7} = m'_{7,4} = 0$	4, 5	0	$m'_{4,8} = m'_{8,4} = 0$
4, 5	0	$m'_{4,9} = m'_{9,4} = 0$	5, 5	0	$m'_{5,5} = 0$
5, 6	0	$m'_{5,6} = m'_{6,5} = 0$	5, 7	0	$m'_{5,7} = m'_{7,5} = 0$
5, 8	0	$m'_{5,8} = m'_{8,5} = 0$	5, 9	1	Read next bit
5, 9	0	$m'_{5,9} = 1, m'_{9,5} = 0$	6, 6	0	$m'_{6,6} = 0$
6, 7	0	$m'_{6,7} = m'_{7,6} = 0$	6, 8	0	$m'_{6,8} = m'_{8,6} = 0$
6, 9	0	$m'_{6,9} = m'_{9,6} = 0$	7, 7	0	$m'_{7,7} = 0$
7, 8	1	Read next bit	7, 8	1	Read next bit
7, 8	0	$m'_{7,8} = 0, m'_{8,7} = 1$	7, 9	0	$m'_{7,9} = m'_{9,7} = 0$
8, 8	0	$m'_{8,8} = 0$	8, 9	0	$m'_{8,9} = m'_{9,8} = 0$
9, 9	0	$m'_{9,9} = 0$			

Decoding process in SMC is shown in Algorithm 3.4. Step 2 to 13 of the algorithm construct each sub-matrix. Transition diagram in Figure 3.2(b) shows the

Algorithm 3.4: Sub-MAT Decoding

Input: Bit-stream, C
Output: Matrix, $\boldsymbol{M}'$

```
1  k ⟵ 0
2  for i ← 0 to Dim  with increment 2 do
3      for j ← 0 to Dim  with increment 2 do
4          if C(k) = 0 then
5              m'_{i,j} ← m'_{i,j+1} ← m'_{i+1,j} ← m'_{i+1,j+1} ← 0
6              k ⟵ k + 1
7          else
8              m'_{i,j} ⟵ C(k + 1)
9              m'_{i,j+1} ⟵ C(k + 2)
10             m'_{i+1,j} ⟵ C(k + 3)
11             m'_{i,j} ⟵ C(k + 4)
12             k ⟵ k + 5
```

construction of sub-matrices. The time complexity of the decoding process is $O(N)$. We claim the following from the encoding and decoding process in SMC.

Claim 3.2. *The process of encoding and decoding in SMC is unique.*

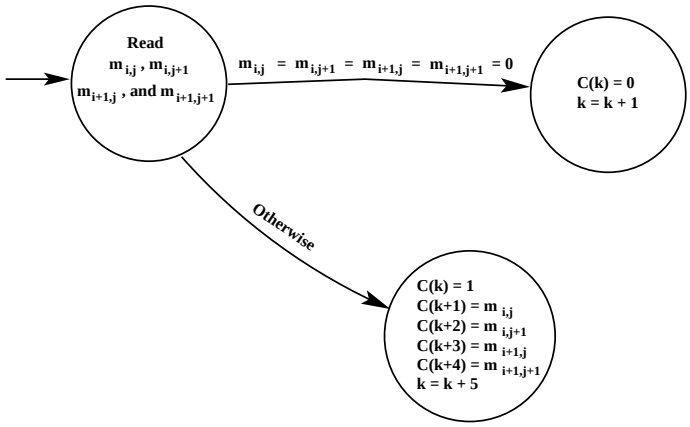
Proof. The encoding and decoding process in SMC is said to be unique, if the following two conditions hold true:

i) The mapping $f : \boldsymbol{M} \rightarrow C$ is one-to-one, and

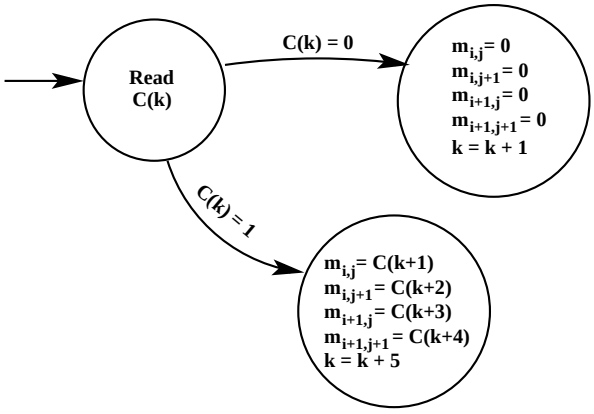
ii) The decoding of bit-stream C is unambiguous

In SMC, the mapping function $f : \boldsymbol{M} \rightarrow C$ generates a bit-stream corresponding to the sub-matrix $\boldsymbol{M}_{i,j}$, as given in Equation 3.9. Figure 3.2(a) shows that the generation of bit-stream is one-to-one. The decoding of the bit-stream is unambiguous as shown in the Figure 3.2(b). □

We explain below the encoding and decoding process in SMC using the example in sub-section 3.2.2. Each sub-matrix of $\boldsymbol{M}$ is given below:



(a)



(b)

Figure 3.2: Transition diagrams showing: (a) Encoding in SMC, (b) Decoding in SMC.

$$\boldsymbol{M} = \begin{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \end{bmatrix}$$

The code generated from each of sub-matrix $\boldsymbol{M}_{i,j}$ of $\boldsymbol{M}$ represented in the form of

Table 3.3: Decoding of bit-stream C, in SMC.

Index *(i, j)*	**Input Bit**	**Action**	**Matrix**
0, 0	1	Read next 4 bits	$m'_{0,0}=0, m'_{0,1}=0, m'_{1,0}=0, m'_{1,1}=1$
0, 2	0	-	$m'_{0,2}=m'_{0,3}=m'_{1,2}=m'_{1,3}=0$
0, 4	0	-	$m'_{0,4}=m'_{0,5}=m'_{1,4}=m'_{1,5}=0$
0, 6	1	Read next 4 bits	$m'_{0,6}=1, m'_{0,7}=0, m'_{1,6}=0, m'_{1,7}=0$
0, 8	1	Read next 4 bits	$m'_{0,8}=0, m'_{0,9}=0, m'_{1,8}=0, m'_{1,9}=1$
2, 0	0	-	$m'_{2,0}=m'_{2,1}=m'_{3,0}=m'_{3,1}=0$
2, 2	0	-	$m'_{2,2}=m'_{2,3}=m'_{3,2}=m'_{3,3}=0$
2, 4	1	Read next 4 bits	$m'_{2,4}=0, m'_{2,5}=1, m'_{3,4}=0, m'_{3,5}=1$
2, 6	0	-	$m'_{2,6}=m'_{2,7}=m'_{3,6}=m'_{3,7}=0$
2, 8	1	Read next 4 bits	$m'_{2,8}=1, m'_{2,9}=0, m'_{3,8}=1, m'_{3,9}=0$
4, 0	0	-	$m'_{4,0}=m'_{4,1}=m'_{5,0}=m'_{5,1}=0$
4, 2	0	-	$m'_{4,2}=m'_{4,3}=m'_{5,2}=m'_{5,3}=0$
4, 4	0	-	$m'_{4,4}=m'_{4,5}=m'_{5,4}=m'_{5,5}=0$
4, 6	1	Read next 4 bits	$m'_{4,6}=1, m'_{4,7}=0, m'_{5,6}=1, m'_{5,7}=0$
4, 8	1	Read next 4 bits	$m'_{4,8}=0, m'_{4,9}=0, m'_{5,8}=1, m'_{5,9}=1$
6, 0	0	-	$m'_{6,0}=m'_{6,1}=m'_{7,0}=m'_{7,1}=0$
6, 2	0	-	$m'_{6,2}=m'_{6,3}=m'_{7,2}=m'_{7,3}=0$
6, 4	1	Read next 4 bits	$m'_{6,4}=1, m'_{6,5}=0, m'_{7,4}=1, m'_{7,5}=0$
6, 6	0	-	$m'_{6,6}=m'_{6,7}=m'_{7,6}=m'_{7,7}=0$
6, 8	0	-	$m'_{6,8}=m'_{6,9}=m'_{7,8}=m'_{7,9}=0$
8, 0	1	Read next 4 bits	$m'_{8,0}=0, m'_{8,1}=0, m'_{9,0}=1, m'_{9,1}=1$
8, 2	0	-	$m'_{8,2}=m'_{8,3}=m'_{9,2}=m'_{9,3}=0$
8, 4	0	-	$m'_{8,4}=m'_{8,5}=m'_{9,4}=m'_{9,5}=0$
8, 6	1	Read next 4 bits	$m'_{8,6}=0, m'_{8,7}=1, m'_{9,6}=0, m'_{9,7}=0$
8, 8	0	-	$m'_{8,8}=m'_{8,9}=m'_{9,8}=m'_{9,9}=0$

a matrix, $\boldsymbol{M}^c$, is shown below:

$$\boldsymbol{M}^c = \begin{bmatrix} 10001 & 0 & 0 & 11000 & 10001 \\ 0 & 0 & 10101 & 0 & 11010 \\ 0 & 0 & 0 & 11000 & 10001 \\ 0 & 0 & 11000 & 0 & 0 \\ 10001 & 0 & 0 & 10100 & 0 \end{bmatrix}$$

The matrix, $\boldsymbol{M}^c$, is read row-wise to generate the bit-stream C. The bit-stream C generated by SMC for the group S_G considered in the example is $\{1000100110001000100101010110100001100010001001100000100010010100 0\}$.

Decoding of the bit-stream C in SMC is shown in Table 3.3.

3.3 Simulation and Results

In this section, the performance of TBC and SMC is analyzed through simulation. Parameters considered for simulation are shown in Table 3.4. We have compared TBC and SMC with the following schemes: *i)* that exchange the identity of each node [25,26,31] (call this as $Scheme_1$), and *ii)* that uses bit-stream of size N [58,59] (call this as $Scheme_2$), and *iii)* that uses Bloom filter. We have assumed that nodes are identified by a unique integer and is represented by sixteen bits.

Table 3.4: Simulation parameters.

Parameter	**Value**
Network size (N)	1024 – 10000
Group size	2.5 – 15 (in % of N)
Number of groups	10 – 40
Maximum number of neighbors	150
Matrix Dimension (Dim)	32 – 100

Table 3.5 shows the space required in terms of number of bits by each scheme to store the group membership information. Size of the group is considered to be 150. $Scheme_1$ and scheme that uses Bloom filter depend only on the group size. Therefore, the number of bits required in these two schemes is constant for network of different size. $Scheme_2$ solely depends on the network size. The number of bits required in $Scheme_2$ increases with the network size. Both TBC and SMC depend on the group size and network size. Therefore, the number of bits required increases marginally with network size.

Next, the size of the group is varied from 2.5% to 15% of N. Tables 3.6, 3.7, and 3.8 show the number of bits required when the network size N, is 1024, 4900, and 10000 respectively. It is observed from the Tables 3.6, 3.7, and 3.8 that SMC requires a lesser number of bits in comparison with other schemes for a given group size. This is because, in SMC and TBC the bit-stream is generated from a sparse matrix representing the group membership information. More the sparseness in the matrix, lesser the number of bits generated.

Then, the comparison of the schemes is made with a metric called space saving percentage. We define the space saving percentage of a scheme with respect to a

Table 3.5: Number of bits required to store group membership information in $Scheme_1$, $Scheme_2$, Bloom filter, TBC and SMC.

N	$Scheme_1$	$Scheme_2$	**TBC**	**SMC**	**Bloom Filter with 1% error**	**Bloom Filter with 0.1% error**
1024	2400	1024	778	737	1440	2160
2500	2400	2500	1550	1175	1440	2160
3600	2400	3600	2124	1426	1440	2160
4900	2400	4900	2793	1813	1440	2160
6400	2400	6400	3520	2176	1440	2160
8100	2400	8100	4374	2592	1440	2160
10000	2400	10000	5400	3100	1440	2160

Table 3.6: Number of bits required to store group membership information in $Scheme_1$, $Scheme_2$, Bloom filter, TBC, and SMC for $N = 1024$.

Group Size (% of N)	$Scheme_1$	$Scheme_2$	**TBC**	**SMC**	**Bloom Filter with 1% error**	**Bloom Filter with 0.1% error**
2.5	410	1024	573	359	246	369
5	819	1024	615	461	492	738
7.5	1229	1024	646	533	738	1106
10	1639	1024	686	604	983	1475
15	2458	1024	758	737	1475	2212

Table 3.7: Number of bits required to store group membership information in $Scheme_1$, $Scheme_2$, Bloom filter, TBC, and SMC for $N = 4900$.

Group Size (% of N)	$Scheme_1$	$Scheme_2$	**TBC**	**SMC**	**Bloom Filter with 1% error**	**Bloom Filter with 0.1% error**
2.5	1968	4900	2646	1715	1176	1764
5	3920	4900	2842	2156	2352	3528
7.5	5888	4900	3038	2500	3528	5292
10	7840	4900	3185	2891	4704	7056
15	11760	4900	3577	3528	7056	10584

base scheme as follows:

$$\frac{\text{Number of bits required in the base scheme - Number of bits required by the new scheme}}{\text{Number of bits required in base scheme}} * 100 \tag{3.10}$$

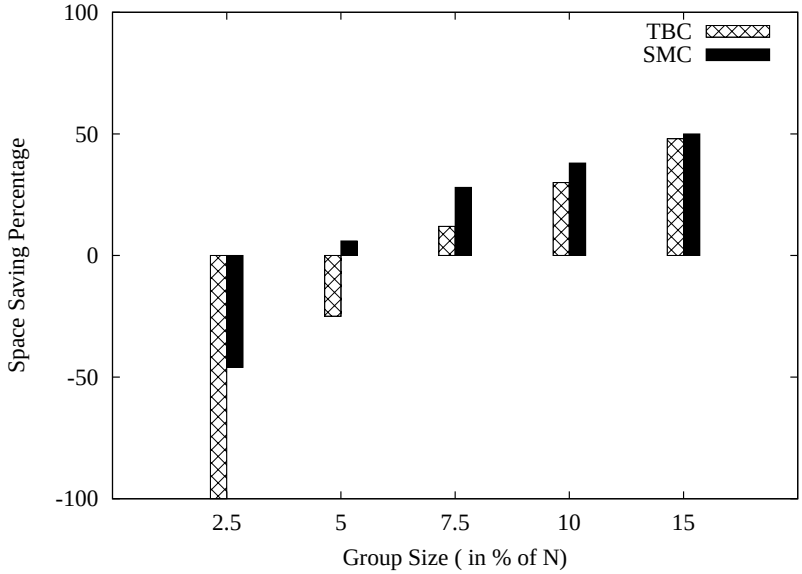
Table 3.8: Number of bits required to store group membership information in $Scheme_1$, $Scheme_2$, Bloom filter, TBC, and SMC for $N = 10000$.

Group Size (% of N)	$Scheme_1$	$Scheme_2$	**TBC**	**SMC**	**Bloom Filter with 1% error**	**Bloom Filter with 0.1% error**
2.5	4000	10000	5400	3400	2400	3600
5	8000	10000	5800	4400	4800	7200
7.5	12000	10000	6100	5200	7200	10800
10	16000	10000	6500	5900	9600	14400
15	24000	10000	7200	7300	14400	21600

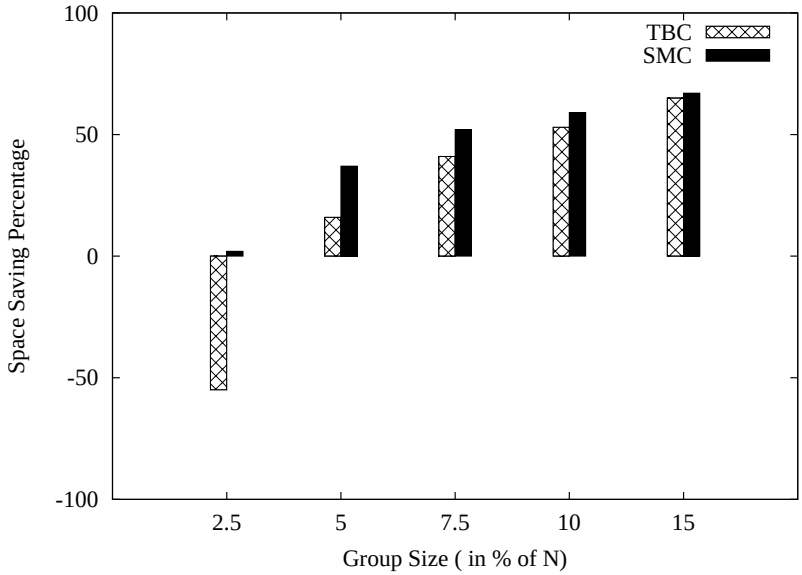
The plot for space saving percentage *vs.* group size of TBC and SMC with respect to Bloom filter is shown in Figure 3.3(a) and 3.3(b) respectively. It is observed from the figures that the space saving percentage is higher in TBC and SMC for larger group size. This is because, the number of bits required to represent group membership information in Bloom filter is directly proportional to the group size.

We have also compared the proposed schemes with Bloom filter for false positive cases. The false positive probability of Bloom filter is computed using Equation 3.1. The false positive probability of TBC and SMC is computed as the sum of the ratio of number of nodes that are falsely detected as a member of the group in each iteration to the total number of nodes in the group divided by the number of iterations. The plot for the false detection probability of Bloom filter *vs.* group size is given in Figure 3.4(a). The value of b is taken to be 750 bits and the number of hash functions k, is set to 5, 10, and 15. It is observed from the figure that the false detection probability increases with the increase in number of hash functions for a fixed value of b. The Figure 3.4(b) shows the plot for false detection probability of Bloom filter *vs.* group size, where the value of b is varied to 250, 500, and 750 bits. The value of k is taken to be 5. It is observed from the figure that the false detection probability of Bloom filter decreases with the increase in the size of b.

Figure 3.5 shows the comparison of SMC with Bloom filter for false positive cases. False detection probability of Bloom filter is plotted varying the value of b to 737, 1175, and 1813 for network size of 1024, 2500, and 4900 respectively. The value of k is considered to be 5. The false detection probability of Bloom filter increases with the increase in group size for a given network size, whereas, SMC do not give rise to false positive cases.



(a) Comparison with Bloom filter (1% error).

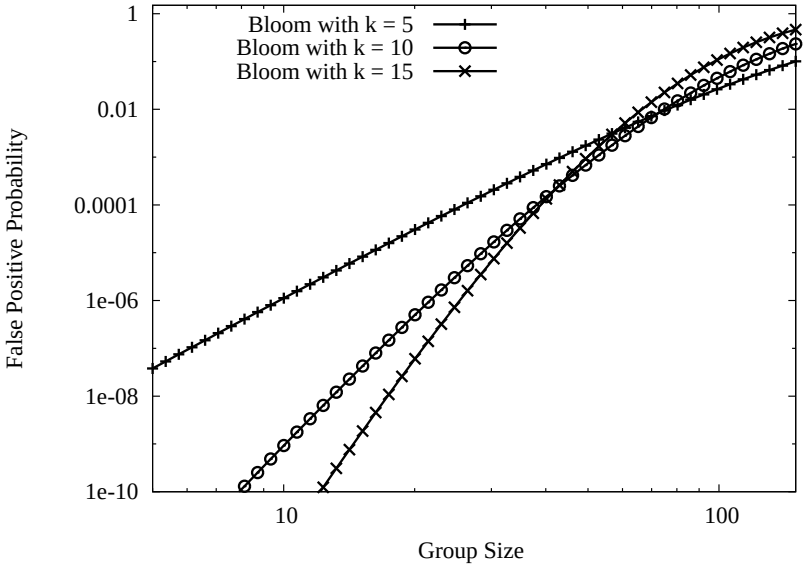


(b) Comparison with Bloom filter (0.1% error).

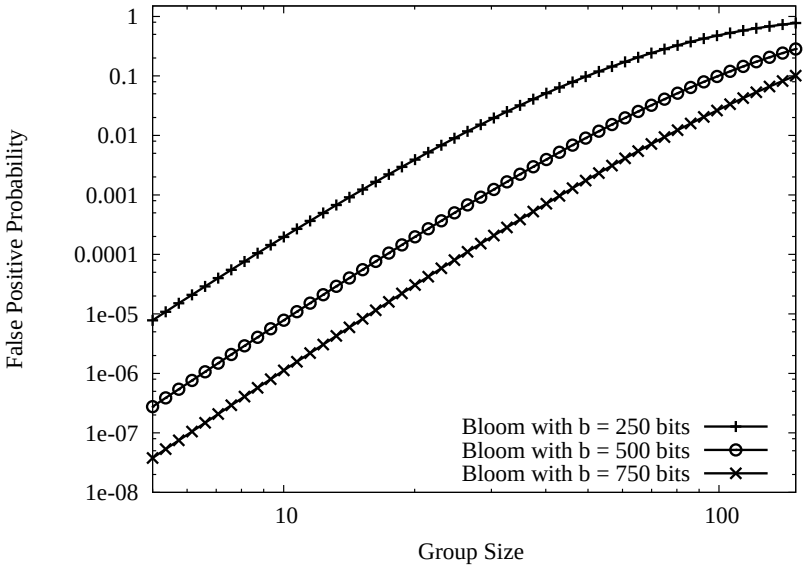
Figure 3.3: Space saving percentage *vs.* Network size in TBC and SMC.

3.4 Summary

In this chapter, various mechanisms for exchanging group membership information among the nodes in WSN are discussed. The strengths and weaknesses in existing mechanisms are identified. Mechanism for exchanging group membership informa-



(a) For different value of k.



(b) For different value of b.

Figure 3.4: False positive probability of Bloom filter *vs.* Group size.

tion should have lower communication and storage overhead. The aforementioned properties are satisfied by the mechanisms that are based on Bloom filter. However, Bloom filter suffers from false positives. In this chapter, we proposed two mechanisms called TBC and SMC. Both TBC and SMC generates a bit-stream from the membership matrix. The encoding and decoding process in each scheme is de-

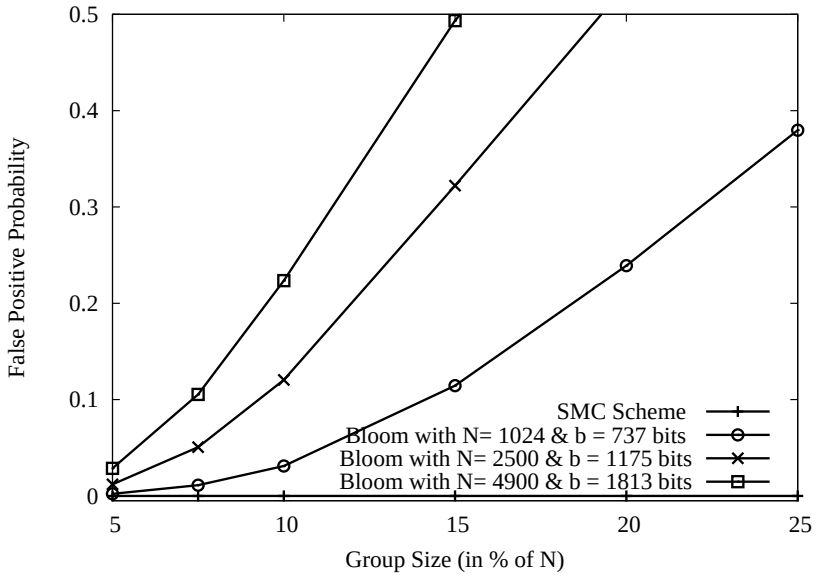


Figure 3.5: False positive probability *vs.* Group size in Bloom filter and SMC.

scribed. We have also shown that the encoding and decoding process in TBC and SMC is unique. TBC and SMC are compared with two trivial schemes and one that uses Bloom filter. The parameters considered for comparison are number of bits required to store group membership information, space saving percentage, and false positive probability. It is found that TBC and SMC do not generate false positive and the space saving percentage is significantly higher compared to Bloom filter. This is because, TBC and SMC are deterministic, whereas Bloom filter determine the membership of a node with some probability. SMC achieves better space saving percentage than TBC; mostly when the group size is relatively small. However, the disadvantage of the proposed scheme is, the space saving percentage is lower, when the group size is large. In this case, the size of the code bit-stream is close to N. The proposed scheme has an additional storage overhead to store the membership bit matrix, but it is less than the storage overhead of other mechanisms that use standard representation to store member IDs.

Next Chapter describes a zone-based replica detection mechanism for WSN. In this scheme, membership information is exchanged among zones to detect replica.

Chapter 4

Zone-Based Node Replica Detection

Most of the node replica detection schemes reported in the literature are location-dependent and probabilistic in nature. It is difficult to guarantee clone-free sensor networks using probabilistic approach. Moreover, location-dependent schemes incur additional memory overhead in storing location-claims.

In this chapter, we have proposed a node replica detection scheme called zone-based node replication detection (ZBNRD). The proposed scheme logically divides the network into number of zones. Each zone has a zone-leader, who is responsible for detecting replica in the network. Replica detection is done at two-levels: *i) intra-zone*, and *ii) inter-zone*. ZBNRD is deterministic and location independent. No memory overhead is associated for storing location information.

4.1 Assumptions

In this section, we describe the assumptions made in the proposed zone-based node replication detection scheme. Table 4.1 shows the notations used in the proposed scheme.

4.1.1 Network Assumptions

The following assumptions are made about sensor networks: *i)* Nodes are connected, static, non-tamper resistant, and are uniformly deployed in the area of observation, *ii)* Communication links are bidirectional, *iii)* There is no centralized trusted entity, *iv)* Nodes are not aware of their position, *i.e,* there is no built-in mechanism to know

Table 4.1: List of notations and symbols.

Symbols	Meaning
N	Size of the network
N_Z	Number of zones in the network
d	Average degree of a node
Z	ID of the zone
n_Z	Average size of zone Z
IDL_Z	ID of the zone-leader of zone Z
ID_i	ID of the node i
$SIG_{SK}(X)$	Digital signature of X signed with secret key SK
$\|\|$	Concatenation operation
H	Cryptographic hash function

the node's physical location, and *v)* Nodes are assigned with a unique ID, prior to their deployment.

4.1.2 Adversary Model

The following assumptions are made about the adversary: *i)* It has the ability to capture any number of sensor nodes, *ii)* Once a node is compromised, the adversary gains full control over the node, *iii)* An adversary can create as many replicas of the captured node as she wishes and deploy in the network, and *iv)* An adversary cannot create a new ID for replica.

4.2 Proposed Scheme

We describe the proposed zone-based node replica detection scheme (ZBNRD) below. In ZBNRD, the network is divided into number of zones similar to SET [25]. However, all members of a zone in ZBNRD may not be within the one-hop neighbor of the zone-leader. Zones are formed dynamically. Identity-based public key crypto-system is used for authenticating messages [48,49]. Each zone in ZBNRD has a zone-leader; whose responsibility is to detect replicas. Zone-leaders are selected apriori before deployment. Each node in the network belongs to exactly one zone. A zone-leader maintains the list of all members in its zone, and the list of zone-leaders

present in the network. ZBNRD operates in two phases: *i) Zone Registration*, and *ii) Replica Detection*. We illustrate below the actions performed in each phase.

4.2.1 Zone Registration

Zone registration phase is initiated as soon as sensor nodes are deployed in the target area. In this phase, nodes register themselves with a zone-leader. Zone registration begins with the broadcast of a zone registration message ZONE_REGD by the zone-leaders. The format of ZONE_REGD message is $< Z$, IDL_Z, $SIG_{SK_{IDL_Z}}$ $(H((Z||IDL_Z)) >$, where Z and IDL_Z are the IDs of a zone and its zone-leader respectively. ZONE_REGD message is an invitation to nodes by a zone-leader to become a member of its zone. Figure 4.1, and 4.2 shows the zone registration process, where the dotted lines indicate the broadcast of ZONE_REGD message, and the solid lines indicate the connectivity. In the above figures, node A, B, C, D and E are the zone-leaders. Figure 4.1 shows the broadcast of ZONE_REGD message to one-hop neighbors, and Figure 4.2 shows the broadcast of ZONE_REGD message by one-hop neighbors of zone-leaders. A node may receive a ZONE_REGD

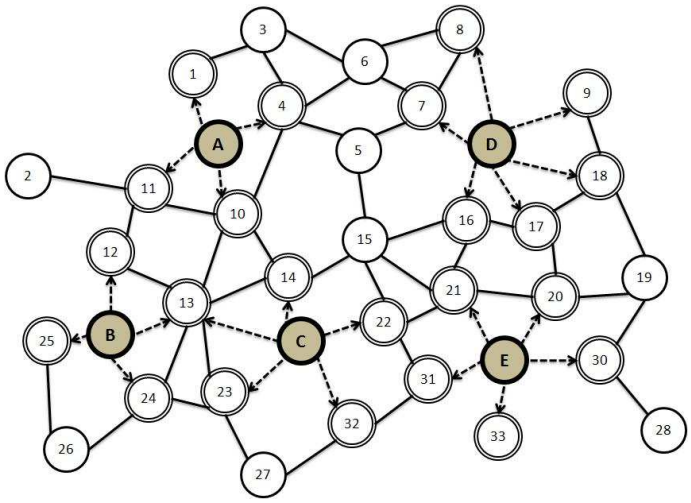


Figure 4.1: ZONE_REGD message broadcast to one-hop neighbors.

message from more than one zone-leaders. However, the node will register to only a single zone whose zone-leader is closer to it. A node becomes the member of a zone by sending a zone join message ZONE_JOIN to the corresponding zone-leader. The format of ZONE_JOIN message is $< ID_m$, IDL_Z, SIG_{SK_m} $(H(ID_m||IDL_Z)) >$, where ID_m is the ID of joining node. A node on receiving the first ZONE_REGD message broadcast it to all its neighbors, and discards the subsequent ZONE_REGD

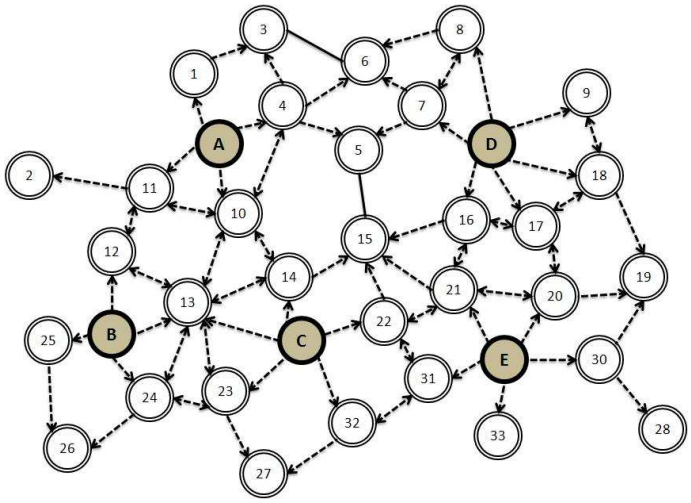


Figure 4.2: ZONE_REGD message broadcast to two-hop neighbors.

messages from the same zone. However, a node broadcast a ZONE_REGD message with a probability, p_{regd}, if the message is from a zone which is different from the zones it has already received. When a ZONE_JOIN message arrives at a zone-leader, it verifies the existence and authenticity of the requested node before adding it to its membership list. ZONE_JOIN reply message from one-hop and two-hop neighbors of zone-leaders is shown in Figure 4.3 and 4.4 respectively. When a zone-leader receives a ZONE_REGD message from another zone-leader, it updates the routing path to it. A case of zone formation at the end of zone registration phase is shown in Figure 4.5.

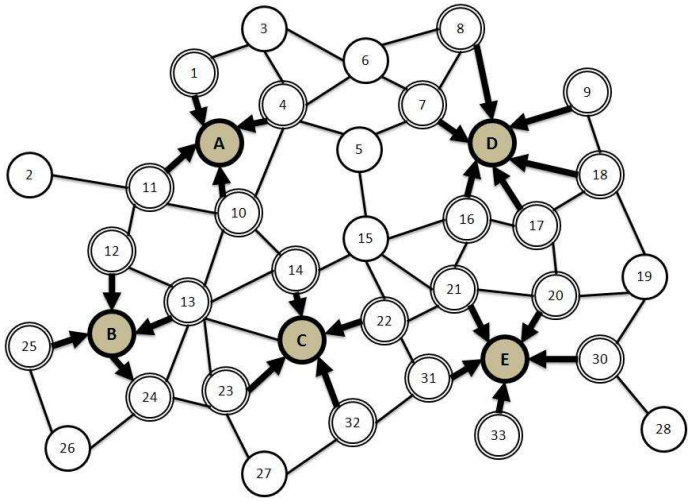


Figure 4.3: ZONE_JOIN reply message from one-hop neighbors.

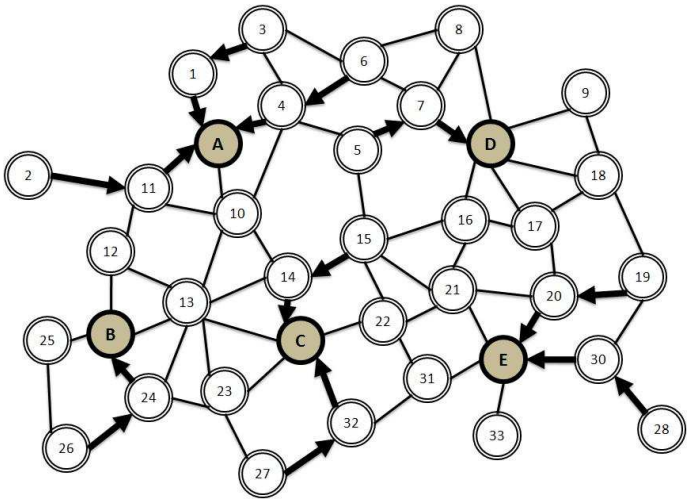


Figure 4.4: ZONE_JOIN reply message from two-hop neighbors.

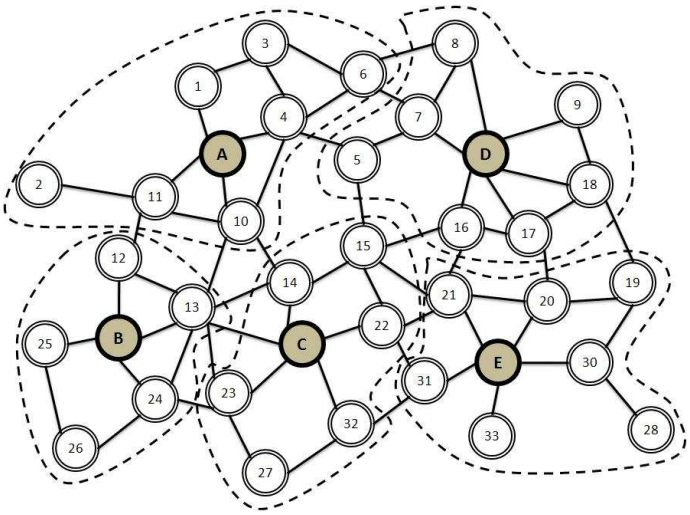


Figure 4.5: A case of zone formation.

At the end of zone registration phase, zone-leaders share their membership list with each other as shown in Figure 4.6. To reduce the message and communication overhead between zone-leaders, we use the SMC scheme as described in Chapter 3. A zone-leader on receiving membership list from other zones, verify for the existence of a replica. The existence of a node's ID in two different zones lead to a conflict. A zone-leader on detecting a conflict initiates a revocation message NODE_REVOKE, which is broadcasted to all its members and zone-leaders. This process will detect replicas that are deployed during the registration phase.

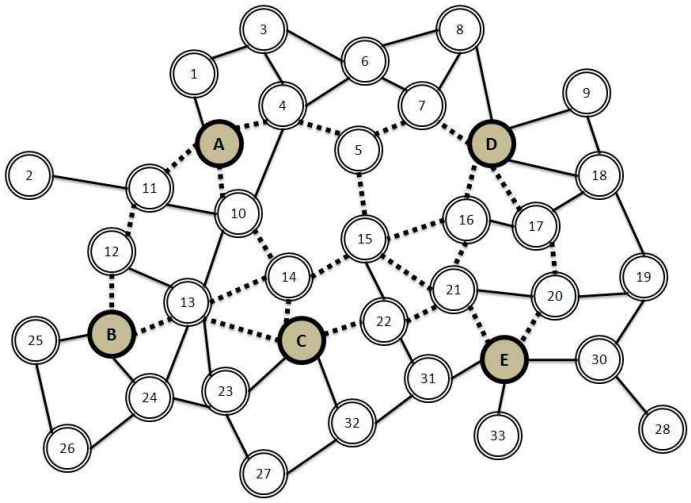


Figure 4.6: Zone membership list sharing path of zone-leaders.

4.2.2 Replica Detection

This section deals with replica detection. An adversary may deploy a replica in the same zone to which the original node belongs or in a different zone. Depending on the location of deployed replica, there are two possible cases of detection: *i) Intra-zone detection:* Node and its replica are members of the same zone, and *ii) Inter-zone detection:* Node and its replica are members of different zones. Both the detection process is explained below.

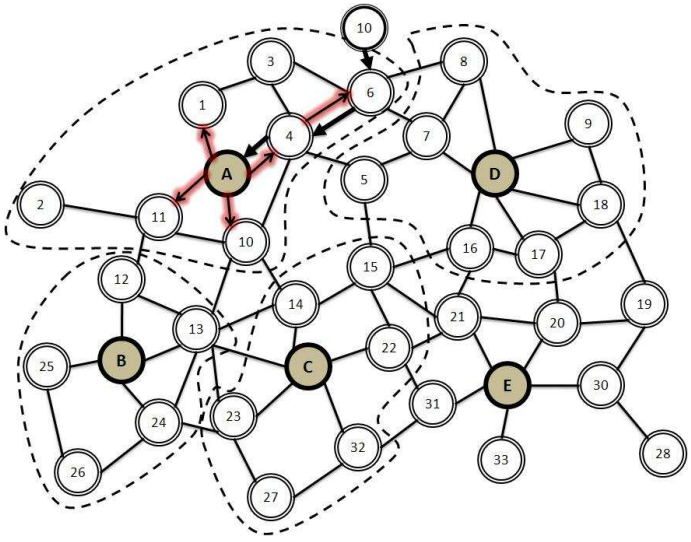


Figure 4.7: Intra-zone replica detection.

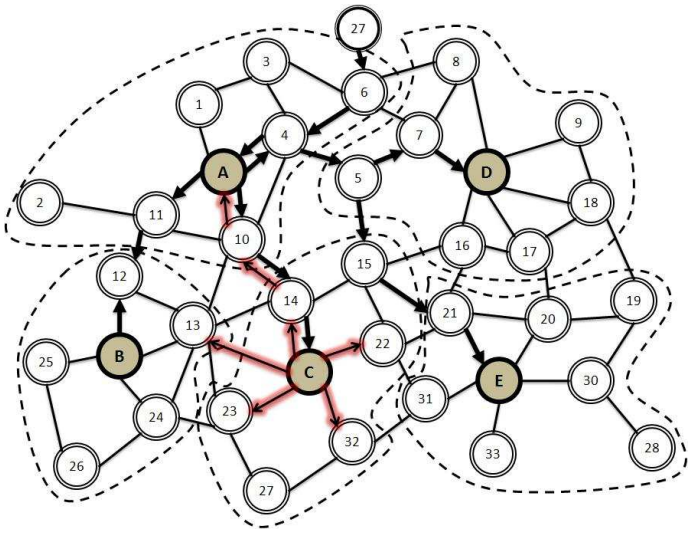


Figure 4.8: Inter-zone replica detection.

i) Intra-zone detection: When a zone-leader receives a ZONE_JOIN message, it checks for the existence of the node in its membership list, and broadcast a revocation message if the node is already a member of its zone. Otherwise, *inter-zone detection* is initiated. Figure 4.7 shows a scenario for *intra-zone detection*. Zone-leader A, on receiving ZONE_JOIN message from replica node 10, checks its membership list. Since, node 10 is already present in its membership list, zone-leader detects node 10 to be a replica. Then, it broadcast a ZONE_REVOKE message for node 10. Finally, node 10 is removed from the network.

ii) Inter-zone detection: This is initiated, when a node that has sent ZONE_JOIN message is not present in the current membership list of the zone-leader. Zone-leader sends a NEW_JOIN message to other zone-leaders and starts a timer. The message format of NEW_JOIN message is $< Z$, ID_{new}, $SIG_{SK_{IDL_Z}}$ $(H(Z||ID_{new}))$ $>$, where ID_{new} is the ID of the node that has newly joined after registration phase. Zone-leaders on receiving NEW_JOIN message verify the existence of the node with ID, ID_{new}, in their membership list. If a conflict is detected, then the zone-leader sends a ZONE_REVOKE message for the new node, ID_{new}, to all zone-leaders and to its own zone members. When the zone-leader that initiated the NEW_JOIN message receives a revocation message, it broadcasts to its members. The node ID_{new} is then revoked from the network. A zone-leader accepts the new node as

its member only when it does not receive a revocation message from any of the other zone-leaders before the timer expires. Figure 4.8 shows the case for inter-zone detection.

In case a zone-leader fails, then the nodes of the failed zone re-join to one of their nearest zone.

4.3 Analysis

In this section, we analyze the performance of ZBNRD.

4.3.1 Connectivity

We have assumed that the sensor network is connected. Therefore, a node in the network will receive a zone registration message from at least one zone-leader. We claim the following:

Claim 4.1. *A node in the network belongs to exactly one zone.*

Proof. We prove our claim by means of contradiction. Let us assume that there exists at least one node say, k, that does not belong to exactly one zone. In this scenario, there will be two cases: *(i)* the node k belongs to more than one zone, or *(ii)* k is not a member of any zone.

Case 1: In our proposed scheme, a node may receive ZONE_REGD message from more than one zone, but responds to only one. Node k, is a member of more than one zone implies that node k has responded to more than one ZONE_REGD message. This violates our proposed protocol that a node replies to only one ZONE_REGD message. Hence, this contradicts our assumption.

Case 2: Node k, is not a member of any zone; this implies that the node k, has not received a ZONE_REGD message from any zone-leaders. This is possible if k is not reachable from any zone-leader. But we have assumed that the network is connected. Therefore, node k, is reachable from every other node in the network. Further, by setting the value of $p_{regd} > \frac{1}{n_Z}$, we can ensure that each node broadcast at least two zone registration message to its neighbors. If the node k, has not received at least one ZONE_REGD message from zone-leaders, this implies that node k, is not reachable which a contradiction to our assumption that the network is connected. Therefore, k is a member of only one zone. □

4.3.2 Security Analysis

We evaluate, the security of ZBNRD based on the following parameters.

i) Replica detection: Assume that node i belongs to a zone Z, and an adversary has deployed a replica i_c, in a zone $Z^{'}$. When the replica i_c, send a ZONE_JOIN message to the zone-leader of $Z^{'}$, it verifies the existence of the replica i_c, by sending a NEW_JOIN message to the zone-leader of Z. The zone-leader of Z on detecting a replica sends a ZONE_REVOKE message to all zone-leaders and its members. Thus, a replica is detected and revoked from the network in ZBNRD.

ii) Secured communication between zone-leaders: Zone-leaders encrypt and digitally sign every message shared among themselves with their private key. Thus, an adversary must have the private key to read and modify the message. Obtaining the private key is a difficult job as keys are assigned at the time of deployment. Therefore, the communication between the zone-leaders are secured.

Table 4.2: Communication and Storage Overhead Comparison.

Schemes	**Communication**	**Memory**	**Location Dependent**
Centralized Detection	$O(N.\sqrt{N})$	$O(d)$	yes
Node-to-Network Broadcasting	$O(N^2)$	$O(d)$	yes
Deterministic Multicast	$O(\frac{N.w.\ln w\sqrt{N}}{d})$	$O(w)$	yes
Randomized Multicast [36]	$O(N^2)$	$O(\sqrt{N})$	yes
Line-Selected Multicast [36]	$O(N.\sqrt{N})$	$O(\sqrt{N})$	yes
RED [32]	$O(d.p.\sqrt{N})$	$O(d.p)$	yes
SDC [22]	$O(d.p.\sqrt{N}) + O(s)$	$O(w)$	yes
P-MPC [22]	$O(d.p.\sqrt{N}) + O(s)$	$O(d.p)$	yes
SET [25]	$O(N)$	N/A	no
NBDS [41]	$O(d.p.\sqrt{N})$	$O(d.p)$	no
Znaidi *et al.* [23]	$O(t^2)$	$O(t)$	no
ZBNRD	$O(N.\sqrt{n_Z}) + O(N_z.\sqrt{N})$	$O(d)/O(n_Z)$	no

4.3.3 Communication Overhead

There are two types of communication cost associated with ZBNRD: *i)* During registration phase, and *ii)* During replica detection.

i) Communication cost during registration phase: The communication cost associated during the registration phase is equal to the sum of the costs associated with ZONE_REGD message, and ZONE_JOIN reply message. Each node broadcasts the first ZONE_REGD message with a probability of *one* and the subsequent messages from different zones with a probability of p_{regd}. The average path length in a network of N nodes is given by $O(\sqrt{N})$ [36]. Therefore, average communication cost to broadcast ZONE_REGD message is $O(d.p_{regd}.n_Z.\sqrt{n_Z})$. The average communication cost associated with ZONE_JOIN reply message is $O(N.\sqrt{n_Z})$. Hence, the average communication cost during registration phase is $O(d.p_{regd}.n_Z.\sqrt{n_Z}) + O(N.\sqrt{n_Z}) = O(N.\sqrt{n_Z})$.

ii) Communication cost for replica detection: For *intra-zone detection*, neighbors communicate with their zone-leaders. Therefore, the average communication cost for *intra-zone detection* is $O(d.\sqrt{n_Z})$. For *inter-zone detection*, a zone-leader communicates with other zone-leaders. Therefore, the average communication cost for *inter-zone detection* is given by $O(N_z.\sqrt{N})$, where N_z is the number of zones in the network. Total communication cost in detecting a replica is $max\{O(d.\sqrt{n_Z}), O(N_z.\sqrt{N})\}$, which is equal to $O(N_z.\sqrt{N})$.

The overall communication cost of ZBNRD is $O(N.\sqrt{n_Z}) + O(N_z.\sqrt{N})$. Average size of a zone is given by $\frac{N}{N_z}$, where $N_z << N$. Therefore, the overall communication cost of ZBNRD is $O(N\sqrt{N})$.

4.3.4 Storage Overhead

In ZBNRD, a zone-leader maintains the list of all its members and additional information regarding other zone-leaders. The memory overhead associated with a zone-leader is $O(N_z + n_Z)$. We have assumed that $N_z \leq n_Z$. Therefore, the memory overhead of a zone-leader is $O(n_Z)$. A node in the network maintains only the information about its neighbors and zone-leader. Therefore, memory overhead for each node is given by $O(d)$.

Table 4.2 shows the comparison of average communication and storage cost of ZBNRD with other existing schemes. Here w is the number of witness nodes, and t is the number of cluster heads defined for network.

Table 4.3: Simulation parameters.

Parameter	**Value**
Area	1000 x 1000 m^2
Network size	1000-10000
Deployment type	Uniformly random
Topology type	Random topology
Communication range	55 meter
Maximum network packet size	20 bytes
Average number of neighbors	40
Number of runs	100
Simulation Time	200 sec

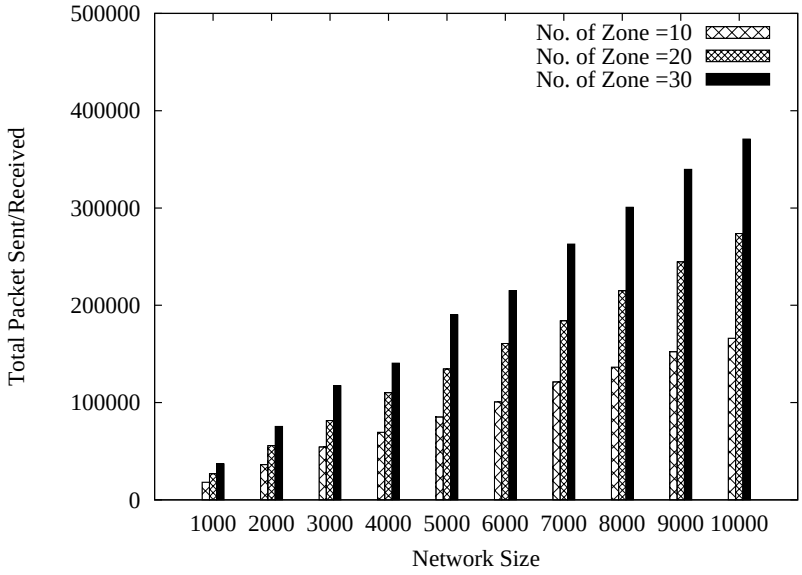


Figure 4.9: Total packets sent/received for detecting replica in the network *vs.* network size in ZBNRD.

4.4 Simulation Results

We performed simulation using Castalia 2-3b [54] that runs on Omnet++ [55]. Parameters considered for simulation are shown in Table 4.3. Replicas are randomly deployed. Simulation was performed varying the network size from 1000 to 10000

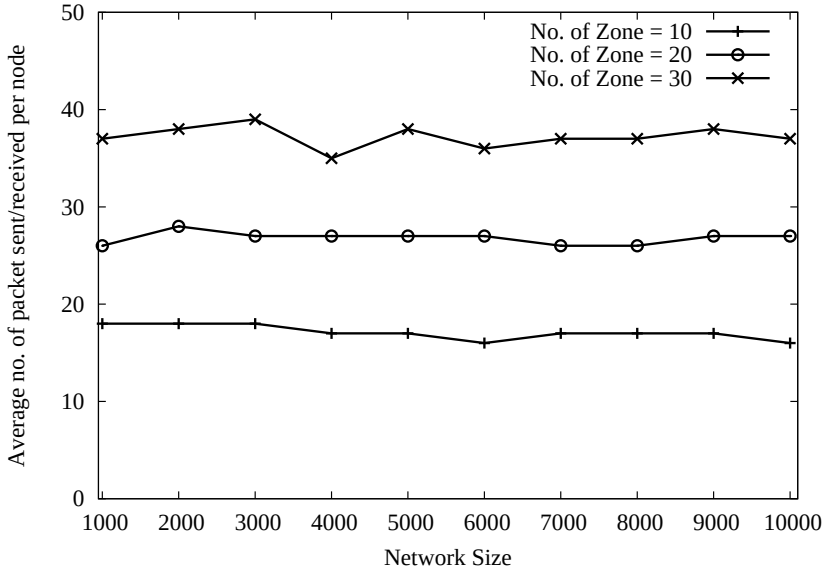


Figure 4.10: Average number of packets sent/received per node for detecting replica *vs.* network size in ZBNRD.

and number of zones in the network to 10, 20 and 30.

The plot for total packets sent/received in detecting a replica is shown in Figure 4.9. It is observed from the figure that the average number of packets sent/received in the network increases with the increase in network size. This is an expected result. It is also observed from the figure that the number of packets sent/received in the network increases with the increase in the number of zones. This is because, with increase in the number of zones, number of zone-leaders also increases. As a result, the number of ZONE_REGD message increases giving rise to the increase in number of packets sent/received in the network.

In Figure 4.10, we plot the average number of packets sent/received per node *vs.* network size. From the figure, it is observed that the number of packets sent/received per node in detecting replica remains almost constant irrespective of the network size. It is also observed from the figure that, for the same network size, the average number of packets sent/received per node increases with the increase in the number of zones in the network. This is because, the number of messages exchanged per node for zone formation is independent of network size. Therefore, the average number of packets sent/received per node remains almost constant with variation in the network size for a fixed number of zones. However, when the number of zones increases, the number of messages required for zone for-

mation also increases. Therefore, the average number of packets sent/received per node increases with the number of zones.

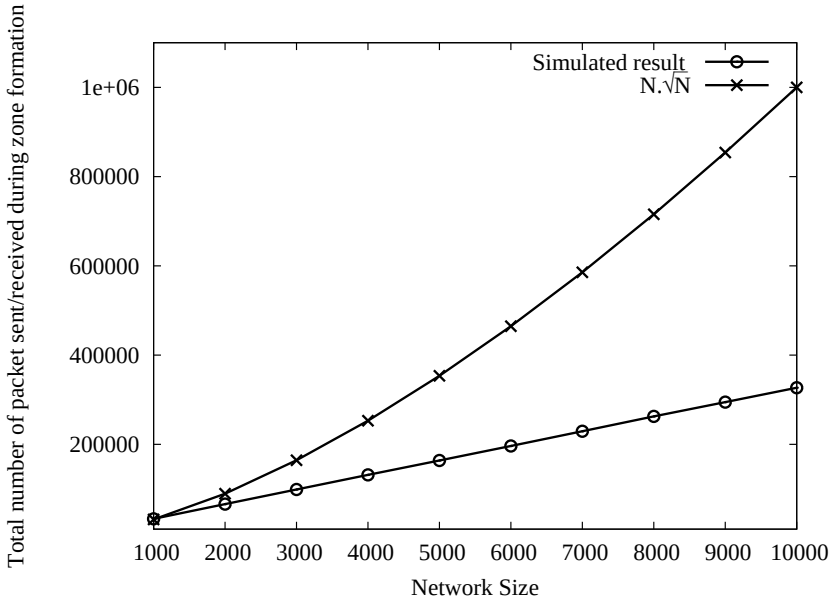


Figure 4.11: Total packets sent/received during zone formation *vs.* network size.

In Figure 4.11, we made a comparison between theoretical value and the simulated one, for the number of packets sent/received during zone formation. Values obtained in the simulation are much lower in comparison to theoretical communication cost.

The plot for maximum path length *vs.* network size varying the number of zones in the network is shown in Figure 4.12. As the number of zone increases there is a marginal decrease in the path length for a network of equal size, as well as with the increase in network size. This is attributed to the increase in the number of zones in the network.

Figure 4.13 shows the plot for time to detect first replica *vs.* network size varying the number of zones. It is observed from the figure that there is a marginal increase in the time to detect the first replica as the network size increases. This is because, in the proposed scheme, the replica detection takes place after zone formation. With increase in the network size, the time for zone formation also increases. As a result, the time to detect replica also increases.

Next, we compared the ZBNRD with a few existing location-dependent and location-independent schemes. The comparison of ZBNRD with location-dependent and location-independent schemes are discussed in sub-section 4.4.1 and 4.4.2 re-

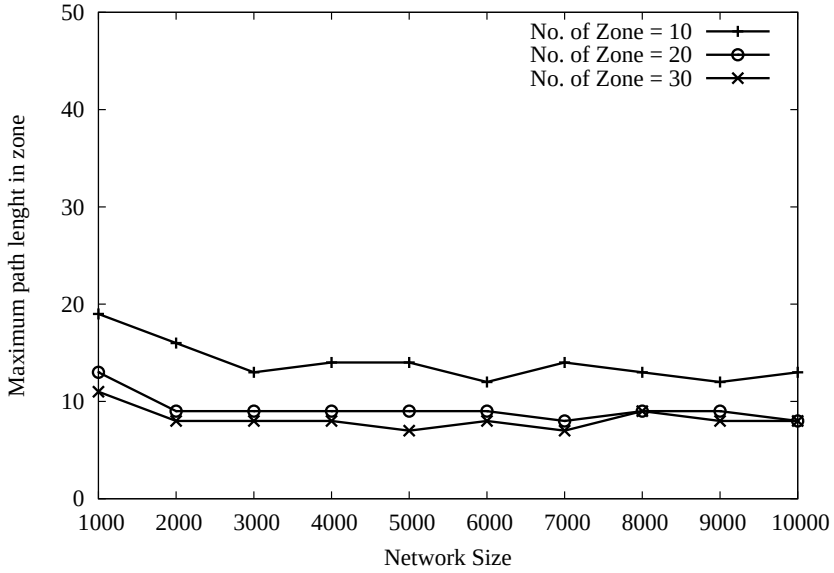


Figure 4.12: Maximum path length within a zone *vs.* network size for different zone size.

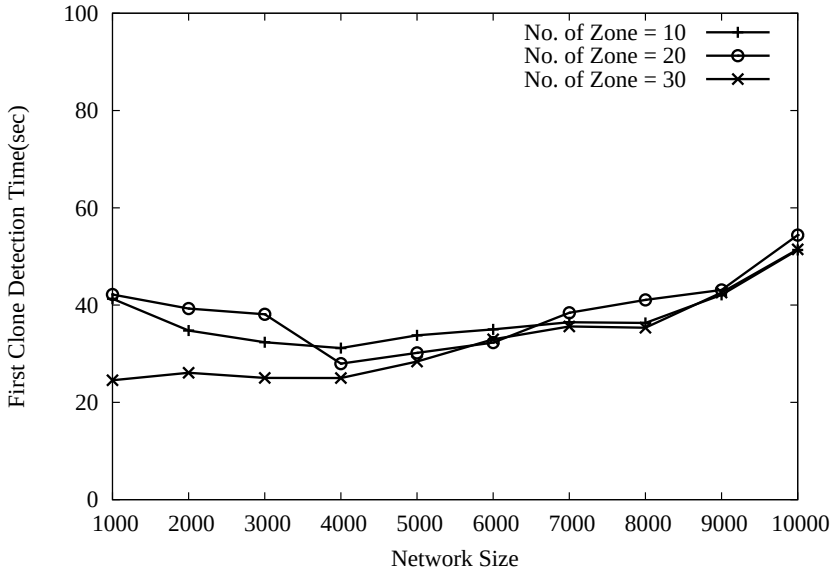


Figure 4.13: Time to detect first replica *vs.* network size for different zone size.

spectively.

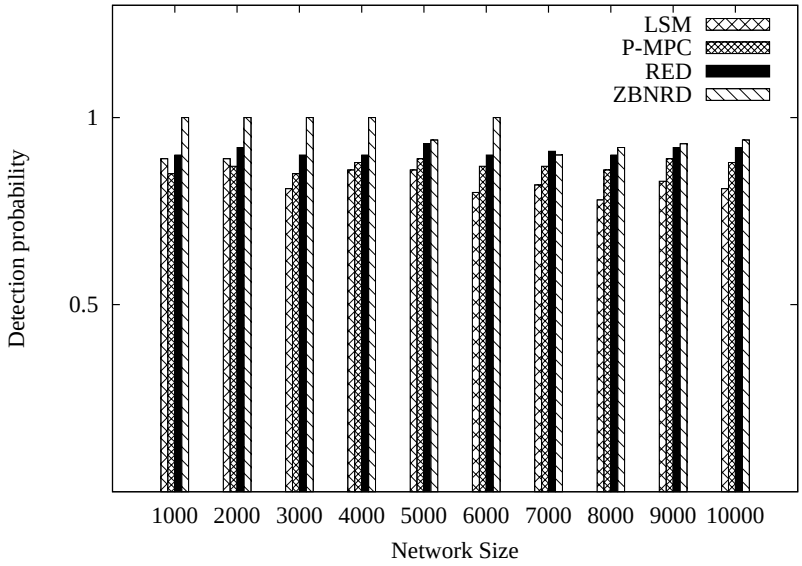


Figure 4.14: Comparison of ZBNRD with location-dependent schemes for detection probability *vs.* network size.

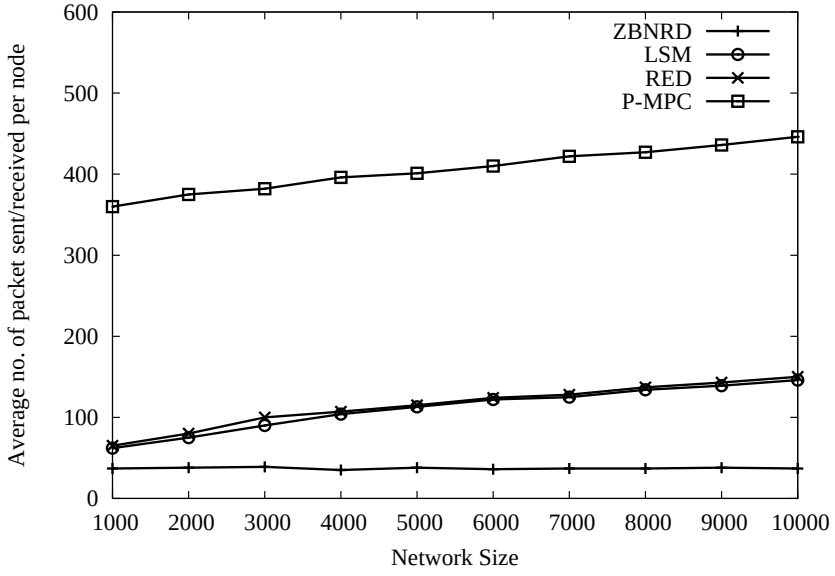


Figure 4.15: Comparison of ZBNRD with location-dependent schemes for average number of packets sent/received per node *vs.* network size.

4.4.1 Comparison of ZBNRD with location-dependent schemes

In this section, we have compared ZBNRD with location-dependent schemes such as LSM [36], P-MPC [22], and RED [32]. Comparison for detection probability

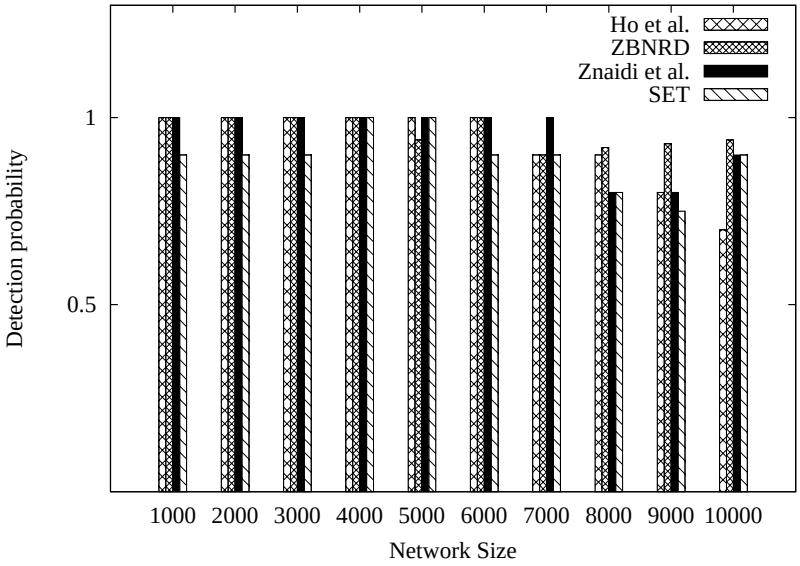


Figure 4.16: Comparison of ZBNRD with location-independent schemes for detection probability *vs.* network size.

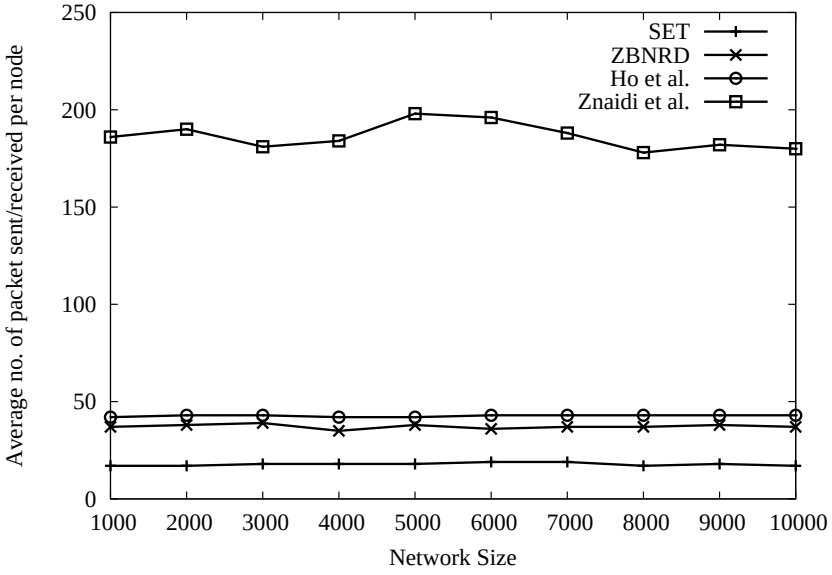


Figure 4.17: Comparison of ZBNRD with location-independent schemes for average number of packets sent/received per node *vs.* network size.

vs. network size is shown in Figure 4.14 and for the average number of packets sent/received *vs.* network size is shown in Figure 4.15. From Figure 4.14, it is observed that ZBNRD has a higher detection probability in comparison to LSM, P-

MPC, and RED. This is because ZBNRD is deterministic, whereas LSM, P-MPC, and RED are probabilistic in nature. There is a marginal decrease in detection probability at higher network size. This is attributed to communication congestion in dense networks. From Figure 4.15, it is observed that ZBNRD has a lower number of packets sent/received per node than other location dependent schemes. This is because, in ZBNRD, each node joins in exactly *one* zone. The number of ZONE_JOIN message sent by a node is only *one*. In other schemes the location claim of a node is send to a number of witness nodes in the network.

4.4.2 Comparison of ZBNRD with location-independent schemes

In this sub-section, we compared ZBNRD with location-independent schemes such as SET [25], Ho *et al.* [41] and Znaidi *et al.* [23]. The plot for detection probability *vs.* network size is shown in Figure 4.16. It is observed from the figure that the detection probability of ZBNRD is almost identical with other location independent schemes. However, there is a marginal drop in detection probability at higher network size. This is due to congestion at higher node density. Figure 4.17 shows the plot for the average number of packets sent/received per node *vs.* network size. It is observed from the figure that ZBNRD has lower number of packets sent/received per node in comparison with [41] and [23], and higher in comparison with SET. This is because, the communication complexity in ZBNRD is lower, compared with [41] and [23], and higher than SET. This is shown in Table 4.2.

4.5 Summary

In this chapter, we proposed a zone-based node replica detection scheme (ZBNRD) for WSN. In ZBNRD, network is divided into number of zones. Each zone has a zone-leader, who is responsible for detecting replicas in the network. ZBNRD operates in two phases: *i) Zone Registration*, and *ii) Replica Detection*. In the zone registration phase, nodes join to one of the zone. A node becomes a member of exactly one zone. At the end of zone registration phase, zone-leaders exchange their membership list. At this stage, replicas deployed during the registration phase are detected. We have compared ZBNRD with a few existing schemes. It is observed that ZBNRD have higher detection probability than other schemes. This is due to the deterministic nature of the ZBNRD, whereas other schemes are probabilistic in nature. For zones of equal size and uniformly distributed zone-leaders, ZBNRD has lower communication overhead. However, for large number of zones, as well as dense

networks, the communication overhead increases. It is also observed that for smaller number of zones, the average path length is higher. This increases end-to-end delay between nodes.

In the next chapter, a replica detection scheme based on node coloring technique is proposed.

Chapter 5

Node Coloring Based Replica Detection

In this chapter, we propose a distributed, location-independent replica detection scheme called Node Coloring Based Replica Detection (NCBRD). In this scheme, each node is assigned with a color (value), which is unique within its neighborhood. Color conflict within the neighborhood is detected as replica. Since, NCBRD is location-independent, it does not incur the additional overhead of sending, receiving, and verifying the location-claims. In addition to improving the detection probability, NCBRD also attempts to minimize the communication overhead.

5.1 Assumptions

This section describes the assumptions made about the network and adversaries.

5.1.1 Network Assumptions

The following assumptions are made about the network: *i)* Nodes are static and uniformly deployed, *ii)* Connectivity is symmetric, *iii)* No centralized trusted infrastructure, *iv)* Nodes are assigned with a predefined color set prior to their deployment, *v)* Identity-based crypto-system is used for secured communication between neighboring nodes, and *vi)* Nodes are equipped with omni-directional antenna.

5.1.2 Assumptions about an Adversary

The following assumptions are made about an adversary: *i)* An adversary has the ability to compromise a subset of nodes, *ii)* She can modify the color assigned to the

captured node, *iii)* Adversary can create as many replica of the captured node as she wishes, and *v)* Adversary does not deploy the replica within the neighborhood of a captured node as this will not lead to any significant benefit for the adversary.

Table 5.1: List of Notations used in NCBRD.

Notation	Meaning
N	Size of the network
CLR	Set of available colors to a node
NBD_i	Set consisting of IDs' of neighboring nodes of node i
$Color_i$	Color of a node with ID i
$Initiator_i$	Initiator of node i
d	Average number of neighbors of a node
$SIG_k(X)$	Signature of X with key, k
SK_i	Secret key of node i

5.2 Node Coloring Based Replica Detection

In this section, we describe the proposed *Node Coloring Based Replica Detection* (NCBRD) technique. Notations used in NCBRD are summarized in Table 5.1. In NCBRD, a color (value) is assigned to each node. Color assignment process ensures that the color assigned to a node is unique within it's neighborhood. A color conflict within the neighborhood of a node is detected as a replica. Color conflict occurs, when either of the following conditions are satisfied: *i)* Two or more neighbors of a node including the node itself claim the same color, and *ii)* The color as claimed by the neighbor of a node exists in the color set of the node, *i.e.*, the node has not assigned the claimed color to any of its neighbor.

Node coloring process is described in sub-section 5.2.1, and replica detection in sub-section 5.2.2.

5.2.1 Node Coloring

In this sub-section, we explain the process of color assignment to a node. Color assignment process is distributive and concurrent in nature. Each node uses color from a finite set of colors, CLR, where $|CLR|$ defines the upper bound on the

number of trusted neighbors a node can have in the network. Node coloring process is initiated by an *Initiator*, which is defined as follows: *Initiator* is a node, which assigns color to its neighboring nodes during the coloring process. It is not a special node. Any node in the network can act as an *Initiator*, provided it has an assigned color. We use the term *Initiator* to distinguish between the color assigner and assignee during the coloring process. We call, the assigner as the *Initiator*. Prior to the deployment, a set of nodes are randomly chosen to act as *Initiator*. These nodes start the color assignment process within the network. A node, which obtains color during the color assignment process, acts as the *Initiator*. Only the *Initiator* assigns color to its neighboring nodes in the proposed scheme. Once an *Initiator* assigns a color, it removes that color from its color set CLR. The color assigning process of an *Initiator* is completed when all its neighbors are colored. The coloring process in the network is completed when the color assigning process of all nodes in the network is over. Nodes are colored only once in their lifetime. A node is assigned with a color during the coloring process. A node that is assigned with a color, will not be assigned with any other color during the coloring process. However, an adversary may be powerful enough to modify the assigned color of a captured node. During the coloring process, a node remains in one of the following three states:

i) *UnColored*: Nodes that are not assigned with any color are in *UnColored* state. At the time of deployment, all nodes other than the predefined *Initiators* are in *UnColored* state.

ii) *Coloring*: A node is in *Coloring* state, when it is in the process of acquiring color from an *Initiator* node.

iii) *Colored*: A node after obtaining color, changes its state to *Colored*. In this state, the node will act as an *Initiator*.

A node during its lifetime changes its state from: *UnColored* $\rightarrow$ *Coloring* $\rightarrow$ *Colored*. The *Initiator* assigns color to only those neighboring nodes that are in *Uncolored* state. During the coloring process, the *Initiator* locally broadcasts a color request message. Let A be the *Initiator*. The color request message broadcast by node A is shown below

$$A \longrightarrow * :< ColReq, A, R_A, SIG_{SK_A}(A|R_A) >$$

where *ColReq* indicates the message type, A is the ID of the initiator node, R_A is a nonce chosen by the *Initiator*, and $SIG_{SK_A}(A|R_A)$ is the signature of the *Initiator*,

signed using its secret key. We refer to the above message type in the remaining text as *ColReq*.

Neighboring nodes of the *Initiator* on receiving a *ColReq* message respond with a color reply message, depending on their states. When an *Uncolored* node receives multiple *ColReq* messages from its neighboring *Initiator*s, it selects an *Initiator* on first-come-first-serve basis and participates in the coloring process. There are three types of color reply message. Let a node B be the neighbor of *Initiator* A. Then, node B responds with one of the following color reply message, depending on its state:

i) Node B in *Colored* state: Node B responds with the following color reply message

$$B \longrightarrow A :< ColRep, B, Color_B, Initiator_B, R_B, \\ SIG_{SK_B}(B|Color_B|Initiator_B|R_A) >$$

where *ColRep* represents the message type, B is the ID of the replying node, $Color_B$ is the color of the replying node, $Initiator_B$ is the ID of the *Initiator* of replying node, R_B is a nonce chosen by the replying node, and SIG_{SK_B} $(B|Color_B|\ Initiator_B|R_A)$ is a signature signed using replying node's secret key. We refer the above message type in the remaining of the text as *ColRep(*$Color_B$*,* $Initiator_B$*)*.

ii) Node B in *Uncolored* state: The color reply message from node, B is of the following type

$$B \longrightarrow A :< ColRep, B, Null, ColKey, R_B, SIG_{SK_B}(B|ColKey|R_A) >$$

where $Null$ indicates that no color is assigned to the replying node, $ColKey$ is the key value assigned to the replying node prior to its deployment, R_B is a nonce chosen by the replying node, and $SIG_{SK_B}(B|ColKey|R_A)$ is a signature signed using replying node's secret key. We refer the above message type in the remaining text as *ColRep(Null)*.

iii) Node B in *Coloring* state: In this state, the replying node B is in the process of acquiring its color from another *Initiator* other than the *Initiator*, A. In this scenario, node B responds with the following color reply message

$$B \longrightarrow A :< ColRep, B, Hold, SIG_{SK_B}(B|R_A) >$$

where $Hold$ indicates that the replying node B is in the process of acquiring color from another *Initiator*. We refer to the above color reply message in the remaining text as *ColRep(Hold)*.

The *Initiator* on receiving color reply message, from its neighbors in *Colored* state verifies the authenticity of colors assigned to them. For color verification, the *Initiator* broadcasts a color verification message within its two-hop neighbor. For example, the *Initiator* A, on receiving color reply message from a neighboring node, say B, in *Colored* state broadcasts the following color verification message within its two-hop neighbor:

$$A \longrightarrow * :< ColVer, A, B, Color_B, Initiator_B, R_A, \\ SIG_{SK_A}(A|B|Color_B|Initiator_B|R_A) >$$

where *ColVer* represents the message type, $Color_B$ and $Initiator_B$ are the color and *Initiator*'s ID of the node under verification, R_A is a nonce chosen by the *Initiator*, and $SIG_{SK_A}(A|B|Color_B|\ Initiator_B|R_A)$ is a signature signed using *Initiator*'s secret key.

On receiving color verification message, nodes within two-hop neighbor of the *Initiator* A, verify the assigned color to node B, *i.e.*, $Color_B$, as well as its initiator, *i.e.*, $Initiator_B$. On successful verification, the *Initiator* of node B sends a color acknowledgement message, *ColAck*, to node A. If, a node within two-hop neighbor of the *Initiator* A, detects a conflict either with the assigned color to node B and/or with its initiator, then it broadcasts a revocation message for B. Thereafter node B is revoked from the network. The node revoked from the network is no more considered as the neighbor of any node.

After receiving *ColAck* message for all neighboring nodes in *Colored* state, the *Initiator* starts coloring its neighbors. First, it checks whether two or more *Colored* neighbors are sharing the same color. If so, then one of them is randomly selected and recorded as neighbor in the neighbor table. Then, the remaining colored neighbors sharing the same color are logically disconnected. After recording the nodes in the *Colored* state, the nodes in *UnColored* state are assigned with a color such that it is unique within one-hop neighbor of the *Initiator*. On completion of color assignment process, the *Initiator* sends a color assigned message to each of its neighbor. The color assigned message is of two types. They are:

i) If the *Initiator*, for example A, assigned a color, say $color_B$ to node B, then

the color assigned message is of the following type:

$$A \longrightarrow B :< ColAsg, A, Color_B, SIG_{SK_A}(A|Color_B|R_B) >$$

We refer to the above type of color assigned message in the remaining text as *ColAsg(Color$_B$)*. The node B on receiving *ColAsg(Color$_B$)* message from A updates its neighbor table. Node A becomes the *Colored* neighbor of B.

ii) If the *Initiator*, for example A could not assign a color to an *Uncolored* node B, then the color assigned message from A to B is of the following type:

$$A \longrightarrow B :< ColAsg, A, Null, SIG_{SK_A}(A|R_B) >$$

where $Null$ indicates that the color assignment process between A and B is unsuccessful. We refer to the above type of color assigned message in the remaining text as *ColAsg(Null)*. The node B on receiving *ColAsg(Null)* message from A discards the message and logically disconnects itself from A. In this case, node A and B do not communicate with each other, even though they are neighbors.

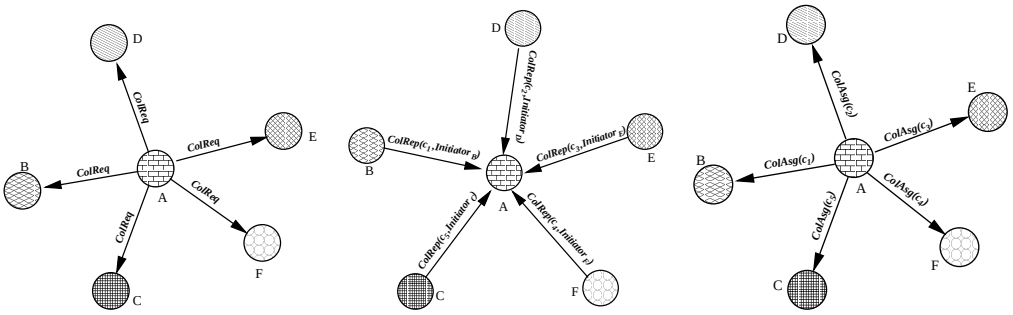


Figure 5.1: All neighbors have distinct color.

The following four scenarios exist between the *Initiator* and its neighboring nodes during the coloring process:

Scenario 1: *All neighbors of the Initiator are in Colored state.*

The following two cases may arise in this scenario: *Case 1* : All neighbors have distinct color. This is depicted in Figure 5.1. *Case 2* : Two or more neighbors have same color. This case is depicted in Figure 5.2. Node coloring process in the above two cases are explained below.

Case 1: *All neighbors have distinct color.*

Initiator on receiving a color reply message say, *ColRep(c_i, Initiator$_i$)* from node

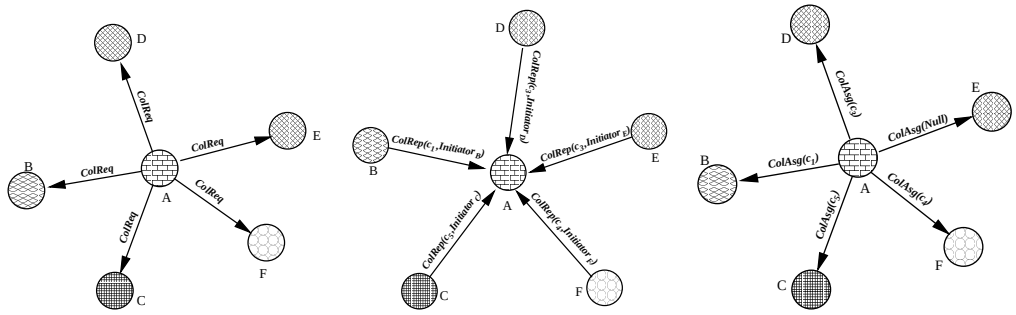


Figure 5.2: Two or more neighbors have same color.

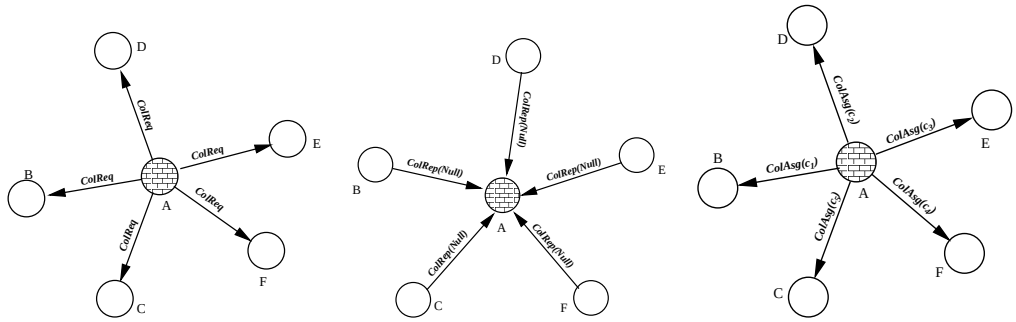


Figure 5.3: All neighbors of the Initiator are in Uncolored state.

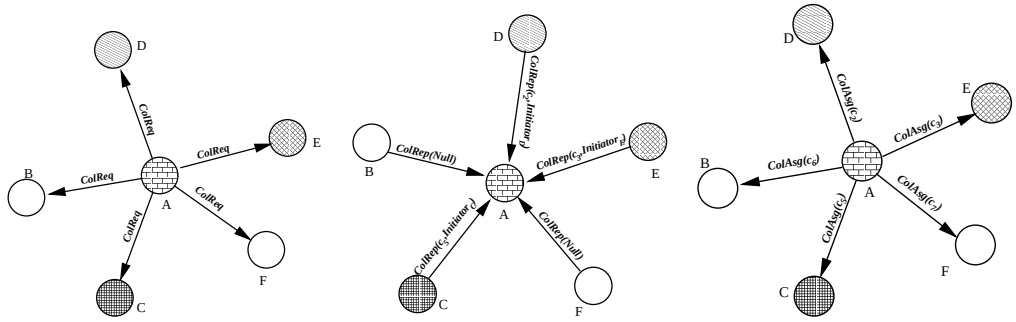


Figure 5.4: Neighboring node C, D, and E of *Initiator* A, are in *Colored* state, while B and F are in *UnColored* state.

i, records it as a colored neighbor in its neighbor table and sends a $ColAsg(c_i)$ to it. In Figure 5.1, the *Initiator* A, records the nodes B, C, D, E and F as its colored neighbors in its neighbor table.

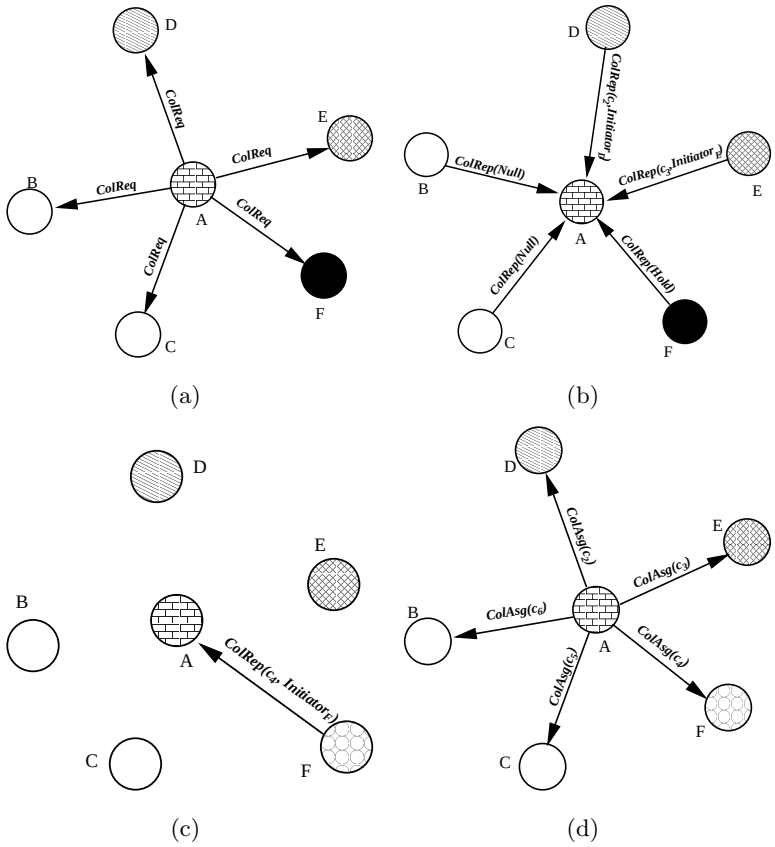


Figure 5.5: The Initiator, A, has the neighbor D, and E in *Colored* state, B and C in *UnColored* state and F in *Coloring* state.

Case 2: *Two or more neighbors have same color.*
In this case, *Initiator* will receive at least two *ColRep* messages, say *ColRep(c_i, $Initiator_i$)* and *ColRep($c_j, Initiator_j$)* from node i and j respectively such that $c_i = c_j$. When two or more nodes report with the same color, then the *Initiator* selects a node at random and records it as a colored neighbor in its neighbor table and sends a *ColAsg(c_i)* message. To the remaining nodes, it sends a *ColAsg(Null)* message and logically disconnect itself from them. In Figure 5.2, nodes D and E have same color c_3. *Initiator*, A, selects node D and records it as a colored neighbor with color c_3. The *Initiator* does not communicate directly with E; even though E is within its communication range.

Scenario 2: *All neighbors of the Initiator are in Uncolored state.*
This scenario is depicted in Figure 5.3. Neighboring nodes of the *Initiator* respond with a *ColRep(Null)* message in response to the *ColReq* message from the *Initiator*, and then the *Initiator* assigns a unique color to each of the neighbors. In Figure

5.3, the *Initiator*, A, on receiving *ColRep(Null)* messages from all neighbors, assigns a unique color to each of them from its color pool, CLR.

Scenario 3: *A few neighbors of the Initiator are in Colored state, while the rest are in Uncolored state.*
Figure 5.4 depicts this scenario. First, the nodes in the *Colored* state are either successfully recorded as colored neighbors or logically disconnected as explained in Scenario 1. Then, the nodes in *UnColored* state are assigned with color as in Scenario 2. In Figure 5.4, node C, D, and E are in *Colored* state, whereas node B and F are in *Uncolored* state. The *Initiator* A first records the node C, D, and E with the color as contained in their *ColRep* message in its neighbor table. Then, it assigns color to the node B and F.

Scenario 4: *At least one of the neighbors of the Initiator is in Coloring state*
This scenario is depicted in Figure 5.5. The *Initiator* waits until it receives a *ColRep* message with a valid color from the nodes in *Coloring* state. On receiveing *ColRep* message from nodes in *Coloring* state, it assigns color to its neighbors as in Scenario 3.

In Figure 5.5(a), node D and E are in *Colored* state with color c_2 and c_3 respectively, node F is in *Coloring* state, and node B and C are in *Uncolored* state. The *Coloring* node F, responds with a *ColRep(Hold)* message in response to *ColReq* message from the *Initiator* A. This is shown in Figure 5.5(b). When the *Initiator* A, receives the *ColRep(Hold)* message from F, it suspends the color assignment process. On receiving the *ColRep*(c_4, $Initiator_F$) from F as shown in Figure 5.5(c), it assigns color to all its neighboring node as shown in Figure 5.5(d). Node coloring process is summarized in Algorithm 5.1.

Claim 5.1. *An* Initiator *assigns a unique color to all its neighbor.*

Proof. We prove this by means of contradiction. Let i be an *Initiator* node. Let m and n be two nodes within the neighbor of i such that they share the same color, *i.e,* $Color_m = Color_n$. Assume that node m is colored prior to n, and c be the color assigned to m. After assigning the color c to m, CLR_i is updated to $CLR_i - \{c\}$. When node i has to assign a color to n, it selects an existing color from its updated color set CLR_i, which does not contain the color, c. Thus, node n cannot be assigned with the color c. This contradicts our assumption that the *Initiator* assigned the same color to m and n. □

Algorithm 5.1: Node Coloring Algorithm.

Input: A node i with its set of neighbor nodes NBD_i
Output: All neighbors of node i have a unique color

```
1  begin
2      Broadcast ColReq message to all neighbors
3      begin
4          for each v ∈ NBD_i do
5              Send ColReq to v
6          end
7      Process ColRep messages of neighbors
8      begin
9          if received a ColRep(Hold) from v ∈ NBD_i then
10             wait until v sends a ColRep(Col,Initiator_v) message
11         else if received a ColRep(Col, Initiator_v) from v ∈ NBD_i then
12             ColoredNbr ← ColoredNbr ∪ {⟨v, Col, Initiator_v⟩}
13         else if received a ColRep(Null) from v ∈ NBD_i then
14             UncoloredNbr ← UncoloredNbr ∪ {v}
15     Color verification of Colored neighbors
16     begin
17         for each v ∈ ColoredNbr do
18             clr ← getColor(v)
19             if clr ∈ CLR_i then
20                 Color_v ← clr
21                 CLR_i ← CLR_i − {clr}
22                 Send ColAsg(clr) to v
23             else
24                 Send ColAsg(Null) to v
25             end
26         end
27     Assign color to UnColored neighbors
28     begin
29         for each v ∈ UncoloredNbr do
30             if CLR_i ≠ {∅} then
31                 Color_v ← x, s.t. x ∈ CLR_i
32                 CLR_i ← CLR_i − {x}
33                 Send ColAsg(x) to v
34             else
35                 Send ColAsg(Null) to v
36             end
37         end
38 end
```

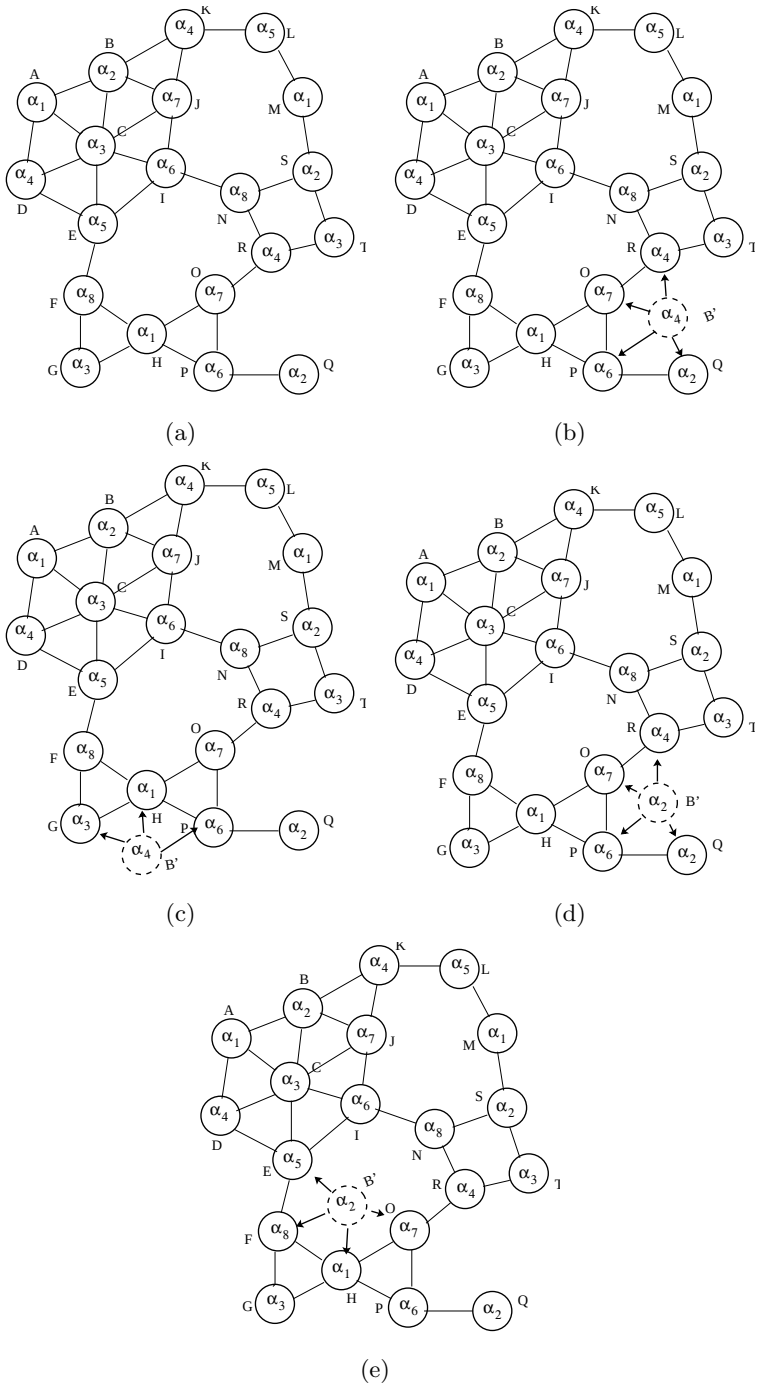


Figure 5.6: Replica Detection.

5.2.2 Replica detection after node coloring process

A color conflict within the neighborhood of a node is detected as a replica. Nodes with conflicting colors are revoked from the network. An adversary can deploy a

replica at any location in the sensor network. When an adversary deploys a replica, the following two scenarios may arise: *i)* The adversary might have changed the color of the replica before deployment. In this case, the replica and its original node have different colors, and *ii)* The adversary might not have changed the color of the replica. In this case, the replica retains the same color as its original node. Figure 5.6 shows the above two situations. We explain below the replica detection in each of the above situations.

Figure 5.6(a) shows a network, where each node is colored uniquely within its neighborhood. In this Figure, $\alpha_1, \alpha_2, ...\alpha_8$ are the colors of color set CLR. Initially, all nodes have the same color set, CLR. When a node assigns any color to its neighbor, then that color is deleted from its CLR. The color encircled in the figure is assigned to the node. For example, node B is assigned with color α_2. Suppose an adversary have captured a node, say B, and deployed its replica B', into the network. Node B and B' have the same identity. The adversary may or may not have changed the color of the replica B'.

i) Replica deployed with changed color: In this case, the adversary has changed the color of the replica prior to its deployment. This situation is depicted in Figure 5.6(b) and 5.6(c) where the replica B' has different color than its original node B. The color of B' is α_4 where as color of B is α_2. Figure 5.6(b) shows a situation, where the replica and at least one of its neighbor share the same color. Figure 5.6(c) shows the situation where the replica and none of its neighbor share the same color. In the proposed scheme, replica is detected during communication. In Figure 5.6(b), to communicate with any of its neighbor O, P, Q, or R, the replica B' have to specify its color and ID. Node R has been assigned with the same color α_4, as that of replica B'. When B' reports its color either to R or O, a conflict with node ID is detected. Node R has been assigned with a color α_4 which conflicts with the color reported by node B'. Node O has recorded during the color assignment process that the color α_4 is assigned to node R. On receiving color information from B', it verifies that the color α_4 is assigned to node R. Again a conflict with node ID arises, and B' is detected as a replica. Suppose the node P and Q in Figure 5.6(b) have not assigned the color α_4 to any of their neighbors. This means, the color α_4 remains unused in their color set CLR. When replica B' reports its color either to P or Q, a color conflict is detected, as the color α_4 reported by B' is neither assigned by P nor by Q to any of their neighbors.

In Figure 5.6(c), none of the neighbors of the replica B' has the same color as that of B'. Suppose the neighbors of B' has not assigned the color α_4 to any of their neighbors. In the above situation B' is detected as a replica, as explained in the previous paragraph. If one or more neighbors of B' have assigned the color α_4 to one of their neighbor, then a conflict with node ID occurs. Thus, B' is detected as a replica.

ii) Replica deployed with same color: Adversary has deployed the replica retaining the same color as that of the original node. This situation is depicted in Figure 5.6(d) and 5.6(e), where replica B' has the same color, α_2 as that of its original node. In Figure 5.6(d), the node B' and one of its neighbor Q has the same color. This situation same as that in Figure 5.6(b). Hence, B' will be detected as a replica.

In Figure 5.6(e), the replica B' and its neighbors does not share the same color. This situation is similar to that shown in Figure 5.6(c). Hence, B' will be detected as a replica.

From *i)* and *ii)* a replica is detected, whenever it is deployed in the network with or without changing its color.

5.3 Analysis

In this section, we evaluate our proposed technique. Parameters considered for evaluation are: *Security, Communication*, and *Storage overhead*.

5.3.1 Security Analysis

i) In the proposed scheme, Identity-based signature scheme is used for message authentication. Therefore, a node's identity cannot be forged. Moreover, Identity-based signature scheme incurs significantly lower computation overhead in comparison to other signature based schemes. Therefore, it is suitable for resource-constrained networks like WSN.

ii) An adversary cannot claim as an *Uncolored* node after being *Colored*. This is because, when a node transforms itself from *Uncolored* to *Colored* state, the *ColKey* of the node is replaced by its assigned color. To claim as an *UnColored* node, it needs the *ColKey*. Since, it is not possible to restore the *ColKey* once

it is replaced by its color, the adversary cannot claim as an *UnColored* node after being *Colored*.

iii) A replica cannot act as the *Initiator*. This is because, during the coloring process each node maintain a list of *ColRep* message it has heard. When the *Initiator* have acquired a color, it must have responded with *ColRep* message to its *Initiator* node. This message is maintained by its neighbor in their *ColRep* message list. On receiving a *ColReq* message from the *Initiator*, neighboring nodes verify the existence of the node identity in the *ColRep* message list. Since, the replica is acting as the *Initiator*, its identity will not be present in the neighbors *ColRep* message list.

iv) A *Colored* node may not act as *Initiator*. However, in this case, the node cannot communicate with any other nodes in the network.

5.3.2 Detection Probability

In this section, we have computed the probability of detecting a replica. Let R be a replica deployed in the network, and $NBD_R = \{nbr_1, nbr_2, \ldots, nbr_d\}$, where $0 < d \leq |CLR|$. The replica R can be detected if one of the following conditions are satisfied:

i) There exists a node $nbr_i \in NBD_R$, such that $Color_{nbr_i} = Color_R$.

ii) There exists a node $nbr_i \in NBD_R$, such that $Color_R \in CLR_{nbr_i}$.

Then, the probability of detecting a replica by a neighboring node of R is given as

$$P_{det} = 1 - [P(A).P(B)] \tag{5.1}$$

where, A is an event that none of the neighbors, nbr_i, of R is assigned with the color, $Color_R$. The probability of A is given by

$$\begin{aligned} P(A) &= \left(\frac{|CLR| - 1}{|CLR|}\right)^d \\ &= \left(1 - \frac{1}{|CLR|}\right)^d \end{aligned} \tag{5.2}$$

B is an event that none of the neighbors, nbr_i, of R have the color, $Color_R$ in their color set CLR. The probability of B is given by

$$P(B) = \left(\frac{d}{|CLR|}\right)^d \tag{5.3}$$

Using Equation 5.2 and 5.3 in Equation 5.1 we get

$$P_{det} = 1 - \left[\left(1 - \frac{d}{|CLR|}\right).\frac{d}{|CLR|}\right]^d \tag{5.4}$$

where, P_{det} represents the detection probability of a node, given the value of d and $|CLR|$.

□

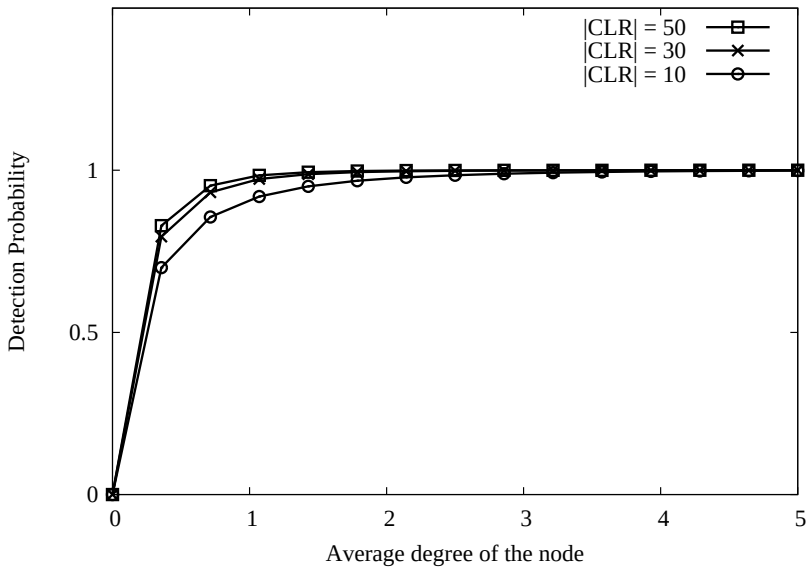


Figure 5.7: Detection probability *vs.* Average degree of the node.

The Figure 5.7 shows the detection probability by varying the average degree d, of a node. It is observed from the figure that the detection probability rapidly approaches to *one* with the increase in d and color set size as well.

5.3.3 Communication and Storage Overhead

In this section, we compute the overall communication and storage overhead associated with the NCBRD technique. The number of messages exchanged during the coloring process is the sum of *ColReq* broadcast by an *Initiator* node, the number

of *ColRep* messages from neighbors to the *Initiator*, and the number of *ColVer* and *ColAsg* messages from *Initiator* to its neighbors. Therefore,

$$\text{Total number of messages exchanged per } \textit{Initiator} = 3d + 1$$

Since, every node in the network can act as an *Initiator*, the overall communication overhead is

$$\begin{aligned} \text{Communication Overhead} &= \text{Number of messages exchanged} \\ &\quad \text{per } \textit{Initiator} \text{ in the Network} \\ &= N.(3d+1) \\ &= O(N.d) \end{aligned} \tag{5.5}$$

For detecting a replica in NCBRD, node uses a color set, CLR, to assign color to its neighbor, where size of $CLR \approx d$. Therefore, the storage overhead associated with NCBRD is $O(d)$.

Table 5.2: Simulation Parameters.

Parameter	**Value**
Network	1000 x 1000 m^2
Network size (N)	1000 – 10000
Deployment type	Uniformly random
Communication range	55 meter
Maximum packet size	20 bytes
Average number of neighbors	40
Number of pre-colored *Initiators*	5% of N
Simulation Time	400 sec

5.4 Simulation and Results

We have simulated the proposed scheme using Castalia-3.2 [54] simulator, that runs on the top of Omnet++ [55]. Parameters considered for simulation are summarized

in Table 5.2. Replicas are randomly deployed. We considered the following parameters of interest: *i)* Coloring time, *ii)* Link loss percentage, *iii)* Number of replicas detected, and *iv)* Number of packets sent/received.

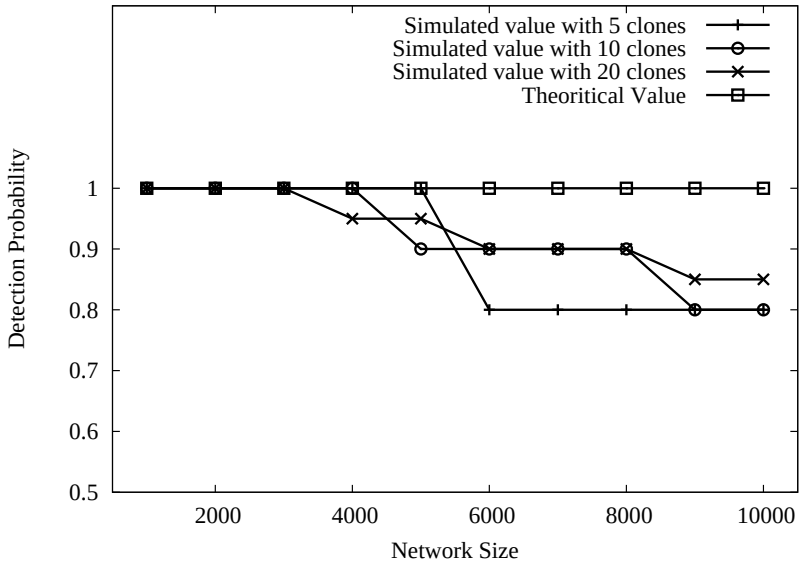


Figure 5.8: Detection probability *vs.* Network size between theoretical value and simulation.

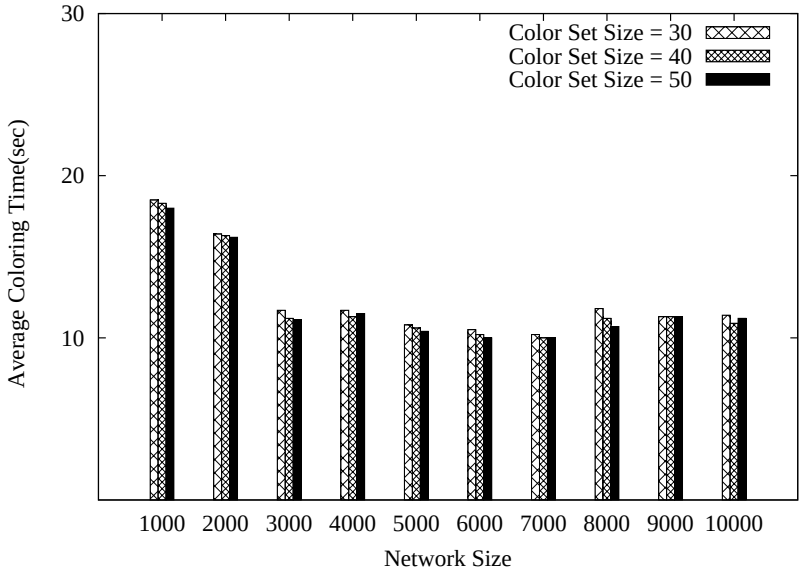


Figure 5.9: Coloring time *vs.* Network size in NCBRD.

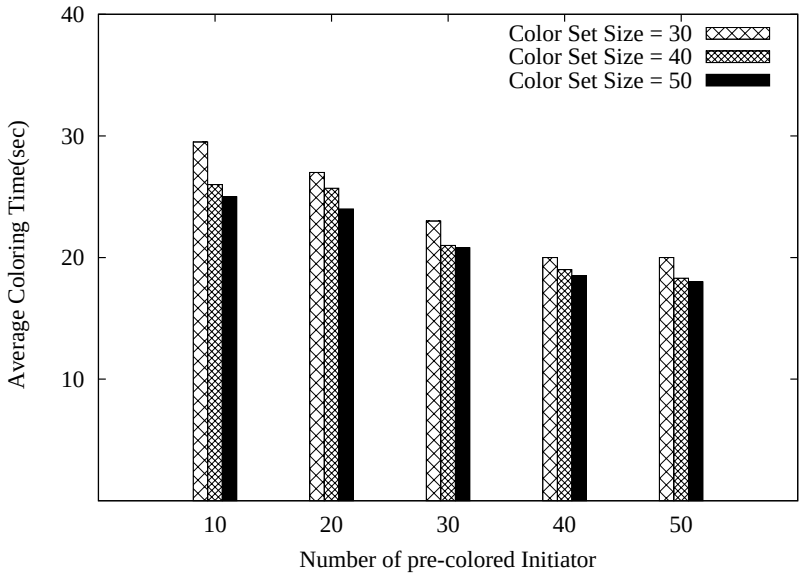


Figure 5.10: Coloring time *vs.* Number of pre-colored *Initiator* for $N = 1000$.

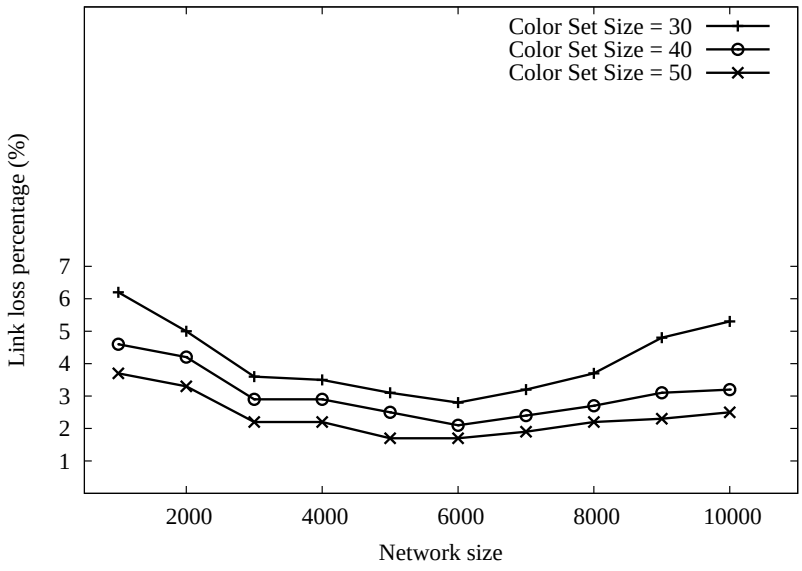


Figure 5.11: Link loss percentage *vs.* Network size in NCBRD.

In Figure 5.8, we compared the detection probability *vs.* network size between the theoretical value and simulated one. As expected the simulated detection probability is lesser than the theoretical value.

The plot for coloring time *vs.* network size is shown in Figure 5.9. Coloring time is the time required to complete coloring process in the network. It is observed

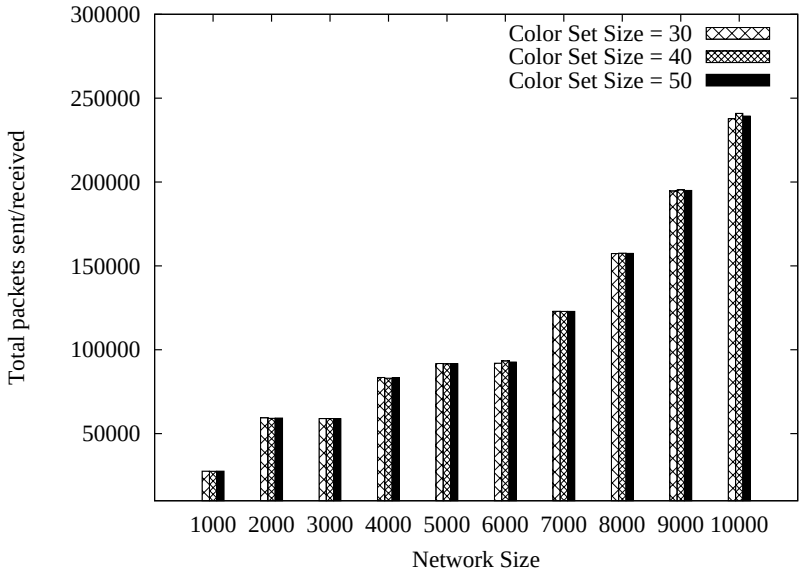


Figure 5.12: Total number of packets sent/received during coloring process *vs.* Network size in NCBRD.

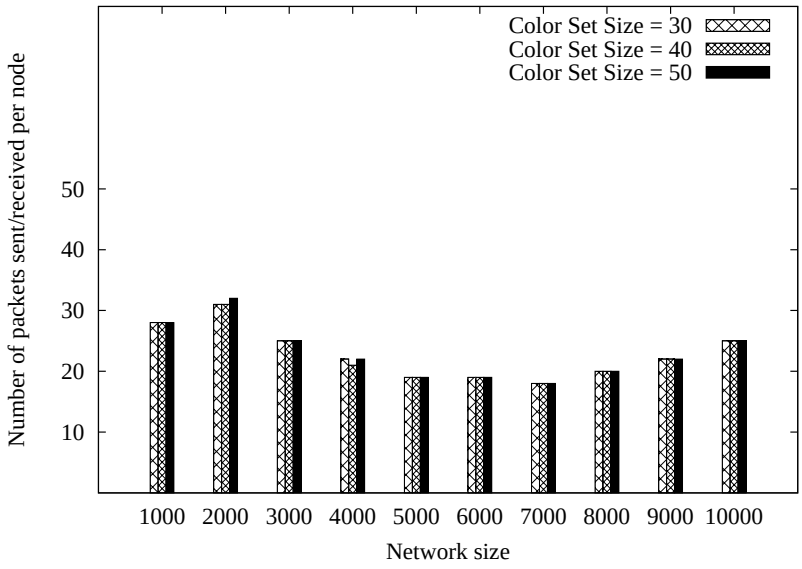


Figure 5.13: Average number of packets sent/received per node during coloring process *vs.* network size in NCBRD.

from the figure that for different size of color set, the coloring time remains almost equal with increase in the network size. It is also observed that the coloring time decreases marginally with increase in the network size due to increase in network

density.

The coloring time depends on the number of pre-colored *Initiators*. The plot for coloring time *vs.* number of pre-colored *Initiators* for a network of 1000 nodes is shown in Figure 5.10. It is observed from the figure that the coloring time decreases with the increase in number of pre-colored *Initiators*. We have observed through simulation that the increase in coloring time is marginal when the number of pre-colored *Initiators* is more than 50. The worst case coloring time is achieved when there is only one pre-colored *Initiator* in the network.

In NCBRD, two neighboring nodes can communicate, if and only if the color assignment between them is successful. For an unsuccessful color assignment, we consider the link between the nodes does not exist, *i.e*, they are logically disconnected even though there exists a physical link. Figure 5.11 shows the percentage of link loss *vs.* network size. It is observed that the link loss percentage increases marginally with increase in the network size. It is also observed from the figure that there is a decrease in the link loss percentage with increase in the size of color set, for a given network size.

The plot for total packets sent/received during coloring process is shown in Figure 5.12. It is observed from the figure that the number of packets sent/received increases with increase in the network size. With increase in the network size, the number of nodes in the network also increases. As a result, the total number of packets sent/received increases.

In Figure 5.13, we plot the average number of packets sent/received per node *vs.* network size during coloring process. From the figure, it is observed that the average number of packets sent/received per node remains almost constant irrespective of the network size and color set size $|CLR|$. The average number of packets sent/received per node is independent of the $|CLR|$. Hence, it remains almost constant with different size of CLR.

We varied the simulation area to 500 $\times$ 500 m^2, 1000 $\times$ 1000 m^2, and 2000 $\times$ 2000 m^2 to evaluate node coloring process. Size of the color set is taken to be forty for evaluation. The comparison of link loss percentage *vs.* network size is shown in Figure 5.14. It is observed from the figure that link-loss is marginally higher in an area of size of 500 $\times$ 500 m^2 and lower in 2000 $\times$ 2000 m^2. Higher link-loss is attributed to higher node density. For equal number of nodes, smaller area will have higher node density than larger areas.

The average number of packets sent/received *vs.* network size for different area size is shown in Figure 5.15. It is observed from the figure that the number of

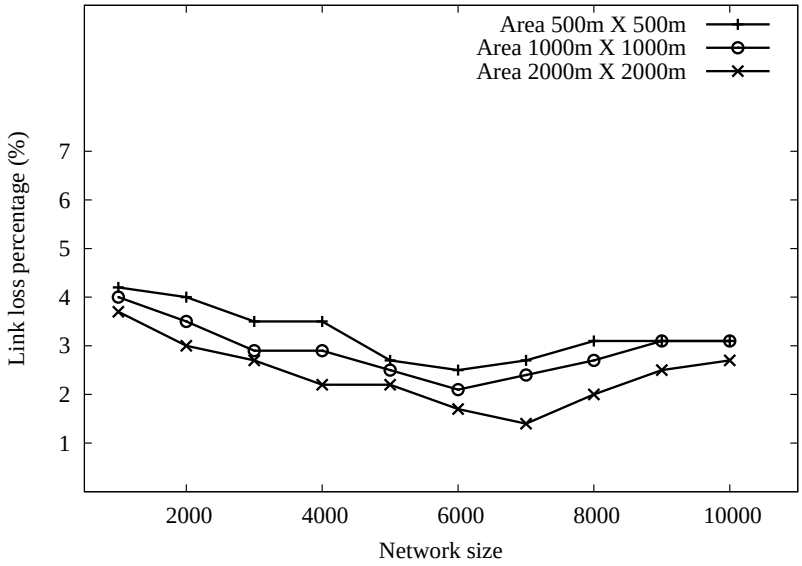


Figure 5.14: Link loss percentage during coloring process *vs.* Network size varying the simulation area size.

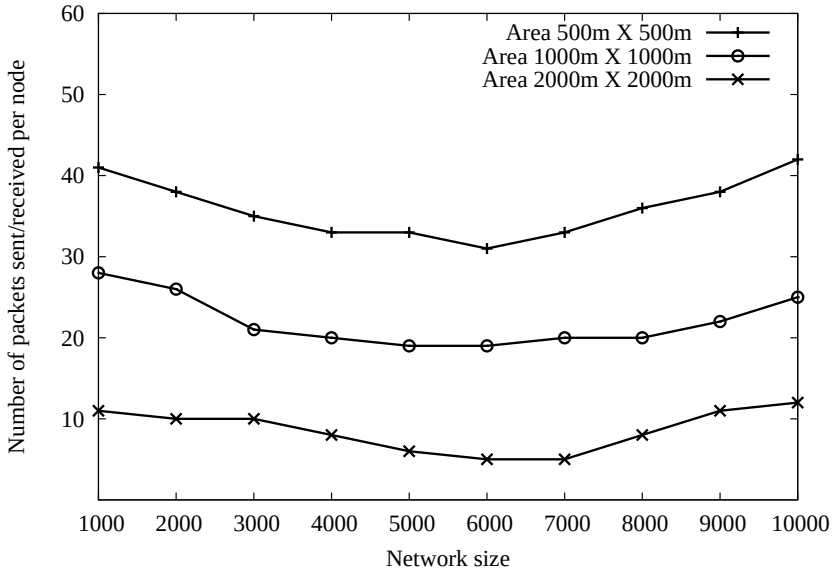


Figure 5.15: Average number of packets sent/received per node *vs.* Network size varying the simulation area size.

packets sent/received is more in an area of size 500 $\times$ 500 m^2. This is because, the density of nodes will be higher in a smaller area than larger area for the same number of nodes. This leads to higher node degree. As a result, the number of

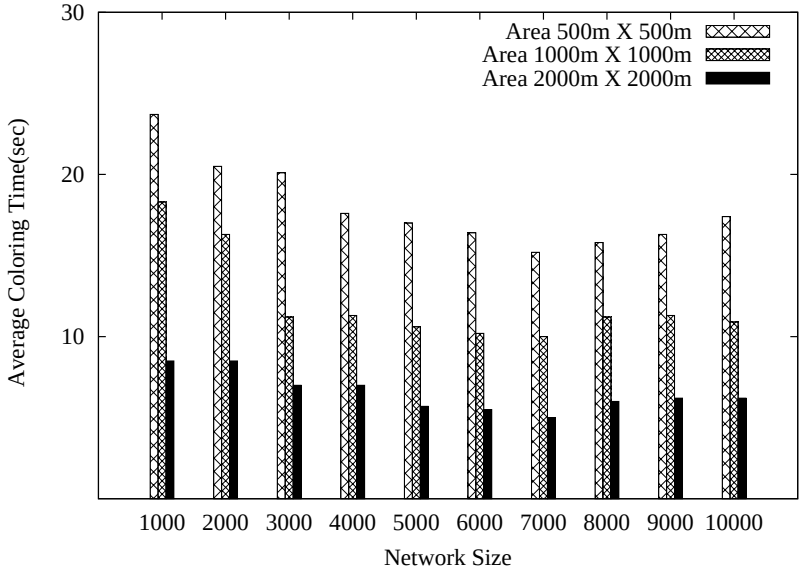


Figure 5.16: Coloring time (sec) *vs.* Network size varying the simulation area size.

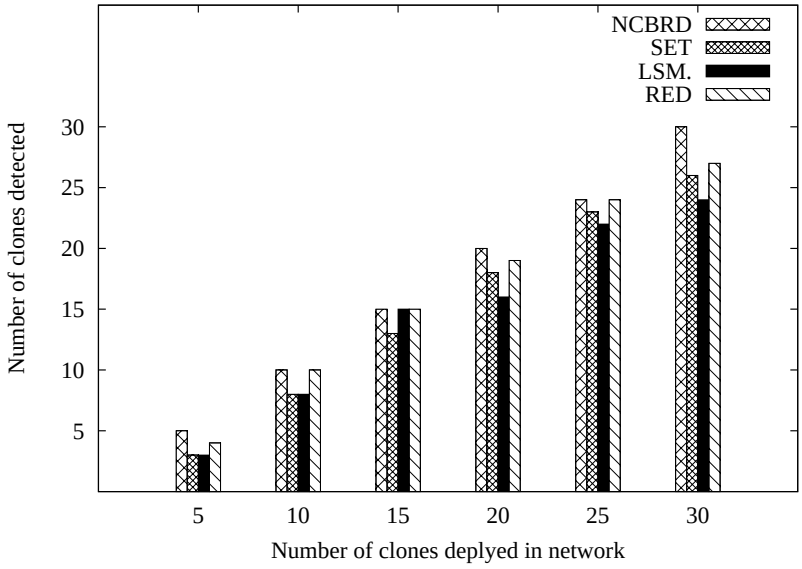


Figure 5.17: Comparison of number of replicas deployed *vs.* Number of replicas detected for network size = 1000.

packets exchanged between the nodes is also higher.

The plot for coloring time *vs.* network size for different area size is shown in Figure 5.16. It is observed from the figure that area of size 500×500 m^2 have higher coloring time in comparison to other areas. This is because of higher node density

in a smaller area. Higher the node density more will be the collision, resulting into more coloring time.

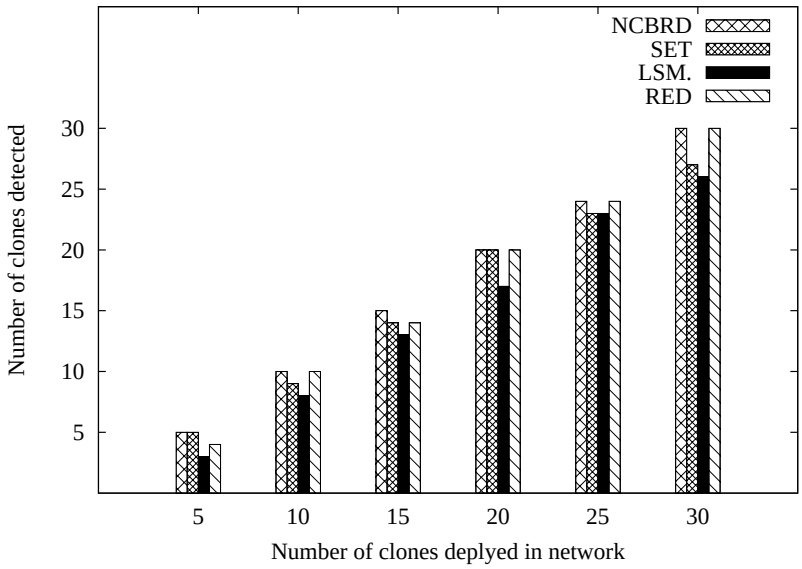


Figure 5.18: Comparison of number of replicas deployed *vs.* Number of replicas detected for network size = 2000.

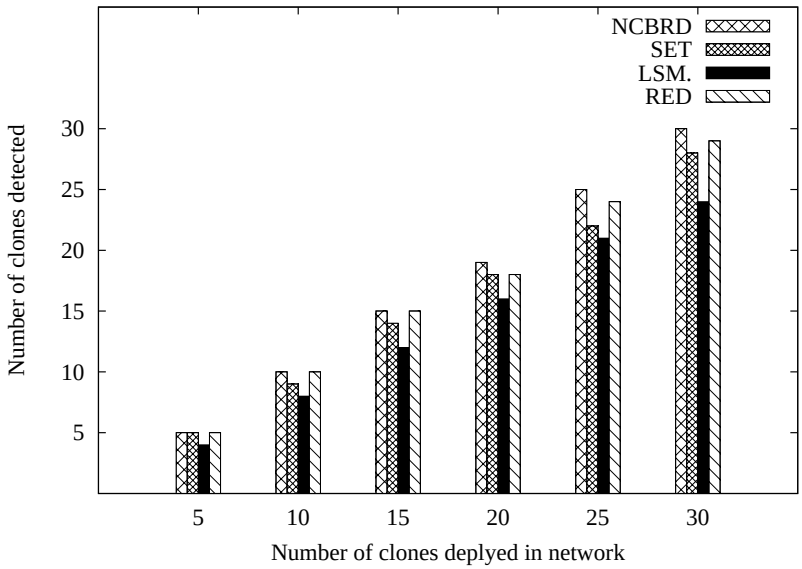


Figure 5.19: Comparison of number of replicas deployed *vs.* Number of replicas detected for network size = 3000.

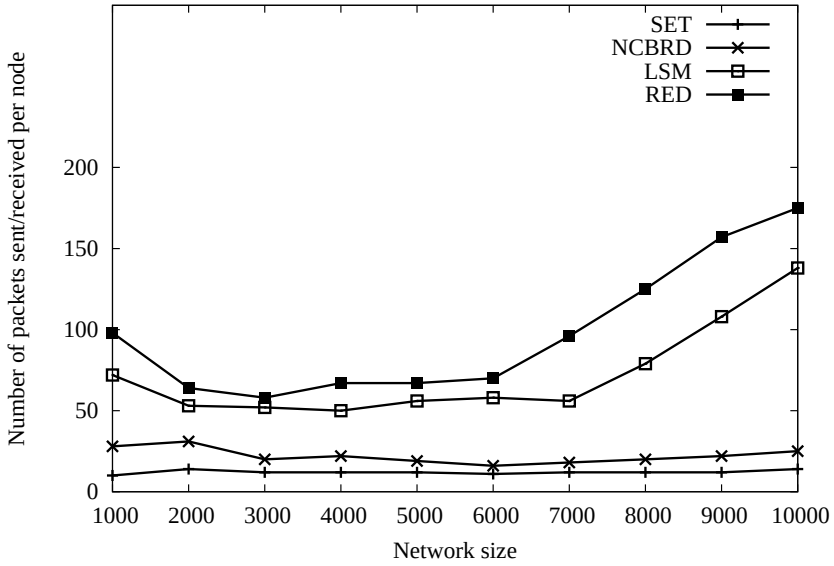


Figure 5.20: Comparison of average number of packets sent/received per node *vs.* Network size.

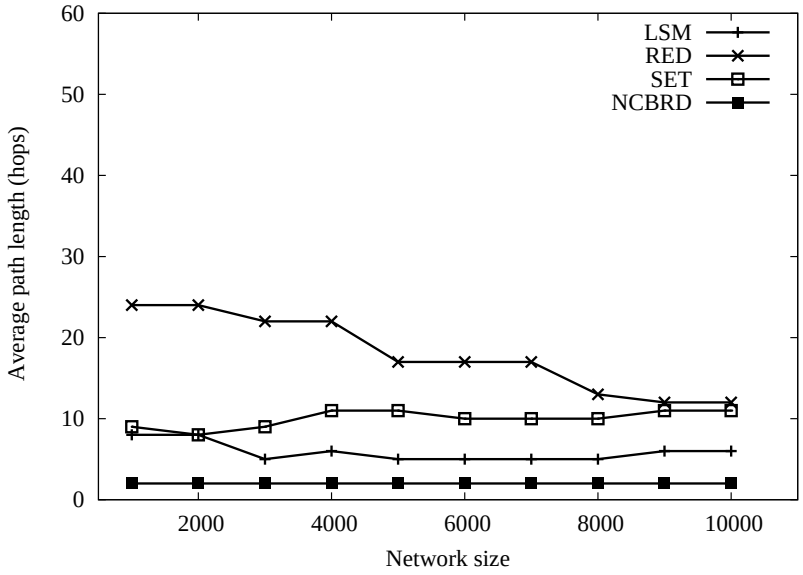


Figure 5.21: Comparison of average path length (in number of hops) *vs.* Network size.

Next, we made comparison of NCBRD with a few existing mechanisms such as LSM [36], RED [32], and SET [25]. The following parameters are considered for comparison: *i)* Number of replica detected, *ii)* Average number of packets sent/received

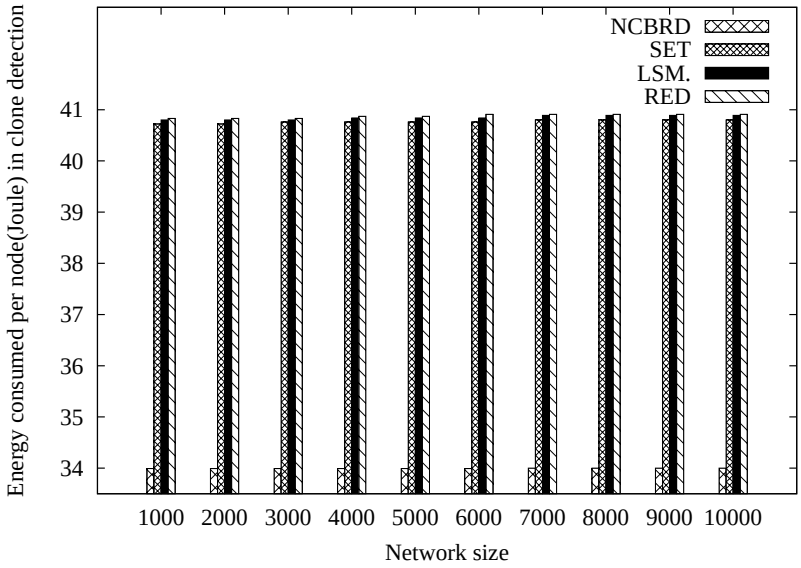


Figure 5.22: Energy consumed per node *vs.* Network size.

per node, *iii)* Average path length, and *iv)* Energy consumed per node.

The plot for number of replicas detected *vs.* number of replica deployed in a network of size 1000, 2000 and 3000 are shown in Figure 5.17, 5.18, and 5.19 respectively. It is observed from the figure that, NCBRD is able to detect all replicas deployed in the network. This is because, neighboring nodes of a replica have unique colors . Once a replica is deployed in the neighborhood of a node, a color conflict arises, which is detected as a replica.

The plot for average number of packets sent/received per node *vs.* network size is shown in Figure 5.20. It is seen from the figure that the number of packets sent/received per node is lesser in NCBRD than RED [32] and LSM [36]. However, the number of packets sent/received per node in NCBRD is marginally higher than SET [25]. This is because, the communication complexity of SET is $O(N)$, whereas in NCBRD it is $O(N.d)$.

Figure 5.21 shows the comparison of average path length *vs.* network size. Average path length of a replica detection mechanism is the average number of hops traveled by a message in replica detection. It is observed from the figure that NCBRD has lower average path length in comparison to other schemes. This is because, in NCBRD, message communication in replica detection is restricted to two-hops only. Whereas, message travels to a comparatively longer path in schemes like LSM, RED and SET.

Finally, the comparison for energy consumption per node in replica detection is shown in Figure 5.22. It is observed from the figure that the energy consumed in detecting a replica is lesser in NCBRD. This is because, in NCBRD, a replica is detected within its neighborhood, resulting into a lesser number of messages exchanged.

5.5 Summary

In this chaper, we proposed a node coloring based replica detection scheme called NCBRD, for wireless sensor networks. In the proposed scheme, each node is assigned with a color, which is unique within its neighborhood. A color conflict in NCBRD is detected as a replica. NCBRD has lower communication and storage overhead. We have compared NCBRD with a few existing schemes such as RED [32], LSM [36], and SET [25]. It is observed that NCBRD outperforms the compared replica detection mechanisms in terms of detection probability, communication, and storage overhead. In NCBRD, the communication for replica detection is restricted to two-hop neighbors only. Whereas the replica detection mechanism in LSM, SET, and RED requires more than two-hop communication. Therefore, the average path length in NCBRD is lower in comparison to LSM, SET, and RED. It is also observed that the NCBRD have higher detection rate in comparison to other schemes. This is because, every node keeps track of the assigned and residual colors in its color set. This helps in detecting a color conflict within its neighborhood. Though NCBRD has higher detection probability, it has certain disadvantages too. It is difficult to add a new node in the network. Moreover, a smaller color set may results in higher link-loss. This mean, more number of neighboring nodes cannot communicate directly. In worst case, it may results into partitioned networks. To ensure minimal link-loss the color set must be chosen judiciously.

Detecting replica in mobile wireless sensor network is more challenging than static network. In the next chapter, we proposed a replica detection mechanism for mobile wireless sensor networks. The proposed mechanism uses residual energy of nodes to detect replica.

Chapter 6

Energy Based Replica Detection

Mobile wireless sensor network (MWSN) is a variation of WSN, where the sensor nodes are mobile. Nodes move within the network to gather information. Battery power of a mobile sensor node is relatively higher than their static counterpart. Mobile nodes have the following advantages over static nodes: *i)* They can reach closer to the target for gathering accurate data, *ii)* They can rearrange their position to connect a disjoint node, and *iii)* They can improve the network lifetime by load balancing. Besides, all advantages that MWSNs have, they also face various challenges such as communication, distributive cooperative control, security etc. [63].

Replica detection schemes proposed for static sensor networks are not applicable for MWSNs as the sensor nodes are mobile in MWSN. A fast and robust mechanism is necessary to detect node replication attack in MWSN. In this chapter, we propose a distributed replica detection scheme for MWSN based on the residual energy of nodes.

6.1 Related Works

A few of the replica detection schemes proposed for MWSN are briefly described below.

Deng *et al.* [64] have proposed two distributed time-location based detection schemes: *i)* Unary-Time-Location Storage and Exchange (UTLSE), and *ii)* Multi-Time-Location Storage and Diffusion (MTLSD). In their schemes, each node maintains multiple time-location claim of the tracked nodes. When two trackers for the same node meet each other, they exchange their claims to verify the feasibility of location-claims. A pair of conflicting time-location claims lead to detection of

replica.

A detection scheme based on node's speed is proposed by Ho *et al.* [65]. In their scheme, a node moving with a speed higher than the pre-defined speed limit is detected as replica. A node that moves to a new location broadcasts the following information to its new neighbors: *i)* current time, and *ii)* location-claim. Each neighbor on receiving the claim message verifies the authenticity of the message. If the message is authentic then it is forwarded to the BS. The BS periodically collects the time and location-claim of each node and computes the speed of the node. If the node is found to be moving at a speed higher than the pre-defined maximum speed limit, then the node is detected as replica. This scheme uses Sequential Probability Ratio Test to detect the replicas using their location-claims.

Deng and Xiong [57] proposed a detection mechanism based on the pair-wise key establishment. In their scheme, a node keeps track of the number of pair-wise keys it has established. Counting Bloom filter is used to store the count of the number of pair-wise keys established. Nodes periodically send their Counting Bloom filter to the BS. The BS on receiving the Counting Bloom filters update the number of pair-wise keys established by each node in a given period. If the number of keys established for a node exceeds a threshold value, then the BS detects it as replica.

Yu *et al.* [66] proposed two schemes called: *i)* eXtremely Efficient Detection (XED), and *ii)* Efficient Distributed Detection (EDD). In XED, each pair of nodes exchange a random number as a challenge between them. When they meet again at some later point of time, they request for the challenge. If a node does not send or an invalid challenge is sent to the other node, then it is detected as a replica. EDD is based on the number of times nodes meet each other in a given time interval. Replica is detected if the number of meetings exceed a pre-defined threshold.

The detection schemes proposed by Deng *et al.* [64] and Ho *et al.* [65] are location-aware schemes. They use either a GPS or an efficient localization technique to determine location. Above schemes are not suitable for applications that mainly deals with the type and frequency of an event within a known territory, independent of the event location. The sensors deployed for these applications mostly do not have built-in location finding mechanism.

Moreover, the schemes proposed by Ho *et al.* [65] and Deng and Xiong [57] are centralized in nature. There are more prone to a single point of failure. The BS is responsible for detecting replica. Thus, BS is overloaded with additional task of replica detection.

An adversary will be able to extract the set of challenges from captured node and

use them in replica, in XED. Therefore, XED may not be suitable in a real world scenario. EDD relies on the number of meetings with a node for replica detection. It also relies on the assumption that the deployment of replicas in the network will increase the number of meetings between the nodes. However, in a random-mobility scenario this is true only when the number of replicas are significantly higher than the legitimate nodes. Moreover, determining the threshold for number of meetings between nodes to detect a replica is difficult, especially in a random-mobility scenario.

6.2 Assumption and Model

In this section we describe the assumptions made about the network and the adversary. Notations used are given in Table 6.1.

Table 6.1: Notations.

Notations	**Meaning**
X	Identity of a node
E_X	Residual energy of node X
t^X	Time instance at node X
$E_X(t^X)$	Residual energy at node X at the time instance t^X
$\|\|$	Concatenation operator
SIG_X	Signature of node X
N	Network size

6.2.1 Network Assumptions

We made the following assumptions about the network: *i)* Nodes are uniformly deployed in the sensor field, and they move freely without any hindrance, *ii)* The speed at which a sensor node moves lies within the range $[v_{min}, v_{max}]$, where v_{min} and v_{max} are the minimum and maximum speed respectively, *iii)* Communication is bidirectional, and *iv)* An identity-based signature scheme as mentioned in [48] is used for message authentication.

6.2.2 Adversary Model

We made the following assumptions about an adversary: *i)* She has the ability to compromise a subset of nodes, *ii)* She can create as many replicas of the captured node as she wishes, and can deploy them at various locations in the network, *iii)* She cannot create a new identity for sensors, *iv)* Clones obey the same protocol suite as that of its original node, and *v)* Adversary is powerful enough to alter both the timestamp and/or residual energy of replicas.

6.2.3 Energy Consumption Model

The energy consumption of a sensor node in MWSN is the sum of its processor energy function (E_{cpu}), transceiver energy function (E_{trans}), sensor energy function (E_{sensor}), and the mobility energy function ($E_{mobility}$) [67, 68]. That is,

$$E_{total} = E_{cpu} + E_{trans} + E_{sensor} + E_{mobility} \tag{6.1}$$

where, E_{total} is the total energy consumption of a sensor node. The processor energy function (E_{cpu}) is defined by the following equation [67, 68]

$$\begin{aligned} E_{cpu} &= E_{cpu\_state} + E_{cpu\_change} \\ &= \sum_{i=1}^{m} P_{cpu\_state}(i).T_{cpu\_state}(i) \\ &\quad + \sum_{j=1}^{n} N_{cpu\_change}(j).e_{cpu\_change}(j) \end{aligned} \tag{6.2}$$

where, E_{cpu_state} is the cpu state energy consumption, E_{cpu_change} is the state-transition energy consumption, $P_{cpu\_state}(i)$ power dissipated by state i, $T_{cpu\_state}(i)$ is the duration of state i, m is the number of processor states, n is the number of state transitions, $N_{cpu\_change}(j)$ is the frequency of state transition j, and $e_{cpu\_change}(j)$ is the energy consumption of one-time state transition j.

The transceiver energy function is the sum of transceiver state energy consumption (E_{trans_state}) and state-transition energy consumption (E_{trans_change}) [67, 68].

$$E_{trans} = E_{trans\_state} + E_{trans\_change} \tag{6.3}$$

The transceiver state energy consumption (E_{trans_state}) is given by

$$\begin{aligned} E_{trans\_state} &= E_{TX} + E_{RX} + E_{idle} + E_{sleep} + E_{CCA} \\ &= \sum_{i=1}^{N_{TX}} V_{tr} I_{TX}.\frac{L_i}{R} + \sum_{i=1}^{N_{RX}} V_{tr} I_{RX}.\frac{L_i}{R} \end{aligned}$$

$$+V_{tr}(I_{idle}T_{idle} + I_{sleep}T_{sleep} + I_{CCA}T_{CCA}) \quad (6.4)$$

where, $E_{TX}, E_{RX}, E_{idle}, E_{sleep}$, and E_{CCA} are the energy consumption of transceiver in transmission, reception, idle, sleep, and Clear Channel Assessment (CCA) state respectively. V_{tr} is the working voltage, I_x is the electric current of state x, where $x \in \{TX,\ RX,\ idle,\ sleep,\ CCA\}$, L_i is the length of the ith packet received/sent, R is the data transferring rate, and N_{TX} and N_{RX} are the total number of packets transmitted and received respectively.

The state-transition energy consumption (E_{trans_change}) is given by [67, 68]

$$E_{trans\_change} = \sum_{j=1}^{n} N_{trans\_change}(j).e_{trans\_change}(j) \quad (6.5)$$

where, $N_{trans\_change}(j)$ is the frequency of transition j, and $e_{trans\_change}(j)$ is the energy consumed by the transceiver during state transition j, and n is the number of transition.

The sensor energy function (E_{sensor}) can be expressed as

$$\begin{aligned} E_{sensor} &= E_{on\_off} + E_{off\_on} + E_{sensor\_run} \\ &= N(e_{on\_off} + e_{off\_on} + V_s I_s T_s) \end{aligned} \quad (6.6)$$

where, e_{on_off} is the one time energy consumption for switching off sensor operation, e_{off_on} is the one time energy consumption of switching on sensor operation, E_{sensor_run} is the energy consumption of sensing operation, V_s and I_s are the working voltage and current of sensors, T_s is the time interval of sensing operation, and N is the number of sensor on and off operations.

The mobility energy function ($E_{mobility}$) can be expressed as [69]

$$\begin{aligned} E_{mobility} &= \sum_{i=1}^{p} e(D_i, T_i) \\ &= \sum_{i=1}^{n} V(D_i, T_i).I(D_i, T_i) \end{aligned} \quad (6.7)$$

where, $e(D_i, T_i)$ is the energy consumed by a node in moving a distance D_i in time T_i, $V(D_i, T_i)$ and $I(D_i, T_i)$ are the voltage and current flows respectively in a sensor node, when it traverses a distance D_i in time T_i, and p is the total number of epochs.

6.3 Energy Based Node Replica Detection

In this section, we discuss the proposed Energy Based Replica Detection (EBRD) scheme for MWSN. A mobile sensor node consumes energy for all operations it perform such as actuation, transmission, computation etc. As a result, the available residual energy of a node decreases with time. In EBRD the available residual energy of nodes is used to detect replicas.

In EBRD, each node acts as a monitoring node for a set of nodes in the network. That is, a node monitors a set of nodes and is monitored by a set of nodes in the network. Nodes maintain the timestamp-residual energy pair, $< t, E(t) >$ for the set of nodes they monitor, where t indicates the timestamp elapsed since a node's deployment, and $E(t)$ indicates the residual energy level of the node at t. The tuple $< t, E(t) >$ is recorded in the energy table maintained at each node. A node locally broadcasts its timestamp-residual energy pair at regular time interval within its one-hop neighbor. A neighboring node on receiving the timestamp-residual energy pair from a node say X, updates its energy table provided the node X is monitored by it. Conflict in the timestamp-residual energy pair for a node is detected as replica by its monitoring node.

The sharing of timestamp-residual energy pair is discussed in sub-section 6.3.1 and the replica detection mechanism is discussed in sub-section 6.3.2.

6.3.1 Timestamp-Residual Energy Sharing

Timestamp is a monotonically increasing parameter whereas the residual energy of a node decreases with time until the node dies. Nodes periodically broadcast their timestamp-residual energy pair. This process continues until the node dies. A node dies when its residual energy level is approximately equal to *zero*.

The timestamp-residual energy pair broadcast from a node X is given below:

$$X \longrightarrow * : \left[\ EU, \ X, \ t^X, \ E_X(t^X), \ R_X, SIG_X(X||t^X||E_X(t^X)||R_X) \ \right] \tag{6.8}$$

where EU indicates the message type and R_X is a nonce chosen by node X.

A node on receiving the timestamp-residual energy pair verifies the authenticity of the message. If the authentication fails, then the message is discarded, else its energy table is updated. The updation of timestamp-residual energy pair and replica detection is explained in the following section.

6.3.2 Replica Detection

In this section, we have explained the replica detection process in detail. To illustrate the replica detection process in EBRD we consider the Figure 6.1. The associated energy table for a few nodes of interest is also shown in the figure.

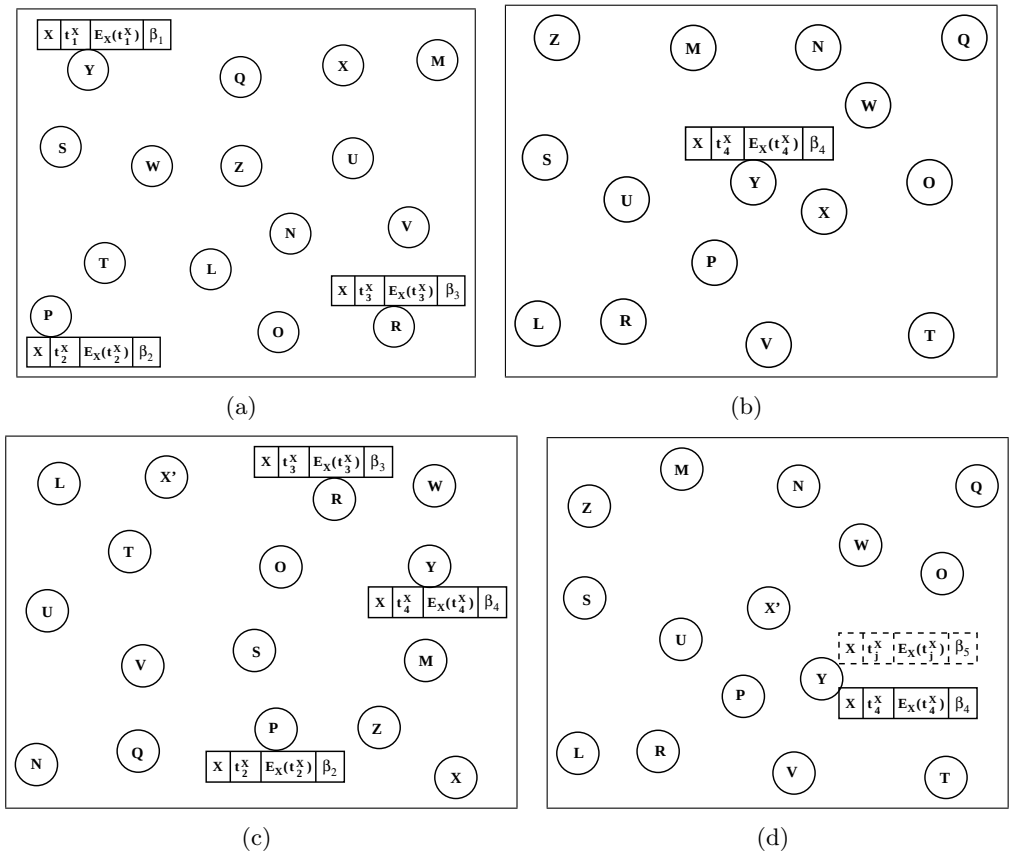


Figure 6.1: Figure to illustrate the replica detection process.

The detailed structure of the energy table is shown in Table 6.2. The *Node ID* field indicates the identity of the monitored node. *Time* and *Energy* indicates the timestamp and residual energy of the monitored node at which it has broadcasted the timestamp-residual energy pair, respectively. *Difference* is the time difference between the monitoring and monitored node at which the corresponding entry is updated in the energy table.

For example, in Table 6.2, X is the identity of the monitored node, t^X is the time at which node X has broadcast its residual energy, $E_X(t^X)$ is the residual energy of node X at time t^X, and t^Y is the time at which node Y has received the timestamp-residual energy pair from node X.

Table 6.2: Energy table

Node ID	**Time**	**Energy**	**Difference (β)**
X	t^X	$E_X(t^X)$	$t^Y - t^X$

Suppose, the node X in Figure 6.1 is monitored by node Y, P, and R. Let us assume that node Y, P and R were in the transmission range of node X in that order at some point of time during its trajectory in the network. The energy tables at node Y, P, and R are shown in Figure 6.1(a). From the energy tables it implies that node Y, P, and R have received the timestamp-residual energy pair $< t_1^X, E_X(t_1^X) >$, $< t_2^X, E_X(t_2^X) >$, and $< t_3^X, E_X(t_3^X) >$ at time t^Y, t^P, and t^R from node X respectively. The time $t_1^X < t_2^X < t_3^X$ and the residual energy $E_X(t_1^X) > E_X(t_2^X) > E_X(t_3^X)$. Further, assume that at time $t_4^X > t_3^X$, node X has broadcasted its timestamp-residual energy pair $< t_4^X, E_X(t_4^X) >$ and node Y is in the transmission range of node X. The updated energy table at node Y is shown in Figure 6.1(b).

Suppose, the node X is captured and its replica is deployed in the network at time t_5^X as shown in Figure 6.1(c). Let X' be the replica of X. Let $E_X(t_5^X)$ and $E_{X'}(t_5^{X'})$ be the residual energy of X and X' respectively at t_5^X.

Depending on the ability of the adversary, the following two scenarios arises: *i)* The adversary will deploy the clone with maximum energy, and *ii)* The adversary will modify the energy level of the captured node and/or the clone before deployment. The replica detection mechanisms in the above two scenarios are explained below.

A. *Clones are deployed with maximum energy.*

An adversary may deploy a clone with maximum energy with an objective to remain in the network for maximum period of time. Since, clones are deployed with maximum energy, $E_{X'}(t_5^X) > E_X(t_5^X)$. We identify the following two cases in this scenario:

Case - I: *Adversary has no capability to synchronize the clone's clock with the original node.*

Suppose node Y is in the communication range of X', when X' broadcast its timestamp-residual energy pair $< t_j^{X'}, E_{X'}(t_j^{X'}) >$ at time $t_j^{X'}$ as shown in Figure 6.1(d). Let $t_i^X > t_5^X$ be the time instant at node X, when the clone X' has broadcasted its timestamp-residual energy pair. Since the clocks are not

synchronized $t_i^X \neq t_j^{X'}$. Node Y performs the following actions, on receiving the timestamp-residual energy pair from X':

i) If $t_j^{X'} < t_4^X$ then X' is detected as clone. This is because, time is a monotonically increasing parameter and moves in the forward direction.

ii) If $t_j^{X'} > t_4^X$ and $E_{X'}(t_j^{X'}) > E_X(t_4^X)$ then X' is detected as clone. This is because, residual energy decreases with time. For $t_j^{X'} > t_4^X$, the residual energy of node X should be less than $E_X(t_4^X)$.

iii) If $t_j^{X'} > t_4^X$ and $E_{X'}(t_j^{X'}) < E_X(t_4^X)$ then node Y computes the expected time of node X as $t^Y + \beta_4$. If $|t_j^{X'} - t^Y| \approx \beta_4$ then the corresponding entry for node X in the energy table at node Y is updated. Since, clones are deployed with maximum energy, this condition will arise only when the timestamp-residual energy pair is received from the original node.

Case - II: *Adversary has the capability to synchronize the clone's clock with the original node.*

In this case, both the clone and its original node have almost equal clock timings. Let $t_i^X > t_5^X$ and $t_j^{X'}$ be the time instance of node X and its clone X' respectively. Then the condition $t_i^X \approx t_j^{X'}$ always holds true. Node Y performs the following action on receiving the timestamp-residual energy pair:

i) If $t_j^{X'} > t_4^X$ and $E_{X'}(t_j^{X'}) > E_X(t_4^X)$ then X' is detected as a clone. This is because, residual energy decreases with time. For $t_j^{X'} > t_4^X$, the residual energy of node X should be less than $E_X(t_4^X)$.

ii) If $t_j^{X'} > t_4^X$ and $E_{X'}(t_j^{X'}) < E_X(t_4^X)$, node Y computes the expected residual energy of X. Let E'_X be the expected residual energy of X and is computed as follows:

$$E'_X = E_X(t_4^X) - e * (t_j^{X'} - t_4^X) \tag{6.9}$$

where e is the minimum energy consumed by a node per unit time. The value of e is computed using the energy model proposed by Zhou *et al.* [67]. If $E_{X'}(t_j^{X'}) > E'_X$, then X' is detected as clone, else the energy table at node Y is updated. This is because, the clone is deployed with maximum energy. Therefore, its residual energy level will be higher than the expected energy level.

Algorithm 6.1: Replica detection for node X when clones are deployed with maximum energy

Input: Timestamp-residual energy pair $< t_j^X, E_X(t_j^X) >$
Output: Updated energy table or detected replica.

```
1  t_i^X ⟵ EnergyTable.getTime(X)
2  E_X(t_i^X) ⟵ EnergyTable.getResidualEnergy(X)
3  β ⟵ EnergyTable.getBeta(X)
4  if t_j^X < t_i^X then
5      X is detected as replica
6  else
7      if E_X(t_j^X > E_X)(t_i^X) then
8          X is detected as replica
9      else
10         /* t^Y - current time at monitoring node Y */
11         if t_j^X − t^Y ≈ β then
12             E'_X ⟵ E_X(t_i^X) − e ∗ (t_j^X' − t_i^X)
13             if E_X(t_j^X) > E'_X then
14                 X is detected as replica
15             else
16                 updateEnergyTable(X, < t_j^X, E_X(t_j^X) >)
17         else
18             X is detected as replica
```

The detection process using above scenario is shown in Algorithm 6.1.

B. *Energy levels of the captured node and/or clone is modified by the adversary.* In this scenario, the adversary attempts to deploy a clone such that the energy levels of the clone and its original node are almost equal. The adversary may do either of the following: *i)* She may recharge the captured node to maximum energy level and deploy both the clone and its original node with maximum energy level. However, the adversary may not adopt this strategy as it takes longer time for charging the battery; *ii)* She may decrease the energy level of the clone to nearly equal to that of the original node and deploy both at the same energy level. The following two cases may arise in this scenario:

Case - III: *Adversary deploys the clone and its original node with the maximum energy level.*

In this case, clones will be detected as explained in Case - I and Case - II.

Case - IV: *Adversary decreases the energy level of the clone to nearly equal to the original node.*

To detect replica, a suspicious counter is maintained by the monitoring nodes for each node they monitor. If the suspicious counter for a node exceeds a threshold value, then that node is detected as replica. In EBRD, nodes periodically broadcast their timestamp-residual energy pair. Let T_{update} be the broadcast interval between two consecutive timestamp-residual energy pair. Suppose node X has broadcast its timestamp-residual energy pair at t_4^X. Then node Y expects to receive the next timestamp-residual energy update from X at time $t^X = t_4^X + n.T_{update}$, for $n \geq 1$. Since, a number of replicas are present in the network, each one would send their timestamp-residual energy updates periodically. Assuming that the nodes are either loosely or not synchronized; the frequency of timestamp-residual energy update from X will be higher. A monitoring node Y will increment the suspicious counter for node X, if $t^X \neq t_4^X + n.T_{update}$. When the counter value exceeds a given threshold, then it is detected as replica.

The detection process for the above scenario is shown in Algorithm 6.2.

Algorithm 6.2: Replica detection for node X when energy level of the captured node and/or its clone is modified by the adversary.

Input: Timestamp-residual energy pair $< t_j^X, E_X(t_j^X) >$
Output: Updated energy table or detected replica.

```
1  t_i^X ⟵ EnergyTable.getTime(X)
2  E_X(t_i^X) ⟵ EnergyTable.getResidualEnergy(X)
3  if t_j^X < t_i^X then
4  |   X is detected as replica
5  else
6  |   if E_X(t_j^X > E_X)(t_i^X) then
7  |   |   X is detected as replica
8  |   else
9  |   |   if t_j^X ≠ (t_j^X + k.T_update) then
10 |   |   |   X.suspectCount ⟵ X.suspectCount + 1
11 |   |   |   if X.suspectCount > suspectThreshold then
12 |   |   |   |   X is detected as replica
13 |   |   |
14 |   |   else
15 |   |   |   updateEnergyTable(X, < t_j^X, E_X(t_j^X) >)
```

Case - V: *Adversary has the ability to synchronize the clock with the original node, and also decrease the energy level of the clone nearly equal to the original node.*

To detect a clone in this case, a monitoring node maintains k recently received timestamp-residual energy pair from each of the node it monitors. That is, if node A is monitored by node B, then the node B stores a timestamp-residual energy vector of length k for node A. This vector contains k recently received timestamp-residual energy pair from node A. We denote this vector as B^A. Each element of the vector B^A stores the timestamp-residual energy pair received from node A. Let B_i^A denotes the i^{th} element of the vector B^A, and $B_i^A.T$ and $B_i^A.E$ denotes the corresponding timestamp and residual energy respectively at B_i^A. When two nodes meet each other, they exchange the identity of nodes they monitor. If there exists one or more nodes monitored by both, then the vector associated with those nodes are also exchanged. A conflict in the timestamp-residual energy vector corresponding to a node at two different monitoring nodes is detected as a replica.

For example, suppose node A, B, C, and D are monitored by X, and D, E, F, and G are monitored by Y. When X and Y meet each other, they exchange the identity of their monitored nodes. Since, the node D is monitored by both, the vector X^D and Y^D corresponding to D at X and Y respectively are also exchanged. Every element of X^D is compared with every other element of Y^D at both X and Y. If a conflict is detected between any pair of elements, then D is detected as replica. In other word, D is detected as replica if any one of the following conditions hold true.

$$\begin{aligned}
(i)\ X_i^D.T = Y_j^D.T \text{ and } & \ X_i^D.E \neq Y_j^D.E \\
(ii)\ X_i^D.T < Y_j^D.T \text{ and } & \ X_i^D.E < Y_j^D.E \\
(iii)\ X_i^D.T > Y_j^D.T \text{ and } & \ X_i^D.E > Y_j^D.E \\
& \text{for any } i, j \leq k
\end{aligned}$$

This is because, if there exists only one node in the network, then the timestamp-residual energy pair broadcast by the node at any instant of time will always be same for every monitoring node receiving the broadcast message. There cannot be different images of the same timestamp-residual energy pair for different monitoring nodes. Algorithm 6.3 summarizes the above scenario of

replica detection.

Algorithm 6.3: Replica detection for node D at node X when nodes are synchronized and energy level of the captured node and/or its clone is modified by the adversary.

Input: Timestamp-residual energy vector Y^D from node Y.
Output: Node D is a replica or no operation.

```
1  X^D ⟵ Energy_Table.getEnergyVector(D)
2  for i ← 1 to k do
3      for j ← 1 to k do
4          if (X_i^D.T = Y_j^D.T) && (X_i^D.E ≠ Y_j^D.E) then
5              X is detected as replica
6              exit
7          if (X_i^D.T < Y_j^D.T) && (X_i^D.E < Y_j^D.E) then
8              X is detected as replica
9              exit
10         if (X_i^D.T > Y_j^D.T) && (X_i^D.E > Y_j^D.E) then
11             X is detected as replica
12             exit
13
14 no operation
15 exit
```

6.4 Analysis

We claim the following:

Claim 6.1. *If t^X_{dep} and $t^{X'}_{dep}$ are the deployment time of a node X, and its clone X' respectively, then the condition $E_{X'}(t) > E_X(t)$ for $t \geq t^{X'}_{dep} > t^X_{dep}$ is always true.*

Proof. Let

$$t = t^X_{dep} + \delta t^X, \text{ and} \tag{6.10}$$

$$t = t^{X'}_{dep} + \delta t^{X'} \tag{6.11}$$

Since, the replica X' is deployed after X, therefore, $t^X_{dep} < t^{X'}_{dep}$. Let

$$t^{X'}_{dep} = t^X_{dep} + \delta t_{diff} \tag{6.12}$$

The residual energy of node X and X' at time t^X_{dep} and $t^{X'}_{dep}$ respectively can be computed using Equation 6.10 and 6.11 as

$$E_X(t) = E_X(t^X_{dep}) - E^c_X(\delta t^X)$$

$$E_X(t_{dep}^X) = E_X(t) + E_X^c(\delta t^X) \tag{6.13}$$

Also,

$$\begin{aligned} E_{X'}(t) &= E_{X'}(t_{dep}^{X'}) - E_{X'}^c(\delta t^{X'}) \\ E_{X'}(t_{dep}^{X'}) &= E_{X'}(t) + E_{X'}^c(\delta t^{X'}) \end{aligned} \tag{6.14}$$

Now, using Equation 6.10 and 6.11, we can write

$$\begin{aligned} t_{dep}^X + \delta t^X &= t_{dep}^{X'} + \delta t^{X'} \\ \Rightarrow t_{dep}^X + \delta t^X &= t_{dep}^X + \delta t_{diff} + \delta t^{X'} \\ &\quad [\text{ Using Equation 6.12 }] \\ \Rightarrow \quad \delta t^X &= \delta t^{X'} + \delta t_{diff} \\ \Rightarrow \quad \delta t^X &> \delta t^{X'} \ [\delta t_{diff} > 0\ ] \\ \Rightarrow \quad E_X^c(\delta t^X) &> E_{X'}^c(\delta t^{X'}) \\ \Rightarrow \quad E_X^c(\delta t^X) &- E_{X'}^c(\delta t^{X'}) > 0 \end{aligned} \tag{6.15}$$

Since, nodes have equal energy at the time of its deployment, *i.e.*, $E_X(t_{dep}^X) = E_{X'}(t_{dep}^{X'})$

$$\begin{aligned} \Rightarrow E_X(t) + E_X^c(\delta t^X) &= E_{X'}(t) + E_{X'}^c(\delta t^{X'}) \\ \Rightarrow \quad E_{X'}(t) &= E_X(t) + E_X^c(\delta t^X) \\ &\quad - E_{X'}^c(\delta t^{X'}) \\ &\quad [\text{ Using Equation 6.13 }] \\ \Rightarrow \quad &> E_X(t) \\ &\quad [\text{ Using Equation 6.15}] \end{aligned} \tag{6.16}$$

Proved. □

6.4.1 Detection Probability

In this section, we analyze the probability of replica detection in EBRD. Let N be the network size, r be the number of replicas deployed in the network and the number of monitoring nodes any node is k. The probability of a node meeting a replica, P_{meet}, is equal to

$$P_{meet} = \frac{r}{N} \tag{6.17}$$

In EBRD, a replica is detected by the monitoring nodes. The probability P_{det} of detecting a replica is given by

$$\begin{aligned} P_{det} &= Pr[\text{ at least one of the } k \text{ monitoring nodes meet a replica}] \\ &= 1 - Pr[\text{ none of the } k \text{ monitoring nodes meet a replica}] \\ &= 1 - \binom{k}{0}(P_{meet})^0.(1-P_{meet})^k \\ &= 1 - (1-P_{meet})^k \\ &= 1 - \left(1 - \frac{r}{N}\right)^k \text{ [Using Equation 6.17]} \\ &\approx 1 - \exp^{-\frac{rk}{N}} \end{aligned} \tag{6.18}$$

□

6.4.2 Communication and Storage Overhead

In EBRD, each node periodically broadcasts its residual energy within one-hop neighbor. Total number of messages broadcast in a periodic interval is N. Therefore, the communication complexity of this scheme is $O(N)$. The storage overhead of EBRD depends on the number of monitoring nodes. According to the Birthday Paradox, $\sqrt{N}$ number of nodes are required, so that a node is monitored by at least two nodes [37]. Through simulation, we observed that $\frac{\sqrt{N}}{2}$ number of monitoring nodes are sufficient to successfully detect replicas. Also, detection probability do not change with increase in the number of monitoring nodes beyond $\frac{\sqrt{N}}{2}$. Therefore, the upper bound on storage overhead is $O(\sqrt{N})$. The comparison of communication and storage overhead of existing replica detection schemes in MWSN is shown in Table 6.3.

6.5 Simulation Results

We have simulated using Castalia-3.2 [54] simulator, that runs on Omnet++ simulation environment [55]. Parameters considered for simulation are summarized in Table 6.4. We have assumed that replicas are randomly deployed by the adversary. EBRD is simulated with two variants: *i)* Replicas are deployed with maximum energy, and we call this as EBRD_1; *ii)* Replicas are deployed with energy nearly equal to the residual energy of original node; we call this as EBRD_2. We have compared the performance of both EBRD_1 and EBRD_2 with a few existing schemes namely EDD [66] and MTLSD [64]. Metrics considered for evaluating the performance are:

Table 6.3: Comparison of communication and storage overhead.

Schemes	**Type**	**Communication**	**Storage**
XED [66]	Distributed	$O(1)$	$O(N)$
EDD [66]	Distributed	$O(1)$	$O(N)$
UTLSE & MTLSD [64]	Distributed	$O(N)$	$O(\sqrt{N})$
Ho *et al.* [65]	Centralized	$O(N\sqrt{N})$	$O(N)$
Deng *et al.* [57]	Centralized	$O(N\sqrt{N})$	-
EBRD	Distributed	$O(N)$	$O(\sqrt{N})$

Table 6.4: Simulation parameters.

Parameter	**Value**
Area	1000 x 1000 m^2
Network size	100 – 1000
Deployment type	Uniformly random
Communication range	20 meter
Pause time	5 sec
Node movement	Random way-point mobility model
Node speed range	2 to 10 meter/sec
Simulation time	800 sec

i) detection probability, *ii)* detection time, *iii)* number of packets sent/received, and *iv)* energy consumed per node.

The comparison of detection probability *vs.* network size is shown in Tables 6.5, 6.6, and 6.7 for 5, 10 and 20 number of replicas respectively. From the above tables it is observed that both EBRD_1 and EBRD_2 have higher detection probability in comparison to EDD and MTLSD. This is because, in EBRD the deployment of replicas results a conflict in the timestamp-residual energy pair of the captured node and this conflict is successfully detected by its monitoring nodes. However, in EDD and MTLSD, it depends on the threshold value of the number of times nodes meet each other, and node speed respectively. In EDD, replica detection depends on the number of meetings, which may vary with network density and node movement.

Table 6.5: Comparison of Detection Probability *vs.* Network size for the number of replicas equal to *Five*.

Network size	**Detection Probability**			
	EDD	**MTLSD**	**EBRD$_1$**	**EBRD$_2$**
100	0.92	1	1	1
200	0.92	1	1	1
300	0.80	1	1	0.84
400	0.80	1	1	0.96
500	0.80	1	1	0.92
600	0.80	0.90	1	0.96
700	0.76	0.90	1	0.84
800	0.60	0.90	1	0.96
900	0.60	0.80	1	0.96
1000	0.60	0.80	1	0.96

Table 6.6: Comparison of Detection Probability *vs.* Network size for the number of replicas equal to *Ten*.

Network size	**Detection Probability**			
	EDD	**MTLSD**	**EBRD$_1$**	**EBRD$_2$**
100	0.90	1	1	1
200	0.90	1	1	1
300	0.90	1	1	0.96
400	0.90	1	1	0.90
500	0.80	1	1	0.96
600	0.80	0.90	1	0.94
700	0.70	0.90	1	0.94
800	0.70	0.90	1	0.96
900	0.70	0.90	1	0.94
1000	0.70	0.80	0.90	0.94

Table 6.7: Comparison of Detection Probability *vs.* Network size for the number of replicas equal to *Twenty*.

Network size	**Detection Probability**			
	EDD	**MTLSD**	**EBRD$_1$**	**EBRD$_2$**
100	1	1	1	1
200	0.90	0.90	0.95	0.96
300	0.90	0.90	0.95	0.90
400	0.90	0.90	0.95	0.89
500	0.90	0.90	1	0.90
600	0.85	0.8	0.95	0.90
700	0.8	0.85	0.95	0.95
800	0.82	0.82	1	0.94
900	0.78	0.82	0.95	0.91
1000	0.78	0.8	0.95	0.91

In our simulation, we observed that a node meets another node for a maximum of *fifteen* number of times for a period of simulation run. Simulation is divided into number of periods. Each period is of 100 seconds. Therefore, we set the threshold value for the number of meetings in EDD to be *fifteen*. At this threshold value, we observed that the detection probability of EDD is lower than EBRD. In MTLSD, the threshold value to detect replica is the node speed, which is computed using the nodes last two consecutive locations. If the replicas move closer to the original node, then the calculated node speed may not exceed the maximum speed limit. As a result, the detection probability is marginally lower in MTLSD.

Figure 6.2 and 6.3 shows the plot for clone detection time *vs.* network size of EBRD$_1$ and EBRD$_2$ respectively. We have defined the clone detection time as the time to detect the first clone after its deployment. It is observed from the figure that for a given network size, the detection time decreases with increase in the number of clones. This is because, as the number of clones increases, the probability that a clone meet its monitoring node at the earliest time also increases. This results in the decrease of clone detection time. With increase in the number of nodes, the number of monitoring nodes for a given node also increases. As a result the probability of meeting a clone with its monitoring node at the earliest time also increases. This

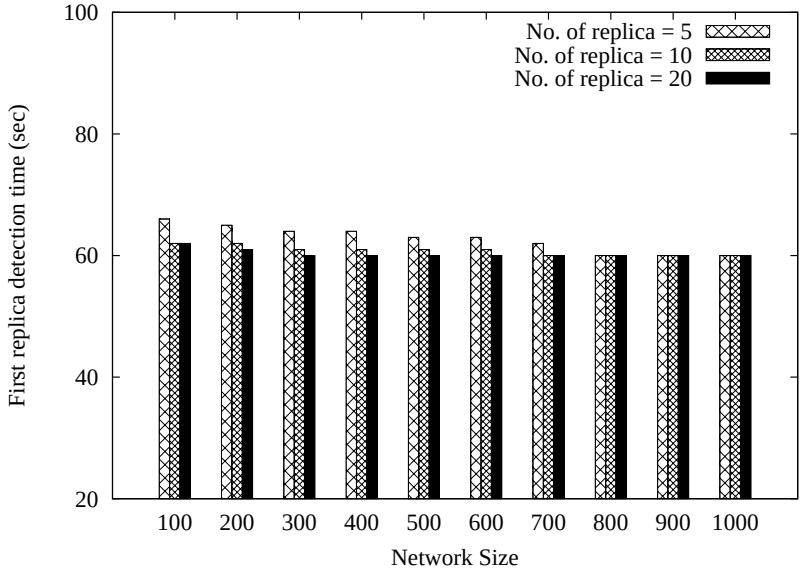


Figure 6.2: Detection time *vs.* Network size for $EBRD_1$.

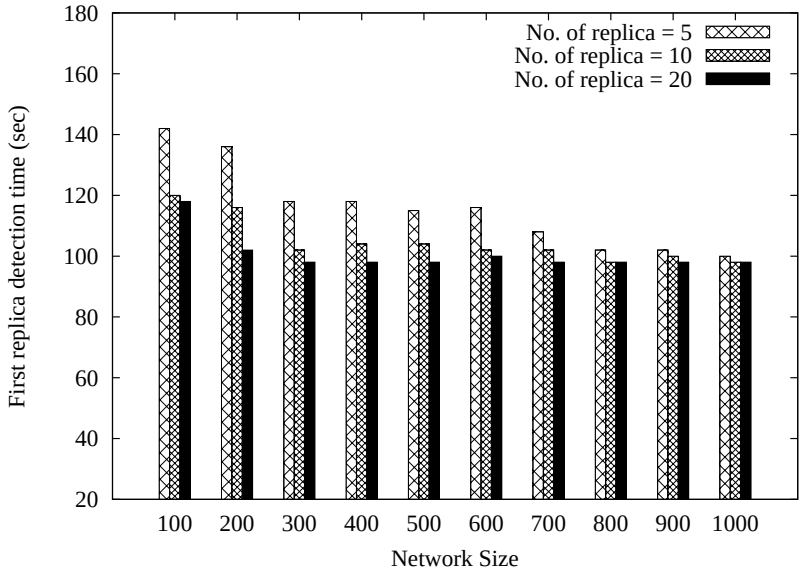


Figure 6.3: Detection time *vs.* Network size for $EBRD_2$.

contribute to the marginal decrease in detection time.

The plot for detection probability *vs.* time is shown in Figure 6.4. This shows the response of each scheme in detecting a clone. From the figure it is observed that the time to detect a clone is significantly lesser in $EBRD_1$ compared to EDD, MTLSD, and $EBRD_2$. This is because, when a clone is deployed with maximum

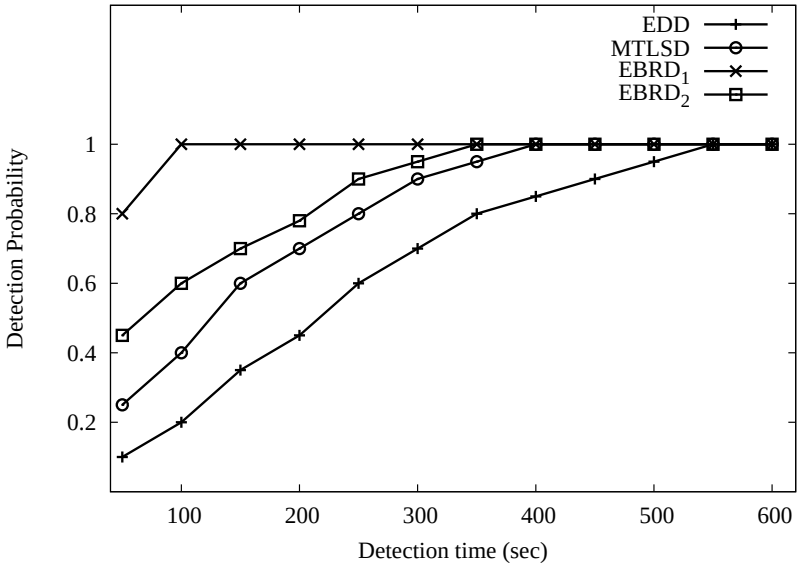


Figure 6.4: Comparison for detection probability *vs.* Detection time.

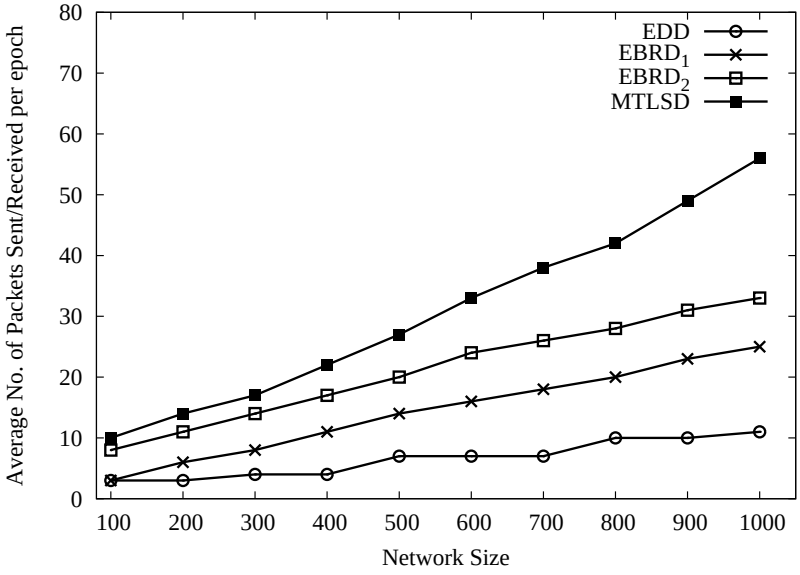


Figure 6.5: Average number of packets sent/received per epoch *vs.* Network size.

energy, its conflicting timestamp-residual energy pair is immediately detected by the monitoring node. EDD needs number of meetings with a node over a period to detect replica. In MTLSD, the diffusion of location-claim within tracker nodes requires comparatively more time than EBRD_1. In EBRD_2, a node is detected as replica when the suspicious counter value of that node reaches the maximum

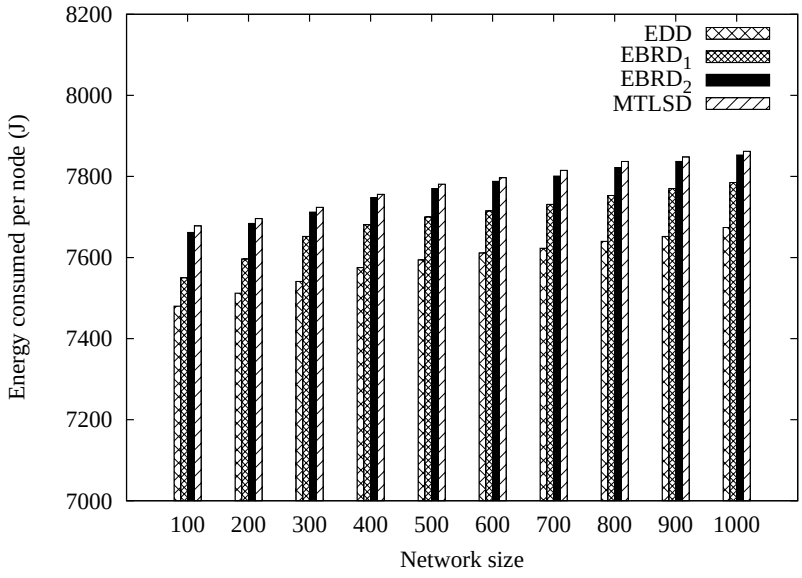


Figure 6.6: Energy consumed per node *vs.* Network size.

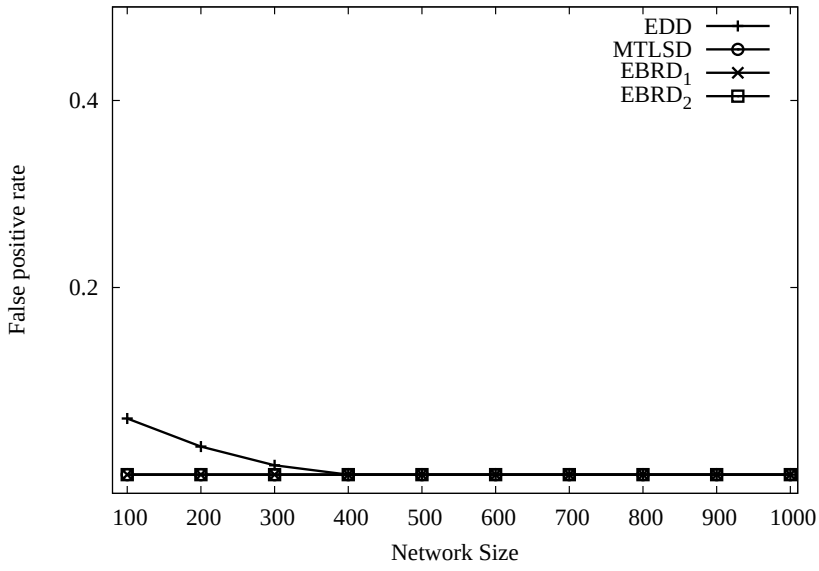


Figure 6.7: False positive rate *vs.* Network size.

threshold. This requires relatively more amount of time than $EBRD_1$.

Figure 6.5 shows the average number of packets sent/received by a node per epoch *vs.* network size. We have considered an epoch as defined in [64, 70]. It is observed from the figure that the average number of packets sent/received per epoch in both $EBRD_1$ and $EBRD_2$ is higher than EDD and lower than MTLSD. This is

because, the number of timestamp-residual energy pairs broadcast per epoch is lower than the number of messages exchanged for diffusion of time-location claim among the tracker nodes in MTLSD, and higher than the number of local connectivity messages per epoch in EDD.

The energy consumed per node in detecting a replica is shown in Figure 6.6. It is observed from the figure that EDD has the lowest energy consumption per node in replica detection. This is because, the energy consumption is proportional to communication overhead. As communication overhead of EBRD is higher than EDD and lower than MTLSD, the energy consumption of EBRD is higher than EDD and lower than MTLSD. It is also observed that the energy consumption in $EBRD_2$ is higher than $EBRD_1$ due to the additional exchange of timestamp-energy vector among the monitoring nodes.

Finally, the plot for false positive *vs.* network size is shown in Figure 6.7. It is observed that the schemes considered for comparison have almost *zero* false positive except EDD which have some cases of false positives for network size from 100 to 300.

6.6 Summary

In this chapter, we proposed a distributed replica detection scheme called energy based replica detection (EBRD). It is based on the residual energy level of nodes. As the time progresses the residual energy level of a node decreases. In EBRD each node is monitored by a set of nodes and each node also act as a monitoring node to a set of nodes. Replica is detected by the monitoring nodes. Conflict in the timestamp-residual energy pair of a node at the monitoring node is detected as replica. The proposed scheme is compared with EDD and MTLSD. The simulation results have shown that EBRD has detection probability higher than ninety percent. This shows that the residual energy level of a node can used to detect replica. The periodic broadcast of residual energy by each node is the only additional communication overhead incurred by EBRD. This communication overhead is significantly lower than MTLSD. But, it is higher than that of EDD which uses only the hello broadcast among the neighbors. The energy consumption is EBRD is relatively higher, because of the additional communication overhead in periodic broadcast of residual energy. This additional communication overhead and energy consumption can be reduced, if the residual energy is broadcasted along with the hello message.

Chapter 7

Conclusions

Work presented in this thesis focuses on the detection of node replication attack in WSN. Most of the replica detection schemes require exchange of information among the nodes. This necessitates an efficient mechanism for information exchange among the nodes. In this thesis, we proposed the followings: *i)* a mechanism for exchanging group membership information among the nodes, *ii)* a zone-based, and a node coloring based replica detection scheme for static WSN, and *iii)* an energy based replica detection scheme for mobile WSN.

We briefly summarize below the original contributions of this thesis, and also highlight the scope for future work.

7.1 Contributions

Nodes exchange their group membership identities for various purpose such as cluster formation, clone detection etc. For a larger and dense network, the exchange of group membership identities is an expensive process. Existing schemes make use of Bloom filter for exchanging membership information. Although, the Bloom filter is memory efficient yet, it suffers from higher probability of false positives. A node may be detected as a member of a group, when it is not a member of that group. In this thesis, we proposed two mechanisms called Transpose Bit-Pair Coding (TBC), and Sub-Mat Coding (SMC) for exchanging group membership information. In the above schemes, a bit-stream is generated from the membership matrix. This bit-stream is used for exchanging membership information among the nodes. The number of bits generated is significantly smaller than the size of membership matrix.

The performance of TBC and SMC are compared with two trivial schemes and Bloom filter. It was observed that TBC and SMC do not generate false positive,

and the space saving percentage is significantly higher compared to Bloom filter. This is because, TBC and SMC are deterministic schemes, whereas Bloom filter is a probabilistic scheme. SMC achieves better space saving percentage than TBC. This is because, SMC efficiently utilizes the sparseness of the membership matrix than TBC. However, the space saving percentage of TBC and SMC decreases, as the group size increases. In the worst case, the size of the bit-stream will be closer to the size of the membership matrix. The proposed scheme requires a bit matrix to store the membership information. Size of the bit matrix is less than the storage overhead associated with other mechanisms that uses the standard format to store membership identities.

Most of the node replica detection schemes that are reported in the literature are location dependent. They use probabilistic approach to forward the location claim. It is difficult to guarantee a clone-free sensor network using probabilistic claim forwarding approach. Location dependent schemes required either GPS enabled node or localization technique to determine node's position. A GPS enabled node increases the network cost. Whereas, a localization technique increases computational and storage overhead. Location independent schemes can reduce the cost and computational overhead associated with location dependent schemes.

Zone-based node replica detection (ZBNRD), Node Coloring Based Replica Detection (NCBRD), Energy Based Replica Detection schemes proposed in this thesis are location independent.

In ZBNRD, network is divided into number of zones. Each zone has a zone-leader and they are responsible for replica detection. It was observed that the ZBNRD have higher detection probability compared to other schemes. This is because, a node belongs to exactly *one* zone. Existence of node identity in more than one zone is detected as replica. Though ZBNRD has higher detection probability, it has some limitations. Communication overhead of ZBNRD increases for higher number of zones, and for larger network size. The end-to-end delay increases, when the number of zones are smaller.

In NCBRD, each node is assigned with a color, which is unique within their neighborhood. A color conflict in NCBRD is detected as a replica. NCBRD outperforms schemes such as RED [32], LSM [36], and SET [25] in terms of detection probability, communication, and storage overhead. Coloring and detection mechanism is performed within two-hop neighbors; therefore, the average path length in NCBRD is lower than other schemes. NCBRD has following disadvantages: *i)* It is not possible to add a new node at a later point of time, *ii)* a small color set may

result in the loss of neighboring links, and may partition the network. To ensure minimal link-loss, the color set must be chosen judiciously.

The above two schemes for replica detection are for static WSN. They cannot be used in MWSN due to node mobility. Energy Based Replica Detection (EBRD) scheme is proposed for MWSN. This scheme is based on the residual energy level of nodes. Each node in EBRD is monitored by a set of nodes, and also acts as a monitoring node to a set of nodes. Replica is detected by the monitoring nodes. Conflict in the timestamp-residual energy pair of a node at the monitoring node is detected as replica. The proposed scheme is compared with EDD and MTLSD. It was observed that the EBRD has the detection probability nearly equal to *one*. Though EBRD has higher detection probability, yet it suffers from higher communication overhead, and has higher detection time than comparative schemes. There is no significant improvement in energy consumption per node.

7.2 Future Directions of Research

We briefly outline the possible future extensions to our work.

The encoding and decoding process of TBC and SMC can be improved further to save space without generating false positives. New data structures may be suggested which will require lesser number of bits to store group membership information.

The communication path length among the zone-leaders in ZBNRD may increase if the number of zones in the network are relatively less. Therefore, a suitable model is needed to optimize the number of zones and path length. Selecting an efficient storage mechanism such as the digest mentioned in [44] may reduce the storage cost.

The NCBRD mechanism can be extended further to establish a trade-off between link-loss percentage and storage cost. The provision for inclusion of new nodes into the network after the coloring process is not considered in NCBRD. A secured mechanism for introducing a new node into the network after coloring process will make the scheme more realistic.

The energy-update mechanism in EBRD can be further modified to reduce the communication cost. A new energy depletion model can be proposed to compute the expected residual energy of a node, more accurately.

Bibliography

[1] Kazem Sohraby, Daniel Minoli, and Taieb Znati. *Wireless Sensor Networks Technology, Protocols, and Applications*. John Wiley and Sons, 2007.

[2] Holger Karl and Andreas Wilig. *Protocols and Architectures for Wireless Sensor Netowks*. John Wiley and Sons, 2005.

[3] Gregory J Pottie and William J Kaiser. Wireless Integrated Network Sensors. *Communications of the ACM*, 43(5):51 – 58, May 2000.

[4] Elliott D. Kaplan. *Understanding GPS: Principles and Applications*. Artech House Inc., Boston, MA, 1996.

[5] Antonio Caruso, Stefano Chessa, Swades De, and Alessandro Urpil. GPS Free Coordinate Assignment and Routing in Wireless Sensor Networks. In *Proceedings of 24th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM' 05*, volume 1, pages 150 – 160, March 2005.

[6] Frank L. Lewis. Wireless Sensor Networks. In *Smart Environments: Technologies, Protocols, and Applications*, pages 11 – 46. John Wiley & Sons, 2005.

[7] Institute of Electrical and Electronics Engineers Inc. IEEE Standard for Information technology - Local and Metropolitan Area Networks - Specific Requirements - Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs). *IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003)*, pages 01 – 320, July 2006.

[8] Chiara Buratti, Andrea Conti, Davide Dardari, and Roberto Verdone. An Overview on Wireless Sensor Networks Technology and Evolution. *Sensors*, 2009(9):6869 – 6896, August 2009.

[9] ZigBee Alliance. ZigBee Specifications, http://www.zigbee.org/Specifications.aspx.

[10] IEEE 1451 Standards, http://www.nist.gov/el/isd/ieee/ieee1451.cfm.

[11] Chee-Yee Chong and Srikanta P. Kumar. Sensor Networks: Evolution, Opportunities, and Challenges. *Proceedings of the IEEE*, 91(8):1247 – 1256, August 2003.

[12] Yong Wang, Garhan Attebury, and Byrav Ramamurthy. A Survey of Security Issues in Wireless Sensor Networks. *IEEE Communications Surveys and Tutorials*, 8(2):02–23, Second Quarter 2006.

[13] Adrian Perrig, John Stankovic, and David Wagner. Security in Wireless Sensor Networks. *Communications of the ACM*, 47(6):53–57, June 2004.

[14] John Paul Walters, Zhengqiang Liang, Weisong Shi, and Vipin Chaudhary. *Wireless Sensor Network Security: A Survey, In Security in Distributed Grid and Pervasive Computing*, chapter 17. Auerbach Publications, CRC Press, 2006.

[15] Chris Karlof and David Wagner. Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures. *Ad Hoc Networks*, 1(2 and 3):293–315, September 2003.

[16] Wazir Zada Khan, Mohammed Y. Aalsalem, Mohammed Naufal Bin Mohammed Saad, and Yang Xiang. Detection and Mitigation of Node Replication Attacks in Wireless Sensor Networks: A Survey. *International Journal of Distributed Sensor Networks*, 2013(2013):01 – 22, 2013.

[17] B. Gowtham and S. Sharmila. Location Traced Hybrid Detection of Node Replication Attack in Mobile Wireless Sensor Network. *IJCA Special Issue on Information Processing and Remote Computing*, IPRC(1):12 – 15, August 2012.

[18] Wen Tao Zhu, Jianying Zhou, Robert H. Deng, and Feng Bao. Detecting Node Replication Attacks in Wireless Sensor Networks: A survey. *Journal of Network and Computer Applications*, 35(3):1022 – 1034, May 2012.

[19] David K. Goldenberg, Arvind Krishnamurthy, Wcsley C. Maness, Yang Richard Yang, and Anthony Young. Network Localization in Partially Localizable Networks. In *Proceedings of 24th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM' 05*, volume 1, pages 313 –326. IEEE, March 2005.

[20] Katayoun Sohrabi, Jay Gao, Vishal Ailawadhi, and Gregory J Pottie. Protocols for Self-Organization of A Wireless Sensor Network. *IEEE Personal Communications*, 7(5):16 – 27, 2000.

[21] Brad Karp and H T Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, MobiCom' 00*, pages 243 – 254. ACM, August 2000.

[22] Bo Zhu, Sanjeev Setia, Sushil Jajodia, Sankardas Roy, and Lingyu Wang. Localized Multicast: Efficient and Distributed Replica Detection in Large-Scale Sensor Networks. *IEEE Transactions on Mobile Computing*, 9(7):913–926, July 2010.

[23] Wassim Znaidi, Marine Minier, and Stephane Ubeda. Hierarchical Node Replication Attacks Detection in Wireless Sensor Networks. *International Journal of Distributed Sensor Networks*, 2013(2013):01 – 12, February 2013.

[24] Chano Kim, Chanil Park, Junbeom Hur, Hanjin Lee, and Hyunsoo Yoon. A Distributed Deterministic and Resilient Replication Attack Detection Protocol in Wireless Sensor Networks. In *Communications in Computer and Information Science*, volume 56, pages 405 – 412. Lecture Notes in Computer Science, Springer Berlin Heidelberg, December 2009.

[25] Heesook Choi, Sencun Zhu, and Thomas F. La Porta. SET: Detecting Node Clones in Sensor Networks. In *Proceedings of Third International Conference on Security and Privacy in Communications Networks and the Workshops, SecureComm' 07*, pages 341 – 350. IEEE, September 2007.

[26] Wibhada Naruephiphat, Yusheng Ji, and Chalermpol Charnsripinyo. An Area-Based Approach for Node Replica Detection in Wireless Sensor Networks. In *Proceedings of 11th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 745 – 750. IEEE, June 2012.

[27] Kai Xing, Fang Liu, Xiuzhen Cheng, and David H.C. Du. Real-time Detection of Clone Attacks in Wireless Sensor Networks. In *Proceedings of The 28th International Conference on Distributed Computing Systems, ICDCS' 08*, pages 17 – 20. IEEE, June 2008.

[28] A. J. Macula. A Simple Construction of d-Disjunct Matrices with Certain Constant Weights. *Discrete Mathematics*, 162(1):311 – 312, December 1996.

[29] Lee-Chun Ko, Hung-Yuan Chen, and Guan-Rong Lin. A Neighbor-Based Detection Scheme for Wireless Sensor Networks Against Node Replication Attacks. In *Proceedings of International Conference on Ultra Modern Telecommunications and Workshops, ICUMT' 09*, pages 01 – 06. IEEE, October 2009.

[30] Sylvia Ratnasamy, Brad Karp, Li Yin, Fang Yu, Deborah Estrin, Ramesh Govindan, and Scott Shenker. GHT: A Geographic Hash Table for Data-Centric Storage. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications, WSNA' 02*, pages 78 –87. ACM, September 2002.

[31] Xiangshan Meng, Kai Lin, and Keqiu Li. A Note-Based Randomized and Distributed Protocol for Detecting Node Replication Attacks in Wireless Sensor Networks. In *Algorithms and Architectures for Parallel Processing*, pages 559 – 570. Lecture Notes in Computer Science, Springer Berlin Heidelberg, May 2010.

[32] Mauro Conti, Roberto Di Pietro, Luigi Vincenzo Mancini, and Alessandro Mei. Distributed Detection of Clone Attacks in Wireless Sensor Networks. *IEEE Transactions on Dependable and Secure Computing*, 8(5):685–698, September-October 2011.

[33] Andrei Broder and Michael Mitzenmacher. Network Applications of Bloom Filters: A Survey. *Internet Mathematics*, 1(4):636–646, June 2002.

[34] Michael Mitzenmacher. Compressed Bloom Filters. *IEEE Transactions on Networking (TON)*, 10(5):604–612, October 2002.

[35] Dawei Xia and Natalija Vlajic. Near-Optimal Node Clustering in Wireless Sensor Networks for Environment Monitoring. In *Proceedings of Canadian Conference on Electrical and Computer Engineering, CCECE' 06*, pages 1825 – 1829, May 2006.

[36] Bryan Parno, Adrian Perrig, and Virgil Gligor. Distributed Detection of Node Replication Attacks in Sensor Networks. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 49–63. IEEE, May 2005.

[37] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, Second edition, 2001.

[38] Chakib Bekara and Maryline Laurent-Maknavicius. A New Protocol for Securing Wireless Sensor Networks Against Nodes Replication Attacks. In *Proceedings of Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, WiMOB' 07*, pages 59 – 66. IEEE, October 2007.

[39] Yuichi Sei and Shinichi Honiden. Distributed Detection of Node Replication Attacks Resilient to Many Compromised Nodes in Wireless Sensor Networks. In *Proceedings of the 4th Annual International Conference on Wireless Internet, WICON' 08*, pages 01 – 08. ACM, November 2008.

[40] Ming Zhang, Vishal Khanapure, Shigang Chen, and Xuelian Xiao. Memory Efficient Protocols for Detecting Node Replication Attacks in Wireless Sensor Networks. In *Proceedings of 17th IEEE International Conference on Network Protocols, ICNP' 09*, pages 284–293. IEEE, October 2009.

[41] Jun-Won Ho, Donggang Liu, Matthew Wright, and Sajal K. Das. Distributed Detection of Replica Node Attacks with Group Deployment Knowledge in Wireless Sensor Networks. *Ad Hoc Networks*, 7(8):1476 – 1488, November 2009.

[42] Zhijun Li and Guang Gong. Randomly Directed Exploration: An Efficient Node Clone Detection Protocol in Wireless Sensor Networks. In *Proceedings of IEEE 6th International Conference on Mobile Adhoc and Sensor Systems, MASS' 09.*, pages 1030 – 1035. IEEE, October 2009.

[43] Thanh Dai Tran and Johnson I. Agbinya. Early and Lightweight Distributed Detection of Node Replication Attack in Sensor Networks. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, pages 01 – 06. IEEE, 2010.

[44] Yingpei Zeng, Jiannong Cao, Shigeng Zhang, Shanqing Guo, and Li Xie. Random-walk Based Approach to Detect Clone Attacks in Wireless Sensor Networks. *IEEE Journal on Selected Areas in Communications*, 28(5):677 – 691, June 2010.

[45] Ralph C Merkle. A Digital Signature Based on A Conventional Encryption Function. In *Advances in Cryptology, CRYPTO' 87*, volume 293, pages 369 – 378. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1988.

[46] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying Hash Functions for Message Authentication. In *Advances in Cryptology, CRYPTO' 96*, volume

1109, pages 01 – 15. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1996.

[47] Carlo Blundo, Alfredo De Santis, Amir Herzberg, Shay Kutten, Ugo Vaccaro, and Moti Yung. Perfectly-Secure Key Distribution for Dynamic Conferences. In *Advances in Cryptology, CRYPTO' 92*, volume 740, pages 471 – 486. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1992.

[48] Adi Shamir. Identity-Based Cryptosystems and Signature Schemes. In *Advances in Cryptology, CRYPTO' 84*, volume 196, pages 47 – 53. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1985.

[49] Clifford Cocks. An Identity Based Encryption Scheme Based on Quadratic Residues. In *Cryptography and Coding*, volume 2260, pages 360 – 363. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2001.

[50] Florian Hess. Efficient Identity Based Signature Schemes Based on Pairings. In *Selected Areas in Cryptography*, volume 2595, pages 310 – 324. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2003.

[51] An Liu and Peng Ning. TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks. In *Proceedings of International Conference on Information Processing in Sensor Networks, IPSN' 08.*, pages 245 – 256. IEEE, April 2008.

[52] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia. LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks. In *Proceedings of the 10th ACM Conference on Computer and Communication Security, CCS' 03*, pages 62 – 72. ACM, October 2003.

[53] Seog Chung Seo, Dong-Guk Han, Hyung Chan Kim, and Seokhie Hong. Perfectly-Secure Key Distribution for Dynamic Conferences. *IEICE Transactions*, 91-D(5):1338 – 1347, 2008.

[54] Athanassios Boulis. *Castalia 3.2, User's Manual*. National ICT Australia Ltd., Australia, 2011.

[55] Andrs Varga and Rudolf Hornig. An Overview of The Omnet++ Simulation Environment. In *Proceedings of 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems and Workshops, SimuTools' 08*, pages 01–10, 2008.

[56] M. Jimeno, K.J. Christensen, and A. Roginsky. Two-tier Bloom Filter to Achieve Faster Membership Testing. *Electronics Letters*, 44(7):503 – 504, March 2008.

[57] Xiao-Ming Deng and Yan Xiong. A New Protocol for the Detection of Node Replication Attacks in Mobile Wireless Sensor Networks. *Journal of Computer Science and Technology*, 26(4):732 – 743, July 2011.

[58] Larry Nyhoff. *Implementing Sets with Bitsets*, chapter 9. Prentice Hall, 2nd edition, 2004.

[59] Dietmar Schtz. Bitstream X-Coder. In *Proceedings of the 13th European Conference on Pattern Languages of Programs (EuroPLoP' 08)*, pages 01 – 08, 2008.

[60] Tossaporn Srisooksai, Kamol Keamarungsi, Poonlap Lamsrichan, and Kiyomichi Araki. Practical Data Compression in Wireless Sensor Networks: A survey. *Journal of Network and Computer Applications*, 35(1):37 – 59, January 2012.

[61] Jonathan Gana Kolo, S. Anandan Shanmugam, David Wee Gin Lim, Li-Minn Ang, and Kah Phooi Seng. An Adaptive Lossless Data Compression Scheme for Wireless Sensor Networks. *Journal of Sensors*, 2012:01 – 20, 2012.

[62] N. Erratt and Y. Liang. Compressed Data-stream Protocol: An Energy-efficient Compressed Data-stream Protocol for Wireless Sensor Networks. *IET Communications*, 5(18):2673 – 2683, December 2011.

[63] Chunsheng Zhu, Lei Shu, Takahiro Hara, Lei Wang, Shojiro Nishio, and Laurence T. Yang. A Survey on Communication and Data Management Issues in Mobile Sensor Networks. *Wireless Communications and Mobile Computing*, 12(16):01 – 18, 2011.

[64] Xiaoming Deng, Yan Xiong, and Depin Chen. Mobility-Assisted Detection of the Replication Attacks in Mobile Wireless Sensor Networks. In *Proceedings of IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 225 – 232. IEEE, October 2010.

[65] Jun-Won Ho, Matthew Wright, and Sajal K. Das. Fast Detection of Mobile Replica Node Attacks in Wireless Sensor Networks Using Sequential Hypothesis Testing. *IEEE Transactions on Mobile Computing*, 10(6):767 – 782, June 2011.

[66] Chia-Mu Yu, Yao-Tung Tsou, Chun-Shien Lu, and Sy-Yen Kuo. Localized Algorithms for Detection of Node Replication Attacks in Mobile Sensor Networks. *IEEE Transactions on Information Forensics and Security*, 8(5):754 – 768, May 2013.

[67] Hai-Ying Zhou, Dan-Yan Luo, Yan Gao, and De-Cheng Zuo. Modeling of Node Energy Consumption for Wireless Sensor Networks. *Journal of Wireless Sensor Network*, 3(1):18 – 23, January 2011.

[68] Alcides Montoya and Demetrio Ovalle. Energy Consumption by Deploying a Reactive Multi-Agent System Inside Wireless Sensor Networks. In *Innovations and Advances in Computer, Information, Systems Sciences, and Engineering*, volume 152 of *Lecture Notes in Electrical Engineering*, pages 925–934. Springer New York, 2013.

[69] Muhammad Tariq, Martin Macuha, Yong-Jin Park, and Takuro Sato. An Energy Estimation Model for Mobile Sensor Networks. In *Proceedings of Fourth International Conference on Sensor Technologies and Applications (SENSORCOMM)*, pages 507 – 5012, July 2010.

[70] Christian Bettstetter, Hannes Hartenstein, and Xavier P'erez-Costa. Stochastic Properties of the Random Waypoint Mobility Model: Epoch Length, Direction Distribution, and Cell Change Rate. In *Proceedings of the 5th ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems, MSWiM' 02*, pages 07 – 14. ACM, September 2002.