New Distributed Byzantine Fault Detection & Data Integrity Scheme for WANET

by

Abdurrahman Elbuni

A thesis

presented to the University of Waterloo

in fulfillment of the

thesis requirement for the degree of

Master of Applied Science

in

Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2017

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Wireless ad-hoc networks (WANET) with multi-hop communication are subject to a variety of faults and attacks, and detecting the source of any fault is highly important to maintain the quality of service, confidentiality, and reliability of an entire network operation. Intermediate byzantine nodes in WANET could subvert the system by altering sensitive routed information unintentionally due to many reasons such as power depletion, software bug, malware, and environmental obstacles. This thesis highlights some of the research studies done in the area of distributed fault detection (DFD) and proposes a solution to detect Byzantine behavior cooperatively. The present research will focus on designing a scalable distributed fault detection (DFD) algorithm to detect byzantine nodes who permanently try to distort or reroute information while relaying a message from one node to another, complimentary to that, a symmetric distributed cryptography scheme will be employed to continuously validates the data integrity of a routed message.

The main hypothesis of the research is that if a wireless ad-hoc network is been divided into N number of groups (classes) with relatively equal number of members, each group of nodes can cooperatively protect the network from every other group. Practically, each group of nodes will be assigned to a distinct shared key; nodes with similar group assignment shall guard the integrity of a routing path by incorporating their own secret message authentication code (MAC) that can be only validated by nodes belonging to the same group contributing to the same routing path. If a node from $Group(i)$ detects a tampering event, it should either store and delay a fault report or embed a fault report to the same routed message and forward it to the Master Node (Destination) if applicable.

Further report message overhead optimization has been devised to reduce the energy cost. Moreover, the empirical results have shown that the more reported evidence the master node can collect, the more accuracy of detection can be reached based on an incremental stream of evidence that contains information about both healthy and unhealthy nodes; so that every healthy report type can justify the unhealthy false report. The heuristic simulation based study considered many different aspects of the system for evaluation such as detection accuracy, fault model, the optimal number of classes, energy consumption, the impact of mobility, and network lifetime. The iGraph network simulation tool has been employed for visualization and graph manipulation, whereas, Python programming language has been utilized in conjunction to implement and simulate the DFD algorithm and generate the results.

Acknowledgements

My thanks and praise first and foremost go to Almighty God for giving me the knowledge, opportunity, and the strength to accomplish this work.

Along the way, several people deserve sincere recognition. I would like first to extend my sincere thanks and gratitude to my advisor, Dr. Otman Basir, for his valuable advice and guidance and for his generous help. His constant support was the secret of achieving all this work. I would like also to extend my appreciation and gratitude to my thesis committee members, Dr. Slim Boumaiza, and Dr. Sagar Naik for sparing their time for reviewing this thesis report.

I would like also to acknowledge the Libyan Ministry of Higher Education and Scientific Research for supporting me financially and logistically to finish this research work.

I would like to thank all my family members. My parents, Dr. Mohamed Samir Elbuni and Awatef Younes, and my brothers and sisters, Ahmed, Anas, Asma, and Ala, who have been the greatest supporters in my life. I am grateful to them for their unconditional love, and support.

A heartful gratitude goes to my beloved wife Masarra for the endless understanding, support, patience and care showed during this important period in my career. Despite her busy life; teaching at school, raising our beautiful children Joud & Omar, and being thousands of miles away from her family, she has always been available to uplift me and congratulate me whenever I achieve a goal. I am by all means indebt to her forever.

Dedication

This thesis is dedicated to Allah, the Most Gracious, the Most Merciful

I praise him for his guidance and blessings.

Table of Contents

List of Tables xi

List of Figures xii

1 Introduction 1

2 Background 4

2.1 WANET Applications 4

2.1.1 Wireless Sensor Network 5

2.1.2 MANET and VANET 5

2.2 Routing in WANET 8

2.2.1 Proactive routing 8

2.2.2 Reactive routing 9

2.3 Distributed Fault Detection in Wireless Sensor Networks 10

2.4 Security Threats in Wireless Ad Hoc Networks 11
2.4.1 Wireless Medium 12
2.4.2 Ad-Hoc Deployment 12
2.4.3 Hostile Environment 12
2.4.4 Resource Scarcity 13
2.4.5 Immense Scale 13
2.4.6 Unattended Operation 13
2.5 Symmetric Cryptography for Confidentiality and Data Integrity 14
2.5.1 Data Confidentiality (Encryption) 15
2.5.2 Data Authentication (MAC) 15
2.5.3 MAC Algorithm Performance for Constrained devices 16
2.6 Byzantine Faults in WANET 16
2.6.1 Byzantine Problem 17
2.6.2 The importance of fault detection 18
2.7 Literature Review 19
2.7.1 Related optimization algorithms 19
2.7.2 Distributed Fault Detection in WANET using localized communication 20
2.7.3 Distributed Fault Detection using statistical learning 21
2.7.4 Distributed Fault Detection using cryptography primitives 22

3 DFD Algorithm **24**

3.1 The Genesis of Self-Policing Ad-Hoc Network Method 24

3.2 Distributed graph coloring algorithm as a candidate solution 28

3.3 Proposed Distributed Fault Detection Process 32

3.3.1 Algorithm Stages 39

3.4 Continues Detection and Update Process 42

3.4.1 Continues Acquittal 43

3.4.2 Report Packet Optimization 45

3.4.3 Decision Making 47

3.5 Security Challenges 47

3.5.1 Node Collusion 47

3.5.2 Probabilistic Byzantine Attack 48

3.5.3 Group key secrecy and freshness 48

4 Simulation and Experimental Results **51**

4.1 Simulation Tools and Technologies 51

4.2 Radio Model and Energy Calculation 52

4.3 Network Model 55

4.4 Fault Model 56

4.5 Mathematical Formulation 57
4.6 Results and Analysis using Heuristic Techniques 58
4.6.1 System Assumptions 58
4.6.2 Accuracy relation with number of byzantine nodes 60
4.6.3 Identifying the Sufficient Segmentation Value 65
4.6.4 Network lifetime and power consumption 70
4.6.5 Message Overhead 72
4.6.6 Algorithm Comparison 72

5 Python Code for the DFD Algorithm **74**
5.1 Simulator Code in Github 74
5.2 Sample Experimentation Code 74

6 Conclusion and Future Work **77**
6.1 Summary and Contributions 77
6.2 Future work 81

References **82**

List of Tables

4.1 Simulation results matrix for Distributed Fault Detection Accuracy 65

4.2 Comparison between various Distributed Fault Detection algorithms 73

List of Figures

1.1 Possible Byzantine faults in WSN: A) Message diversion; B) Message distortion of tempering; C) Flooding attack; D) Routing loop. 2

2.1 Routing messages using multi-hop communication. 7

2.2 Energy consumption of MAC algorithms based on AES-128 on a TelosB sensor mote. 17

3.1 The notion behind applying Graph Coloring is to be able to make each class guard the network from every other class 26

3.2 The notion behind applying Graph Coloring is to be able to make each class guard the network from every other class 29

3.3 Segmenting network into three different groups (classes) 33

3.4 Random network deployment after segmentation 33

3.5 Probability of finding a path consist of nodes belonging to the same group. Network size N=300, and S=3 36

3.6 Probability of finding a path consist of nodes belong to same group 37

3.7 Random 10x10 network deployment with 3 different groups (classes) . . . 38

3.8 Data integrity calculation and validation process, assuming that the network is divided into two classes. 40

3.9 Data integrity calculation and validation process, assuming that the network is divided into two classes. 41

3.10 First scene, shows two messages been routed respectively from (source1 and source2) to the destination. Four faulty nodes has been detected during communication, and one faulty node has been intersected and reported by two different paths. 43

3.11 Second scene, shows a third message initiated from $Src3$ constructed a new relaying path routing a message to *Destination*; the new path highlighted in blue has given an acquittal to a node that formerly was tagged as Byzantine node from a path highlighted in orange 3.10. 44

4.1 Network and Routing Simulation Tools 51

4.2 Radio energy consumption model 52

4.3 20x20 Random graph network representation with 1/3 of the total number of nodes been randomly distributed as Byzantine faulty nodes 54

4.4 Presenting different set of network typologies that can be used for simulation 55

4.5 Accuracy measurement of Distributed Fault Detection (DFD) in network size of 400 with 50 Byzantine nodes. Using 5 different classes with variant communication hop length 61

4.6 DFD with 100% accuracy. Graph visualization for the network with 50 detected Byzantine nodes, aided by 3 different class segmentation, and an average contribution of 37.5 routed message per node. Triangle shape means that are proven to be healthy 62

4.7 Accuracy measurement of Distributed Fault Detection (DFD) in network size of 400 with 133 Byzantine nodes. Using 5 different classes with variant communication hop length 63

4.8 Accuracy measurement of 91% for the DFD, having 133 byzantine nodes, with 3 classes, and an average contribution of 37.5 message per node. Normal nodes highlighted in square shapes are falsely tagged as byzantine by the algorithm 64

4.9 Probability of finding a path consist of nodes belonging to the same group. Network size N=400, and S=3 66

4.10 Heat-map graph visualization that depict the performance of detection using different number of classes 67

4.11 DFD with 95% accuracy. Graph visualization for a network with 1600 nodes and 533 simulated Byzantine nodes, aided by 3 different class segmentation, and an average contribution of 42 routed message per node. 69

4.12 Cumulative power consumption for the entire network (400 Nodes / 133 Byz / 3 Classes / 68,010 messages) 70

4.13 Network lifetime graph (400 Nodes / 133 Byz / 3 Classes / 68,010 messages) 71

4.14 Metadata overhead size graph with average size of 50 bytes per route for a large network with the following specifications (400 Nodes / 133 Byz / 3 Classes / 68,010 messages) 72

Chapter 1

Introduction

A wireless ad hoc network (WANET) is an uncentralized wireless network. The network is ad hoc because it does not rely on a dedicated infrastructure, such as routers, and switches in wired networks or access points in managed (infrastructure) wireless networks. In contrast, each node participates in routing by transmitting data for other nodes, so the determination of which nodes transmitting data is made dynamically by network connectivity. In addition to the classic routing, ad-hoc networks can use broadcasting to forward information.

Over the past two decades distributed wireless sensor networks (WSN) have been the focus of considerable research for both civil and military applications. Sensors are constrained in onboard energy supply. Therefore, efficient management of the network is crucial to extend the life of the system. Failures and disturbance are inevitable in wireless sensor networks executing in hostile environments and due to their unattended deploy-

ment. Faulty sensors or data distortion can cause network infrastructure to needlessly spread erroneous or even totally false information through the entire system. Such events usually cost additional power consumption of key elements of WSN (e.g. Cluster Heads). The main aim of this thesis is to explore and propose a distributive algorithm that can ensure data integrity for large-scale WSN in the presence of uncertainties and disturbances. The algorithm proposed should be capable of addressing any Wireless Ad Hoc Network (WANET) that employs multi-hop communication protocol; moreover, it should compliment reducing and isolating the influence of nodes that function incorrectly to tamper, or inject, or divert critical routed information. See Figure (1.1).

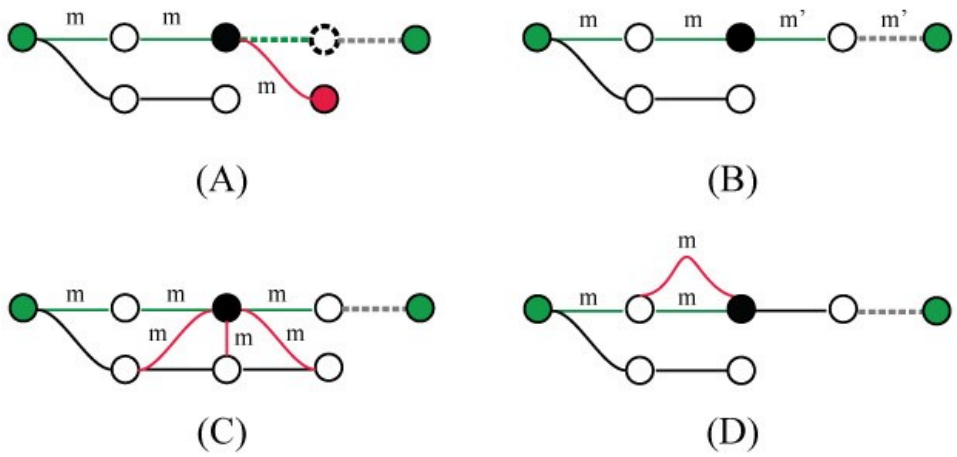

Figure 1.1: Possible Byzantine faults in WSN: A) Message diversion; B) Message distortion of tempering; C) Flooding attack; D) Routing loop.

A significant threat to WANET is Byzantine attacks where some authenticated nodes turn into unfaithful network peers. These compromised nodes would be a source of disruption and confusion to a local or global data aggregator center. Byzantine attacks may, in

general, refer to various types of Byzantine behaviors [17], [5], the focus in this thesis is on Byzantine attacks that contribute to faults like data diversion, data falsification, and other kinds of attacks shown in Figure (1.1). For attacks of this nature, compromised devices are reprogrammed and then forced to send tampered data to the Master Node to undermine the inference performance of the network. The Byzantine attackers could unknowingly mar the performance of wireless systems.

Chapter 2

Background

2.1 WANET Applications

The decentralized kind of wireless ad-hoc networks makes them fit for a variety of applications where central nodes cannot be utilized efficiently and may not improve the scalability of networks compared to decentralized wireless managed networks, although, in many applications technical and functional limitations to the overall volume of such networks have been identified. Insignificant configuration and fast deployment make ad hoc networks suitable for emergency applications like military conflicts and natural disasters. The existence of dynamic and adaptive routing protocols allows ad hoc networks to be developed rapidly. Wireless ad-hoc networks can be classified by their application as follow:

2.1.1 Wireless Sensor Network

Wireless Sensor networks (WSN) is a term used to refer to a heterogeneous system combining tiny sensors and actuators with general/special-purpose processors. Sensor networks are assumed to grow to include hundreds or thousands of static, low-power, low-cost, or mobile nodes.

Sensor networks are useful in different fields, including military, environmental monitoring, natural disaster, surveillance, and information gathering from hostile places. In addition of WSN monitoring applications, WSN also facilitates remote controlling of physical environments. Sensors are key players in various applications: measuring humidity, pressure, brightness, flow, temperature, mechanical stress, and health signs. Areas such as disaster forecast, environment control, health care, military command & control benefit greatly from this emerging technology.

WSN is a sub-category of Wireless Ad-Hoc Networks WANET, which make it part of our study and research. What makes WSN applications a potential candidate for using our distributive data integrity protocol, is that it highly relies on multi-hop communication to a focal data gathering node, sometimes called Fusion Center.

2.1.2 MANET and VANET

With the proliferation of inexpensive, miniaturized, and more powerful mobile devices, mobile ad hoc networks (MANETs) and vehicular ad hoc networks (VANETs) have become one of the fastest advancing areas of research. This new kind of self-organizing network

(SON) mixes wireless communication with a high-degree node mobility. Contrary to typical wired networks, they have no static infrastructure (base stations, centralized management points, etc.). The overall stance of nodes forms an arbitrary topology. This flexibility in movement makes them suitable for many applications such as transportation, and military applications, where the network topology may alter rapidly to reflect a forces operational movements, and disaster recovery operations, where the existing static infrastructure may be nonoperational.

Traditional networks use specific nodes to carry out basic functions such as network management, packet forwarding, and routing. In ad hoc networks, these functions are commonly performed distributively by all network nodes. Nodes on MANETs use multi-hop communication: nodes that are close to each others radio range can communicate through wireless links, whereas those that are away from each other must rely on intermediate nodes to function as routers to relay messages. Contrary to fixed Wireless Sensor Nodes, mobile nodes can move, leave, and join a network; thus, routes required to be updated rapidly due to the dynamic network topology. For instance, node S can communicate with node D by using the shortest path S-G-F-D as shown in Figure (2.1) (the dashed lines show the direct links between the nodes). If node G moves out of range from node S, it has to search for a different route to node D (S-T-K-F-D). A decent amount of new protocols has been developed for finding/updating the routes and ensure communication between end points (but no protocol has been accepted as standard, because of the high contrast in trade-offs). However, these new routing protocols, based on collaboration between nodes, are susceptible to new forms of attacks. Unfortunately, many routing protocols for MANETs do not consider security in design. Mainly because MANET features such as the lack of

central points, the dynamic topology, and the existence of highly constrained nodes impose a complex challenge for security.

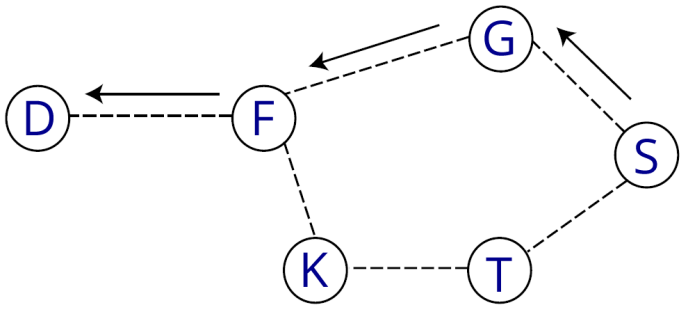

Figure 2.1: Routing messages using multi-hop communication.

Much study and research have been done to detect attacks against MANET routing protocols, including researches done on secure routing protocols and intrusion detection systems (IDSs). Nevertheless, for applied reasons, the presented solutions typically focus on a few particular security vulnerabilities since devising a complete solution is nontrivial. If we are to devise more general solutions, we must have a thorough understanding of possible vulnerabilities and security threats against WANET that explicitly embodies VANET. Moreover, in the following section, we will further describe some of the commonly used MANET routing protocols and highlight some of the security threats.

2.2 Routing in WANET

2.2.1 Proactive routing

This type of protocols maintains current destination and route lookup table by periodically broadcasting routing tables throughout the network.

1. Respective amount of data for maintenance.

2. Slow reaction on restructuring and failures.

One of the famous proactive protocols is "Optimized Link State Routing Protocol" OLSR [7].

Distance vector routing

Distance-vector protocols are functioning by determining the direction and distance to any link in a network. Direction usually abstracted to what is the next hop address and what exit channel should be used; whereas distance is a measure of the cost to reach a certain node in terms of hop counts and other variable information. The cheapest cost route between any source and destination nodes is the route path with minimum distance. Each device maintains a vector (table) of minimum distance path to every node. The price cost of reaching a destination is calculated using dynamic route metrics. For instance, RIP [12] uses the hop count of the destination while IGRP takes into account other network details such as node delay and available bandwidth.

2.2.2 Reactive routing

This kind of protocol identifies a route on demand by flooding the network with Route Request messages. Here are some of the mostly used reactive protocols.

Ad hoc On-Demand Distance Vector

The Ad hoc On-Demand Distance Vector (AODV) [28] routing protocol designed for use by non-stationary nodes in an ad-hoc network. It provides rapid adaptation to dynamic memory overhead, link conditions, low processing, low network utilization, and determines unicast routes to destinations within the mobile ad-hoc network. It uses destination sequence numbers to always guarantee loop freedom (even in the face of abnormal delivery of routing control packets), evading problems (such as "counting to infinity") associated with classical distance vector protocols.

Dynamic Source Routing

Dynamic Source Routing (DSR) [15] is a routing protocol for wireless mesh networks. It is similar to AODV in that it builds a route on-demand when a source node seeks one. However, it leverage source routing instead of depending on the routing table at each intermediate node.

Determining source routes demand to record the address of each intermediate device between the source and destination during route path discovery. The registered path addresses cached by nodes processing the route discovery packets. The discovered paths

are used to route messages. To achieve source routing, the routed packets must contain the address of each node the packet visits. This may result in high overhead cost for long paths or large addresses, like IPv6. To avoid using source routing, DSR optionally defines a flow id option that allows packets to be routed on a hop-by-hop basis.

2.3 Distributed Fault Detection in Wireless Sensor Networks

Wireless sensor networks (WSNs) have gained global interest in recent years, especially with the proliferation in Micro-Electro-Mechanical Systems (MEMS) technology which has expedited the development of smart miniaturized sensors. These sensors are tiny, with constrained processing and computing resources, and they are cheap compared to typical commercial sensors. These sensor nodes can gather information from the surrounding environment and, based on some decision model; they can compress and transmit the sensed data to the user.

Unlike traditional networks, a WSN has its design and resource constraints. Resource constraints include a restricted amount of energy, low bandwidth, limited processing and storage, short communication range. Design limitations are application dependent and are based on the monitored environment. The environment plays a major role in determining the network topology, the size of the network, and the deployment scheme. The size of the network could vary with the monitored environment. An ad hoc deployment is preferred over pre-planned deployment when the environment is inaccessible by humans or when

the network is comprised of hundreds of nodes. For indoor environments, fewer nodes are required to form a network in a limited space whereas outdoor environments may demand more nodes to cover a broader area. Obstructions in the environment can also limit communication between nodes, which in turn affects the network connectivity (or topology).

2.4 Security Threats in Wireless Ad Hoc Networks

The quick advances in wireless communication and electronics have open a path for the development of low cost, low power, and multifunctional wireless sensor nodes which consist of communication, sensing phenomenon, and data processing components. These small sensor nodes can conveniently be deployed into a designated area to form a wireless network and perform specific functions. Recently accelerated research in this area, wireless sensor networks have been applied in various areas, such as environment and habitat monitoring, activity monitoring, disaster management, and emergency response.

The nature of large, ad-hoc, wireless sensor networks presents significant challenges in designing security schemes, and the hostility of the targeted environment could act as a catalyst for transforming nodes from a normal healthy state into a weak faulty state. A wireless sensor network is a special network which has many constraints compared to a traditional computer network, thus designing an efficient and scalable distributed fault detection (DFD) is not an easy task. Here are some of the main challenges that Ad-Hoc wireless network could impose.

2.4.1 Wireless Medium

The wireless medium is less secure because of its broadcast and open nature which makes eavesdropping easier. Moreover, the wireless medium allows an attacker to intercept valid packets and easily inject malicious ones. Any communication can be intercepted, modified, or reflected by an attacker. Although this problem is not targeting sensor networks only, traditional solutions have to be modified and adapted to be applied to sensor networks efficiently. [24]

2.4.2 Ad-Hoc Deployment

The ad-hoc kind of sensor networks has no specific pre-defined structure. The network topology is always subject to changes due to node failure, addition, or mobility. Nodes can be deployed remotely from an airplane, so nothing is known of the topology before deployment. Since nodes may get corrupted or failed, an ad-hoc network must support self-configuration. Security schemes such as the one will be proposed in this thesis must be able to operate within this dynamic environment.

2.4.3 Hostile Environment

One of the challenging factors is the hostile environment in which sensor nodes function. Constrained nodes could be destructed or captured by attackers. Since nodes could live in a hostile environment, attackers can physically acquire these devices. Attackers may capture a node, physically disassemble it, and extract valuable information (e.g. cryptographic

keys). The highly hostile environment impose a serious challenge for security researchers.

2.4.4 Resource Scarcity

The extreme resource limitations of sensor devices present significant challenges to resource-hungry security mechanisms, such in the case of using public cryptography, or digital signature. The hardware constraints require extremely reliable security algorithms in terms of computational complexity, bandwidth, and memory space; which is not a simple task. Energy is the most demanded resource for sensor networks. Wireless communication is especially expensive in terms of power. Thus, security mechanisms must give special effort to be communication efficient to sustain network life time for a prolonged time. [33]

2.4.5 Immense Scale

Furthermore, the proposed scale of sensor networks presents a serious challenge for security mechanisms. Simply networking hundreds of thousands of nodes has proven to be a delicate task. Applying security over such size of networks is equally challenging. Security mechanisms must be scalable to very large networks while maintaining high computation and communication efficiency.

2.4.6 Unattended Operation

Depending on the function of the particular sensor network, the sensor nodes may be left unattended for extended periods of time. There are three main cautions to unattended

sensor nodes [33]:

- **Exposure to Physical Attacks**: The sensor may be deployed in an environment open to adversaries, inclement weather, and so on. The probability that a sensor suffers a physical attack in such an environment is therefore much higher than the conventional PCs, which is located in a secure place and mainly faces attacks from a network.
- **Managed Remotely**: Remote management of a sensor network makes it virtually impossible to detect physical tampering and physical maintenance issues.
- **No Central Management Point**: A sensor network should be a distributed network without a central management point; this will increase the vitality of the sensor network. However, if inaccurately designed, it will make the network organization difficult, inefficient, and fragile.

Perhaps most importantly, the longer that a sensor is left neglected, the more likely that an adversary has compromised the node.

2.5 Symmetric Cryptography for Confidentiality and Data Integrity

With the limited computation resources available for wireless ad-hoc networks, we cannot afford to use asymmetric cryptography and so it is better to use symmetric cryptographic primitives to construct a distributed fault detection algorithm. [29]

2.5.1 Data Confidentiality (Encryption)

A sensor network should not expose sensor readings to neighboring networks and intruders. In many applications (e.g., key distribution) nodes would communicate highly sensitive information. The conventional approach for keeping sensitive data secret is to cipher the data with a secret key that only appointed receivers to possess, hence obtaining confidentiality. Given the perceived communication models, secure channels between nodes and sinks should be setup so that later the network can bootstrap other secure channels as necessary.

2.5.2 Data Authentication (MAC)

Message authentication is an essential security tool for many applications in sensor networks. Since an adversary can easily forge messages, the sent data would be used in any decision-making process originates from a trusted source need to be ensured by the receiver. Technically, data authentication enables a receiver to verify that the claimed sender sent the data.

In the two-way communication scenario, data authentication can be accomplished through a purely symmetric mechanism; the source and the destination nodes share a secret key to compute a message authentication code (MAC) of all communicated messages. When a message with a correct MAC reaches the destined node, the destination would know that the designated source node must have sent it. This form of authentication cannot be applied to a broadcast setting, without placing much stronger trust model between the network nodes. If one sender wants to send valid data to mutually untrusted

receivers, using a symmetric MAC is insecure; as any one of the receivers knows the MAC key, and hence, could impersonate the sender and send tampered messages to other receivers. Hence, typically asymmetric mechanism is used to achieve authenticated broadcast communication. One of the contributions will be proposed is to construct authenticated broadcast from symmetric primitives only by using probabilistic key distribution scheme among the nodes.

2.5.3 MAC Algorithm Performance for Constrained devices

(Lee, et. al, 2010) [18] have studied several MAC algorithms. We have shown that the memory requirements of the block-cipher-based MACs are slight compared to those of a hash-based MAC. However, the hash-based MAC has a shorter computation time and therefore consumes less energy. Besides, when using XXTEA, HMAC could be a better match than a block cipher MAC, because XXTEA cannot use different operation modes and has not yet been proven to be secure when combined with a block-cipher MAC. Admittedly, HMAC uses a significant amount of memory, but this requirement could be reduced if a more memory-efficient algorithm is used. See Figure [2.2].

2.6 Byzantine Faults in WANET

Distributed wireless systems are subject to a variety of faults and attacks. In this thesis, we consider general (Byzantine) faults [11], i.e. a defective node who may exhibit irrational behavior. In particular, a faulty node may transform its local state from normal into corrupt

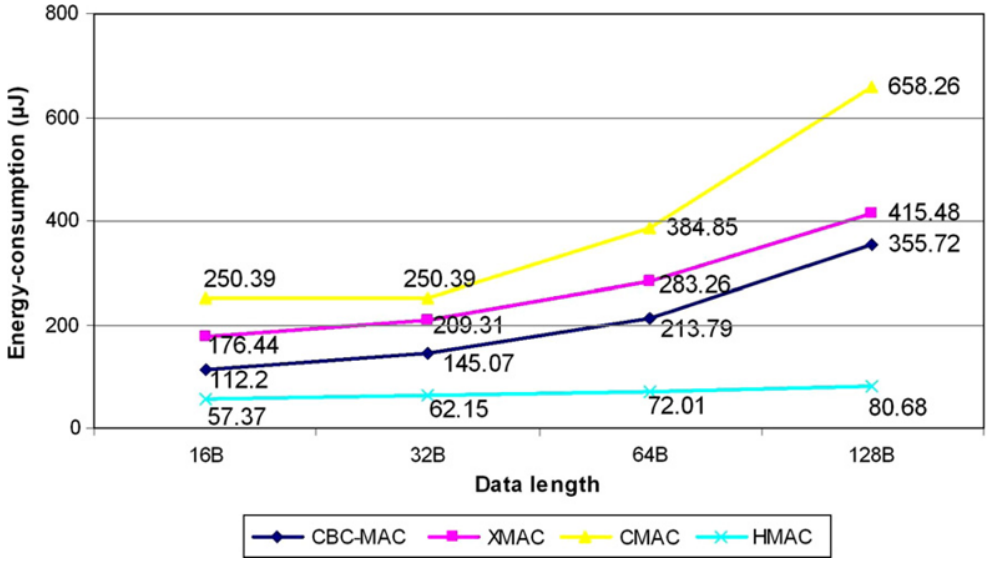

Figure 2.2: Energy consumption of MAC algorithms based on AES-128 on a TelosB sensor mote.

state and start sending random messages, including specific messages intended to undermine the system. Several security attacks, such as censorship, freeloading, misrouting, and data corruption, can be modeled as Byzantine faults.

2.6.1 Byzantine Problem

According to (Lamport, et al, 1982) [17] reliable computer systems must handle malfunctioning behaviors that provide contradictory information to different parts of the system. This circumstance can be expressed in terms of a group of generals of the Byzantine army stationed with their troops around an enemy region. Communicating only by messenger; the generals must agree upon a common engagement strategy. Still, one or more of them

may be traitors who will try to mislead the others. The problem is to design an algorithm to ensure that the loyal generals will reach into an agreement. It is proven that using only verbal messages; this issue is solvable if and only if more than 2/3 of the generals are loyal; consequently, a single traitor can puzzle two loyal generals. With unforgeable written messages, the problem is solvable for any number of generals and possible traitors.

2.6.2 The importance of fault detection

If a practical system can tolerate up to f simultaneous faults, Byzantine Fault Tolerance (BFT) algorithm cannot become usable with less than $3f + 1$ nodes. The corresponding view of this point is that BFT requires the number of faulty nodes in the system to settle below 33% at all times, while a correct node can reliably detect faults regardless of the number of faulty nodes.

Also, fault detection bypasses the agreement required by BFT, and it enables huge aggregation of state, messages, and processing associated with detection.

Detection systems do not only require fewer resources than BFT, but they also have some practical advantages that serve distributed systems whether or not they use BFT. Detection enables a convenient response to faults, In a system that does not use BFT, once correct nodes obtain evidence of a fault, they can stop communicating with the faulty node and thus isolate it. Also, correct nodes can initiate recovery, e.g. by creating additional replicas of any objects affected by the fault, or by alerting a human operator who can repair the faulty node. Timely repair can also help BFT based systems to stay within their bound for the number of concurrent faults.

Detection provides a deterrent. The mere presence of a detection system can reduce the likelihood of certain faults. For instance, it can discourage censorship in peer to peer systems by creating a disincentive to cheating since a faulty node risks separation and removal from the system. Furthermore, if the system maintains a binding from node identifiers to real world principals, then the owner of a faulty node can be exposed and held responsible. Reducing the frequency of certain faults also benefits BFT based system, allowing it to more easily maintain its error bound.

2.7 Literature Review

In this section, the related work on the fault and Byzantine node detection will be briefly discussed. The primary emphasis in this thesis is to present different distributed and large scale fault detection techniques and methods.

For the past few years many Researches have proposed different algorithms to detect hard and soft faults in WANET. Nevertheless, as it is commonly known, there are many formidable challenges in Wireless Ad-Hoc Networks that made all of the algorithms proposed have a tradeoff between scalability, energy consumption, reliability, and efficiency.

2.7.1 Related optimization algorithms

Other works of literature addressed Graph Coloring problem in dense WSN, some of these publications emphasize the importance of Graph Coloring applications in WSN [3, 2], the solution for this NP-Complete Decision Problem can be used to solve critical assignment

problems for WSN. In [3] (Amdouni, et al, 2011) have proved that the graph coloring is NP-Complete, and then they represented a 3-hop distributed algorithm that is optimized for dense networks; and also showed how the coloring algorithm can use regularity properties to achieve a periodic color pattern with the optimal number of colors. They proposed a Vector Method that provides the optimal number of colors among all periodic h-hop coloring. The Vector Method for dense wireless sensor networks will be assessed and evaluated in this paper to calculate fault detection accuracy.

2.7.2 Distributed Fault Detection in WANET using localized communication

An early study of securing routing protocol against Byzantine attacks was proposed by (Awerbuch, Baruch, et al, 2002) [4], the authors have proposed an on-demand routing protocol for ad hoc wireless networks that exhibit resilience to Byzantine failures generated by individual or colluding nodes. The authors have used the adaptive probing technique that detects a faulty link after $log(n)$ faults have occurred; where n is the distance of the path. The faulty links are then excluded by increasing their weights and by using an on-demand route discovery protocol that finds a minimum weight path to the destination. However, the study finds that it is hard to design a scheme that is resistant to a broad number of adversaries; thus, this technique is only applicable to a MANET with a limited scale of faulty nodes.

Lee and Choi [19] have introduced a fault detection algorithm in which each sensor node performs a local comparison between its sensed data with its neighbors data at a specific

time instant and stores the comparison locally. This process is repeated for a constant c times, and at each time comparison results are stored in a vector. Then, each sensor node identifies its fault status by evaluating the data stored in the vector.

In the same manner, in [25, 14, 6] the comparison based soft fault detection algorithms are discussed. In those approaches, each sensor node compares its sensed data with the received data from the neighbors and shares the results with them unlike keeping in a local table [19]. Then, based on majority voting in the neighborhood, each sensor is tagged with a name likely fault free or likely faulty. Each likely fault free sensor node has identified as fault free sensor node by some rigid criteria. Finally, the remaining likely fault free or likely faulty sensor nodes are determined to be fault free or faulty with the help of the known fault-free sensor nodes or its own tendency value respectively. The major disadvantage of these distributed approaches is high message complexity of C_N, where N is the number of sensor nodes in WSNs and c is the number of times each sensor node shares message with the neighbors. Further, all the above comparison based fault detection algorithms depend on a threshold value. The appropriate choice of the thresholds is an important problem because; the accuracy of the algorithm depends on the threshold value.

2.7.3 Distributed Fault Detection using statistical learning

In [22], Luo et al. presented the Bayesian framework for finding the faulty sensor nodes in WSNs. Here, each sensor node sends a query to the sink node to know how much noise is present in its area. The noise calculation is based on the Bayesian and the Neyman-Pearson performance criteria. After knowing own regions noise, each sensor node estimates

the same for the neighbors. This scheme needs multi-hop communication for estimating its noise which may change during the diagnosis period. Further, multihop communication adds high traffic in the network, which consumes more energy and makes the algorithm energy inefficient.

Other literature used statistical techniques for distributed fault detection (DFD), such as the research proposed by (Panda, et al, 2015) [26]; where each sensor node initially tests the presence of faulty sensor nodes in its neighbor. If detects any fault, then it tries to forecast the probable fault status of them. The authors used NeymanPearson (NP) statistical detection method; where the sensor nodes share the predicted fault status of the neighbors with them. Then, every sensor node uses a data fusion scheme to infer the final decision on its fault status. This literature has been compared to [6, 14], and the comparison result shows that the proposed scheme largely enhance the performance parameters for large scale sparse sensor networks as compared to that of existing algorithms. The enhancement measured that there is an 8% improvement in detection accuracy and 34% boost in false alarm rate as compared to the existing algorithms.

2.7.4 Distributed Fault Detection using cryptography primitives

Using security premises to serve Byzantine Fault Tolerant routing also has been a hot research subject recently, (Xu, Jiawei, et al, 2015) [34] has proposed Byzantine Fault-Tolerant routing for WSN based on Fast Elliptic Curve (ECDSA) algorithm. The authors have applied Byzantine Fault Tolerant (BFT) protocol on a large scale clustered WSN, where their experiment reduces the number of rounds proportional to 1/3, which brings the net-

work load to equilibrium. Moreover, they have used Fast ECDSA public key cryptography technique to enhance the communication and to secure the BFT algorithms. Although BFT protocols are proved to be efficient and scalable, they still require more energy and computational power to converge into an agreement in WSN environment. Also due to constrained energy that WSN has, it is known that symmetric key based security is favorable and more flexible in terms of calculation complexity over public key based security, but their application varies according to their setup complexity trade-offs.

Chapter 3

DFD Algorithm

3.1 The Genesis of Self-Policing Ad-Hoc Network Method

Wireless multi-hop networks is a decentralized sort of networks that consist of multiple wireless nodes acting as routers. The emergence and advancements of fault detection and discovery are mandatory to isolate the faulty behaviors and rejecting them from spreading. The discovery and isolation of faulty nodes enhances the ad-hoc network reliability and quality of service. So far, many failure detection and recovery techniques have been proposed such as route switch and multiple path detection that relies on stop failure model, where the only way the system would fail is by having a crashed node. Nevertheless, assuming that faulty wireless nodes never transmit any messages does limit the area where the introduced methods can be applied.

Our proposed technique circumvent this by transforming nodes into different types of

guarding officers. Each node would act as a checkpoint that signs a routed message with a distinct signature that nobody can identify and validate except officers belonging to the same unit. By segmenting the network into N number of units we can assign a distinct secret and shared key for each unit (Class) to use it in special security operations. By following the proposed self-policing scheme, we can implement a distributed fault detection algorithm that can identify soft and permanent Byzantine faults with an increasing accuracy using Symmetric Security scheme; which quite analogical to replacing Digital Signature, but without applying the expensive setup and installation cost of public security schemes. One of the most unpredictable Byzantine faults in Wireless Networks is the one that tampers with sensitive routed information and forward them as genuine and authentic packets. This sort of arbitrary node behavior make it so difficult to be realized; because of its unpredictability and lack of common fault patterns.

If we consider an ideal formation of self-policing network scheme for a multi-hop wireless network, see Figure 3.1. The Figure depicts a network that has every single node surrounded by nodes belonging to a different group (Unit). In this case, any message can be routed through any random path with a perfect alternation of router types.

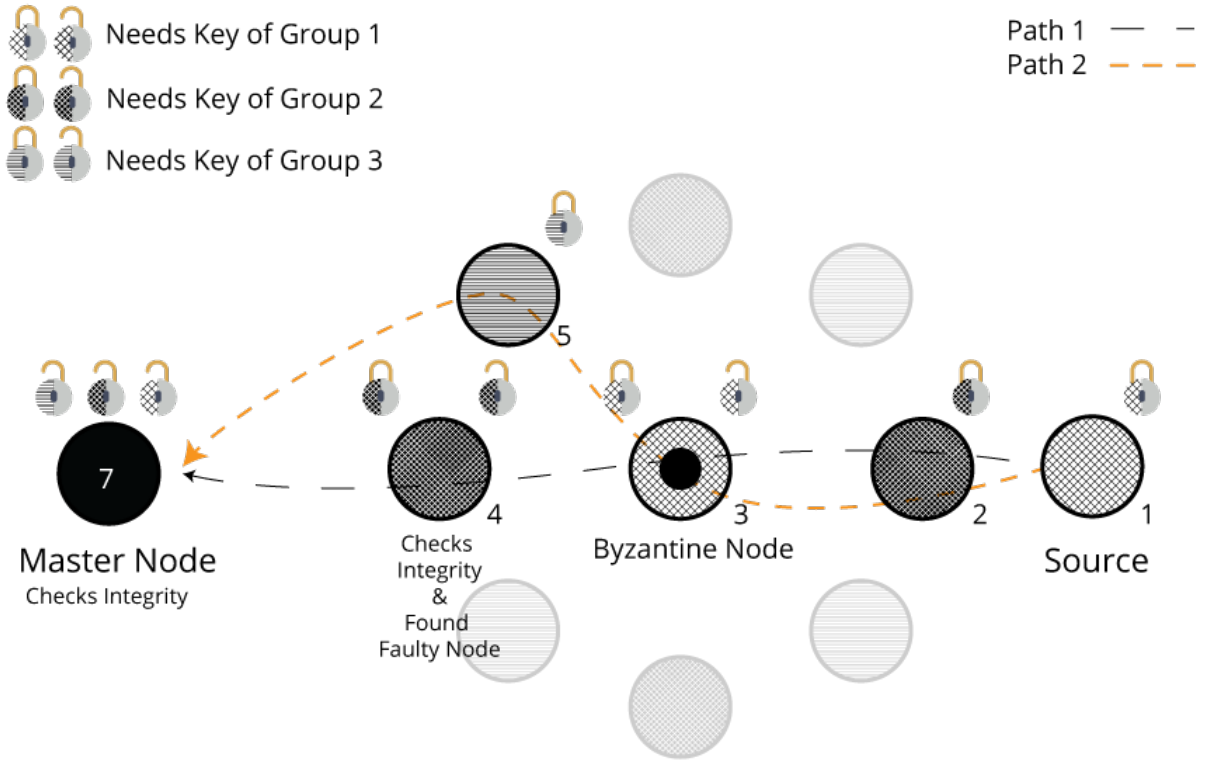

Figure 3.1: The notion behind applying Graph Coloring is to be able to make each class guard the network from every other class

However, assuming that all nodes are original and are not controlled by any intelligent attacker; let us consider the scenario depicted in the Figure 3.1. The first message will be sent from $Node1$ to $Node7$ through $Path1$, assuming that self-policing scheme is applied; every path should consist of nodes with alternating secret shared keys. Each secret key is only known to nodes that belong to the same group; in the scenario given the network segmented into three groups. So the journey of the message starts at $Node1$ and then will be continued to $Node2$ that belongs to $Group2$. Then, using its Group shared key $Node2$

will check the integrity of the message if possible, which means if the message has passed a node that belongs to the same group and been signed (MAC) by this node. Regardless, $Node2$ should update the message with a new (16, or 32) bits Message Authentication Code

$$updateGroup3HashField_{16bit} \leftarrow MAC(message, key3)$$

The resulted MAC should be stored in a dedicated metadata field reserved for the corresponding group after done verification. Right after that, the message will arrive into $Node3$, which we will assume that it is exhibiting a faulty state and not intelligently controlled by any external force. The reason of $Node3$ faulty behavior could be either due to a power depletion, malware, software bug, hardware fault, or any other environmental effects. Now $Node3$ will tamper [1] with the information and route it to $Node4$.

Now $Node4$ will repeat the process that $Node2$ has done, but in this case $Node4$ will find the $Group2$ field already been updated with a Hash MAC; thus, it will run a comparison task by recalculating the MAC and compare it with the one that has already been sent. In this case, the MAC comparison will show inequality, hence $Node4$ will report both $Node3$ and $Node2$ as alleged faulty nodes, taking into consideration that $Node2$ might be the one who tampered with the message right after it has the data integrity evaluation. Finally, $Node7$ which is the main Sink Node that assumed to know all of the keys would also repeat the verification and update process for the received message; along with processing any

[1] Note, if we replace Message Authentication Code function with reversible cryptography function, it will be impossible for an intermediate node to change a single bit in the message without breaking the key; assuming that a network is segmented into three groups

accusation list received from the routed path nodes to infer a decision.

In the described scenario we assumed that the network is ordered in an ideal formation, which is impractical and proven to be an NP-Complete problem (Vertex Coloring Algorithm) [3]. In the next section, we will explain how this problem has been simplified to the level that will make the DFD algorithm decently scalable and accurate at the same time.

3.2 Distributed graph coloring algorithm as a candidate solution

Distributed Graph Coloring Algorithm [3] could facilitate the employment of self-policing network scheme, which help the network to identify faulty nodes by dividing the wireless network preferably into two groups (Classes) or more. See Figure 3.2.

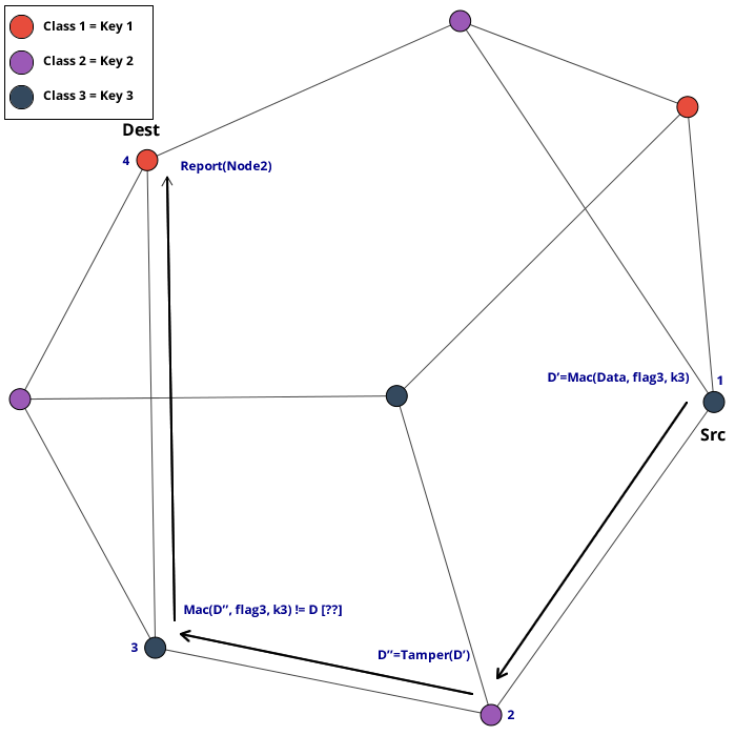

Figure 3.2: The notion behind applying Graph Coloring is to be able to make each class guard the network from every other class

Using Graph Coloring algorithm will ensure that every communication path will have the maximum number of variant guarding nodes contributing in protecting the message integrity; where every routed message can be validated by N_k number of nodes.

$$C = C_1, C_2, .., C_k \qquad s.t.(k << n)$$

$$k, n \in Z^+$$

$$argmin(k) \quad while \quad k > 0$$

where k is the number of colors (classes), and n is the number of nodes in a wireless network, and N_k is the number of nodes belongs to group K.

To leverage Graph Coloring algorithm to improve data integrity of the complete routed information inside a network, we need to distribute the same unique key for every single node belongs to class C_i, so that all nodes in that class can calculate and verify their originated messages using that key. Thus, distributive message authentication code (MAC) will be used to ensure a high level of security against forgery, and at the same time distributing the fault detection processes among n number of nodes. However, for our fault detection algorithm, and according to the Two-General Problem definition [17].

In the literature, both centralized and distributed soft fault detection methods are available [25, 6, 14]. The centralized algorithms required more communication because all the sensor nodes send the information to the fusion center to infer fault detection; whereas in our distributed case, each sensor node gathers fault evidence and inject them into forwarded packets utilizing its position as a router, which reduces the bootup and electronic communication overhead. It is evident from the operation of WANETs that the communication requires much higher energy than computation [25]. According to (Shebli, et al. 2007) [30], "The energy consumed for the calculation operation is very low as compared with the communication energy. The energy needed to transmit 1 KB over a 100m distance is approximately equivalent to the energy necessary to carry out 3 million instructions at a speed of 100 million instructions per second (MIPS)". Therefore, distributed fault detection approach using symmetric data integrity check will be highly valuable.

Dis-Advantages of Graph Coloring

Although finding an efficient Graph Coloring algorithm could help us reach into the perfect alternative group formation we need in self-policing scheme; however, Graph Coloring algorithm is an NP-Complete problem [3] that would not be practical to apply in constrained wireless networks as in case of WSN.

3.3 Proposed Distributed Fault Detection Process

We will first define communication and test graphs. A communication graph of a multi-hop wireless network can be represented as undirected graph $G = (V, E)$, where V represents the set of sensor nodes in the network and E represents the set of edges connecting sensor nodes. Two nodes V_i and V_j are said to have an edge in the graph if the distance $d(V_i, V_j)$ between them is less than l (transmission range). The communication graph can be a test graph in our fault detection if two nodes with an edge connecting them are compared. If some of the edges are not involved in the fault detection or ignored based on the previous test results, a test graph in our fault detection can be a subgraph of the communication graph. Figure 3.7 shows a randomly generated mesh graph with 100 nodes including a single Master Node (Cluster Head) at the top left corner of the mesh graph; the Master Node should be responsible for collecting data from the network or sub-network. Since the Master Node has a unique function that requires higher amount of energy in contrast to the other nodes, it could have a permanent power source or a higher capacity of energy storage.

Distributed Data Validation and Fault Detection Algorithm

To overcome challenges such as (Key Management, Key distribution, Mobility, Deployment complexity, Key Freshness, Computational and Communication complexity); we designed an algorithm with best efforts to make it as much scalable, efficient and practical as possible, by leveraging the natural power of randomness to make it stand in defiance against installation and key distribution complexity.

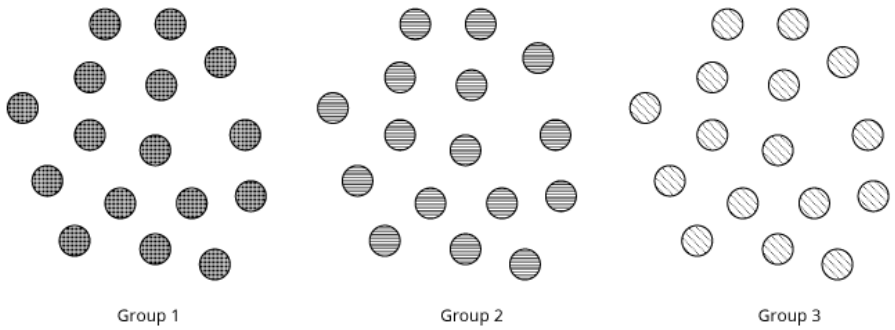

Figure 3.3: Segmenting network into three different groups (classes)

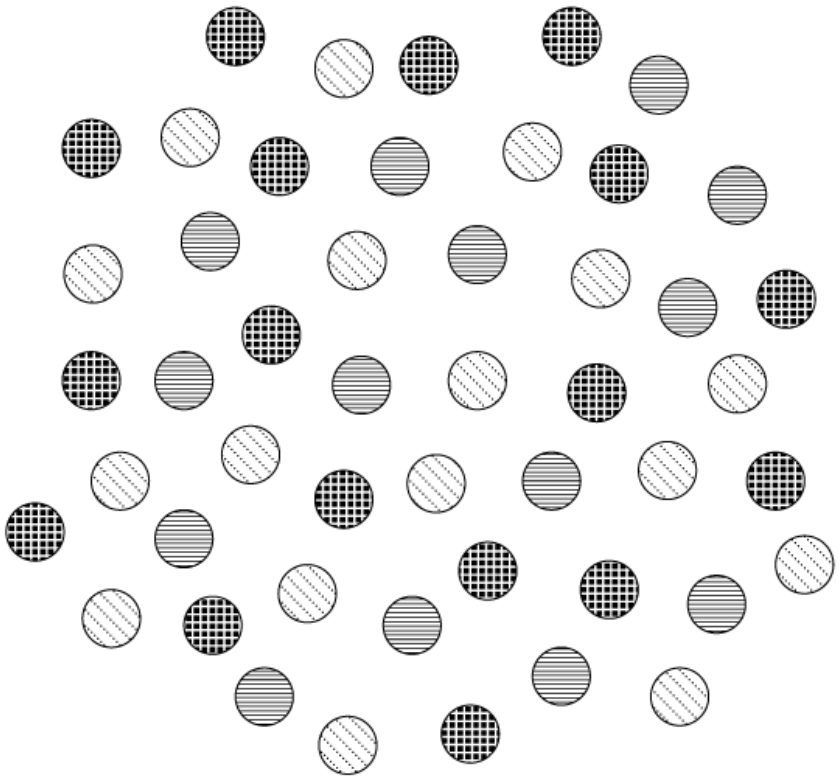

Figure 3.4: Random network deployment after segmentation

As already mentioned, it is hard to find an efficient algorithm that will organize an ad-hoc wireless network to an optimal formation to make every single node surrounded by nodes with different classes; which would be an ideal topology for a self-policing network model. However, the argument here is that if we segment a wireless network into three groups or more (K) and shuffle it randomly before deployment, see Figures [3.3][3.4], can we get an approximately perfect alternation of groups in any single constructed path of communication. Using statistic we will try to calculate the probability of finding a path that consist of k number of consecutive nodes that belong to the same group; so if we can prove that the determined probability can reach into negligible percentage, we can rely on it as a fact for algorithm operation.

Looking at Figure [3.6], let's denote the with the following variable names:

$$N \rightarrow \text{Network size, the total number of nodes} \tag{3.1}$$

$$G_n \rightarrow \text{The number of nodes inside of each group} \tag{3.2}$$

$$S \rightarrow \text{The number of segmenting groups} \tag{3.3}$$

$$k \rightarrow \text{The number of consecutive nodes belonging to the same group} \tag{3.4}$$

Using the combination function for calculating the probability of selecting k number of consecutive nodes with the same class would be:

$$
\begin{aligned}
Pr(k) &= C_{kn}^{G} / C_{k}^{N} \\
Pr(k) &= G_n!(N-k)!/N!(G_n-k)! \\
G_n &= N/S
\end{aligned}
\tag{3.5}
$$

Using equation [3.5] we can calculate the probability of finding k number of consecutive nodes by applying the following values.

$$N \rightarrow 300$$

$$S \rightarrow 3$$

$$G_n \rightarrow \text{N/S=100}$$

From equation [3.5], its notable that the more greater the number of segmentation groups S was, the more smaller G_n we will gain. Which yields a very small value of probability of finding k number of consecutive similar node occurrences; and, this means that the more segmentation we apply to the network will make it almost impossible to find even three consecutive nodes that belong to the same group if we set our probability threshold to ($<= 0.01$). However, increasing the number of segmentation will come with a cost to our algorithm, as more keys need to be distributed and more hashes need to be incorporated inside of the routed packets. So an optimization problem need to be solved using the formula given, to identify the optimal number of classes for any given network size.

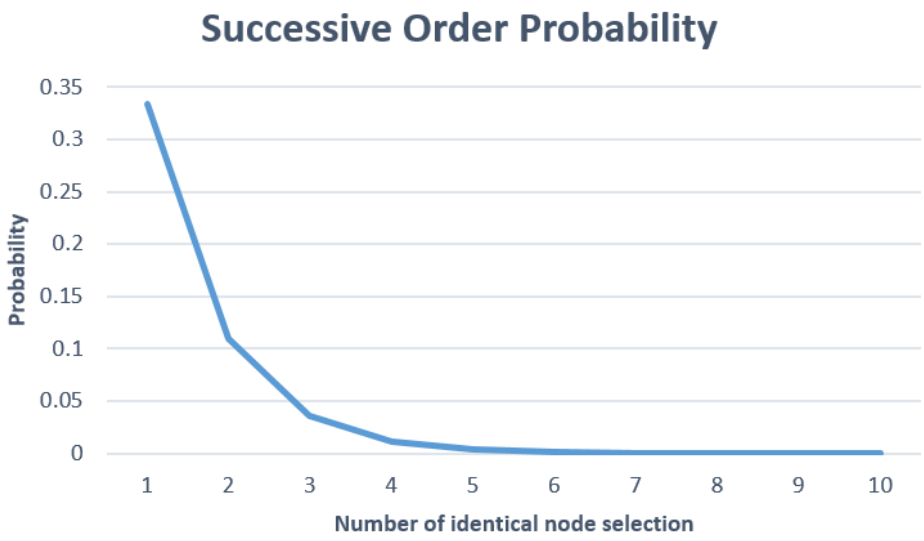

Figure 3.5: Probability of finding a path consist of nodes belonging to the same group. Network size N=300, and S=3

From Figure [3.5], we can clearly see that above the possibility of finding three consecutive nodes, the value of probability becomes negligible. At $k = 4$ the probability calculated was (0.0118), and at $k = 5$ the probability calculated was (0.0038).

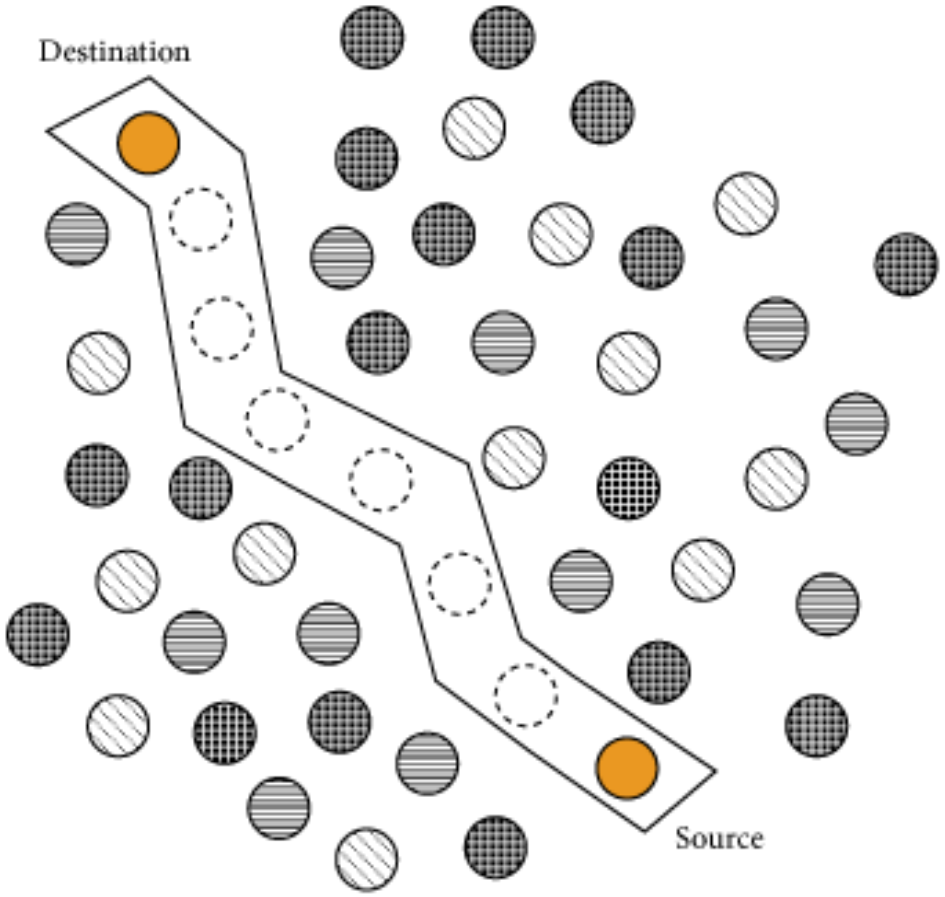

Figure 3.6: Probability of finding a path consist of nodes belong to same group

The optimal K number of groups would be approximately determined in the experimentation section later; however, using Equation [3.5] to decrease the probability of successive equals to a certain extent would be sufficient to determine the best segmentation value S. For instance, if we want to deploy a WSN that consist of 100 nodes and the optimal number of classes determined was $K = 3$, then, the network should be divided into three equal number of groups. Each group will contain 33 nodes excluding one node that will act as

the master node or Fusion Center (FC), see Figure 3.7. So a pre-deployment configuration must be done in advance; so that at least every single node will be updated with a shared group key that will be entirely secret and hidden from other different groups.

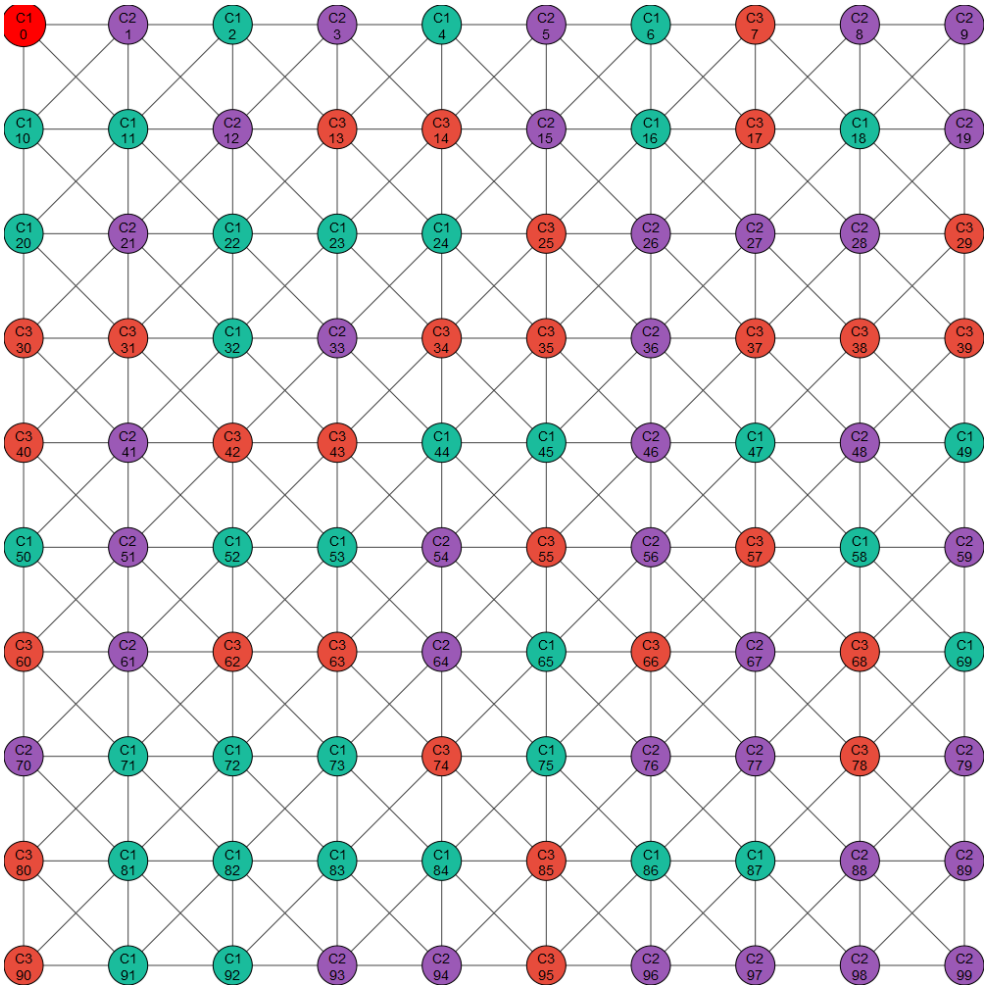

Figure 3.7: Random 10x10 network deployment with 3 different groups (classes)

Now assuming that a Wireless Ad-Hoc network will communicate through a multi-hop communication protocol, any message sent from an arbitrary source will be carried to the

destination through intermediate wireless routing nodes. While the message is traveling from source to destination, it will be parsed, diagnosed and updated if necessary by every single node to maintain the data integrity, and detect any tampering incident done since the message visited the last node belonging to the same current node group. As mentioned eariler, the alternation of classes while relaying the message to the destination will help protect the message and detect any faulty node as described in Algorithm ??.

3.3.1 Algorithm Stages

Parsing Routed Packet

In this stage, the node should parse four main lists, a latest similar class hash, and the payload message from the packet. The information that should be processed are:

- $hopIDList \rightarrow$ In this list the routing nodes will store their encoded ID's in the packet to inform the next routing nodes about the history of contributing routing path since the creation of a message.
- $hopClassList \rightarrow$ In this list every routing node will store its Class number to inform all of the next contributing routing nodes about the previous class chain. Note that if the optimal class number was found to be $K = 3$, then we only need 2 bits to encode the class type (Group) number.
- $accusedByzList \rightarrow$ Any new possible findings of faulty nodes would be recorded and registered in this list to be reported to the main Master Node.

- *whiteNodeList* → For those healthy nodes who routed message successfully without applying any distortion to the relayed message, they will be registered into this list to be reported whenever found. [2]

- *latestSimilarClassHash* → This variable should parse and store the last hash value overwritten to the location dedicate for the nodes with corresponding similar group type.

- *message* → Payload message.

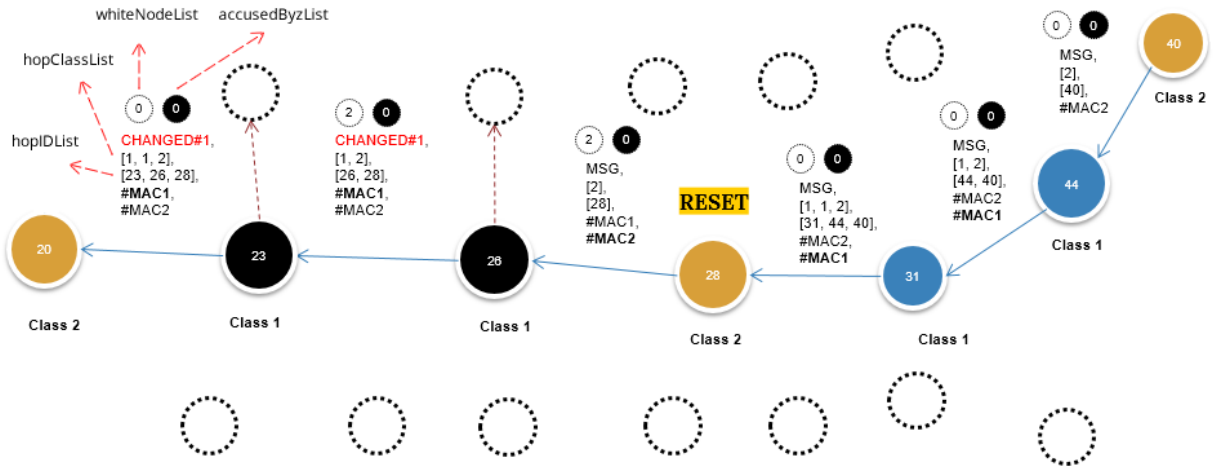

Figure 3.8: Data integrity calculation and validation process, assuming that the network is divided into two classes.

[2] As an extra optimization procedure, each node should retain incremental and non-duplicated records of white node list, so that there will be no need to resend the same report over and over again to the Master Node

Check Packet Integrity

Illustrating a sample packet representation shown in Figure 3.9 that consist of dynamic and static segments, assuming that $K = 2$ for the number of classes defined for the Wireless Ad-Hoc Network. The first part on top of the Figure shows how a node belongs to (Group 1) should calculate the Message Authentication Code for the dynamic packet segment that belongs to its class (1) post-fixed with $C1$. Then. after the MAC is calculated it will be overwritten to the fixed store location for group 1 (HMAC1).

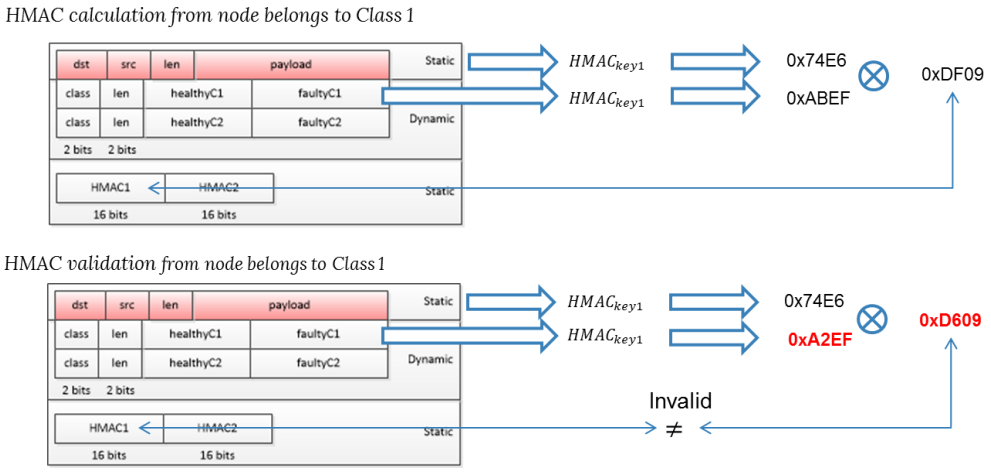

Figure 3.9: Data integrity calculation and validation process, assuming that the network is divided into two classes.

Now after a node with group 1 has updated the $MAC1$; the next routing node should repeat the process of checking the data integrity, and if it is not valid then report all intermediate nodes including the node that updated $MAC1$ itself and add them to the fault accusation list.

Update Packet Integrity

Updating data integrity for each group should happen right after detecting the tampering event. So if the calculated $MAC1$ in Figure [3.9] was not equal to the received $MAC1$; then a report should be created followed by updating the corresponding MAC location with the new locally calculated MAC. However, what if the calculated $MAC1$ was wrong? The algorithm should take care of this incident using the continues report and update process, as we will see in the next section.

3.4 Continues Detection and Update Process

Whether we have a WANET with multi-clusters and multi-cluster heads, or a WANET with a single cluster head, the algorithm assumes that the nodes should always be able to report node health information to the single nearest cluster-head that has the property mentioned in subsection 4.6.1.

The proposed DFD algorithm tries to elicit nodes to guard the network by reporting other nodes status either as faulty or normal. And, the continuation of this process will eventually form different intersected paths that rectify previous false detection reports; especially if we assume that network has mobility [3]. So a continues dissemination of nodes status to the master node will be processed as long as the DFD algorithm is enabled.

[3] As will be supported in Simulation chapter, one of the advantages of the proposed DFD algorithm that it strives for mobility; not only that it improves fault detection, but yet, it improve security in case encryption was applied using the proposed network segmentation scheme

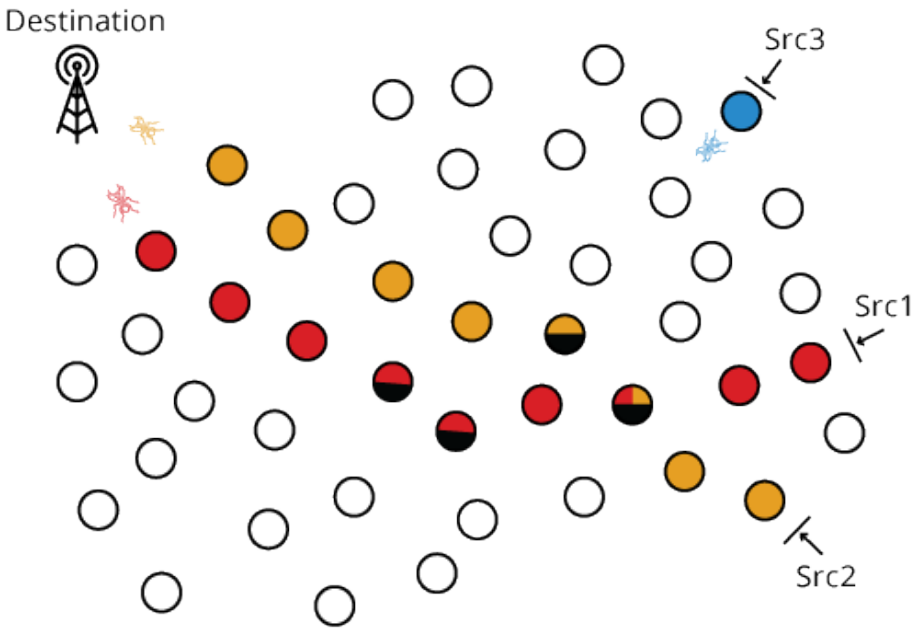

Figure 3.10: First scene, shows two messages been routed respectively from (source1 and source2) to the destination. Four faulty nodes has been detected during communication, and one faulty node has been intersected and reported by two different paths.

From the first scene shown in Figure 3.10, two messages are routed from $Src1$ and $Src2$ respectively, the first path painted in red identified two faulty nodes; whereas the second crossed the same faulty node. Thus it complements the same discovery that node one has found.

3.4.1 Continues Acquittal

The continues fault detection, acquittal and report process will facilitate the final decision inference. where we an incremental evidence reports will help removing any previously

falsely accused nodes; because we are assuming that Byzantine nodes will generate permanent faults, thus, any positive health reports about a node will discharge any previous accusations. For instance in the scenario shown in Figure 3.11, a new routing path highlighted in blue have discovered that a previously accused faulty node to be a healthy router; thus, it has reported this discovery to the master node to drive the online inference system to be more accurate and precise.

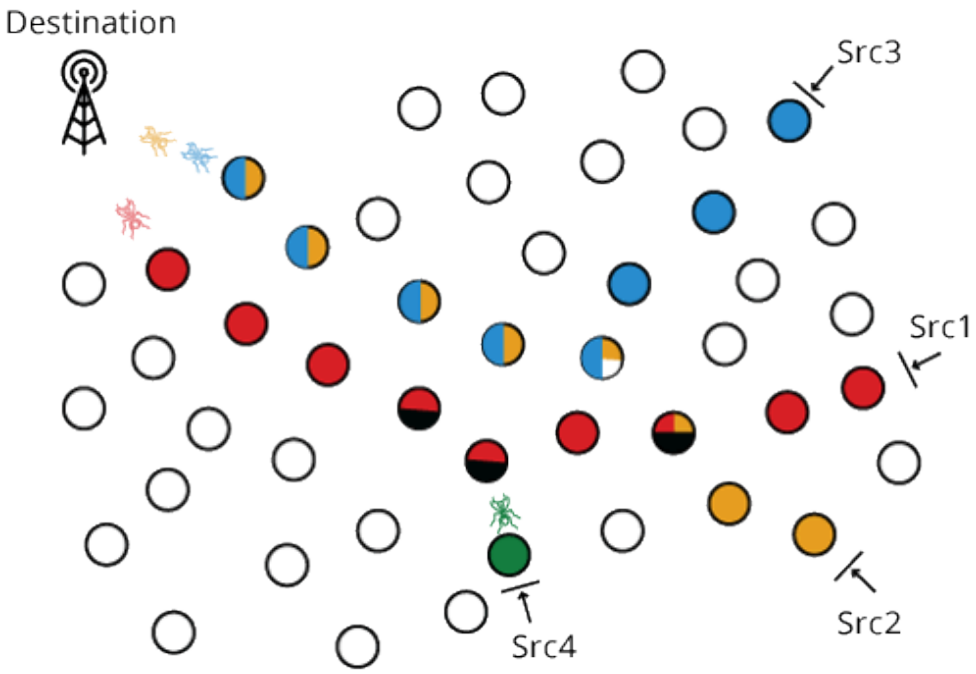

Figure 3.11: Second scene, shows a third message initiated from $Src3$ constructed a new relaying path routing a message to *Destination*; the new path highlighted in blue has given an acquittal to a node that formerly was tagged as Byzantine node from a path highlighted in orange 3.10.

3.4.2 Report Packet Optimization

Detecting Byzantine faults along a routing path that consist of multiple classes needs an updated ordered record about every single class the message had traversed through since its origination. So recording nodes ID's and Classes for routing nodes is mandatory. This information will be used along the way by every single node to determine when is the last time a node belongs to the same class had updated the integrity of the message; and. If the integrity check was violated, what set of nodes should be accused of a violation.

Here is an example, imagine a message been routed from node 7 to 6, 5, 4, 3, 2, and 1. And this routing path is having the nodes that have been assigned to the following set of classes respectively [1, 2, 3, 1, 3, 2 , *]. Now if we retrospect the message transmission, when $Node4$ received the message, it has to be able to contrive the following tracking information from reading the metadata added inside of the original packet:

$$key_{Class1} = 0xF8A223CF$$

$$hopTrackingIDList = [7, 6, 5, 4]$$

$$hopTrackingClassList = [1, 2, 3, 1]$$

$$lastStaticMAC_{Class1} = 0xFD12BA65$$

Since $Node4$ belongs to $Class1$, it can validate the MAC as follows:

$$MAC_{key1}(Message) \rightarrow 0xFD12BA65$$

From the yield MAC, the message seems to be tampered by one of the routers, hence, according to the proposed DFD algorithm, $Node4$ will find it safe to accuse [7, 6, and 5] with minimum probability of 1/3 of the defendant list is faulty.

About optimization, now the result that $Node4$ has reached can either be stored locally and be reported at later suitable time; or it can be embedded into the packet report metadata and broadcast it to the next adjacent nodes. In the case of WSN, the packet size is very constrained and it should be wise to buffer the findings and report it individually at a later time suitable time.

However, in case our network does allow us to add information to the header of the packet, the algorithm does one further optimization of removing all of the stored tracking lists once a successful fault detection occurs. So for our illustrated example when the message gets forwarded to $Node3$, the tracking should have already been cleared by $Node4$ at this moment of time. So $Node3$ should see no tracking information and should start updating tracking information all over again:

$$hopTrackingIDList = [3]$$

$$hopTrackingClassList = [3]$$

3.4.3 Decision Making

The continues dissemination of integrity metadata from every node to the master node will consolidate the historical evidence set; in which it enables the master node to infer decisions easily based on negative and positive report frequencies. Inferring decision logic could be customized based on the combination of the available input and network assumptions, for instance, let's consider the following scenarios:

- A scenario (A) assumes that any node reaches a faulty state would have a fault ratio of 100%. In this case reading a single healthy report about the node (V_i) entails an absolute discharge for (V_i) from any previous accusations have been charged against it before.

- A scenario (B) where the fusion center infers decision-based on the frequency of negative and positive reports. Here a probabilistic model can be built to infer decisions, for instance, Evidence theory ($Dempster - Shafer$ [35]) can be used to synthesize a belief function.

3.5 Security Challenges

3.5.1 Node Collusion

In decentralized networks, malicious node collusion poses severe security threats, because the collusion of Attackers will not only help to achieve a faster convergence rate toward

their desired objectives, but also provide them with an opportunity to support each other and distort the default network trust model. However, information exchange among the Byzantine attackers should be identifiable due to the multi-hop characteristic of a decentralized wireless network, which may accelerate the exposure of attack behaviors.

3.5.2 Probabilistic Byzantine Attack

The next critical problem is when to attack. Selection of attack opportunity will influence attack gains and risks. If an attacker reports falsified results all the time, such an attacker will use every attacking opportunity no matter whether attacks work or not. In consequence, they obtain significant but yet a transient attack performance; as frequent falsification makes them prone to be detected by defense mechanisms. In practice, an attacker may not misbehave all the time, especially if it is in pursuit of long-term profits. Nevertheless, an attacker can complicate matters by using a probabilistic attack where he launches attacks with a certain probability, making it extremely hard to identify a malicious invader based on communication pattern analysis.

The probabilistic attack can improve stealthiness of attack behaviors, and is easy to be modeled and analyzed.

3.5.3 Group key secrecy and freshness

It is often desirable to frequently change the key in a cryptography system. And in case of our algorithm the same desire is maintained. If a foreign attacker wants to forge a message,

then he has to find both the algorithm suite the system is using and a collision for each MAC a group will generate for a message. The simplest attack could be made is to find a collision for only one MAC if the location of the attacker was one hop away from the message originator.

Although, active attacks are not under the thesis scope; however, if an attacker is actively trying to forge messages all the time, the DFD algorithm should be able to expose him and report him as faulty node too. Unless the attacker has been able to extract all group keys, there is always a chance to identify an invader who tries to corrupt the system.

Key freshness could be applied to our scheme from a central master node that is pre-deployed with all of the group keys. A master node or any other mandated node could act as the key manager for a sub-network or an entire network; and this dedicated node can update group keys periodically. If a network has been divided into three groups then the key manager need to broadcast only three ciphered key messages to the network so that corresponding groups only will be able to interpret the new key update message and execute it.

Algorithm 1: Distributed Byzantine Fault Detection Algorithm

Function Parse_Verify_Update (*packet*)

Input : Extract the following parameters from packet:
$hopIDList \longleftarrow packet['routedHopList']$
$hopClassList \longleftarrow packet['routedClassList']$
$accusedByzList \longleftarrow packet['byzNodes']$
$whiteNodeList \longleftarrow packet['whiteNodes']$
$latestSimilarClassHash \longleftarrow secureUnlock(packet['latestClassHash'])$
$message \longleftarrow packet['payload']$

$macHash \longleftarrow MAC(message, privateClassKey)$
if *current Node is not the Source* **then**
  if $macHash != latestSimilarClassHash$ **then**
    /* If the current class MAC is not equal to the last similar class MAC, then select all of the intermediate nodes and put them in fault accusation list */
    $byzNode \longleftarrow augmentAccusationList(hopIDList, hopClassList)$
    /* Check and clear any redundant accused faulty nodes */
    $byzNode \longleftarrow clearRedundancy(accusedByzList)$
    /* Discharge or remove white nodes exist inside of the recent accusation list */
    dischargeWhiteAccusedNodes(byzNode, whiteNodeList)
  else
    /* If MAC is equal, then this mean that the payload is still intact. Then we want to label all of intermediate routing nodes since last similar class node as white nodes. */
    $newWhiteNodes \longleftarrow identifyWhiteNodes()$
    updateWhiteNodeList(newWhiteNodes, whiteNodeList)
    $newWhiteNodes \longleftarrow clearRedundancy(newWhiteNodes)$
  end
else
  return *packet*;
end

Output: $packet \longleftarrow updatePacket(packet, newWhiteNodes, byzNode)$
forwardToNextHop(packet)

End Function Parse_Verify_Update ;

Chapter 4

Simulation and Experimental Results

4.1 Simulation Tools and Technologies

Figure 4.1: Network and Routing Simulation Tools

Simulating our network has been done using the (The iGraph library package for simulating complex network research) [8]. Our main simulation and development environment is

based on Python language, it's the language that we used to write the DFD algorithm and implement the wireless multi-hop communication protocol. iGraph and PyCrypto along with other libraries have been imported to facilitate our implementation.

MatPlotLib and Matlab have been used to analyze the results and generate visualized graphs.

4.2 Radio Model and Energy Calculation

For our experiments, we used a 400-node network where nodes were randomly distributed in a triangle network topology with equal distance apart; and, with an Aggregator node (FC) labeled with the number (0), see Figure (4.3).

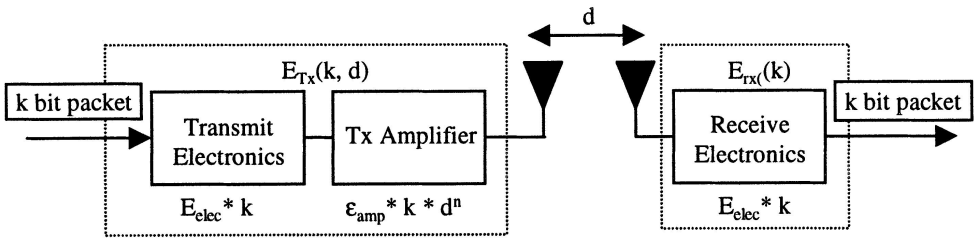

Figure 4.2: Radio energy consumption model

Assuming a simple model for the radio hardware energy dissipation as presented in [13]; where the transmitter consumes energy to run the radio electronics and the power amplifier, and the receiver consumes energy to run the radio electronics, as shown in Figure (4.2). For the experiments presented in this thesis, both the free space (d^4 power loss) and the multipath fading (d^2 power loss) channel models were used, depending on the distance

between the transmitter and receiver. Power control can be used to invert this loss by appropriately setting the power amplifier, if the distance is less than a threshold d_0, the free space (fs) model is used; otherwise, the multipath (mp) model is used. Thus, to transmit an l-bit message a distance, the radio will expend

$$E_{T_x}(l,d) = E_{T_x-elec}(l) + E_{T_x-amp}(l,d)$$

$$= \left\{ \begin{array}{ll} lE_{elec} + l_{\in_{fs}} d^2, & d < d_0 \\ lE_{elec} + l_{\in_{mp}} d^4, & d \geq d_0 \end{array} \right\}$$

and to receive this message, the radio expends:

$$E_{R_x}(l) = E_{R_x}(l) = lE_{elec}$$

The electronics energy (E_{elec}), depends on factors such as the digital coding, modulation, filtering, and spreading of the signal, whereas the amplifier energy, $\in_{fs} d^2$ or $\in_{mp} d^4$, depends on the distance to the receiver and the acceptable bit-error rate. For the experiments described in this paper, the communication energy parameters are set as: nJ bit, pJ bit m , and pJ bit m .

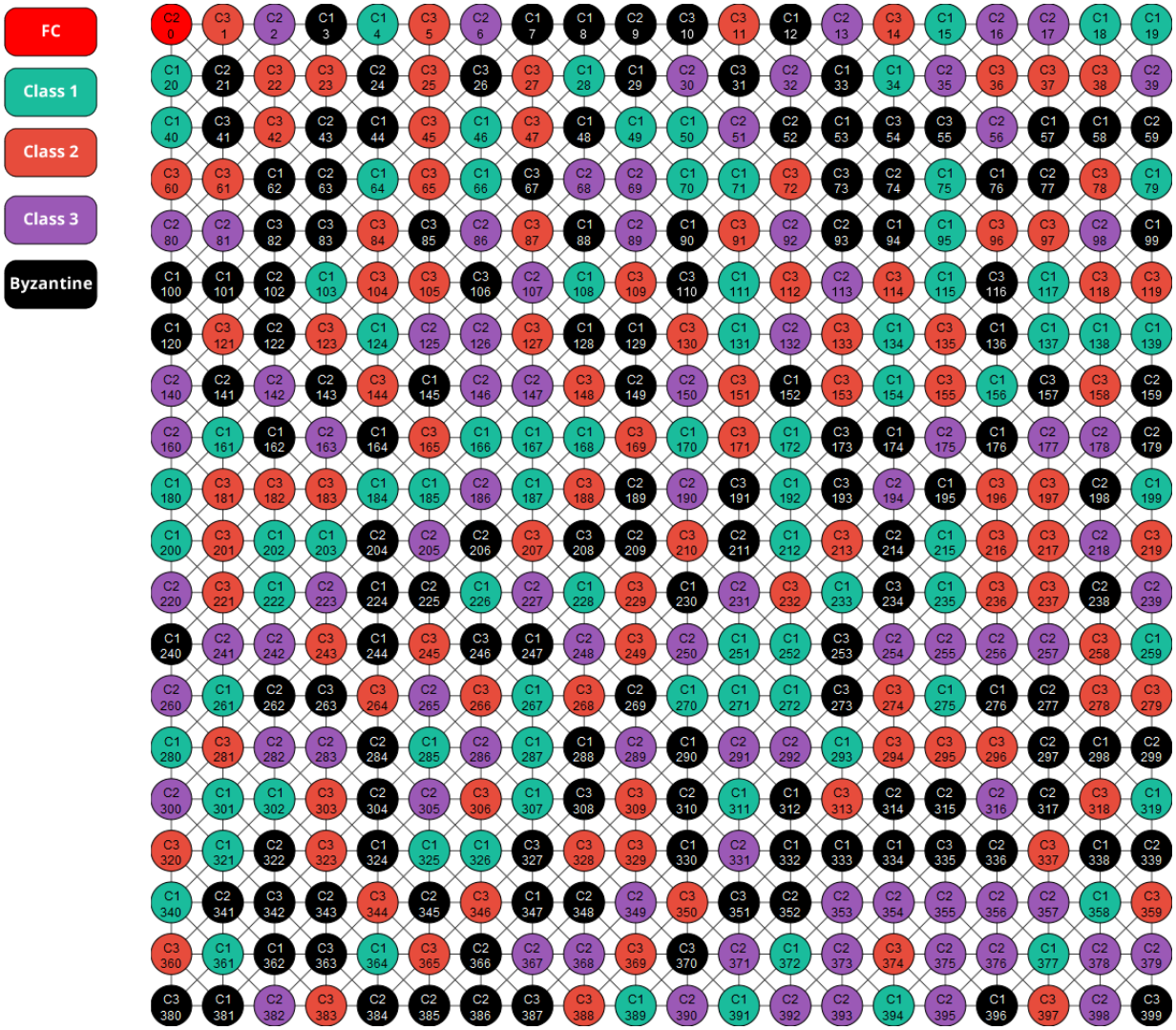

Figure 4.3: 20x20 Random graph network representation with 1/3 of the total number of nodes been randomly distributed as Byzantine faulty nodes

4.3 Network Model

We assume nodes are randomly deployed in the desired area and all nodes have a common transmission range. The area is assumed to be entirely covered by the nodes. As shown in Figure (4.3), this square area is entirely covered by wireless network nodes. The dark circles in the figure represent faulty sensors where it generates random faults represented in the Fault Model Section [4.4], and all other circles represent healthy nodes with a different class assignment.

Class assignment is evenly and randomly distributed among a variable number of classes. In the Figure (4.3), three different classes have been assigned to network nodes before deployment. In which each class should have a distinct shared key. See Figure [4.3] for simulation network visualization, although the simulation is showing a network with mesh-grid topology, but we are assuming to have a triangle based topology, where every node is situated to have the same distance from all of the other neighbors, see Figure [4.4].

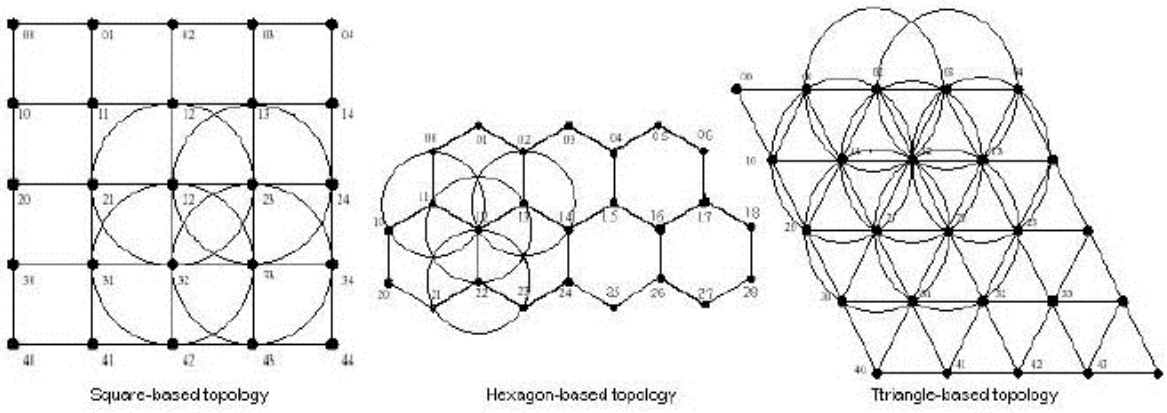

Figure 4.4: Presenting different set of network typologies that can be used for simulation

4.4 Fault Model

In our simulation, we will assume that all the nodes have the same communication range. Wireless nodes can be randomly deployed or placed in predefined locations. Nodes with faulty software and permanent communication faults are to be identified. Nodes which generate incorrect data or fail in communication intermittently are treated as usable nodes, and thus are diagnosed as fault-free. Mobile nodes with malware could participate in the network operation since they are still capable of routing information. Only those nodes with a permanent fault in communication (including lack of power) are to be identified from the network.

Here are the random fault models that will be generated from nodes while routing a message. The algorithm should be able to withhold with any of the following listed faults:

Model #	Description	Protection Type
1	Tempering with payload	HMAC
2	Forge fault report	Continues Report Update
3	Clear Tracking Information	Continues Report Update
4	Tampering with Tracking Information	Continues Report Update
5	Tampering with report information	HMAC

4.5 Mathematical Formulation

In our experimentation we define a limited search space that consist of three variable input:

$$X_i \rightarrow \text{Variable number of byzantine nodes} \tag{4.1}$$

$$C_j \rightarrow \text{Variable number of classes} \tag{4.2}$$

$$M_k \rightarrow \text{Variable number of routed messages} \tag{4.3}$$

$$(i, j, k) \in Z^+ \tag{4.4}$$

We do want to scan for a local maximum solution by running a simulation script. Thus, an Integer Linear Programming (ILP) formulation has been set in our script as follow:

$$\begin{aligned} \underset{A}{\text{maximize}} \quad & \text{Accuracy}(X_i, C_j, M_k) \\ \text{subject to} \quad & (X_i, C_j, M_k) \in Z^+, \\ & 250 \succeq X_i \succeq 0 \\ & 10 \succeq C_j \succeq 1 \\ & 250,000 \succeq M_k \succeq 10 \end{aligned}$$

where we need to maximize the accuracy from variant set of input. For a network with 400 nodes, we will change the number of classes within this range [1-10], and inject byzantine nodes within this range [1-250], and finally simulate routing [10-250,000] messages to study what is the effect of these three variables on the overall detection accuracy of our algorithm.

4.6 Results and Analysis using Heuristic Techniques

Encoding and simulating the problem described in the previous section in an optimal manner requires an excessive computational power. We need to study the DFD characteristic for different sizes of networks, in addition to applying permutation with an unlimited set of other input variations. Some of the input are (classes, routing, location of a master node, mobility, etc). So the best effort can be made here is to heuristically try to discover approximated weights and parameters that will lead into gaining optimal results.

In our experiment we have reduced the size of the search space to the following set of input for practical reasons:

$$Classes \rightarrow [1 - 10]$$

$$NumberofByzantineNodes \rightarrow [25, 50, 70, 100, 133, 150, 170, 200, 250$$

$$OverallSimulatedHopCount \rightarrow [10, 100, 1000, ..., 150000, 200000, 250000]$$

4.6.1 System Assumptions

Here are some important base assumptions that have been used in throughout the simulation:

- The DFD algorithm is designed to detect intermediate routing nodes that have permanent soft and hardware faults (Byzantine); so the assumption is that every source

node originates a message would be only diagnosed whenever it participates in routing a new message as an intermediate node.

- The routing protocol used will be similar to AODV protocol [28], where a node will broadcast RREQ and once it receives RREP the routing path with fastest response rate should have already been entailed. So for our simulation, we are generating random paths from source to destination adding a Gaussian path shifts to the distance vector direction.

- The group shared keys should be configured and stored secretly inside of every node using cryptography premise.

- One or master nodes that collect and report information should have the knowledge of the group secret keys; for instance in the case of having multi-clusters, the cluster head should be granted with this knowledge. So, in case a network have been divided into three separate groups $k = 3$, then, three keys should already be deployed securely to every Master Nodes to help them compile the final packet integrity and infer decision.

- The proposed algorithm could be applied to any ad-hoc wireless network, MANET, VANET, or WSN. In case it has been applied to WSN, then the packet size usually is limited and does not exceed 200 bytes, so caching and reporting delay mechanism should be employed.

4.6.2 Accuracy relation with number of byzantine nodes

In this section, we will discuss the impact of the routing path length toward the overall Byzantine detection accuracy. Looking at Figure [4.6], evidently the accuracy has a proportional relation with the number of hop counts for the routed message. In the graph shown [4.6], the number of hop count is the cumulative value of how many nodes the message had visited before it reached its destination.

The **Random Permutation** has been used to generate all of the results; the process can be conceived from the following Algorithm.

Algorithm 2: Generating random network setups, and injecting different set of byzantine nodes and plotting a graph for each network instance

```
1 Function GeneratePlots (networkSize = 400)
  Input :
          classes → [1 − 10]
          byzantineNodes → [25, 50, 70, 100, 133, 150, 170, 200, 250
          simulatedHopCount → [10, 100, 1000, ..., 150000, 200000, 250000]
2 while class in classes do
3 |   while byz in byzantineCount do
4 |   |   while hopCount in simulatedHopCount do
5 |   |   |   continuouslyPlotAccuracy(class, byz, hopCount)
6 |   |   end
7 |   end
8 end
  Output: Plotting Accuracy Graph
9 End Function GeneratePlots ;
```

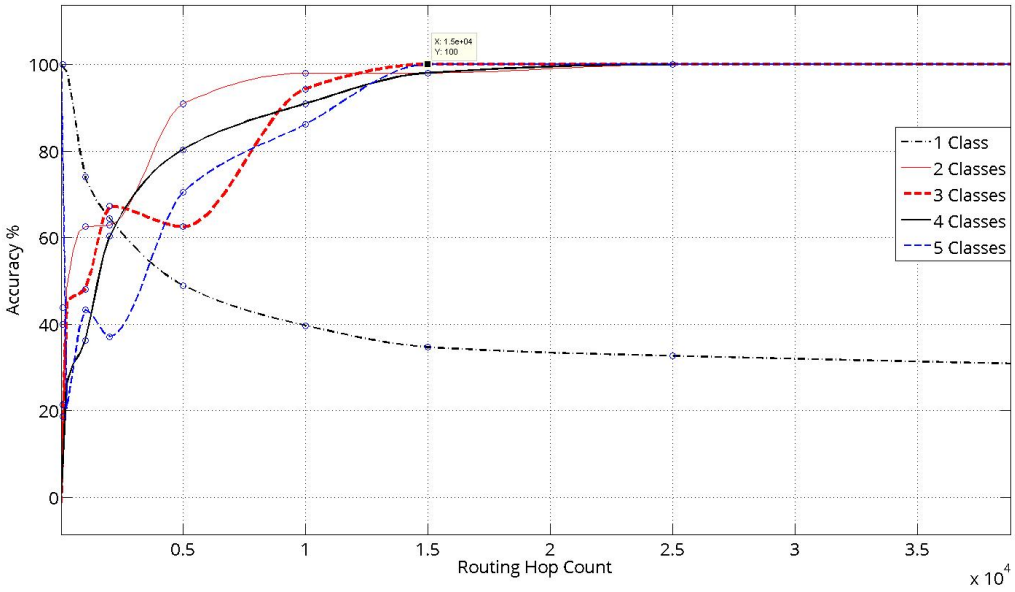

Figure 4.5: Accuracy measurement of Distributed Fault Detection (DFD) in network size of 400 with 50 Byzantine nodes. Using 5 different classes with variant communication hop length

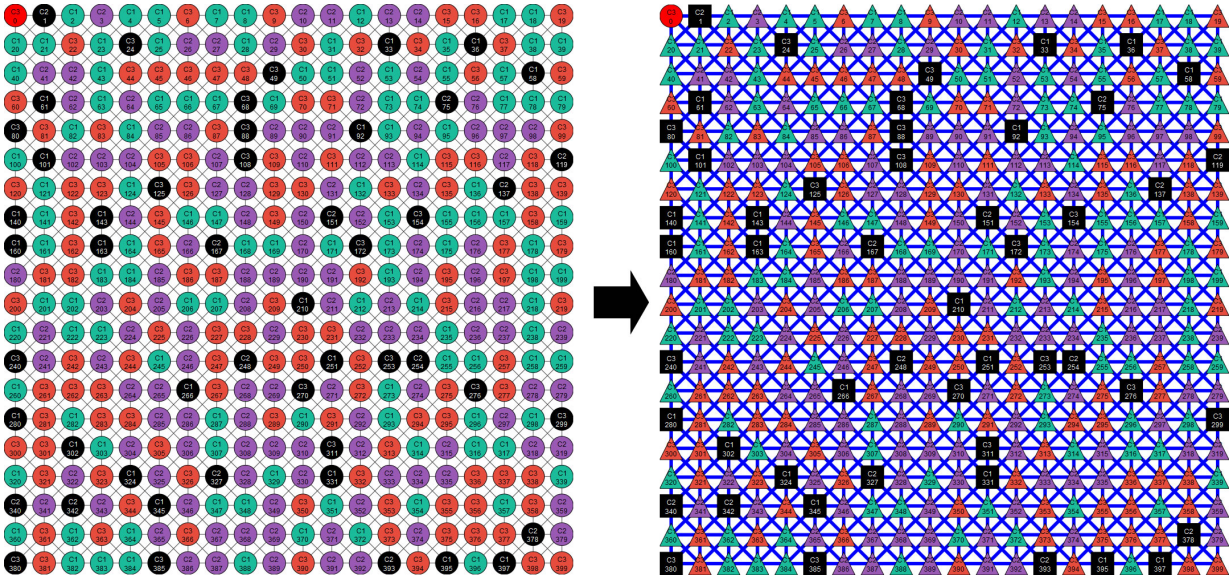

Figure 4.6: DFD with 100% accuracy. Graph visualization for the network with 50 detected Byzantine nodes, aided by 3 different class segmentation, and an average contribution of 37.5 routed message per node. Triangle shape means that are proven to be healthy

The Figure [4.6] shows a dramatic increase in overall detection accuracy as more reports are accumulated in the Master node. With the presence of 50 Byzantine nodes, it took an overall number routed hop count of (15,000) for a wireless network with 400 nodes; which statistically means that it required an average contribution of routing 37.5 messages per node to reach a detection accuracy of 100%. In our experiment, Master node has been placed in the top left corner of the network to gain maximum routing path length.

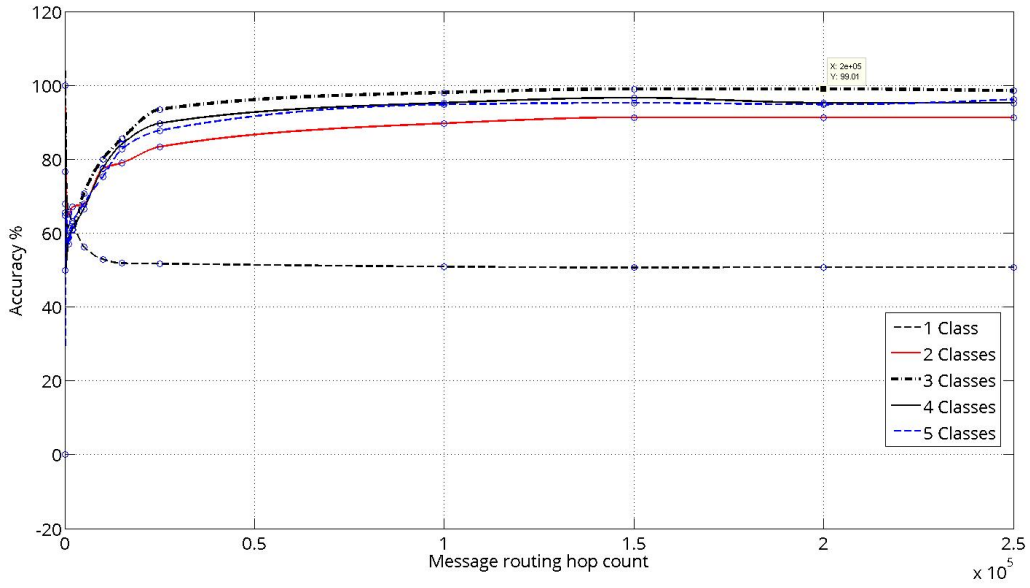

Figure 4.7: Accuracy measurement of Distributed Fault Detection (DFD) in network size of 400 with 133 Byzantine nodes. Using 5 different classes with variant communication hop length

Increasing the number of Byzantine nodes to (133), which is almost one-third of the size of the network, will give us another results with degraded accuracy if we compare it to the same routing hop count with the previous scenario. By looking into Figure [4.7], we can clearly see that we gained less accuracy at $1.5x10^4$ using 3 segmentation classes. The accuracy measured was 91.1% at (15,000) routing hop count. And the maximum measured accuracy was 99.01% at (100,000) routing hop count, which translates into 250 message contribution per node. However, having 133 is quite exaggerated and considered to be the worst case scenario.

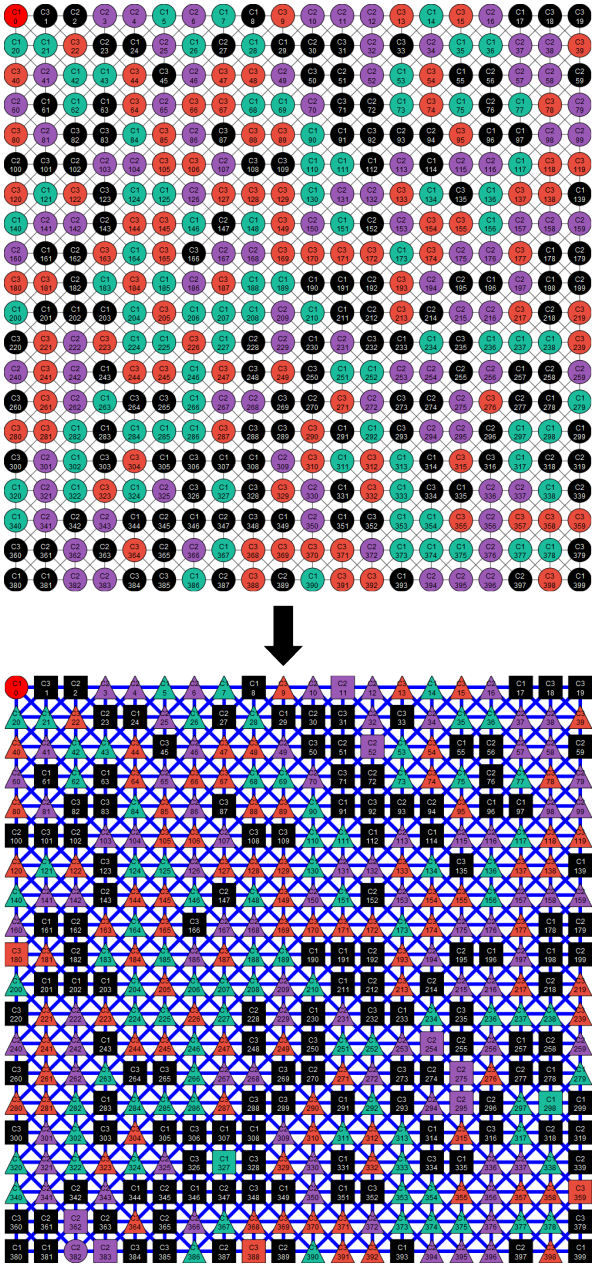

Figure 4.8: Accuracy measurement of 91% for the DFD, having 133 byzantine nodes, with 3 classes, and an average contribution of 37.5 message per node. Normal nodes highlighted in square shapes are falsely tagged as byzantine by the algorithm

Results generated have been gathered, and concisely been compiled in Table [4.2]. Notably, we can see that the Message contribution per node is increasing proportionally to the number of simulated Byzantine nodes; and in contrast the detection accuracy decreases as the number of Byzantine increases.

Mesh Grid 400	Message Contribution Per Node	Accuracy	Classes
25 Byzantine	25	100%	2
100 Byzantine	62.5	100%	2
133 Byzantine	75	97%	3
200 Byzantine	250	97%	3
250 Byzantine	375	95%	9

Table 4.1: Simulation results matrix for Distributed Fault Detection Accuracy

4.6.3 Identifying the Sufficient Segmentation Value

Based on our simulation, segmenting a network with 400 nodes into three even groups has always shown to be effective, at least for our experiment scope and problem search space defined. Moreover, we can easily validate this finding by using the probability equation [3.5]. Here is the successive order probability chart for having 400 nodes segmented into three groups, Figure [[4.9]. From the graph shown, we can see that the maximum cost of accusation length of a single report could be identified by trapping nodes is four, with a probability of (0.0118).

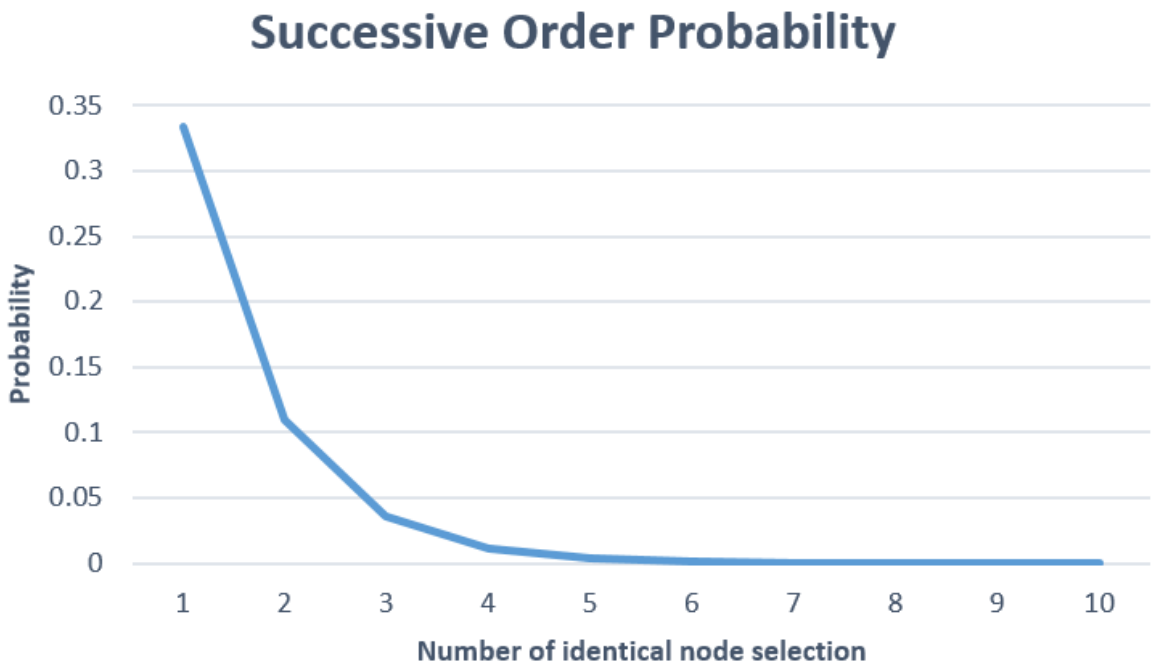

Figure 4.9: Probability of finding a path consist of nodes belonging to the same group. Network size N=400, and S=3

Figure [4.10] presents an average of accumulated simulation results using 5 group segmentation. Except for the case of having a single group, all of the other group segmentation [2-5] has shown exceptional progressive performance as the number of routed messages increases. For the case of having a single group, the inference model will have the highest uncertainty value of 0.5 which has not been projected in the graph, because we fixed the accuracy range within [80%-100%] to compare which class has provided a constant rapid increase in precision.

Having more than three groups will come with a cost for our algorithm, as it implies

that we will have to store more hash values inside of the forwarded packets according to the number of classes been used. Nevertheless, practically segmenting our network into a minimum odd number of classes (3), it seems to be the least cost that will make the proposed DFD algorithm efficiently effective. Read Section [3.3.1] to check the report packet representation.

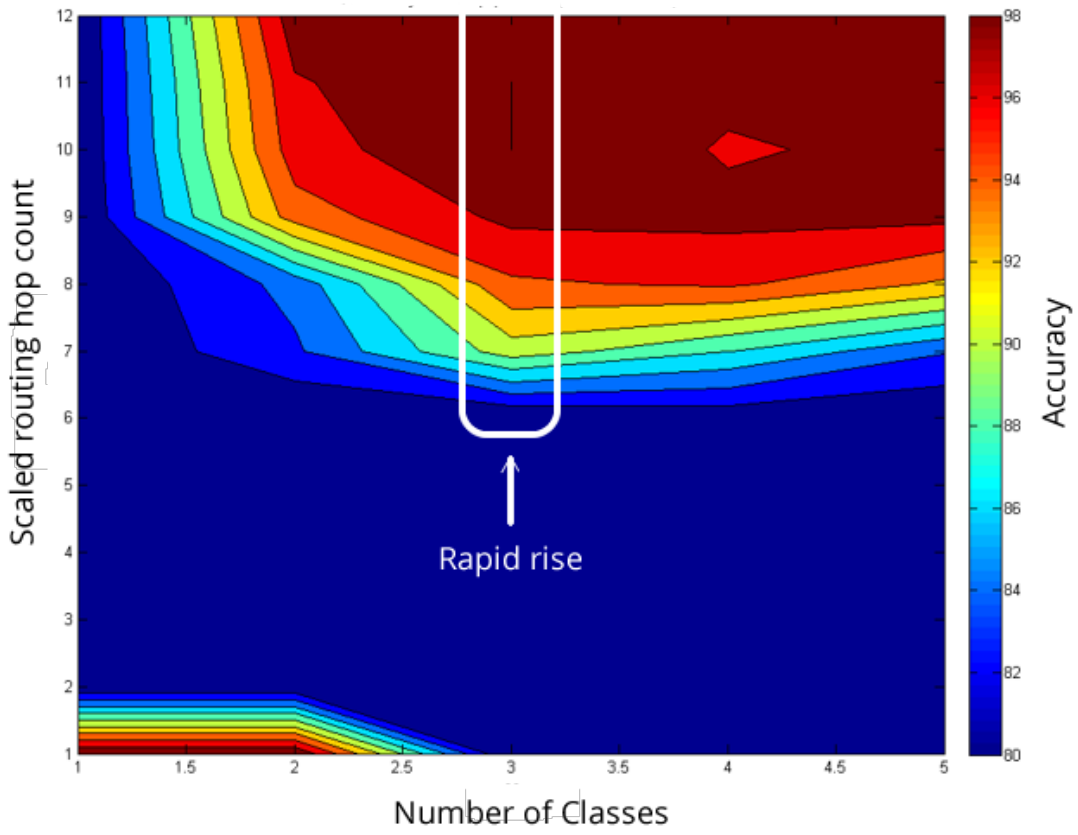

Figure 4.10: Heat-map graph visualization that depict the performance of detection using different number of classes

Hypothetically speaking, segmenting a Wireless network into three group should be optimal with the size of 400 nodes. Mainly, because the number of classes should be fully minimized in our experiment, not only to reduce the cost of report message overhead; but, also because we need to increase the probability of having multiple pairs of nodes from the same group (Class) to be presented in any random routing path been discovered. In contrast, applying the next minimal option of segmenting the network into two groups would under-fit the system; because it will increase the possibility of generating node clustering effect [1]. To prove our hypothesis a further analysis has been done for two different network sizes (10x10, 20x20, and 40x40), these networks have 33% of their nodes simulated as Byzantine nodes. See Figure [4.11] to observe the result of $40x40$ network segmented optimally with three different classes out of 9 other segmentation variation simulated from 1 to 10.

[1] Clustering effect, is when the network have multiple nodes belonging to the same group connected to each other. This effect will make the accusation report precision become lower.

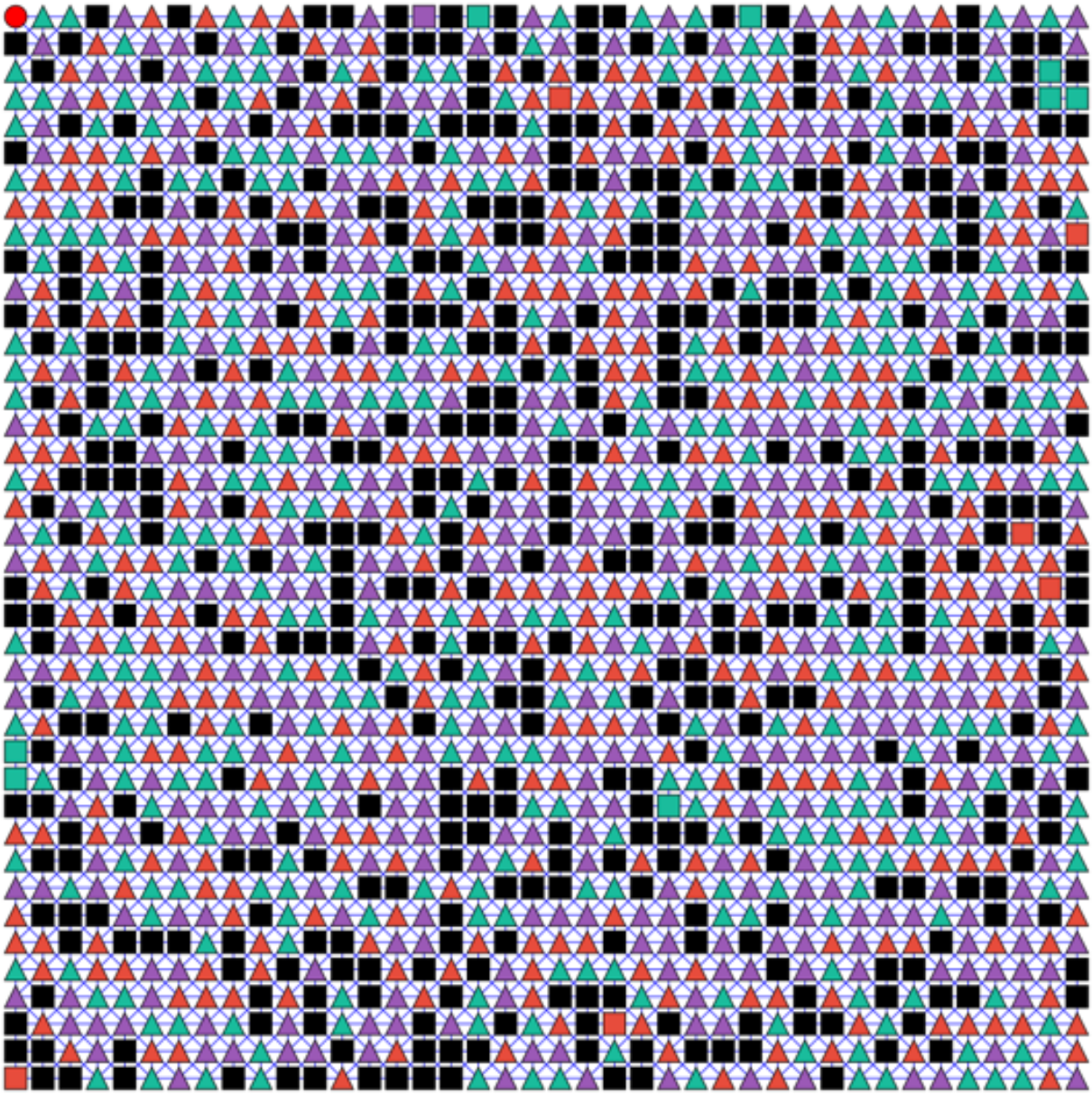

Figure 4.11: DFD with 95% accuracy. Graph visualization for a network with 1600 nodes and 533 simulated Byzantine nodes, aided by 3 different class segmentation, and an average contribution of 42 routed message per node.

4.6.4 Network lifetime and power consumption

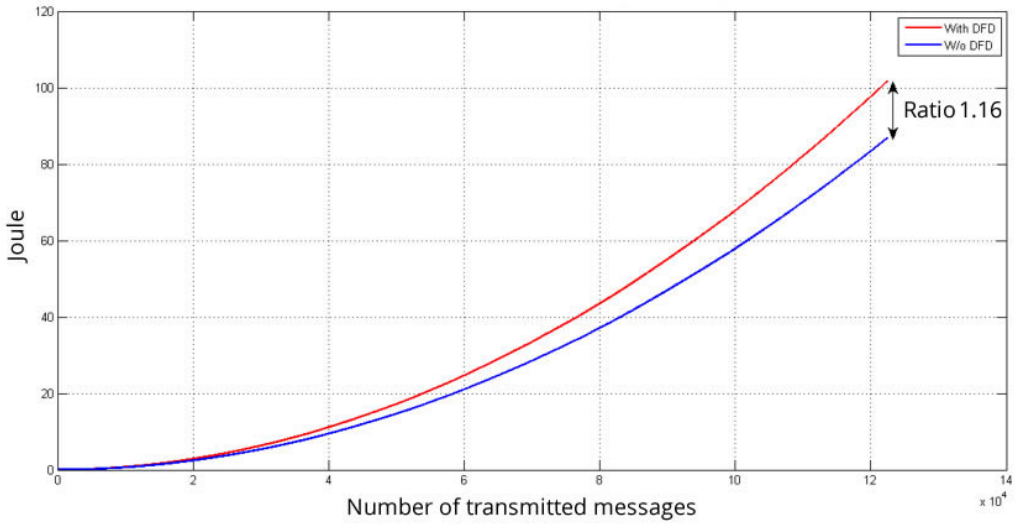

Figure 4.12: Cumulative power consumption for the entire network (400 Nodes / 133 Byz / 3 Classes / 68,010 messages)

In the simulation, the power capacity of each node has been assumed to be 1 joule so that the overall network power capacity would be 400 Joules'. Running the worst case scenario experiment of having 133 Byzantine nodes, the power consumption of the entire network is represented in Figure [4.12]. Corresponding to the radio model described in Section [4.2], the overall power consumption of the network to send $68,010$ messages with an average of 10 hops per message, has taken 116% more power applying the DFD algorithm. Off course, this measurement does not address the individual power consumption of each node, which obviously becomes crucial for the nodes surrounding the Master Node which will have the highest routing contribution percentage; which implies that those nodes will drain most of

their energy before others.

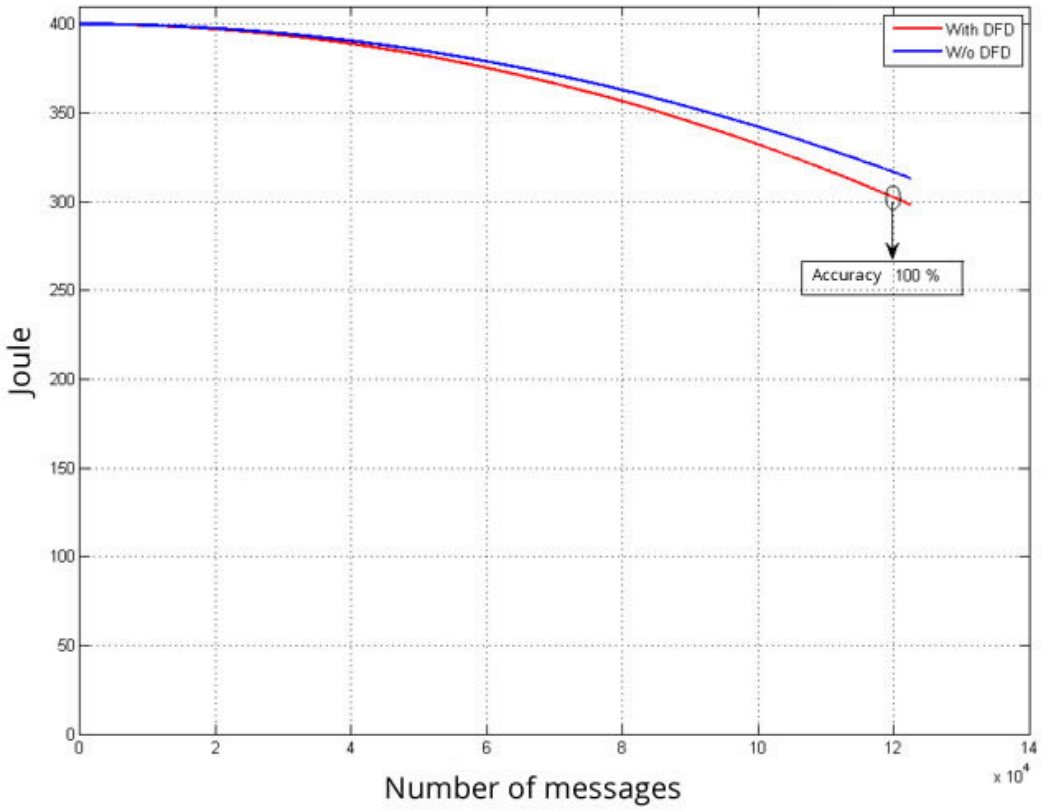

Figure 4.13: Network lifetime graph (400 Nodes / 133 Byz / 3 Classes / 68,010 messages)

The reverse graph has been plotted for the entire network life time. See Figure [4.13].

4.6.5 Message Overhead

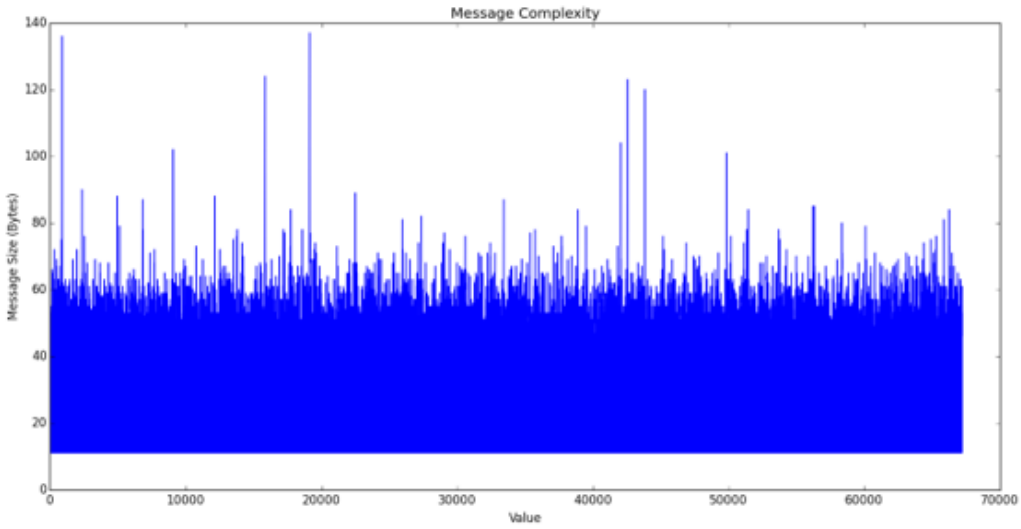

Figure 4.14: Metadata overhead size graph with average size of 50 bytes per route for a large network with the following specifications (400 Nodes / 133 Byz / 3 Classes / 68,010 messages)

4.6.6 Algorithm Comparison

In the following table, we will highlight some of the Distributed Byzantine Fault detection proposed in the literature and try to show their strength and weaknesses along our proposed DFD algorithm.

Algorithm	Communication Complexity	Target Application Domain	Advantages	Disadvantages
Proposed DFD	Adding Encoded Metadata utilizing forwarded messages whenever possible	WANET including WSN	- Can reach 100% accuracy even if 33% or more of the nodes were faulty. - Low power consumption with symmetric lightweight key hashing	- Does not include fault probability in tests. Assuming faults are permanent. - Difficulty to identify faulty Nodes reside in the edge of a static network.
PK DFD [26]	Extra Local Message Exchange	WSN	- High detection accuracy with less false alarm	- It is assumed that all the fault free sensor nodes measuring the same value - The algorithm is susceptible to the degree of neighboring nodes
JSA [6]	Extra Local Message Exchange	WSN	Accuracy is over 97% even when 25% nodes are faulty	- Not scalable due to exchange information between neighbors - Focusing on hardware faults only
Fast ECDSA [34]	Extra Local Message Exchange	WSN	- Complexity reduction for message exchange between cluster heads. - Smaller key size with better security using ECC	- High computation from cluster heads to calculate and validate Public Key Cryptography parameters - Certification Authority is required

Table 4.2: Comparison between various Distributed Fault Detection algorithms

Chapter 5

DFD Algorithm - Simulation Source Code

5.1 Simulator Code in Github

The source code for the proposed simulation has been publicly posted to the engineering community for further enhancement and validation proposes.

https://github.com/aelbuni/distributed-byzantine-detection-algorithm

5.2 Sample Experimentation Code

All figure properties can also be manipulated from the source code. Here's an example:

```
import numpy as np
```

```
import decisionNodeWithGraph as DN
import randomNetworkGenerator as GEN
import recursiveRandomRouting as DFD
import matplotlib.pyplot as plt
import utility as util

# =============== Initialize parameters ===========
sinkLocation = 0
meshSize = 10
numClasses = 3
numMessages = 900
numberHops = 10
numByzantine = np.floor((meshSize**2)/5) # 1/5 of the entire network size
mobility = False

g = GEN.generate_random_wanet_mesh_graph(meshSize, numClasses, numByzantine, 40, 12, si

byzNodes, healthyNodes, byzListFormat, messageSizeComplexity, routedHopCount, energyCon

for testCommunication in range(0, numMessages):
    if mobility:
        g = util.random_mobility(g, meshSize**2);
    byzNodes2, healthyNodes2, byzListFormat2, messageSizeComplexity2, routedHopCount2,
```

```
    # print "Message" + str(testCommunication) + " : " + str(g.vs[0]["msg"])
    byzNodes.extend(byzNodes2)
    healthyNodes.extend(healthyNodes2)
    byzListFormat.extend(byzListFormat2)
    routedHopCount += routedHopCount2
    messageSizeComplexity.extend(messageSizeComplexity2)
    energyConsum.extend(energyConsum2)
    energyConsumWithout.extend(energyConsumWithout2)

# no line highlights (false)
util.plot_save_graph(g, "graph.png", healthyNodes)

DN.result_analysis(g, healthyNodes, byzNodes, routedHopCount, messageSizeComplexity, me
```

Chapter 6

Conclusion and Future Work

6.1 Summary and Contributions

Wireless ad-hoc networks promise attainable and valuable solutions to many problems in different fields. Sensor technology today is advancing relatively fast from research contexts to industrial application contexts, and with increasing interest in Internet of Things (IoT) and the rising concern about security, data secrecy, and system reliability; research and study for providing new distributive and cooperative security schemes come with tremendous value. Most wireless applications, if not all, must be provided with security as an intrinsic feature. Also, fault tolerance and self-healing capabilities will help boost the network performance and its service reliability.

The motivation behind this thesis is to design a distributive and cooperative security scheme for multi-hop communication networks that is suitable to be applied for constrained

and mobile wireless network. Consequently, the aim was to relax the problem as much as possible to reduce the cost of implementation, operation, and deployment. Stochastic network segmentation and deployment has been employed to bypass formidable challenges such as initial key distribution, network self-organization, digital signature, key management, the level of security communication overhead, and lastly the data integrity. Here is a detailed list of the complexity reduction that our algorithm have utilized:

- **Initial Key Distribution** initial key distribution after deployment is a challenging problem that can compromise the confidentiality of a network if it does not be handled correctly. One of the secure techniques is to use public cryptography and rely on having a centralized certificate authority to grant keys for nodes, which is not suitable for constrained ad-hoc networks because of its cost. Our technique is based on pre-grouping and pre-distribution of keys before deployment, which is quite manual; but, in return it serves the kind of network devices that have limited resources in terms of energy, computational power, and memory.

- **Key Management** Special shared permutation function should be stored inside of each group nodes so that they can use it to derive fresh keys periodically from the Master key. The frequent update for the group keys will help the network to be more secure against adversaries and intruders.

- **Level of Security** One of the advantages of the proposed stochastic deployment and distributive security scheme, is that if a message authentication code has been generated using $16bit$ security level; this means that the overall network level of security should be the individual group security level multiplied by how many groups

does the whole network has. So if a network has 3 groups with $16bit$ MAC algorithm, then based on probability, an adversary should be required to break the three group's MAC with cumulative security of $48bit$ to be able to forge a message or generate a collision. Unless the malicious node has been placed right after the originator of the message of course; so raising the base security level to a sufficient point is a necessary task.

- **Communication Overhead** Numerous other DFD algorithms proposed relied on adding extra local communication to the neighbors to do fault discovery and reporting; which, based on our Radio Model it requires energy to boot the transmission electronics, and to transmit the message wirelessly. Nevertheless, this cost has been reduced in our proposed research to the level of adding an optimized non-repeated metadata to the routed messages to report healthy and unhealthy nodes. Also, the algorithm could be turned on and off depending on network orchestration demands.

- **Data Integrity** Digital Signature is our of options when we consider Wireless Sensor Network, for instance, however, new studies have proposed fast public Fast Elliptic Curve Cryptography (ECSDA) to secure fault tolerance algorithms. Though, it is known that symmetric key based security is more favorable and flexible in terms of calculation complexity over public key based security.

The proposed scheme and DFD algorithm have the following contributions:

- The stochastic deployment and network segmentation scheme has simplified many problems and relaxed them to overcome real world implementation and production

challenges.

- Proposed a new ground (Self-policing scheme) for stochastic distributive security, which could be extended to be applied for encryption with an augmented level of security based on the number of groups.

- Highly compatible with mobility; and yet mobility does increase the DFD accuracy as it brings nodes from the edge of a network into the center to expose them to be diagnosed by other neighboring nodes frequently.

- The proposed distributive and the cooperative algorithm have been able to reach 100% of detection accuracy for permanent soft and hard faults; with 37.5 message contribution per node. The simulated network size was 400 nodes with 100 Byzantine nodes assuming construction of best effort random paths from source to destination, which has been inspired from AODV routing protocol.

- The total complexity of the message overhead has been reduced to have an average of 50 bytes overhead size without involving any encoding, local caching or duplication removal optimization, which can reduce the cost further to at least 30 bytes of average overhead size. Low security of MAC has been used ($16bit$), relying on the stochastic deployment and augmentation of security levels of at least three groups, which should raise it theoretically to $40bit$ level of total level of security.

6.2 Future work

The work presented in this thesis can still have various extensions and directions for future work. One main extension is to add an isolation mechanism whenever Byzantine node is found, so a technique such as Exclusion Basis System (EBS) [9] can be utilized to cure the network of any harmful faulty behavior. Also, lots of enhancements can be made to the decision-making center, where event based action can be defined and activated; along with enhancing the inference model, orchestration and adding other feedback testing criteria to be certain about any reported faults.

A second possible extension is to emulate the algorithm on top of Ad-hoc On-Demand Distance Vector (AODV) multi-hop routing protocol, as it would be highly suitable ground for further simulation studies. Moreover, storing and delaying reports and retaining reporting records locally to avoid report duplication should be applied to optimize the algorithm further and make it more efficient.

To summarize, Byzantine behavior that is hard to be identified and tracked seem to introduce conflicts; however, introducing fault discovery is essential for isolating and fixing any formidable faults for any ad-hoc wireless network, whether it was mobile, or vehicular, or a sensor network. The results of this thesis provide a good starting point for a deeper study of secure data routing protocols.

References

[1] Aymen Abid, Abdennaceur Kachouri, Awatef Ben Fradj Guiloufi, Adel Mahfoudhi, Nejah Nasri, and Mohamed Abid. Centralized knn anomaly detector for wsn. In *Systems, Signals & Devices (SSD), 2015 12th International Multi-Conference on*, pages 1–4. IEEE, 2015.

[2] Cedric Adjih, Ichrak Amdouni, and Pascale Minet. Vcm: the vector-based coloring method for grid wireless ad hoc and sensor networks. In *Proceedings of the 15th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, pages 213–222. ACM, 2012.

[3] Ichrak Amdouni, Pascale Minet, and Cédric Adjih. Node coloring for dense wireless sensor networks. *arXiv preprint arXiv:1104.1859*, 2011.

[4] Baruch Awerbuch, David Holmer, Cristina Nita-Rotaru, and Herbert Rubens. An on-demand secure routing protocol resilient to byzantine failures. In *Proceedings of the 1st ACM workshop on Wireless security*, pages 21–30. ACM, 2002.

[5] Huimin Chen, Vesselin P Jilkov, and X Rong Li. Optimizing decision fusion in the presence of byzantine data. In *Information Fusion (FUSION), 2014 17th International Conference on*, pages 1–8. IEEE, 2014.

[6] Jinran Chen, Shubha Kher, and Arun Somani. Distributed fault detection of wireless sensor networks. In *Proceedings of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks*, pages 65–72. ACM, 2006.

[7] Thomas Clausen and Philippe Jacquet. Optimized link state routing protocol (olsr). Technical report, 2003.

[8] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal, Complex Systems*, 1695(5):1–9, 2006.

[9] Mohamed Eltoweissy, M Hossain Heydari, Linda Morales, and I Hal Sudborough. Combinatorial optimization of group key management. *Journal of Network and Systems Management*, 12(1):33–50, 2004.

[10] Gaurav Gupta and Mohamed Younis. Fault-tolerant clustering of wireless sensor networks. In *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, volume 3, pages 1579–1584. IEEE, 2003.

[11] Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel. The case for byzantine fault detection. In *HotDep*, 2006.

[12] Charles L Hedrick. Routing information protocol. 1988.

[13] Wendi B Heinzelman, Anantha P Chandrakasan, and Hari Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on wireless communications*, 1(4):660–670, 2002.

[14] Peng Jiang. A new method for node fault detection in wireless sensor networks. *Sensors*, 9(2):1282–1294, 2009.

[15] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile computing*, pages 153–181. Springer, 1996.

[16] Ryszard Klempous, Jan Nikodem, Lukasz Radosz, and Norbert Raus. Adaptive misbehavior detection in wireless sensors network based on local community agreement. In *14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07)*, pages 153–160. IEEE, 2007.

[17] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[18] Jongdeog Lee, Krasimira Kapitanova, and Sang H Son. The price of security in wireless sensor networks. *Computer Networks*, 54(17):2967–2978, 2010.

[19] Myeong-Hyeon Lee and Yoon-Hwa Choi. Fault detection of wireless sensor networks. *Computer Communications*, 31(14):3469–3475, 2008.

[20] Xiang-Yang Li, Peng-Jun Wan, and Ophir Frieder. Coverage in wireless ad hoc sensor networks. *IEEE Transactions on Computers*, 52(6):753–763, 2003.

[21] Junhai Luo and Zan Cao. Distributed detection in wireless sensor networks under byzantine attacks. *International Journal of Distributed Sensor Networks*, 2015:222, 2015.

[22] Xuanwen Luo, Ming Dong, and Yinlun Huang. On distributed fault-tolerant detection in wireless sensor networks. *IEEE Transactions on Computers*, 55(1):58–70, 2006.

[23] J-P Martin, Lorenzo Alvisi, and Michael Dahlin. Small byzantine quorum systems. In *Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on*, pages 374–383. IEEE, 2002.

[24] Tahir Naeem and Kok-Keong Loo. Common security issues and challenges in wireless sensor networks and ieee 802.11 wireless mesh networks. *3; 1*, 2009.

[25] Meenakshi Panda and PM Khilar. Energy efficient soft fault detection algorithm in wireless sensor networks. In *Parallel Distributed and Grid Computing (PDGC), 2012 2nd IEEE International Conference on*, pages 801–805. IEEE, 2012.

[26] Meenakshi Panda and PM Khilar. Distributed byzantine fault detection technique in wireless sensor networks based on hypothesis testing. *Computers & Electrical Engineering*, 48:270–285, 2015.

[27] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 27(2):228–234, 1980.

[28] Charles Perkins, Elizabeth Belding-Royer, and Samir Das. Ad hoc on-demand distance vector (aodv) routing. Technical report, 2003.

[29] Adrian Perrig, Robert Szewczyk, Justin Douglas Tygar, Victor Wen, and David E Culler. Spins: Security protocols for sensor networks. *Wireless networks*, 8(5):521–534, 2002.

[30] F Shebli, I Dayoub, A Okassa M'foubat, A Rivenq, and JM Rouvaen. Minimizing energy consumption within wireless sensors networks using optimal transmission range between nodes. In *Signal Processing and Communications, 2007. ICSPC 2007. IEEE International Conference on*, pages 105–108. IEEE, 2007.

[31] Norihiro Sota and Hiroaki Higaki. Byzantine failure detection in wireless ad-hoc networks. In *Sarnoff Symposium, 2015 36th IEEE*, pages 173–178. IEEE, 2015.

[32] Aditya Vempaty, Onur Ozdemir, Keshav Agrawal, Hao Chen, and Pramod K Varshney. Localization in wireless sensor networks: Byzantines and mitigation techniques. *IEEE Transactions on Signal Processing*, 61(6):1495–1508, 2013.

[33] John Paul Walters, Zhengqiang Liang, Weisong Shi, and Vipin Chaudhary. Security in distributed, grid, and pervasive computing. *Wireless sensor network security: A survey*, 2006.

[34] Jiawei Xu, Keda Wang, Chao Wang, Feng Hu, Zhenhua Zhang, Shiyi Xu, and Jie Wu. Byzantine fault-tolerant routing for large-scale wireless sensor networks based on fast ecdsa. *Tsinghua Science and Technology*, 20(6):627–633, 2015.

[35] Ronald R Yager. On the dempster-shafer framework and new combination rules. *Information sciences*, 41(2):93–137, 1987.

[36] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer networks*, 52(12):2292–2330, 2008.