DEFENCE AND TRACEBACK MECHANISMS IN OPPORTUNISTIC WIRELESS NETWORKS

By

MAJEED ABDULLA AL-AJEELY
(MSc Mobile Computing)

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY AT DEAKIN UNIVERSITY MELBOURNE, AUSTRALIA DECEMBER, 2015

© Copyright by Majeed Abdulla Alajeely, 2015
I am the author of the thesis entitled Defence and Traceback Mechanisms in Opportunistic Wireless Networks submitted for the degree of Doctor of Philosophy.

This thesis may be made available for consultation, loan and limited copying in accordance with the Copyright Act 1968.

'I certify that I am the student named below and that the information provided in the form is correct'

Full Name: ..........MAJEED ABDULLA AL-AJEELY...............................................

Signed: .................................................................

Date: .................................................12/2/2016..........................................................
I certify the following about the thesis entitled “Defence and Traceback Mechanisms in Opportunistic Wireless Networks” submitted for the degree of Doctor of Philosophy

a. I am the creator of all or part of the whole work(s) (including content and layout) and that where reference is made to the work of others, due acknowledgment is given.

b. The work(s) are not in any way a violation or infringement of any copyright, trademark, patent, or other rights whatsoever of any person.

c. That if the work(s) have been commissioned, sponsored or supported by any organisation, I have fulfilled all of the obligations required by such contract or agreement.

d. That any material in the thesis which has been accepted for a degree or diploma by any university or institution is identified in the text.

e. All research integrity requirements have been complied with.

'I certify that I am the student named below and that the information provided in the form is correct'

Full Name: .....MAJEED ABDULLA AL-AJEELY..........................................................

Signed: ........ .......................................................... ................................

Date: ..........16/2/2016..........................................................
Acknowledgements

I would like to express my special appreciation and thanks to my supervisor Dr. Robin Doss for his continuous follow up and valuable feedback on my work. I would like to thank you for encouraging my research and for allowing me to grow as a researcher. Your advice on both research as well as on my career have been invaluable.

A special thanks to my family. Words cannot express how grateful I am to my wife Asma’a for supporting me for everything, and especially I can’t thank you enough for encouraging me throughout this experience. To my beloved sons and daughter (Yousef, Adam, Noah and Sarah), I would like to express my thanks for all of you for being such a good and always cheering me up.

I would also like to thank my father and mother (God rest their souls) for all of the sacrifices that they made for me and my brothers and sisters.

Finally I thank my God, for letting me through all the difficulties. I have experienced Your guidance day by day. You are the one who let me finish my degree. I will keep on trusting You for my future.
Publications

Journals

Conferences
1. Majeed Alajeely, Asma’a Ahmad and Robin Doss “Malicious node detection in OppNets using hash chain”, ICCSNT 2015, 4th International Conference on Computer Science and Network Technology, IEEE, December 2015, Harbin, China. (Chapter 5), To Appear
Table of Contents

List of Tables v
List of Figures ix
Abstract x

1 Introduction 1
  1.1 Motivation .................................................... 2
  1.2 Research Objective ........................................... 3
  1.3 Research Problems ............................................ 4
  1.4 Contributions ................................................ 4
  1.5 Overview of the Thesis Organization ...................... 5

2 Literature Review 7
  2.1 Overview on Routing Protocols in OppNets ............... 9
    2.1.1 Replication Family ...................................... 9
      2.1.1.1 Flooding Technique ................................ 9
      2.1.1.2 Coding Technique .................................. 11
    2.1.2 Forwarding Family ..................................... 12
      2.1.2.1 Basic Technique .................................. 12
      2.1.2.2 Prediction Technique ............................... 13
      2.1.2.3 Time Technique .................................... 15
      2.1.2.4 Buffer Management Technique ...................... 15
      2.1.2.5 Social Relationship Technique .................... 17
    2.1.3 Hybrid Family ......................................... 19
      2.1.3.1 Utility Replication Techniques .................. 19
2.1.3.2 Improved Spray and Wait Technique ............... 21
2.1.3.3 Improved Epidemic Technique ...................... 21
2.1.3.4 Coding Technique ................................. 22
2.2 OppNets Routing Protocols Summary ..................... 22
2.3 Evaluating Routing Protocols in OppNets ................. 26
   2.3.1 Metrics Used for Evaluation ....................... 26
   2.3.2 Framework of Evaluation .......................... 26
   2.3.3 Simulation Settings ............................... 27
   2.3.4 Simulation Results ............................... 28
      2.3.4.1 Scenario 1 .................................. 28
      2.3.4.2 Scenario 2 .................................. 30
      2.3.4.3 Scenario 3 .................................. 33
2.4 Security and Trust in OppNets ............................ 34
   2.4.1 Defence against Wormhole Attacks .................... 35
   2.4.2 Defence against Blackhole Attacks .................... 36
   2.4.3 Defence against Dropping and Selective Dropping Attacks .. 37
   2.4.4 Defence against Selfish Attack ....................... 41
   2.4.5 Defence against Sybil Attacks ....................... 42
   2.4.6 Anti-Localization Techniques ....................... 43
2.5 Security Solutions Based on Intrusion Detection System .... 44
2.6 Trust Management Evaluation ............................. 45
2.7 Trust Strategies in OppNets .............................. 49
   2.7.1 Social-Based Trust Strategies ....................... 49
   2.7.2 Reputation-Based Trust Strategies .................... 58
   2.7.3 History-Based Trust Strategies ....................... 69
2.8 Summary ............................................... 70

3 Catabolism Attack and Anabolism Defence ............... 72
   3.1 Introduction ........................................ 73
   3.2 Overview on Catabolism Attack and Anabolism Defence .......... 74
      3.2.1 Assumptions .................................. 75
      3.2.2 Anabolism Defence (Time-based Approach) .......... 76
# TABLE OF CONTENTS

3.3 Mathematical Model ................................................. 79  
   3.3.1 Detection of Fake Packets .................................... 80  
      3.3.1.1 Optimization of the Number of Packets ................. 83  
   3.3.2 Malicious Node Detection .................................... 84  
3.4 Simulation Settings ................................................. 89  
   3.4.1 Simulation Results and Analysis .............................. 91  
      3.4.1.1 Destination-based Approach ......................... 91  
      3.4.1.2 Node by Node-based Approach ....................... 97  
3.5 Summary ............................................................... 102  

4 Anabolism Defence - Merkle Tree Approach 104  
  4.1 Introduction ....................................................... 104  
  4.2 Overview on the Merkle Tree Approach Mechanism ............... 105  
     4.2.1 Assumptions ............................................. 105  
     4.2.2 Merkle Based Defence and Traceback Method ............ 106  
  4.3 Mathematical Model .............................................. 110  
  4.4 Simulation Settings .............................................. 114  
     4.4.1 Simulation Results and Analysis ....................... 116  
  4.5 Summary ............................................................. 124  

5 Anabolism Defence - Hash Chain Approach 125  
  5.1 Introduction ....................................................... 125  
  5.2 Overview on the Hash Chain Approach Mechanism ............... 126  
     5.2.1 Assumptions ............................................. 127  
     5.2.2 Attack and Defence Scenario ............................ 127  
  5.3 Mathematical Model .............................................. 132  
  5.4 Simulation Settings .............................................. 138  
     5.4.1 Simulation Results and Analysis ....................... 139  
  5.5 Summary ............................................................. 145  

6 Conclusion and Future Works 147  
  6.1 Summary of Thesis ............................................... 147  
  6.2 Discussion .......................................................... 150
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3 Future Work</td>
<td>151</td>
</tr>
<tr>
<td>Bibliography</td>
<td>153</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Summary of Routing Protocols in OppNets- Replication Family . . . 23
2.2 Summary of Routing Protocols in OppNets- Hybrid Family . . . . . 23
2.3 Summary of Routing Protocols in OppNets- Forwarding Family-1 . . 24
2.4 Summary of Routing Protocols in OppNets- Forwarding Family-2 . . 25
2.5 A comparison of trust strategies . . . . . . . . . . . . . . . . . . . . . 50
List of Figures

2.1 OppNets family tree .................................................. 9
2.2 Scenario 1: Delivery rates ........................................... 29
2.3 Scenario 1: Delivery delay ............................................ 30
2.4 Scenario 1: Overhead ratio .......................................... 31
2.5 Scenario 2: Delivery rates ........................................... 31
2.6 Scenario 2: Delivery delay ............................................ 32
2.7 Scenario 2: Overhead ratio .......................................... 32
2.8 Scenario 3: Delivery rates ........................................... 33
2.9 Scenario 3: Delivery delay ............................................ 34
2.10 Scenario 3: Overhead ratio ........................................... 34

3.1 Catabolism attack where a malicious nodes (t15) drops and injects a
new packet ................................................................. 75
3.2 Packets Path with Packet Creation Time .......................... 76
3.3 Different sequence packets dropping/injecting in 2 malicious nodes path 79
3.4 Same sequence packets dropping/injecting in 2 malicious nodes path . 80
3.5 Probability of at least one packet surviving all malicious hops in a
path with 2 malicious nodes ............................................. 82
3.6 Probability of at least one packet surviving all malicious hops in a
path with 5 malicious nodes ............................................. 82
3.7 Probability of at least one packet surviving all malicious hops in a
path with 20 malicious nodes .......................................... 83
3.8 Number of packets needed to achieve $\alpha = (0.85, 0.90, 0.95)$ in a path
with 2 malicious nodes ................................................. 84
3.9 Number of packets needed to achieve $\alpha = (0.85, 0.90, 0.95)$ in a path with 5 malicious nodes ........................................ 85
3.10 Number of packets needed to achieve $\alpha = (0.85, 0.90, 0.95)$ in a path with 20 malicious nodes ................................. 86
3.11 Probability of detecting malicious nodes in a path with 2 malicious nodes ......................................................... 88
3.12 Probability of detecting malicious nodes in a path with 5 malicious nodes ......................................................... 88
3.13 Probability of detecting malicious nodes in a path with 20 malicious nodes ......................................................... 89
3.14 Probability of accurate detecting of malicious nodes with 2 malicious nodes in a path ............................................. 90
3.15 Probability of accurate detecting of malicious nodes with 4 malicious nodes in a path ............................................. 91
3.16 Probability of accurate detecting of malicious nodes with 6 malicious nodes in a path ............................................. 92
3.17 Probability of accurate detecting of malicious nodes with 8 malicious nodes in a path ............................................. 93
3.18 Fake packet detection accuracy as the number of malicious nodes increases ......................................................... 94
3.19 Fake packet detection accuracy based on the mobility of nodes .... 95
3.20 False negative rate as the number of malicious nodes increases .... 96
3.21 Nodes detection accuracy as the number of malicious nodes increases ................................................................. 97
3.22 Fake packet detection accuracy as the number of malicious nodes increases ......................................................... 99
3.23 Fake packet detection accuracy based on the mobility of nodes .... 99
3.24 Network Traffic Reduction ......................................................... 100
3.25 False negative rate as the number of malicious nodes increases .... 101
3.26 Nodes detection accuracy, our approach with Epidemic vs ACK and network coding techniques .................................. 102

4.1 Packet Dropping/Injecting at Malicious Node (t15) ..................... 106
4.2 Merkle tree, “H07” is the root value ................................. 107
4.3 Packets with Merkle root value “H07” ................................. 107
4.4 Fake packets conditions that we need to achieve ...................... 110
4.5 Two malicious nodes sending to each other in one message path 110
4.6 Probability of receiving at least one fake packet ($\beta$) in a path with 1
malicious nodes ............................................................ 113
4.7 Probability of receiving at least one fake packet ($\beta$) in a path with 2
malicious nodes ............................................................ 114
4.8 Probability of receiving at least one fake packet ($\beta$) in a path with 5
malicious nodes ............................................................ 115
4.9 Number of packets needed to achieve $\beta = (0.85,0.90,0.95)$ in a path
with 1 malicious nodes ................................................... 116
4.10 Number of packets needed to achieve $\beta = (0.85,0.90,0.95)$ in a path
with 2 malicious nodes ................................................... 117
4.11 Number of packets needed to achieve $\beta = (0.85,0.90,0.95)$ in a path
with 5 malicious nodes ................................................... 118
4.12 Node detection accuracy for different OppNets protocols .............. 118
4.13 Node detection accuracy based on the mobility of nodes .............. 119
4.14 Comparison of malicious node detection accuracy: Our approach (Epi-
4.15 Comparison of malicious node detection accuracy: Our approach (PRoPHET)
4.16 Comparison of malicious node detection accuracy: Our approach (Spray
4.17 Comparison of malicious node detection accuracy: Our approach (Max-
4.18 Comparison of malicious node detection accuracy: Our approach (First
4.19 False negative rate as the number of malicious nodes increases ........ 123
4.20 Nodes detection accuracy percentage vs. Simulation time .............. 123
5.1 Packet Dropping/Injecting at Malicious Node (t15) ..................... 127
5.2 Packets with hash chain ............................................... 128
5.3 Path with two malicious nodes sending to each other ............... 130
5.4 Probability of receiving at least one fake packet ($\beta$) in a path with 1 malicious nodes ..................................................... 135
5.5 Probability of receiving at least one fake packet ($\beta$) in a path with 2 malicious nodes ..................................................... 135
5.6 Probability of receiving at least one fake packet ($\beta$) in a path with 5 malicious nodes ..................................................... 136
5.7 Number of packets needed to achieve $\beta = (0.85,0.90,0.95)$ in a path with 1 malicious nodes ..................................................... 137
5.8 Number of packets needed to achieve $\beta = (0.85,0.90,0.95)$ in a path with 2 malicious nodes ..................................................... 137
5.9 Number of packets needed to achieve $\beta = (0.85,0.90,0.95)$ in a path with 5 malicious nodes ..................................................... 138
5.10 Nodes detection accuracy for different OppNets protocols .......... 140
5.11 Nodes detection accuracy based on the mobility of nodes .......... 141
5.16 Comparison of malicious node detection accuracy: Our approach (First Contact) vs. ACK [1] vs. Network coding techniques [2] .......... 144
5.17 False negative rate as the number of malicious nodes increases .... 144
5.18 Nodes detection accuracy percentage vs. Simulation time .......... 145
Abstract

Opportunistic networks (OppNets) refer to a number of wireless nodes opportunistically communicating with each other in a form of “Store-Carry-Forward”. This occurs when nodes come into contact with each other without proper network infrastructure. OppNets use wireless technologies, such as IEEE 802.11, WiMAX, Bluetooth and other short range radio communication and find applications in domains such as disaster management, wildlife tracking and rural communication. In OppNets, there is no end-to-end connection between the source and the destination nodes with frequent partitions and long delays being common.

Security is a major challenge in Opportunistic Networks (OppNets) because of their characteristics, such as open medium, dynamic topology, no centralized management and absent clear lines of defence. Packet dropping attacks are one of the major security threats in OppNets since neither source nodes nor destination nodes can have the knowledge of where or when the packet will be dropped. This attack is one of the most difficult denial of service (DoS) attacks to detect and can lead to degradation of network performance and obstruction of the propagation and delivery of sensitive data.

In this thesis, we have identified a novel attack in OppNets, a special type of packet dropping attack where the malicious node(s) drops one or more packets (not all the packets) and then injects new fake packets instead. We name this novel attack as the Catabolism attack and propose a novel attack detection and traceback approach against this attack referred to as the Anabolism defence. As part of the Anabolism defence approach we have proposed three techniques: time-based, Merkle tree based and Hash chain based techniques for attack detection and malicious node(s) trace-
back. We provide mathematical models that show our novel detection and traceback mechanisms to be very effective and detailed simulation results show our defence mechanisms to achieve a very high accuracy and detection rate.
Chapter 1

Introduction

Opportunistic networks (OppNets) refer to a number of wireless nodes opportunistically communicating with each other in the form of “Store-Carry-Forward” when they come into contact with each other without proper network infrastructure. OppNets aim to establish reliable connectivity where there is no end-to-end connection between a source and destination node. The nodes in OppNets usually have high mobility, low density, limited power, short radio range, and are often subject to different kinds of attacks by malicious nodes. Due to these characteristics, OppNets present significant research challenges.

With the proliferation of wireless mobile devices, OppNets are being used in a wide variety of areas including disaster recovery, military deployment, and wildlife. OppNets encompass different technologies, such as ad hoc Networks, wireless sensor networks, peer-to-peer (P2P) systems, grid networks and delay tolerant networks (DTN). However, unlike DTNs, routing in OppNets must be opportunistic. In DTNs, when a message is to be sent, an existing end to end route is first investigated; if none is found, the message is then sent opportunistically. Whereas in OppNets, the message is always sent opportunistically, and an existing end to end path is never identified. OppNets work by exploiting human relationships and interactions with various mobile devices/users to build strong and secure protocols.

Researchers have studied the characteristics of OppNets such as their contact duration times [3], mobility [4], storage-delay, and energy-delay [5] as these characteristics affect the overall network performance and routing protocols. One of the routing challenges in OppNets is the high mobility of nodes and the work in [3]
focused on and observed the inter-contact time and contact duration between two nodes transmitting opportunistically in order to determine the capacity of the opportunistic network, such as the amount of data that can be transferred between two nodes during their contact times. The authors discovered that the distribution of inter-contact time over a large time range can be approximated to a power-law. Under power-law conditions, redundant transmissions could be used to significantly improve transmission schemes that usually perform badly.

Routing in OppNets is a challenging problem, due to features such as frequent partitions, long delays and no end-to-end path from source to destination. Many protocols have been developed to accommodate these features. Generally, replication-based protocols have higher delivery rates and lower delays over forwarding-based protocols because of the multiple copies in the network. However, the cost incurred on network resources from replication-based protocols is higher than the cost incurred from forwarding-based protocols. Hybrid protocols inherit some features from both replication and forwarding families to come up with new protocols that have better performance in term of delivery rate, delays and cost. Researchers are yet to reach agreement on the best routing approach for OppNets.

1.1 Motivation

As outlined earlier, OppNets are dependent on other nodes for successful packet/data delivery. Consequently, packet dropping attack is one of the major security threats in OppNets. It can be classified as a denial of service attack (DoS) where one or more malicious nodes drop all or some of the packets. This attack is one of the most difficult DoS attacks to detect since neither the source node nor the destination node has the knowledge of where or when the packet will be dropped. Packet dropping can degrade the performance of the network and may obstruct the propagation of sensitive data. It is a significant challenge to deal with such an attack since the unreliable wireless communication and resource limitations can result in communication failure and result in the wrong prediction about the presence of a packet dropping attack. Moreover, a node’s resources, such as energy and bandwidth can be the real reasons behind packet dropping. A power shortage or communication
failure such as physical damage can make a node unavailable. It may be difficult to recognize whether packets were dropped due to a security attack or for non-security reasons. Further, Dropping packets can lead to an increase in the number of packet retransmissions, transfer time, response time and network overhead. However, there is no ambiguity about the malicious behaviour if the node drops some legitimate packets and then injects fake packets to replace them. In this case, the malicious node obviously has enough resources to do this.

Existing packet dropping defence mechanisms, such as the multipath routing-based mechanisms\[6\][7][8], reputation-based mechanism\[9\], data provenance-based mechanism \[10\], acknowledgement-based mechanisms \[11][1\], are inefficient as in OppNets we have no end to end connections and usually have no alternative paths from the sender to the destination or vice versa. Network coding-based mechanisms\[2\] are also inefficient as the destination nodes are required to have a copy of all neighbours packets/messages so it can decode its message, which is difficult to achieve in OppNets. Watchdog and pathrater mechanisms\[12][13][14\] are inefficient for detecting this type of attack as the detection idea is based on the calculation of the total number of transmitted/received packets. Encryption techniques \[15\] are inefficient as well, as we require the use of a secret key which is difficult to manage in OppNets since we have no centralized management.

1.2 Research Objective

Packet dropping attack is one of the major security threats in OppNets since neither source nodes nor destination nodes have the knowledge of where or when the packet will be dropped. In the Catabolism attack malicious nodes selectively drop some packets (not all the packets) and instead of them inject fake packets in order to maintain the original total number of packets originated from the sender node. In this case, the malicious node obviously has enough resources to carry out this malicious behaviour. Developing solid techniques for detecting the Catabolism attack and then tracing back to the malicious nodes that triggered this attack are the main research objectives in this thesis.
1.3 Research Problems

When one or more malicious nodes selectively drop some legitimate packets and inject fake packets as replacement then the existing packet dropping mechanisms will be inefficient for detecting this attack and identifying the malicious nodes. Therefore, we can summarize this problem as below:

- How to accurately detect this attack, stop fake packet propagation and traceback to the malicious nodes using the available packet information without presetting the OppNets nodes?
- How to accurately detect this attack, stop fake packet propagation and traceback to the malicious nodes even when the malicious node has the ability to change/modify the entire content of the packets?
- How can the efficiency of fake packet detection and malicious node traceback be increased to suit the energy needs of resource-constrained networks even when the malicious node has the ability to change/modify the entire contents of the packets?

1.4 Contributions

To the best of our knowledge, this is the first attempt to identify this type of attack and to propose a detection and malicious node traceback mechanism. The main contributions of this work are:

1. To identify a novel attack in OppNets (Catabolism Attack) where malicious nodes drop some packets and then inject fake packets instead.

2. To propose an efficient defence mechanism (Anabolism Defence) where the malicious nodes can check the received packets to detect the attack, and then traceback and identify the malicious nodes that triggered this attack.

We have developed different traceback mechanisms to deal with this attack, including the:
• Time-based technique: This technique relies on the packet creation time (PCT) of each packet and the contact time between nodes. There are two phases in this technique. The first phase is to find the fake packets that has different creation time, and the second phase is to find the malicious nodes by comparing that fake creation time with the nodes contact times so we can identify the malicious nodes. Destination nodes can run this technique to detect this attack and traceback the malicious nodes. To improve this mechanism, we further calibrate this technique so any legitimate node can run it to detect the attack, stop fake packet propagation and then detect the malicious nodes.

• Merkle tree technique: This technique relies on the Merkle tree hashing techniques where the sender node uses Merkle tree techniques to calculate the message/packets root value and then include it in each packet. Legitimate nodes compare the existing Merkle root values in each packet to detect the attack and then traceback the malicious nodes.

• Hash chain technique: This technique relies on the hash chain mechanism where the sender node hashes all the packets in a chaining manner and then includes these hashes in each packet. Legitimate nodes check the existing hashes values in each packets to detect the attack and then traceback the malicious nodes.

1.5 Overview of the Thesis Organization

The rest of this thesis is organized as follows. Chapter 2 provides a broad description about the early research and history of opportunistic networks and then an overview of the available OppNets routing protocols and their classification including replication-based, forwarding-based and hybrid-based protocols. An evaluation study has been done on six routing protocols (Epidemic, PRoPHET, MaxProp, Spray and Wait, Direct Delivery and First Contact) in terms of complexity, and scalability. We also discuss secure routing techniques in OppNets and different defence mechanisms against various types of attacks like Blackhole, Wormhole, Dropping and Sybil attacks. Anti-localization techniques, Selfishness, Trust and the impact of the intrusion detection systems on OppNets are
also discussed.

In Chapter 3, we present a novel attack and traceback mechanism against a special type of packet dropping attack where the malicious nodes selectively drop some packets (not all the packets) and instead of them inject fake packets so it can maintain the original total number of packets originated from the sender node. We call this novel attack a Catabolism attack and we call our novel traceback mechanism against this attack Anabolism defence. In Chapter 3, we propose a time-based technique for detecting this attack and traceback to the malicious nodes.

In Chapter 4, we present our Merkle tree based technique for attack detection and traceback the malicious nodes. This technique is a node-based technique and requires a sender node to construct a Merkle tree and then include the Merkle root value in each packet header before forwarding these packets to the destinations.

In Chapter 5, we present our hash chain-based technique. This technique is also a node-based mechanism and required the sender node to construct a hash chain based on the packets content and then include appropriate chain values into packet headers.

In Chapter 6 we conclude our thesis by summarizing the major work we have done and the major contributions and we provided possible avenues for future research.
Chapter 2

Literature Review

OppNets began with applications like ZebraNet [16], a wireless mobile sensor network used for tracking animals in wildlife areas, with sensors operating as peer-to-peer networks once attached to the animals’ necks. Each sensor has a GPS, memory, wireless transceiver, and a CPU. The sensors exchange data with a mobile base station using flooding-based techniques where nodes send data to the base station and if the data is delivered successfully it is deleted. If not, data is passed to a neighbour node with a higher probability of reaching the destination based on the node’s historical communication record. In ZebraNet, storage, bandwidth, and energy tradeoffs were taken into account to achieve a better performance. However, ZebraNet assumes the same speed and direction for node movements. Another project was CenWits [17], a search and rescue system, used in emergency situations in wild areas where hikers, skiers, or climbers carry small Radio Frequency (RF) sensors with them on their journey. Each sensor has a unique ID and a GPS receiver. When any two sensors come in to their communication range, they exchange their presence as witness information along with the witness information for all other sensors they have met. By using this information, they can find the movement path and various locations of the sensors. In addition, access points are fixed in different known locations and are connected to servers through satellites. These access points are used to collect information on sensors that are on the same communication range. CenWits can be used in more scenarios than ZebraNet [16] because their sensors are assumed to move in different directions at different speeds. Each sensor is assigned to a specific group to eliminate the problem of battery life and memory as sensors don’t need to
be awake all the time. A further advantage is by using the four-phase hand-shake protocol technique in which sensors transmit only as much as the other sensors are willing to receive.

The underwater sensor network project [18] has sensing and mobility features, where nodes communicate with each other using acoustic modems. This project was developed for different water environments like lakes, rivers, and oceans. Nodes use TDMA protocols for communication to schedule messages, and use 3D distribution localization algorithms for network self-localization without the need for an external clock source like a GPS. A robot is used to retrieve the data from the nodes by traveling around and downloading the nodes information, and then relocating the sensor nodes. Another underwater networks project in [19] fixes radio frequency antennas on whales to transmit data to SWIM stations, SWIM stations communicate with other SWIM stations or satellites. All data collected during one day can be encapsulated into one packet, and whales or SWIM stations can potentially reject packets they have already received as each packet has a different identifier. Each packet also has a time stamp with its creation time and time to live (TTL). Packets are discarded from the whales’ tags once their TTL expires. No synchronization between whales is needed as the local clock is used to maintain timing. TurtleNet [20] is another project for turtle tracking.

In [21], one of the real-world mobility traces undertaken by the study was the opportunity of data transfers between wireless devices carried by users. Haggle project [22] is an open-source architecture for OppNets. The haggle system consists of a single event queue and a set of managers that create and act on various system events such as new neighbour discovery or incoming data from the local applications or the network. A study on the storage-delay and energy-delay tradeoffs in frequently disconnected paths between the source and the destination nodes has been completed in [5]. In this study nodes store the message and carry it till they reach the destination, or forward the message to another intermediate node who then forwards it to the destination node.
2.1 Overview on Routing Protocols in OppNetS

Routing protocols in OppNets is based on the idea of “Store-Carry-Forward” as there is no end-to-end connection between sources and destinations. Routing protocols in OppNets can be classified in different ways [23]. To structure our discussion in this work we classify them into three broad families: Replication, forwarding, and hybrid as in Figure 2.1. We describe each of them below providing examples for each family.

![OppNets family tree](image)

Figure 2.1: OppNets family tree

2.1.1 Replication Family

Replication based protocols work on the principle of duplicating the message on the network. So at any given time there is more than one copy of the message on the network. This kind of protocol achieves good delivery rates but can waste network resources. The replication family can be further classified based on the use of flooding or coding techniques.

2.1.1.1 Flooding Technique

Replication based protocols can be based on flooding techniques [24][25] aiming to replicate multiple copies of a message in the network. Epidemic routing [24] is a flooding based protocol. When two nodes are in range, the node with the smaller identifier begins sending its summary vector to the node with the larger
identifier, including a table of messages on the node’s buffer. After the summary vector exchange, each node can determine if the other node has new unseen messages. The nodes can request the global unique ID of unseen messages as long as they have enough buffer space (nodes forward messages with fixed probability). Nodes receive and carry messages even if there is no path to the destination at that time. Epidemic routing uses a simple FIFO (first in first out) scheme for managing node buffers. When the buffer is full, and the node is no longer able to store new messages, the node discards the first message that has remained the longest time in the buffer. Epidemic routing is used as a benchmark to compare other protocols as Epidemic routing achieves 100% message delivery but with a maximum amount of resources used and the highest congestion.

Broadcasting messages using flooding techniques like Epidemic routing [24] is one of the reasons for the increase in network overhead, causing redundancy, contention, and collision. Authors in [26] proposed a number of techniques to reduce the broadcasting of messages by presenting five schemes, the Probabilistic Scheme, Counter-Based Scheme, Distance-Based Scheme, Location-Based Scheme, and the Cluster-Based Scheme.

Spray and Wait [25] technique has two phases; in the first phase, the source node “sprays” a predefined number of copies to the network, and then in the “wait” phase the nodes perform direct delivery to the destination.

History Based Prediction Routing (HBPR) protocol [27] utilizes the behavioural information of the nodes to find the best next node for routing. HBPR selects the next hop for the message based on three parameters; the stability of nodes’ movements, prediction of the direction of future movement using Markov predictors and the perpendicular distance of the neighbouring nodes from the line of sight of the source and destination. HBPR performs better than the Epidemic routing protocol in terms of the number of messages delivered and the network overhead ratio.

Agent-based MORP [28] protocol aims to increase the throughput rate and save node energy in DTNs and OppNets. The agent node minimizes the retransmissions in the network by monitoring the Successful Packet Transmission-Reception rate. Agent-based MORP uses the stateless approach, where the forwarding nodes are divided into probable relay regions and the routing paths are established on demand. Agent-
based MORP can decrease the load from the source node and establishes a link between member nodes in the network.

### 2.1.1.2 Coding Technique

Replication protocols can also use coding techniques [29] [30] [31] [32] to improve the throughput, efficiency and scalability of the networks. Network Coding (NC) and Erasure Coding (EC) can be used as efficient techniques to encode the original packets into a data stream of encoded packets. Original packets are reconstructed when the destination node obtains a certain number of encoded packets. NC allows the intermediate node to encode the received packets and uses a small size of encoded blocks to achieve high delivery rates, while EC allows the source node to only encode the packets and the messages together to achieve a large size of encoded blocks with high transmission rates and a low overhead ratio [29]. In [30], EC is used to design a forwarding algorithm. Constant overhead and best/worst case delay performance can be maintained even when a large number of intermediate nodes are used. Messages are encoded into small blocks on the senders side, and then send to a two hop relay till they reach the destination. A reconstruction process occurs at the destination to obtain the portion from the encoded small blocks.

Hybrid Erasure Code (H-EC) routing techniques [31] is a modification on the work of [30]. H-EC has the strength of EC based routing techniques where the encoded block is duplicated then transmitted to the next hop. The original block forwarding is similar to the techniques used in [30], while the copied block is forwarded to the remainder of the nodes encountered after the original block is sent out.


ORWAR [33] is a quota-based replication protocol and is one of the resource efficient routing protocols in DTNs where messages are classified into three priority classes; low, medium, or high. Message utilities are assigned based on these three priorities; 1 for low, 2 for medium, and 3 for high. A message is ranked based on the ratio of message utility and message size which is called the per bit utility metric. The
number of retransmissions can be reduced by calculating the maximum transmittable message size \( S_{\text{max}} \), by using the data transfer rate, and contact time window. Also, the contact time window can be calculated by using the context of mobile nodes (speed, direction of movement, and radio range). When two nodes are on the same contact range they forward half of the message copies and keep the rest. Messages are deleted from the lower buffer queue where the queue should include information such as the sizes and the utilities of messages.

### 2.1.2 Forwarding Family

In the forwarding family, messages are forwarded from node to node after selecting the best node as the next hop in the routing process. Sender nodes transmit a single copy of the messages rather than replicating it. As a result, forwarding-based protocols waste less network resources but the delivery rates are lower than replication-based protocols. The forwarding family of protocols can be further classified as basic, prediction-based, time-based, buffer management-based and social relationship-based techniques.

#### 2.1.2.1 Basic Technique

In the forwarding family, the most basic and simple routing protocols are Direct Delivery (DD) \[34\], Direct Transmission \[35\] and First Contact (FC) \[36\]. DD only uses one hop instead of a number of hops, meaning the source node directly forwards its message to the destination node. Grossglauser et al., \[34\] proposes a two hop relaying scheme where if possible, nodes directly replicate the generated message to the destination node or to a randomly selected intermediate node. In FC, the message is forwarded to any next hop node if this node has not previously carried this message and is directly deleted from the buffer of the sender node.

The Direct Transmission scheme \[35\] is one of the simplest possible routing protocols where a message is directly transmitted from the source to the destination node. If the source is not close to the destination, it holds the message until it moves close enough to deliver the message to the destination. Direct Transmission adopts the forwarding/single copy routing technique. The strength of this technique is its
minimum overhead cost on the network as there is only one transmission per message. However, this technique has the highest expected delays in delivery.

In the First Contact algorithm (FC) [36], the message is forwarded to the next hop randomly. The message stays in the buffer and gets forwarded when the node carrying it encounters the first node it meets even when it has zero knowledge about it.

### 2.1.2.2 Prediction Technique

Forwarding-based protocols might also use prediction [35] [37] [38] [39] [40] [41] [42] to get information about the best intermediate nodes that can forward the message. Seek and Focus [35] is a prediction-based protocol and has two phases. The first phase is the seek phase where the sender performs random forwarding to neighbour nodes with parameter $P$. The second phase is the focus phase where the forwarding is based on the utility of the neighbour node which is based on the recent encounter time. A node is selected to be the next hop when its utility value is more than the threshold. A timer is also used in Seek and Focus to switch from the focus to the seek phase.

The Spray and Focus protocol [37] also has two phases. Based on a summary vector and forwarding tokens, the spray phase begins by spraying a predefined number of copies into the network. In the focus phase, each relay node uses a utility based scheme to forward its copy to another more suitable relay node. Based on a set of timers, such as the timers to record the age of the last encounter that records the time since two nodes previously met, a forwarding decision is made. The node will not forward any message to another node unless the utility of that node is greater than its own utility plus the utility threshold.

Similar to the Seek and Focus and the Spray and Focus protocol, the scheme in [38], also has two phases. In the first phase, the source node sprays half of its message copies and keeps the other half for itself. When there is only one copy left on the node’s buffer, the node switches to direct transmission or utility-based routing, which is the second phase. The second phase is when the decision to forward is based on the age of last encounter timers. Node $A$ forwards its message to the intermediate node $B$ destined to node $D$, if and only if $UB(D) > (UA(D) + U_{th})$, where $UB$ and
\(UA\) are the utility of nodes A and B respectively and \(U_{th}\) (utility threshold) is a parameter of the algorithm.

In Predict and Spread (PreS) [39], a Markov chain is used for the node mobility pattern model and also for obtaining the social characteristics of nodes. PreS uses an adapted binary spraying scheme for designing a multi-copy routing protocol. PreS is based on the assumption that nodes usually move around main venues. Therefore on a campus network, nodes usually move around classrooms, cafeterias, and labs. Nodes can exchange messages only if they are on the same main venue. However, node connections between different venues is not considered.

The Context-aware Adaptive Routing (CAR) protocol [40] uses the Kalman filter base prediction technique and utility theory for selecting the next hop. In CAR, the delivery probability of a node is periodically calculated by the node itself based on its own context information such as the mobility of the node, and battery level. Nodes broadcast their delivery probabilities and routing information to all reachable nodes through DSDV (Dynamic Destination-Sequenced Distance Vector) synchronous routing. Through synchronous routing, CAR selects the next hop among the reachable neighbour nodes by forwarding the message to the node with the highest chance of delivering the message to its destination.

The work in [41] uses the location information of nodes to present a mobile trace based routing protocol. The next hop is selected based on its direction. Each node has its own trace file containing regularity in movement. By using this trace file the direction of nodes can be predicted, and then the best hop towards the destination node can be selected. Each node informs other nodes of its presence by sending out a beacon message sporadically containing a node ID, node location, and a timestamp. However, declaring the location to the malicious node can cause a serious threat.

To alleviate large scale problems, geographical routing is proposed by the EASE (Exponential Age SEarch) algorithm [42]. A node’s routing decisions can be based on the destination’s geographical coordinate, with each node maintaining a table of time and location of the last encountered node. The forwarding decision to the intermediate node will be based on the time and location of the last encounter with the destination, and any packet information such as the number of nodes the packet came through.
2.1.2.3 Time Technique

Forwarding-based protocols can also use time varying shortest path-based techniques [36] [43] [44]. In [36] four routing algorithms are proposed based on such techniques, Minimum Expected Delay (MED), Earliest Delivery (ED), Earliest Delivery with Local Queue (EDLQ), and Earliest Delivery with All Queues (EDAQ). In MED, the overhead cost of the next hop is calculated as the sum of the average waiting time, propagation delay, transmission delay, and the proactive routing approach is used for message routing. In ED, Dijkstra’s shortest path algorithm is used for path calculation at the source node without consideration to the intermediate node buffer size, even though messages might be dropped with a limited buffer. In EDLQ, the next hop delay is calculated using local queue occupancy and the route can be recalculated in each hop. In EDAQ, instantaneous queue sizes can be calculated using the queuing oracle. The source node can calculate the best route for the message, and the capacity for all nodes on that path are reserved at the time of message to ensure adequate time for message movement and accurate prediction of queuing in the network.

A prediction of the future link uptime is proposed in the Delay Tolerant Link State Routing (DTLSR) protocol [43]. Each node has a view of the network state by maintaining a graph and using Dijkstra’s algorithm to select the shortest route for the message. Each node should belong to an administration area, with the link state protocol instance only working in this area. Farther, each administrator area has endpoint identifiers used as communication gateways.

In DTN Hierarchical Routing (DHR) [44], the contact information between nodes is combined to achieve an aggregation level. Nodes above this level can maintain information about the time invariant hierarchical network, and the nodes below this level maintain information about time variance based on the shortest path construction.

2.1.2.4 Buffer Management Technique

Some forwarding-based protocols use buffer management and congestion control based techniques [45] [46] [47] [48] [49] to improve performance because nodes have finite storage and may carry messages for a long time. The message can be forwarded
to other nodes on the network, with the node receiving the message either accepting or rejecting the message if it doesn’t have enough buffer space. Nodes may place certain restrictions on the total number of messages and size of message on their buffers, and may discard some messages from the buffer after some time to free up buffer space.

In TTL Based Routing (TBR) [45], each node has a priority list or schedule for the messages that will be forwarded/dropped on its buffer based on the message’s TTL, hop count, message replication count, and message size. This mechanism efficiently utilizes node buffers where each node maintains a list of delivered messages. The destination node inserts the ID of a delivered message into the delivered list, and when nodes meet each other, acknowledgment messages are exchanged including the list of delivered messages. According to the list, delivered messages will be deleted from their buffers. Messages with an earlier deadline have priority to be forwarded or delivered. This type of protocol, or TTL based protocols aim to enhance message delivery rates.

In [46], a method for calculating the number of message copies in a certain time by maintaining a list in each node is proposed. A new cache management strategy is also proposed based on message priority. Message priority is defined based on message properties, such as the number of messages copies, elapsed time, and TTL. An Enhanced Buffer Management Policy (EBMP) was proposed in [47] to maximize the delivery of messages and reduce delivery delay. The utility value of each message can be calculated by using message properties such as the number of copies of each message in the network, message age, and the remaining TTL of each message.

In [48], the message joint scheduling and dropping mechanism is proposed to optimize average delivery rates and average delivery delays. The theory of encounter-based message dissemination is used where each node in the network maintains a list of encountered nodes and the state of each message is carried by these nodes as a function of time. Nodes send the list of the updated messages since the last exchange, and after some time all nodes will get the same history of global network information. However, this work is based on the assumption that all messages have the same size.

In [49], a study of the prioritization schemes of messages is done using real measurements. Because of the path explosion phenomenon, the authors claim that techniques
of assigning high priority to messages with a low delegation number, and a lower priority for a high delegation number, performs better in terms of balancing delivery rates, delay, and network overhead.

Resource availability [50] [51] is one of the essential services that should be provided in OppNets. In [50], a novel schema has been proposed for data caching by choosing some nodes that can be easily accessed by other nodes on the network. These Network Central Location (NCL) nodes are chosen based on their central location to store data and to make it available to the requester’s node. A probabilistic selection metric is used to select the best node to be the NCL, with a coordination strategy between them to improve the trade off between caching overhead and data accessibility.

A new resource location algorithm has been proposed in [51] where the individuality of OppNets is taken into account by employing the caching of resource meta data and proactive resource announcements to improve the discovery rate. Any kind of supply, aid, function or task can be physical resources and the nodes periodically advertise themselves and their resources. A message dissemination mechanism has been proposed for resource location based on existing OppNets routing protocols (Epidemic routing, and Spray and Wait).

2.1.2.5 Social Relationship Technique

Some forwarding based protocols use social relationship information [52] [53] [54] [55] [56] [57] for selecting the best next hop. Context Information Prediction for Routing in OppNets (CiPRO) [52] is a routing protocol that uses relationship information. When two or more nodes are on the same transmission range, the sender sends a control message $H_m$ to all the first hop neighbours containing hashed value (evidence/value) pairs which is the node’s profile that contains information about the node itself, such as name, residence address, workplace, nationality and hobbies. The first hop neighbours then compare their hashed values with the received hashed value to calculate the encounter probability with the destination. Following this, they broadcast their $H_m$ to the other first hop neighbours so they can also calculate their encounter probability with the destination and then return it to the first hop node. This node then selects the higher probability value from its neighbours and
returns it to the sender, who will use this information to send the message content. PROPICMAN [53] is a social context-based routing protocol for intermittently connected mobile ad hoc networks (MANETs). PROPICMAN has the ability to route with no knowledge about the neighbours. In PROPICMAN, nodes do not need to send their information to other neighbours for routing issues. Instead, the sender selects the best neighbour based on the highest message probability to reach the destination. The sender achieves this probability by sending the message header to two hop neighbours containing some information that the sender knows about the destination. According to this information, the neighbour will calculate their delivery probability by predicting its mobility based on the behaviour of repeating patterns at different times during the day, week, and month. This means each node can compute the delivery probability and build its own profile, which is an instantiation of a common set of evidence/values. Any node then receiving the header will compare the hashed pairs of evidence/values with its own value so it knows the highest delivery probability and share it in a hidden format. Also, the content of the message is designed to be hidden so none of the intermediate nodes can read it, except the destination.

PeopleRank [54] is similar to Google’s mechanism of PageRank. Higher weight is given to nodes if they are socially connected to other important nodes on the network. When two nodes meet each other they exchange their current PeopleRank values and the number of social graph neighbours they have, they then update their PeopleRank values. Therefore, the more the nodes meet each other, the more their rank value increases.

BUBBLE Rap [55] is a social-based forwarding algorithm. In this protocol, context information is the social communities that nodes belong to. Each node belongs to at least one community with local and global ranking across the whole system. Based on the patterns of contacts between nodes, the communities are automatically defined and labeled. When a source node wants to send a message to the destination node, it begins looking for nodes belonging to the same community as the destination node. If such nodes are not found, it will try to forward the message to sociable nodes which have more chances of meeting with the community of the destination node.

SimBet [56] uses the concept of centrality/similarity to calculate the centrality of
nodes, which is the structural importance of the node. SimBet uses three methods to achieve centrality of nodes (Freeman’s degree, closeness, and betweenness measures). SimBet utility is calculated based on these values for each node,

\[ S_{im}B_{et}U_{tit,n}(d) = \alpha S_{im}U_{tit,n}(d) + \beta B_{et}U_{tit,n}, \]

where \( S_{im}U_{tit,n}(d) \) is a similarity utility, \( B_{et}U_{tit,n} \) is a betweenness utility and \( \alpha, \beta \) are tunable parameters where \( \alpha + \beta = 1 \). According to this utility, a bridge node will be selected between a group of nodes to connect with the bridge of neighbour nodes to the destination. At the end only bridge nodes will be used for communication. However, SimBet prevents its forwarding behaviour if the utility metrics of encountered nodes are equal.

FRESH [57] is based on the idea that the node that I met 5 minutes ago is probably closer to me than the node I met 5 hours ago. In FRESH, nodes maintain a record of all nodes recently encountered. Nodes forward their messages to neighbouring or intermediate nodes if these nodes have encountered the destination more recently than the node itself and so on, until the message reaches its destination.

### 2.1.3 Hybrid Family

In hybrid family, the protocols aim to combine both the forwarding and replication based mechanisms to develop new and powerful protocols. The hybrid family of protocols includes schemes such as Utility Replication [58][59][60][61][62][63], Improved Spray and Wait [25][64][65], Improved Epidemic [66] and Coding techniques [67][68].

#### 2.1.3.1 Utility Replication Techniques

Hybrid protocols can use the utility replication based idea to select the best neighbour. [58] [59] [60] [61] [62] [63].

The main idea of the delegation forwarding algorithm [58], is the assumption that each node has an associated quality metric. If a node encounters another node with a quality metric higher than other nodes they have seen, it will transmit the message to the encountered node. However, this will add overhead on the network as the node may have to carry this message for a long time.

PRoPHET [59] is similar to Epidemic routing [24]. Nodes exchange summary vectors when they meet each other, and this contains delivery predictability values which are built in each node. If the node has previously visited a specific location several times.
before, then it will most likely revisit that location again. This delivery predictability ages with time and has a transitive property. After exchanging summary vectors, the source node transfers messages to other nodes if the delivery predictability of other nodes is higher. PRoPHET achieves good delivery rates with less network overhead. Modified PRoPHET is proposed in [60] where average delivery predictabilities are used instead of delivery predictabilities to solve the problem of jitter. In PRoPHET, the delivery predictabilities increase when two nodes regularly encounter each other, and decrease if these nodes have no chance of encountering each other due to network disruption. If these nodes encounter each other again after some time, then delivery predictabilities return to the previous value. This results in a routing jitter problem. RAPID [61], uses a utility function to assign a utility value to each packet. Packet utility value is based on the average delay metric. RAPID first replicates packets with the highest increase in utility. There are four steps in RAPID. The first is the initialization step which involves the exchange of metadata to help estimate packet utilities. In the second step, the direct delivery step, packets are intended for immediate transmission to neighbour nodes. In the third step, packets are replicated according to a marginal utility. The RAPID protocol ends in the fourth and final termination step, when contacts are broken or all packets are replicated.

In DTC [62], the sender node selects the next hop node based on the node utility value that is computed by using a number of variables, such as most frequently noticed, future plans, power and rediscovery intervals.

PREP [63], which stands for PRioritized Epidemic, is a variant of Epidemic routing. PREP aims to fix the area where Epidemic routing was weak. When the load on the network increases, Epidemic routing starts to drop messages to accommodate storage or bandwidth, thus affecting delivery rates. Therefore, PREP aims to determine which messages to drop and which to transmit, and it does this using a message prioritizing scheme and hence, the name PRioritized Epidemic. It works by prioritizing messages based on their current overhead costs to a destination, and their expiry time. It then uses this priority information to determine the messages that should be discarded or transmitted. Upon an encounter, messages closest to their destinations are assigned a higher priority over messages which are farthest from their destinations. When nodes encounter each other, messages with higher
priorities are transmitted, while messages with low priorities are dropped because they have to work harder to reach their destinations.

### 2.1.3.2 Improved Spray and Wait Technique

Hybrid protocols that utilize and improve the Spray and Wait protocol [25] for selecting the best neighbour, such as HiBOp [64] and EBR [65] have been proposed. HiBOp [64] is based on the Spray and Wait protocol where the source node sprays a number of message copies with a utility forwarding approach using context information. HiBOp calculates the delivery potential of nodes and looks for nodes that have an increasing match with the known context attributes of the destination. A high match means a high similarity between nodes and destination contexts, consequently, the message is handed to the destination’s community.

EBR [65] aims to control and limit the flooding of messages. EBR is similar to the Spray and Wait protocol, only that it uses previous contact history in the spray phase to spray messages in the network. Messages are forwarded to nodes with a high encounter rate. EBR implements security measures against a black hole denial of service attack.

### 2.1.3.3 Improved Epidemic Technique

Improved Epidemic protocols such as MaxProp [66] also lay within the hybrid family. The core idea of MaxProp is to prioritize both the schedule of packets transmitted and dropped as MaxProp assumes limited storage and bandwidth. Each node has a vector of the total number of nodes in the network so when two nodes meet they exchange their vectors and then use these vectors to estimate the shortest path to the destination node. These priorities are based on the likelihood of path to peers and are also based on a number of complementary mechanisms including historical data, hop count, acknowledgments, a head start for new packets, and any lists of previous intermediary nodes. MaxProp uses broadcasted acknowledgments to update the encountered nodes about the delivered messages so they can delete them from their buffers.
2.1.3.4 Coding Technique

Hybrid protocols can also use coding techniques to select the next best hop from neighbour nodes [67]. The main idea of network coding is to reduce network overhead. This is a process of allowing and encouraging intermediate nodes to mix data in the network so they can send out packets with linear combination of information received earlier. Tuples of encoding vectors and information vectors received will be stored and checked on the node’s buffer, and the node will then find the packets originally intended for it. Source vectors are grouped to generations in order to limit the size of the matrix stored on the nodes buffers. In addition, there will be one matrix per generation and it is required that combined packets should be from the same generation. Hash functions are used by nodes in order to determine which generation to insert in a given packet.

The RED algorithm [68] is based on erasure coding and encounter prediction techniques. RED has two parts; the data transmission part which is responsible for the decision of when and where to forward data based on delivery probabilities. The second part is message management which is responsible for the optimal erasure coding parameters based on its delivery probability to accomplish the required data delivery rates while reducing network overhead.

2.2 OppNets Routing Protocols Summary

Tables 2.1, 2.2, 2.3 and 2.4 provide a summary of the important characteristics of routing protocols in OppNets based on their families. These characteristics have been drawn from the research undertaken by respective authors. Characteristics include routing techniques, storage, bandwidth, metrics, buffer management policy, complexity, and scalability. In regards to scalability and complexity, the values are based on the work of the authors. For example, if the scalability or complexity is low, this means the authors have tested their protocol under low scalability or complexity. This excludes Epidemic routing, PRoPHET, MaxProp, Spray and Wait, Direct Delivery and First Contact protocols where the values for these are based on our evaluation results in real simulation scenarios that will be outlined in Section 2.4.
Table 2.1: Summary of Routing Protocols in OppNets- Replication Family

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Routing Technique</th>
<th>Storage</th>
<th>Bandwidth</th>
<th>Metrics</th>
<th>Buffer policy</th>
<th>Complexity</th>
<th>Scalability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epidemic routing [24]</td>
<td>Flooding</td>
<td>Limited</td>
<td>Not Mentioned</td>
<td>Delivery Rate, Delivery Delay</td>
<td>FIFO</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>Spray and Wait [25]</td>
<td>Flooding</td>
<td>Unlimited</td>
<td>Unlimited</td>
<td>Delivery Rate, Delivery Delay</td>
<td>Not mentioned</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>HBPR [27]</td>
<td>Flooding</td>
<td>Limited</td>
<td>Not Mentioned</td>
<td>Delivery Rate, Overhead ratio</td>
<td>Not mentioned</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>Agent-based MORP [28]</td>
<td>Flooding</td>
<td>Limited</td>
<td>Not Mentioned</td>
<td>Throughput Rate, Energy levels</td>
<td>Not mentioned</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>Network Coding [29]</td>
<td>Coding</td>
<td>Unlimited</td>
<td>Limited</td>
<td>Delivery Rate, Overhead</td>
<td>Not mentioned</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Erasure Coding [30]</td>
<td>Coding</td>
<td>Unlimited</td>
<td>Not Mentioned</td>
<td>Data rate, Data latency, Data Overhead</td>
<td>Not Mentioned</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>H-EC [31]</td>
<td>Coding</td>
<td>Limited</td>
<td>Not Mentioned</td>
<td>Delivery Rate, Delivery Delay</td>
<td>Not Mentioned</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>ORWAR [33]</td>
<td>Coding</td>
<td>Limited</td>
<td>Unlimited</td>
<td>Delivery Rate, Latency, Overhead</td>
<td>Priority</td>
<td>Medium</td>
<td>Low</td>
</tr>
</tbody>
</table>

Table 2.2: Summary of Routing Protocols in OppNets- Hybrid Family

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Routing Technique</th>
<th>Storage</th>
<th>Bandwidth</th>
<th>Metrics</th>
<th>Buffer policy</th>
<th>Complexity</th>
<th>Scalability</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRoPHET [59]</td>
<td>Utility Replication</td>
<td>Limited</td>
<td>Not Mentioned</td>
<td>Delivery Rate, Delivery Delay</td>
<td>FIFO</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>Modified PRoPHET [60]</td>
<td>Utility Replication</td>
<td>Limited</td>
<td>Not Mentioned</td>
<td>Delivery Rate, Delivery Delay</td>
<td>FIFO</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>RAPID [61]</td>
<td>Utility Replication</td>
<td>Limited</td>
<td>Limited</td>
<td>Delivery Delay, Utility function</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>DTC [62]</td>
<td>Utility Replication</td>
<td>Not Mentioned</td>
<td>Not Mentioned</td>
<td>Delivery Rate, Delivery Delay</td>
<td>Not Mentioned</td>
<td>Not Mentioned</td>
<td>Not Mentioned</td>
</tr>
<tr>
<td>PREP [63]</td>
<td>Utility Replication</td>
<td>Limited</td>
<td>Limited</td>
<td>Delivery Rate, Delivery Delay</td>
<td>Priority</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>HiBOp [64]</td>
<td>Improved Spray and Wait</td>
<td>Limited</td>
<td>Not Mentioned</td>
<td>Buffer occupation, Message Loss, Overhead, Delay</td>
<td>FIFO</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>EBR [65]</td>
<td>Improved Spray and Wait</td>
<td>Limited</td>
<td>Limited</td>
<td>Delivery Rate, Delivery Delay, Network Overhead</td>
<td>Buffer size</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>MaxProp [66]</td>
<td>Improved Epidemic</td>
<td>Limited</td>
<td>Limited</td>
<td>Delivery Rate, Delivery Delay, Network Overhead</td>
<td>Buffer size</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>Network Coding [67]</td>
<td>Coding</td>
<td>Not Mentioned</td>
<td>Not Mentioned</td>
<td>Delivery Ratio, Overhead, Packet Delay</td>
<td>Not Mentioned</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>RED [68]</td>
<td>Coding</td>
<td>Limited</td>
<td>Limited</td>
<td>Delivery Ratio, Overhead, Delivery Delay</td>
<td>Buffer size</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>
Table 2.3: Summary of Routing Protocols in OppNets - Forwarding Family-1

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Routing Technique</th>
<th>Storage</th>
<th>Bandwidth</th>
<th>Metrics</th>
<th>Buffer policy</th>
<th>Complexity</th>
<th>Scalability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct Delivery (DD) [34]</td>
<td>Basic</td>
<td>Not Mentioned</td>
<td>Not Mentioned</td>
<td>Throughput Packet Delivery</td>
<td>Not Mentioned</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Direct Transmission [35]</td>
<td>Basic</td>
<td>Not Mentioned</td>
<td>Not Mentioned</td>
<td>Network Overhead</td>
<td>Not Mentioned</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>First Contact (FC) [36]</td>
<td>Basic</td>
<td>Limited</td>
<td>Limited</td>
<td>Average Delay Delivery Rate</td>
<td>Storage limitation</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Seek and Focus [35]</td>
<td>Prediction</td>
<td>Not mentioned</td>
<td>Not mentioned</td>
<td>Delivery Delay Delivery Rate</td>
<td>Not Mentioned</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Spray and Focus [37]</td>
<td>Prediction</td>
<td>Limited</td>
<td>Limited</td>
<td>Delivery Delay Delivery Rate</td>
<td>TTL expiration</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Algorithm in [38]</td>
<td>Prediction</td>
<td>Unlimited</td>
<td>Unlimited</td>
<td>Delivery Delay Transmissions</td>
<td>Not Mentioned</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>PreS [39]</td>
<td>Prediction</td>
<td>Not Mentioned</td>
<td>Not Mentioned</td>
<td>Delivery Ratio Delivery Latency</td>
<td>Not Mentioned</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>CAR [40]</td>
<td>Prediction</td>
<td>Limited</td>
<td>Not Mentioned</td>
<td>Delivery Ratio Delivery Delay</td>
<td>Buffer overflow</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Algorithm in [41]</td>
<td>Prediction</td>
<td>Not Mentioned</td>
<td>Not Mentioned</td>
<td>Delivery Ratio Overhead</td>
<td>Not Mentioned</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>EASE [42]</td>
<td>Prediction</td>
<td>Unlimited</td>
<td>Unlimited</td>
<td>Network Overhead</td>
<td>Not Mentioned</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>MED [36]</td>
<td>Time</td>
<td>Limited</td>
<td>Limited</td>
<td>Average Delay Delivery Rate</td>
<td>Storage limitation</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>ED [36]</td>
<td>Time</td>
<td>Limited</td>
<td>Limited</td>
<td>Average Delay Delivery Rate</td>
<td>Storage limitation</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>EDLQ [36]</td>
<td>Time</td>
<td>Limited</td>
<td>Limited</td>
<td>Average Delay Delivery Rate</td>
<td>Storage limitation</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>EDAQ [36]</td>
<td>Time</td>
<td>Limited</td>
<td>Limited</td>
<td>Average Delay Delivery Rate</td>
<td>Storage limitation</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>DTLSR [36]</td>
<td>Time</td>
<td>Not Mentioned</td>
<td>Not Mentioned</td>
<td>Message Completion Delivery Delay</td>
<td>Not Mentioned</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>DHR [44]</td>
<td>Time</td>
<td>Not Mentioned</td>
<td>Not Mentioned</td>
<td>Delay ratio Hop count Overhead</td>
<td>Not Mentioned</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>
### Table 2.4: Summary of Routing Protocols in OppNets- Forwarding Family-2

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Buffer Management</th>
<th>Limited</th>
<th>Not Mentioned</th>
<th>Delay ratio</th>
<th>Median Latency</th>
<th>Replication and Size</th>
<th>Overhead</th>
<th>TTL, hop count, size</th>
<th>Priority</th>
<th>High</th>
<th>Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBR [45]</td>
<td>Buffer Management</td>
<td>Limited</td>
<td>Not Mentioned</td>
<td>Delay ratio</td>
<td>Median Latency</td>
<td>Overhead</td>
<td>High</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm in [46]</td>
<td>Buffer Management</td>
<td>Unlimited</td>
<td>Unlimited</td>
<td>Delivery Rate</td>
<td>Delivery Delay</td>
<td>Overhead</td>
<td>Priority</td>
<td>High</td>
<td>Low</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EBMP [47]</td>
<td>Buffer Management</td>
<td>Limited</td>
<td>Not Mentioned</td>
<td>Messages delivery</td>
<td>Delay</td>
<td>Hop count</td>
<td>Utility value</td>
<td>Low</td>
<td>Low</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm in [48]</td>
<td>Buffer Management</td>
<td>Limited</td>
<td>Limited</td>
<td>Delivery Rate</td>
<td>History</td>
<td>Medium</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm in [49]</td>
<td>Buffer Management</td>
<td>Limited</td>
<td>Unlimited</td>
<td>Success Rate</td>
<td>Delay</td>
<td>Network Overhead</td>
<td>Priority</td>
<td>High</td>
<td>Low</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TBR [45]</td>
<td>Buffer Management</td>
<td>Limited</td>
<td>Not Mentioned</td>
<td>Delay ratio</td>
<td>Median Latency</td>
<td>Overhead</td>
<td>High</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm in [46]</td>
<td>Buffer Management</td>
<td>Unlimited</td>
<td>Unlimited</td>
<td>Delivery Rate</td>
<td>Delivery Delay</td>
<td>Overhead</td>
<td>Priority</td>
<td>High</td>
<td>Low</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EBMP [47]</td>
<td>Buffer Management</td>
<td>Limited</td>
<td>Not Mentioned</td>
<td>Messages delivery</td>
<td>Delay</td>
<td>Hop count</td>
<td>Utility value</td>
<td>Low</td>
<td>Low</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm in [48]</td>
<td>Buffer Management</td>
<td>Limited</td>
<td>Limited</td>
<td>Delivery Rate</td>
<td>History</td>
<td>Medium</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm in [49]</td>
<td>Buffer Management</td>
<td>Limited</td>
<td>Unlimited</td>
<td>Success Rate</td>
<td>Network Overhead</td>
<td>Delay</td>
<td>Priority</td>
<td>High</td>
<td>Low</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CiPRO [52]</td>
<td>Social Relationship</td>
<td>Unlimited</td>
<td>Unlimited</td>
<td>Overhead and Delay</td>
<td>Not Mentioned</td>
<td>Medium</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROPICMAN [53]</td>
<td>Social Relationship</td>
<td>Unlimited</td>
<td>Unlimited</td>
<td>Overhead and Delay</td>
<td>Not Mentioned</td>
<td>Medium</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PeopleRank [54]</td>
<td>Social Relationship</td>
<td>Not Mentioned</td>
<td>Not mentioned</td>
<td>Delivery Rate</td>
<td>Network Overhead</td>
<td>Not Mentioned</td>
<td>Medium</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bubble Rap [55]</td>
<td>Social Relationship</td>
<td>Not Mentioned</td>
<td>Not Mentioned</td>
<td>Delivery Rate</td>
<td>Network Overhead</td>
<td>Delete once received</td>
<td>Medium</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SimBet [56]</td>
<td>Social Relationship</td>
<td>Unlimited</td>
<td>Unlimited</td>
<td>Delivery Rate</td>
<td>Delivery Delay</td>
<td>Network Overhead</td>
<td>Not mentioned</td>
<td>High</td>
<td>Medium</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRESH [57]</td>
<td>Social Relationship</td>
<td>Unlimited</td>
<td>Unlimited</td>
<td>Latency</td>
<td>Overhead</td>
<td>Network Overhead</td>
<td>Not mentioned</td>
<td>Low</td>
<td>High</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chapter 2 Literature Review

2.3 Evaluating Routing Protocols in OppNets

We have evaluated six OppNets routing protocols - Epidemic routing [24], PRoPHET [59], MaxProp [66], Spray and Wait [25], Direct Delivery [34] and First Contact [36] in terms of complexity, scalability, and limitation of the node’s type. In these evaluations, complexity defines how efficiently the protocol functions when the overhead on the network increases, and scalability is the ability of the protocol to scale and deal with the network as it grows and as the number of nodes increases.

2.3.1 Metrics Used for Evaluation

Metrics used to evaluate these six routing protocols in OppNets are:

1. Delivery Rate: The ratio of the number of successfully delivered messages to the total number of messages generated.

2. Delivery Delay: The average duration between the time a message is generated at the source and the time the message or its copy is received at the destination.

3. Overhead ratio: The number of copies per generated message in the network.

High performance often means a high delivery rate and low delivery delay, while maintaining a low network overhead [69].

2.3.2 Framework of Evaluation

We have employed three scenarios for the purpose of this evaluation. Below is an overview of each scenario:

- Scenario 1 aims to test the complexity of the 6 protocols. For this scenario, overhead is added to the network by increasing the message sizes and reducing the intervals of message generation. Based on the results of scenario 1, protocols were rated depending on their ability to deal with complexity. We divided the delivery rate and delivery delay scales to three ranges (low, medium and high) starting from the minimum to the maximum value. By comparing a protocol’s performance in each simulation run to these ranges we can classify
the behaviour of the protocols; then protocols can be rated with either high, medium, or low complexity.

- Scenario 2 aims to test the scalability of the 6 protocols. They will be tested by increasing the number of nodes and network area. Based on the results of scenario 2, protocols were rated depending on their ability to deal with scalability. We divided the delivery rate, delivery delay and overhead ratio scales to three ranges (low, medium and high) starting from the minimum to the maximum value. By comparing protocol’s performance in each simulation run to these ranges we can classify the behaviour of the protocols; then protocols can be rated with either high, medium, or low scalability.

- Scenario 3 aims to test if results differ by using single group types as opposed to multiple group types. The scenario will be tested on 4 different node groups, one composed of pedestrians only, cars only, trams only, and one group with all 3 types.

### 2.3.3 Simulation Settings

To perform the above scenarios we used the ONE simulator [70]. The following settings apply to all scenarios, however any specific changes in the settings will be outlined for each individual scenario in the following section.

The simulation is designed to last for 12 hours, with 0.1 seconds of update intervals. Bluetooth is chosen for connectivity with a transmit range of 10 meters, and a transmit speed of 2 Mbps. Each point on the graph represents the average of 30 simulation runs. There are 18 active nodes composed of cars and pedestrians. Pedestrians and cars have up to 10 MB of RAM for storage. Pedestrians move at random speeds between 0.5 and 1.5 m/sec, with cars only driving on roads and moving at speeds between 10 – 60 Km/hr, and wait times of 0 – 120 secs. Map based movement is used for pedestrians and cars, with a network area of $4500m^2 \times 3400m^2$. Nodes move randomly on roads and walkways with a movement warm-up of 1000 seconds. Further, there are 3 groups of trams, with 2 trams in each group. Since trams have bigger buffers in their communication devices, their buffers have up to 60 MB of RAM. Map route movement is used for trams to follow a constructed tram
line. Trams drive at speeds of $7 - 10$ m/sec with a wait time of $10 - 30$ secs at each configured stop. In addition to the Bluetooth interface, a group of trams uses the high speed interface with a transmit range of 1000 meters and a transmit speed of 10 Mbps.

Messages are generated every 15 to 25 seconds per node, with message sizes between 250 Kb and 950 Kb, and a message time to live of 3 hours.

### 2.3.4 Simulation Results

The simulation settings outlined in the previous section will be used for each scenario, however specific changes will be made to reflect what each scenario aims to achieve.

#### 2.3.4.1 Scenario 1

Scenario 1 is designed to test the impact of delivery rates and delivery delays by increasing the size of messages; with one new message generated every 15 to 25 seconds. A simulation was run 30 times for each protocol as follows: small sized messages (10KB to 20KB), medium sized messages (50KB to 700KB), and large sized messages (900KB to 3MB).

Based on the data collected from the simulation reports for scenario 1, the graphs in Figure 2.2, 2.3 and 2.4, show the delivery rates, delivery delays and overhead ratio for the 3 message sizes.

In Figure 2.2, with small message sizes, the delivery rates were higher for Epidemic routing, Spray and Wait and MaxProp, and lower for PRoPHET, Direct Delivery, and First Contact. With medium sized messages, the delivery rates dropped to almost half for all protocols, where MaxProp had the highest delivery rates. With large sized messages, delivery rates were low for the 6 protocols.

In Figure 2.3, delivery delays were lower with small sized messages, and higher for all protocols with both medium and large sized messages. The delivery delay of large sized messages was higher than small and medium sized messages because there were hardly any messages delivered for large sized messages.

Figure 2.4, shows the overhead ratio of the six protocols, Epidemic routing and PRoPHET incur a high network overhead as they are flooding the network with multiple message copies. MaxProp, Direct Delivery and Spray and Wait incur a low
network overhead as they do not flood the network with multiple message copies. First Contact incurs a medium network overhead because the message is forwarded to the next hop randomly and stays on the buffer till it gets forwarded when the node carrying it encounters the first node it meets even when it has zero knowledge about it.

From the results of scenario 1, delivery rates decreased as message sizes increased and the intervals of message generation decreased. With large sized messages, delivery rates dropped to almost 0 for all 6 protocols. This is because the buffer space of nodes is continuously consumed, and the need to drop messages to free buffer space is required. To evaluate each protocol in terms of complexity, we divided the delivery rate scale to three ranges, Low (starting from 0 to 0.33), Medium (starting from 0.34 to 0.66) and High (starting from 0.67 to 1). The Epidemic routing protocol for example, performed 0.85 (High) in the small message size, 0.91 (High) on the medium message size and 0.91 (High) in the large message size, so Epidemic routing achieved three High values in delivery rate. In terms of delivery delay, Epidemic routing achieved two Medium and one Low. Overall, Epidemic routing has the following ranges: 3-High, 2-Medium and 1-Low. So, the average behaviour of the Epidemic routing protocol will be Medium in term of complexity. Based on these calculations and the behaviour of the protocols (Epidemic [24], PRoPHET [59], MaxProp [66], Spray and Wait [25], Direct Delivery [34] and First Contact [36]) we can say, these 6

![Delivery rate](image)

**Figure 2.2: Scenario 1: Delivery rates**
2.3.4.2 Scenario 2

Scenario 2 was designed to last 6 hours with the update interval increased to 1 second. Scenario 2 was made to test the impact of delivery rates, delivery delays, and overhead ratio by varying the number of nodes with a network area of $8500m^2 \times 7400m^2$ (almost double) for the 6 protocols. The simulation was run 30 times for each protocol as follows: Small (306 nodes composed of 200 pedestrians, 100 cars, and 6 trams), medium (909 nodes composed of 600 pedestrians, 300 cars, and 9 trams), and large (1812 nodes composed of 1200 pedestrians, 600 cars, and 12 trams).

Based on the data collected from simulation reports, Figures 2.5, 2.6, and 2.7 show the delivery rates, delivery delays and overhead ratio of the 3 different number of nodes and an increase in the network size as outlined in scenario 2.

In Figure 2.5, as the number of nodes increases in the network, the delivery rates also increases for all protocols. They increase for Epidemic routing, PRoPHET, Spray and Wait, MaxProp, First Contact and Direct Delivery. With an increasing number of nodes, chances for messages to reach their destinations are higher as each sender can find an intermediate node faster to hand the message towards the destination and sometimes the sender can meet the destination node directly. In Figure 2.6,
as the number of nodes increases in the network, the delivery delay decreased for Epidemic routing, PRoPHET and MaxProp as all of them use the epidemic principle for choosing intermediate nodes. Delivery delay is increased for Spray and Wait (in medium scalability), Direct Delivery, and First Contact as the network area is increased in this scenario to almost double so the chance of meeting the destination will be less.

In Figure 2.7, as the number of nodes increases in the network, the overhead ratio increases for Epidemic routing and PRoPHET as the network is flooded with multiple message copies. Overhead ratio remains very low for MaxProp, Direct Delivery and
Spray and Wait as they do not flood the network with multiple message copies. The network overhead for First Contact slightly increased for medium and large scalability because the message is forwarded to the next hop randomly. The message stays on the buffer and gets forwarded when the node carrying it encounters the first node it meets even when it has zero knowledge about it.

Based on these results, as the network size increase in terms of area and number of nodes, Epidemic routing and PRoPHET maintain high delivery rates with a decrease in delivery delays. However, an increase in the network overhead was also observed, and so we rated them with medium scalability. Although MaxProp and Spray and
Wait did not achieve delivery rates as high as Epidemic routing or PRoPHET, it incurred a very low network overhead, so we rated them with medium scalability. First Contact and Direct Delivery achieved small delivery rates with high delivery delays, and a high network overhead for First Contact so we rated them with low scalability.

2.3.4.3 Scenario 3

This scenario aims to test if results differ by using single group types as opposed to multiple group types. As a result we can discover which type of nodes improves connectivity in OppNets. In this scenario, the simulation was run as follows: Nodes were composed of 100 pedestrians, 50 cars and 9 trams; nodes were composed of only 100 pedestrians; nodes were composed of only 50 cars; and nodes were composed of only 9 trams.

Based on the data collected from the simulation reports, Figures 2.8, 2.9, and 2.10 show the delivery rates, delivery delays, and network overhead of the 4 node groups as outlined in scenario 3.

![Figure 2.8: Scenario 3: Delivery rates](image)

From the 3 Figures, we can see that trams have the highest delivery rates with the lowest delivery delays and network overhead for all 6 protocols. This is most likely due to the longer contact times trams have, the large buffers, and the high speed interface that 3 out of 9 trams use in addition to the Bluetooth interface. There-
fore, trams can carry and exchange information faster, and improve connectivity in OppNets.

### 2.4 Security and Trust in OppNets

Security can be defined as the process, provisioning and management of ensuring the confidentiality, integrity, non-repudiation, access control, availability, and authentication of network systems. Confidentiality, is to protect any sent data by making it unreadable by other nodes except the destination node. As data passes through an open medium or different intermediate nodes until reaching destination, encryption
Chapter 2 Literature Review

can be used to protect data in OppNets. Integrity, is to protect data from modifications, dropping, or re-sending to an alternative destination through a malicious attack. Non-repudiation is the protection from malicious nodes that deny sent data using signatures for sent data. Access control, is to control the system from unauthorized access to network resources by malicious users. Availability, is to ensure network resources are available to legitimate users. Authentication, is the process of detecting the impersonation by identifying legitimate users from malicious users or nodes. However, in OppNets authentication will be difficult to implement compared to infrastructure based networks as there is no central management or authority.

OppNets can be vulnerable to different types of attack. We can classify these attacks as internal and external attacks [71][72][73], where internal attacks come from nodes inside the domain or part of the network. These kinds of attacks have a high impact on the network since malicious nodes can gain privileges to access network resources. External attacks come from nodes outside the network or they are nodes that don’t belong to the domain of the network. The impact of this type of attack is less than internal attacks because nodes have less information about network resources. Different types of attacks can affect routing in OppNets, such as Wormhole attacks, Blackhole attacks, Selfish attacks, Sybil attacks and Selective Dropping attacks. Attack descriptions and the mechanism used to defend against them is detailed below.

2.4.1 Defence against Wormhole Attacks

The idea of a Wormhole attack [74] is that malicious nodes record packets at specific location on the network and “tunnel” to other locations on the network and retransmit them from that location. Malicious nodes claim a short path in the network to certain destinations so they can attract traffic or other nodes. The packet leash technique [75] has been proposed as a defence against Wormhole Attacks. Some information can be added to packets to restrict the maximum allowed transmission distance. Geographical leashes and temporal leashes can be used, with the receiver of the packet ideally within a certain distance from the sender in geographical leashes. In temporal leashes the packet should have an upper bound on its lifetime so it can restrict the maximum distance travelled. However, this kind of technique needs se-
cure and tight time synchronization. A method for detecting and isolating Wormhole attacks was proposed in [76]. This is a modification of the AODV protocol where the source node sends a route request to a destination and receives all available routes with the number of hops. These routes are used later as a reference for each other in order to find malicious nodes. The proposed method works in three steps, by using route redundancy, routes aggregation and calculating the round-trip time (RTT) for all listed routes. In order to detect any malicious nodes and isolate them we do a comparison between RTT and the number of hops for all routes. However, this method will not be efficient in OppNets because there is no end to end connection and it is difficult to find more than one route to the destination.

2.4.2 Defence against Blackhole Attacks

In Blackhole attacks, malicious nodes silently drop or discard all or some of the received packets all of the time or some of the time. Malicious nodes can advertise themselves as having a valid route to some of the popular destinations on the network. In [77], a defence against Blackhole attacks was proposed, where an attacker can fake its contact history with some popular destinations and raise its value of delivery likelihood to the maximum value. The idea of an “encounter ticket” (ET) is proposed as evidence of the encounters of nodes. However, a malicious node can still fake its contact history with a destination by a one-time tailgate attack, where the malicious node collects redundant ETs by tailgating the destination once, then moving around the data source to intercept the data. However, even with the author’s technique of ignoring redundant ETs generated within a short interval, it may not work efficiently in case of a multi-tailgate attack, where an attacker moves in and out of the connection range with the destination. This technique can only detect an attacker when claiming non-existent encounters and cannot handle packet dropping in a Blackhole attack. In [78], malicious nodes bloat their competency of meeting a destination node so it can intercept data from other senders. A watchdog mechanism was proposed to monitor the behaviour of neighbour nodes with the absence of an end to end connection. In this mechanism, a watchdog with a positive feedback message (PFM) is used to inform the sender that the next hop will truthfully forward data to other nodes. When node A sends the message to node B, node A
will monitor the forwarding behaviour of node B in terms of evidence of the PFM created by other nodes like node C which received a message from node B. It will then generate a PFM and send it to node A telling it that B successfully forwarded the message. If node A does not get this PFM, then node B will be registered as suspicious till the PFM arrives and the trust/reputation system is built according to this mechanism. In this case, each node will have a trust value record for other nodes and they will exchange this value when they meet each other to make an indirect reputation plus a direct reputation. This trust value is derived and integrated with the probability of meeting the destination to achieve a final evaluated forwarding competency for a node. However, a PFM is sent using epidemic routing. Therefore, it adds more overhead on the network. A method of securing the packet delivery history of contact between nodes is proposed in [79]. Nodes can detect Blackhole attacks by checking these records. In case of an encounter between two nodes, both nodes record the number of exchanged packets between them and use their private key to create a secure record. Neighbour nodes can do sanity checks by checking the historic packet records of other nodes, thus detecting more Blackhole attacks. Each node has a private key (RK) and public key (PK) pairs, with each node owning the public keys of other nodes. However, the method of manually pre-loading all keys into the nodes during a network setup phase or using a key distribution scheme can be difficult to apply in OppNets.

2.4.3 Defence against Dropping and Selective Dropping Attacks

In normal network operation, packets can be dropped according to predefined rules such as resource limitation as in [9] where a packet dropping policy, dropping mechanism and performance analysis is proposed according to the packet’s weight. This weight is calculated based on inter-contact time between nodes. However, in dropping attacks or selective dropping attacks, malicious nodes drop all or some of their received packets. It is difficult to detect an attacker since both source and destination do not know when or where the dropping takes place and also since the malicious node is part of the network domain. Acknowledgement based mechanisms can be used for detecting packet dropping attacks [80], [11]. These mechanisms are based on the authenticated acknowledgment from the intermediate nodes and the destination.
within a specific time period. Source or destinations can detect a malicious node. In [2], a mitigation scheme to evaluate the impact of a selective packet dropping attack is proposed by using network coding. In this scheme, the destination node should measure the delivery ratio and return it to the sender. The sender starts to dynamically adjust the redundancy factor to mitigate against the degradation in the delivery ratio caused by the attack. Theoretical analysis and simulations show the impact of packet dropping on routing performance. The impact of non-cooperative action, like selfishness or message non-forwarding in the routing performance reduces the delivery cost, while the behaviour of dropping messages increases the delivery cost. The work in [1] is a proposed mechanism for detecting packet dropping attacks, where intermediate nodes acknowledge the reception of the packets. Source nodes use this acknowledgment to construct a Merkle tree and then compare the value of the tree root with precalculated values. If these values are equal then there is no packet dropping in that path; otherwise there is packets dropping. However, this technique can detect the path with a malicious node and then look for alternative paths for retransmission. Thus, this technique results in network overhead. This technique also cannot detect the exact malicious node in the path. In E-HSAM (Enhanced Highly Secured Approach against Attacks on MANETs) [6], a security improvement mechanism is proposed where packets that go through a path with a malicious node are redirected to alternative paths. However, this variety is not always available in OppNets as there is no end to end connection and no alternative paths available all the time.

Authors in [81] proposed a packet dropping detection mechanism based on cooperative participation at the network-bootstrapping phase. Alternative routing is used for avoiding malicious nodes or a non-trusted path. However, this solution leads to network overhead.

The impacts of TCP packet dropping attacks and detection methods is explored in [82]. Three dropping mechanisms are investigated; Periodic packet dropping (PerPD), Retransmission packet dropping (RetPD) and Random packet dropping (RanPD). Statistical based analysis is (TDSAM) used for the detection of this kind of attacks. However, only a detection technique is proposed in this work without any defence mechanism.
Authors in [10], proposed a detection mechanism for the packet dropping attack based on data provenance to identify malicious nodes. The characteristics of the watermarking-based secure provenance transmission mechanism and the inter-packet timing characteristics are exploited to achieve this goal. There are three stages for this technique: detect lost packets using the distribution of the inter-packet delays, identify the presence of the attack by comparing the empirical average packet loss rate with the natural packet loss rate of the data flow path and identify the malicious path or link and then isolate it by transmitting more provenance information along with the sensor data. However, this technique is not very accurate since it does not detect the exact malicious node in the entire path or link.

In [12], two techniques were used to improve throughput - watchdog and pathrater. In the watchdog stage, a sender node detects the misbehaving node by overhearing the neighbour node and comparing its message transmission with the saved copy on its buffer and checks whether it’s matching. If matched, then the node is not malicious and the message copy on the buffer is deleted. If nothing is heard for a certain time, the watchdog will increment the failure tally of that neighbour node. If that tally exceeds the threshold value, this node will be recorded as a misbehaving node. Each node running the pathrater phase determines the best path with the highest metric by combining the information from watchdog with the link reliability data, then calculates the best path. According to the information from watchdog and pathrater each node will build a rating table for other known nodes on the network so it can be used in future transmissions. However, the watchdog technique is not that efficient if there are ambiguous collisions, receiver collisions, limited transmission power, false misbehaviour or collusion.

TBDSR [14] is an enhancement of the Dynamic Source Routing protocol where each node maintains a trust table of opinion toward other nodes with categories of either belief, disbelief or uncertainty. At the beginning stage of the network each node has the default opinion (uncertain) toward other nodes but after some successful or failed transmissions between nodes this opinion changes gradually to either belief or disbelief. Nodes can also exchange trust information about each other then use a trust combination algorithm to calculate new opinions by combining all the recommendations.
The work in [83] has used the same idea of watchdog mechanism. A node can act as a monitor node if it can overhear the radio signal of two nodes on the same routing path, and oversees the data transmission that comes from these two nodes. When these two nodes send packets to each other, the monitoring path node backs up the data packets, and then oversees the transmission behaviour of the receiver node. If the receiver node truly sends out the packet to other nodes, then the monitor node removes the packet from its buffer. When the monitor node finds out the packets has been stored in its buffer over a threshold time, it attempts try to find alternative path to the destination and retransmit the data packet.

To solve the weakness of watchdog, ExWatchdog was proposed [13], to enhance the intrusion detecting system for discovering malicious nodes. ExWatchdog has the ability to detect malicious nodes that partitions the network by falsely reporting other node as malicious. Each node builds a table with the number of received packets and the number of forwarded packets. When a node receives a report about a misbehaving node, the source of communication starts to send a message to the destination to check if the number of received and forwarded packets are equal. If they are equal, the node that reported the other node as malicious is actually malicious and if not equal, the report is correct.

In [84], a trust establishment scheme is proposed to expand the consistency for packet forwarding in presence of malicious nodes. Nodes are used first hand and recommendation information is used to collect an opinion about other nodes. The Bayesian approach is used to build a trust scheme. And the trust value of this approach is calculated based on the assumption of probability distribution.

The authors in [85] have proposed a reputation-based mechanism for detecting packet dropping attacks. This mechanism uses direct observation and indirect or second hand information to calculate a full reputation weight. Nodes can be excluded from the network if they have a low reputation weight. To prevent fault tolerance, the authors use a reputation system based on history information of nodes. To improve the performance, fuzzy logic is used to achieve a more precise detection accuracy.

In [15], a mechanism for detecting and tracing packet dropping in malicious nodes with selfish or black hole behaviour was proposed. Nodes are required to keep a record of the packets sent and received during data exchange when they come into
contact with each other and verify the confidentiality by using the initiators public key. An encryption algorithm is used to detect the malicious nodes during the data exchange.

2.4.4 Defence against Selfish Attack

Selfish nodes may use network services, but refuse to cooperate with other nodes. For example, selfish nodes may not forward or route messages due to the limitation of battery life or resource consumption. Defence against this type of attack can be classified as barter-based and credit-based. In [86], a barter-based mechanism to stimulate selfish nodes to cooperate is proposed. This system consists of two parts; a reputation system and a virtual payment or rewarding scheme part. When two nodes are on the same transmission range they start sending a description of their messages on the buffer. They can then agree on which messages will be exchanged, with each message sent one by one from each side in preference order (primary/secondary message). If one side cheats, the transaction is directly disrupted and the worst scenario will be the deferment of one message. After each message interaction the nodes receive a score and they accumulate these scores to obtain their total score at the end (this part represents the rewarding schema). Game theory is used to achieve this barter based mechanism where the nodes are divided between two players; “Crowd” player, represents the majority of nodes and “Deviator” player, represents a small group that deviates from the behaviour of normal nodes. However, this techniques relies on the assumption that selected subsets of messages must be the same size and connection time should be enough to exchange all agreed messages which is not practical. Also, there is no clear picture of the network behaviour in case a node has no messages or fewer messages than the second side. In MobiCent [87], a Credit-Based Incentive System is proposed. Each node, client / receiver pays for message delivery using a payment scheme involving two algorithms. The first is a payment set selection algorithm which decides the relays to be paid. The second algorithm is a payment calculation algorithm which decides how much should be paid to each selected relay and how much the client should be charged. As a result, nodes will forward packets without adding phantom links or waste any contact opportunity unless the reward is not sufficient or it is the decision of an underlying
routing protocol. This technique, however, is not that strong because the sender can flood the network as he is not involved in the payment schema. This kind of strategy is not effective if the majority of nodes have selfish behaviour.

### 2.4.5 Defence against Sybil Attacks

A Sybil attack is the ability of a malicious node to create a number of fake IDs while dropping received packets. In a Sybil attack, it is difficult to identify the real node causing the packet dropping since malicious nodes use different IDs to communicate with neighbours. In [88], a definition and taxonomy to the Sybil attack is proposed, showing the types of defence such as the resource testing as an old technique, and a new technique including radio resource testing, verification of key sets for random key predistribution, registration, position verification and code attestation.

In [89], a defence against Sybil attacks in opportunistic networks is proposed. Key management is presented to enable the bootstrapping of local, topology-dependent security associations between nodes besides discovery of the neighbourhood topology by using pseudonym certificates and encapsulated signatures. There are two phases for key management. The first when nodes generate a public/private key pair and then sends the public key to the identity manager server so it can be used to generate unique pseudonyms and a certificate for each node. The second phase is where opportunistic communication and the security associations bootstrapping occur. However, using an identity manager server is not practical in OppNets due to mobility of the nodes and opportunistic connections.

Another defence against a Sybil attack is proposed in [90] by the design of a reputation-based system - explicit and implicit social trust establishment. This trust relies on two factors. The first is contact quality between nodes and the trustworthiness of the nodes’ opinions. To establish social trust, nodes combine explicit and implicit social trust where explicit social trust is built from “Friend Ties” whenever they meet via secure pairing. A friend list is built in each encounter and then saved in a friendship graph. Implicit social trust is built from contact time and relies on the familiarity and the similarity of the nodes. Familiarity means the accumulated contact time and similarity means the degree of familiarity for the two nodes match.
2.4.6 Anti-Localization Techniques

Detection of node’s location can be a serious threat by tracking path and movement of nodes in OppNets. One of the techniques used to hide the location of nodes is ALAR (Anti Localization Anonymous Routing) [91]. ALAR uses good techniques to protect a sender’s location privacy by dividing the message to a number of encrypted segments and sending each one of these to a different neighbour. The decryption key is kept on the last segment so the receiver doesn’t know the message content unless he receives all the segments. ALAR achieves the minimization of a sender’s location privacy and maximizes message delivery. The weakness of this technique is that routing performance is influenced by the setting of the number of segments and the number of neighbours or receivers. Additionally, the routing performance is degraded in terms of delivery ratio and delivery latency as the two parameters increase. In [92], any node should be a member of at least one group and each group has a set of nodes. Public/private key-pair is assigned to each group, and nodes are distributed randomly to groups. Each node maintains a keychain of a public/private key pair, plus a copy of all other nodes and groups of public keys. It also maintains the private key of its group, where each node uses group public keys to encrypt messages for other groups, and uses the groups private keys to decrypt messages for groups of which it is a part of. TPS (Threshold Pivot Scheme) use a secret sharing technique to divide the message, which is considered a secret, into multiple shares, then sends the secret to the destination through a number of independent paths. In this scenario, the author protects the content of the message from individual intermediary nodes. The sender encrypts the message with a receiver’s public key and seals both the message and receiver’s address before sending them. While the message travels through the network, each node checks whether to decrypt it. If yes, then this node is the message destination and the message can be reconstructed. However, with a Sybil attack, multiple pseudonymous nodes can copy, create and then intercept a sufficient number of shares. In [93] the authors have reviewed the application of $k$-anonymity for Location based Services (LBS) and its recent advancements. They have recognized three perspectives for the applicability of $k$-anonymity for LBS: the application of $k$-anonymity based on the architecture, based on the algorithms for anonymization, and based on the types of $k$-anonymity (according to the different
query processing techniques). Hence, the review has been done within the framework of these perspectives. This review can arm the privacy providers with the latest techniques and possible modifications in their present techniques.

2.5 Security Solutions Based on Intrusion Detection System

An Intrusion Detection System (IDS) can be a good technique for increasing security in OppNets. In [94], a ferry-based intrusion detection and mitigation (FBIDM) scheme was proposed. In FBIDM, special nodes called “Ferries” are used to collect information from other nodes by passing them in fixed routes and stopping at a fixed stop points. “Secret” encrypted messages are broadcast by ferry nodes so any genuine node can understand or decode it and then start sharing information like encounter and delivery predictability with other nodes. Ferry nodes then compare this information to decipher any malicious nodes and inform the genuine nodes to update their blacklist. Mutual correlation detection scheme (MUTON) [95] is a detection scheme based on the same idea as FBIDM [94] but with the modification of taking transitive properties into consideration. In MUTON, special nodes called “Ferries” are used to collect information from other nodes by passing them in fixed routes and stop in fixed stop points. “Secret” encrypted message are broadcast by ferry nodes so any genuine node can understand or decode the message and start sharing information like Delivery Encounter Table (DET), Delivery Probability Table (DPT), and Transitive Information Table (TIT) with other nodes. DET and DPT are inherited from FBIDM, and a MUTON created TIT table is used to record transitive information. In FBIDM, a cross-checking of delivery probabilities between two nodes is used to detect malicious node. However, for the duration of the detection process MUTON uses associations inherited from consecutive encounter events between nodes and calculates the sanity of the node based only on the information on that node. The intrusion detection system in [96] uses the same techniques as used in wired networks where nodes observe traffic sent by one hop neighbours and compares the observed values of some metrics, such as the unconditional packet dropping ratio and selective random packet dropping ratio with original values observed from the
past to detect anomalous behaviours. This intrusion detection system means nodes will be in promiscuous mode and will process all monitored packets, thus it is energy consuming. Additionally, there can be an insufficient number of neighbours that can be used as monitoring nodes in sparsely connected networks.

2.6 Trust Management Evaluation

In order to evaluate the trustworthiness of other nodes in OppNets, a trust management model [97] [98] [99] [100] [101] [102] [103] [104] [105] can be an efficient solution to increase the security levels of the network. In [97], a trust model was built to evaluate the forwarding behaviour of neighbour nodes and the model was applied to opportunistic routing in ad hoc networks. From this work, [97] proposed the minimum cost routing algorithm (MCOR). The MCOR framework has three layers; trust management, trusted opportunistic forwarding model, and trusted minimum cost opportunistic routing. The upper layer is used for the initialization of trust relationships, trust recommendations, trust computations, trust judgments, and trust updating. The middle layer is used for selecting the effective forwarder with the least cost distance to the destination from the trusted neighbour forwarding list. The bottom layer contains the trusted forwarding list and trusted minimum cost opportunistic routing. Probe packets are used to evaluate links delivery probabilities between nodes and passive acknowledgement modes used by nodes so they can observe the forwarding behaviour of their neighbours. Each node observes its neighbour’s behaviour by comparing their trust degree value with a threshold value. Then the node initializes the cost of trusted opportunistic routes and updates the trusted forwarding list in all nodes. Nodes with trust degree values less than the threshold are considered as malicious nodes.

In [98], a framework is presented to quantitatively measure trust, model trust propagation, and defend trust evaluation systems against malicious attacks. The idea is each node maintains a trust record, and when the source node needs to find a route to the destination it first tries to find as many routes to the destination as it can. The source node then tries to find the packet-forwarding trustworthiness of nodes on these routes from its own trust record or through recommendations. At the end, the
source node will select the trustworthy route to the destination and then updates the trust records based on its monitoring of the route quality. This trust record is also used for detecting malicious nodes. However, these methods are not efficient in OppNets as there is no end to end connection and its difficult to find more than one route to the destination.

In [99], a quantitative measure and a model trust propagation information theoretic framework is presented by two trust models, entropy-based model and probability-based model. We can establish trust relationships in two ways. One through direct observations and the second through recommendations from other nodes. Uncertainty is used to represent trust, and entropy is used to measure uncertainty. In the entropy-based trust model, node B monitors the behaviour of node C then makes a recommendation to node A. In the probability-based model, the probability values of trust relationships are used to calculate concatenation and multipath trust propagation. When a sender wants to establish a route to the destination, it should find multiple routes to the destination. The sender should then try to find the trustworthiness of the routes from its own trust record or through the recommendations. The sender chooses the trustworthy route to transmit the message and updates the trust records based on the monitoring of route quality. The trust records can also be used for detecting malicious nodes.

In [100], a trust management model is built in two parts; a “subjective trust evaluation model” part where nodes are evaluated using an analytic hierarchy process theory and fuzzy logic rules, and a “trusted routing model” part where new fuzzy trusted dynamic source routing (FTDSR) protocol is proposed to avoid malicious nodes. Three types of trust values are used by FTDSR; the historical trust value, current trust value, and the path trust value. In FTDSR, the source node checks the local routing cache before sending it to the destination. If path trust value is less than the requirement for data packet transmission, the sender initiates a route discovery process to the destination. Discovered route entries will be created and inserted into the routing cache of the sender, with the sender selecting the smallest hop count path in qualified routes. In case of an equal hop count, then maximum path trusts will be selected. The forwarding node can detect malicious nodes in the route discovery process according to its local trust record list. Each node maintains
a black list for malicious nodes so it will not forward or send messages to malicious nodes. However malicious nodes can receive the broadcast from other nodes and can track nodes locations. Malicious nodes also can move to another subnetwork as a newcomer with a new trust value.

Building a trust management system based on ontology or structural frameworks to organize information is proposed in [101]. A reputation system is used to classify users reliability by collecting direct and indirect information about other users. The reputation system uses a range of values or classes to evaluate other nodes. These values or classes are very untrustworthy, untrustworthy, no opinion, trustworthy, and very trustworthy.

In [102], trust management in OppNets is proposed to deal with the mechanism of trust evaluation, collection, and propagation. The semantic Web solution called COTTON is proposed to provide an architectural basis for representing trust and trust management for handling the ability of discovery tasks. COmposite Trust and Trust management in Opportunistic Networks (COTTON) uses a semantic service discovery process by applying this process on seed nodes in two steps. The first step is the Helper Registry step that stores a list of all identified services. The second step is the Helper Advertisement step which defines the abilities and services provided by a foreign node and a helper or seed node. The seed node and distributed control center nodes which are a subset of the seed nodes take care of overall operations in OppNets. The two nodes integrate with each other by downloading a copy of the current ontology and the helper registration from control center nodes to the new joined helper. The matchmaking agent is also deployed on the helper. In this case, nodes check and monitor each other and send feedback to control center nodes for integration and classification to build a reputation database for nodes. This reputation is sent back to all nodes in an epidemic way. According to that, classification nodes will or won’t be invited to join the network as a helper. However, using an epidemic way to send feedback will increase the network overhead.

Secure MANET Routing with Trust Intrigue (SMRTI) is a reputation-based trust model proposed in [103]. Evidence of trustworthiness is captured in three ways; direct interactions with neighbour nodes, observing interactions of neighbours and through recommendations from other nodes. Direct Reputation is based on the
captured and quantified evidence from one-to-one interactions of nodes. In Observed Reputation, evidence is captured from the interactions of the nodes from neighbours. Recommended Reputation is derived from the route of packets. A reputation rating will be created for each node by quantifying the captured evidences. According to this rating, nodes will be classified as malicious or not. However, in an opportunistic network we cannot totally rely on this technique because we might not be able to see other nodes due to the mobility and the absence of the end to end connection.

In [104], a modification to the existing routing metric ETX has been proposed called E2TX where trustworthiness and ETX are integrated. By using E2TX, a new opportunistic routing design called TOR has been proposed, with the initial value of trust set to 0.5 for newly joined nodes. Probe packets are then sent between nodes so they calculate the link quality metric and then a combined metric E2TX will be derived by each node. Source nodes will collect all of these E2TX of its neighbour nodes and calculate the prioritization for selecting the relay node after which the trust value is updated. The trust value for a neighbour node should be more than the threshold value, otherwise it will be recorded as a malicious node. However, this technique may not be efficient in the case of malicious nodes selectively dropping packets.

User’s risk assessment in opportunistic encounters when they share their information is discussed in [105]. By using the definition of privacy rules, the trust relation can be built by users so they can refuse or accept other user’s request. Ontology called PrOHHand (Privacy Ontology For Handover) is proposed to define a trust management system by defining a common vocabulary for a community of researchers and for software agents that need information about privacy policies in a wireless and ubiquitous environment. In SOHand (Service Oriented Handover), service providers and users are responsible for managing privacy policies. Three types of privacy agreements are defined in SOHand; User-Provider Agreements, Provider-Provider Agreements and User-User Agreements. A direct and indirect reputation system is used by users to evaluate the behaviour of others. However, to design a secure system we should not rely on the ability of users as some users do not realise the value of risk and some malicious nodes have the ability to easily cheat others.
2.7 Trust Strategies in OppNets

Trust strategies in OppNets can be built based on Social, Reputation and history information as shown in Table 2.5. A combination of all the previous strategies can be used as well. Trust is especially difficult when mobile users find themselves in a new surrounding without established trust or reputation available. The first time a node meets with another node, it doesn’t trust it. But the second time it meets the same node, how does it know if it should trust it or not? On what basis is a new node accepted into the network? Trust can be established in 2 ways: Direct trust - sometimes called first hand trust - established through direct observations of other node’s behaviour from past records, and indirect trust - sometimes called second hand trust - established through trust propagation from recommendations from other nodes. In a network, when node A wants to send a message to node B, node A has 3 choices: Node A fully trusts node B and is sure that node B will perform its job as required, node A doesn’t fully trust node B and is not sure if node B will perform the job, node A does not trust node B. How does node A make the choice?. Trust is important to authenticate a node’s identity and avoid malicious nodes. Methodologies used to measure trust include but are not limited to, rating experiences, reputation models, recommendations, past experiences, word of mouth, weighting, probability, encounter based trust, voting methods, cluster based methods, honey bee mating, beacon nodes, public key cryptography, digital signatures, Bayesian networks, chain optimization, social networks, iterative algorithms, acknowledgments, watchdog monitoring nodes, neural networks, game theory, fuzzy logic, confident and core, swarm intelligence, and directed and undirected graph.

2.7.1 Social-Based Trust Strategies

Social network information is used to build trust systems. In [106], a study and analysis of the trade-off between trust and success delivery rates in OppNets was proposed by adopting a real-trace driven approach. Number of trust social based filters including common interests, common friends and the distance in the social graph were applied on an epidemic protocol to achieve a reasonable trade-off between trust and the success rate by achieving more than a 35% success rate compared to
Table 2.5: A comparison of trust strategies

<table>
<thead>
<tr>
<th>Strategies</th>
<th>Techniques</th>
<th>Methodology</th>
<th>Advantage</th>
<th>Disadvantage</th>
<th>Suitability with OppNets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Social-Based</td>
<td>Filters [106]</td>
<td>Common interests</td>
<td>Fair trade-off between trust and success rate</td>
<td>Filters can be break</td>
<td>Suitable</td>
</tr>
<tr>
<td></td>
<td>Opinion [107]</td>
<td>Trust and recommendation</td>
<td>Good protection and scalability</td>
<td>Needs recommendations</td>
<td>Not suitable</td>
</tr>
<tr>
<td></td>
<td>Chain optimization [108]</td>
<td>Stochastic Petri net technique</td>
<td>Identify the optimal length of trust chain</td>
<td>Rely on the path length</td>
<td>Not suitable</td>
</tr>
<tr>
<td></td>
<td>Friendship, Familiarity and similarity [90]</td>
<td>Explicit and implicit trust</td>
<td>Doesn’t need end to end connection</td>
<td>Needs larger encounter duration</td>
<td>Suitable</td>
</tr>
<tr>
<td>Position [109]</td>
<td>Nodes position</td>
<td>Good Blackhole attacks defence</td>
<td>Weak techniques with mobility</td>
<td>Suitable with limitation</td>
<td></td>
</tr>
<tr>
<td>Markov model [110]</td>
<td>Level factor and con/Adherence</td>
<td>Good accuracy rate</td>
<td>Not working with uncommon information</td>
<td>Suitable with limitation</td>
<td></td>
</tr>
<tr>
<td>Reputation-Based</td>
<td>Bayesian [111]</td>
<td>Reputation and trust rating</td>
<td>Addressing lying nodes</td>
<td>Needs recommendations</td>
<td>Suitable</td>
</tr>
<tr>
<td>Watchdog [112]</td>
<td>Direct and indirect observation</td>
<td>Periodically evaluate other nodes</td>
<td>Nodes should be on the same transmission range</td>
<td>Note suitable</td>
<td></td>
</tr>
<tr>
<td>Acknowledgments [113]</td>
<td>Reputation building</td>
<td>Reputation expire with the age</td>
<td>Sender needs destination feedback</td>
<td>Suitable with limitation</td>
<td></td>
</tr>
<tr>
<td>Ontology [101]</td>
<td>Similarities and differences</td>
<td>Consider direct and indirect reputation</td>
<td>Not consider trust changing</td>
<td>Suitable with limitation</td>
<td></td>
</tr>
<tr>
<td>Certifiicates (CA) [114]</td>
<td>trustworthiness</td>
<td>Dealing with offline and online stages</td>
<td>Malicious nodes can fake CA</td>
<td>Suitable with limitation</td>
<td></td>
</tr>
<tr>
<td>Fuzzy Recommendation [115]</td>
<td>Packet forwarding patterns</td>
<td>Direct and indirect trust calculation</td>
<td>Nodes should be on the same transmission range</td>
<td>Not suitable</td>
<td></td>
</tr>
<tr>
<td>Activity [116]</td>
<td>Cooperation, honesty, and similarities</td>
<td>Direct and indirect trust calculation</td>
<td>Inactive nodes have less opportunities</td>
<td>Suitable</td>
<td></td>
</tr>
<tr>
<td>History-Based</td>
<td>Filters [117]</td>
<td>Encounter Frequency, Duration, Location and Behaviour</td>
<td>Good accuracy rate</td>
<td>Nodes misbehaviour is not considered</td>
<td>Suitable</td>
</tr>
<tr>
<td>Trusted authority [118]</td>
<td>Nodes interactions history</td>
<td>Good accuracy rate</td>
<td>History records can be fake</td>
<td>Not suitable</td>
<td></td>
</tr>
</tbody>
</table>

an untrusted environment where 10% of the nodes refused to cooperate because of the absence of trust. However, it is difficult to rely on these filters for trust criteria since malicious nodes can break these filters.

IRONMAN [119] is an incentive mechanism to detect and punish selfish nodes in OppNets using social network information. IRONMAN works when the sender node sends a message to the destination node through an intermediate node. The sender node has the expectation the intermediate node will pass this message to the destination. To verify that, when a sender meets its destination he will check if he received the message from the intermediate node. If not, he knows the intermediate node was selfish by withholding the message, and then he will decrement that nodes rating. Similarly, if this intermediate node passes the message to the destination, he will have a rating increment. When two nodes meet each other, they prove their encounter ticket, then store their local rating of encounters so they can use it to calculate their node trust score by adding the local rating to the foreign rating. When receiving messages, nodes check the trust score of the sender whether it is more or less than the threshold. If less, the receiver discards the message if the selfish node
is the source of that message and informs the sender it has been detected as a selfish
node. The receiver also lets the selfish node know it and will not accept the message
till it reaches the threshold score, and gives her a chance to do this by passing her
message forward to the destination so it can improve her rating score. This method
does not punish nodes that are given messages to forward. It only punishes nodes
that do not pass on a message to a destination.

In [107], the authors use social networks to propose a trust model. When a peer
in the network wants to work with an unknown peer, it first asks its friends for an
opinion. When a malicious peer is found in the network, friends inform each other
to eliminate the malicious peer from the network. Probability is used to define trust
and recommendation values. The trust value is how much 2 peers trust each other,
and the recommendation value is how much a peer friend recommends a peer to
other peers. The more positive recommendations a peer gets, the higher its trust
value increments. The network starts with legitimate peers and new peers can only
join the network when there is an existing peer in the network that can recommend
it. This condition helps prevent malicious peers from joining the network from the
beginning.

The authors in [120] also use social networks to build a trust based model. The model
is designed to help a peer find a trusted peer to share information with. Before a
peer can trust another peer for the first time, it starts with the propagation stage
where it collects proofs of trust from other nodes who trust that peer, and based
on the collected proofs, it calculates the trust value of that peer. In this case, the
peer uses indirect trust to calculate the trust value of another peer. Next time it
meets the same peer, it can calculate the trust value using proofs from direct trust.
The model uses a trust-aware propagation algorithm (which implements the Markov
process) which includes direct trust, indirect trust, and path trust. Based on those 3
factors, a peer’s trust value dynamically changes. Because of this dynamic feature,
the model cannot keep a fixed record of mutual trust or prevent future betrayal.

In [108], a model with trust chain optimization based on the stochastic Petri net
technique (bipartite graph) and social networks is used to measure the trust and
social values of multiple nodes in a path. When nodes evaluate each other’s trust
they combine social trust with QoS trust to compute the total trust value of a node.
Social trust is measured from direct and indirect trust derived socially from own experiences with other nodes, or the reputation of the node in the social network. QoS trust is measured from a node’s ability to provide good services and conduct positive interactions with other nodes in the network. When trust is measured in a path, its distance and number of nodes affects the computed trust, where longer chains of nodes in a path weakens the trust value of a path.

In [121], and prior in [122], an encounter based trust management model for DTNs to deal with both malicious and selfish misbehaving nodes is proposed. They use both social trust and traditional Quality of Service (QoS). They use the properties of healthiness and unselfishness to measure the social trust of a node, and the properties of connectivity and energy to measure the trust value of an encountered node. These 4 factors are composed to make a trust metric to assess the trust of each node in a DTN. The trust metric is affected by direct trust information such as encounter events, and indirect trust information such as recommendations.

In [90], secure routing in OppNets relies on social trust between users by designing a systems using explicit and implicit trust. Explicit trust is built from the level of friendship among friends meeting each other. Implicit trust relies on the familiarity (encounter duration) and the similarity (extent to which 2 familiar nodes stayed in the same location) of nodes, the larger the encounter duration, the more trust that peer gains, implicit trust helps in determining if a node is legitimate or not.

Another social-based trust system is presented in [109] where nodes in the network are grouped based on their social position in the network. Each group is assigned a value that indicates the importance of the group in the social network. The destination receives the message forwarded to it from the source node through intermediate nodes, it uses the number of hops to measure the trust value of each intermediate node. The trust value of nodes that participated in forwarding the message is updated.

The authors in [123] present a peer to peer trust model, PStrust that uses probability and statistics. The system aims to solve the limitations of the classical trust models (Eigentrust and Peertrust) in which the global trust value is not precisely calculated and does not reflect the real trust value of a peer. Peers return feedback once a transaction is completed. If the peer is legitimate, it gives honest feedback. If the
peer is not legitimate, it gives false feedback. PStrust addresses the issue (which is not addressed in Eigentrust and Peertrust) of a malicious node that once gains a high trust by providing a good service in the network, starts to give false feedback about other peers. The system relies on honest feedback and hypothesis testing is used to eliminate false feedback. In real large complicated networks it might be difficult to put PStrust in practice for a couple of reasons, one of which is the complexity of storing the global value of trust.

In [124], the authors use probability to estimate the trust value of nodes sending honest information to other nodes. The authors develop a model that utilizes legitimate nodes in the network. Legitimate nodes trust values are exchanged in the network, until the trust value of legitimate nodes is spread in the network. Initially, each node in the network is assigned a fixed trust value. Nodes then start to exchange information. When a node receives the correct information from another node, it updates the trust value of the other node. To confirm a trust value of a node, a node may request from a node information that it already has and can compare with, making this request a couple of times it can confirm that the node is indeed honest. Each node in the network keeps a small list of well-behaved nodes. Nodes with trust value meeting the required threshold go into the good list. Nodes can then share their small list of well-behaved nodes until each node has a large list of the well behaved nodes in the network.

The work in [110], improves the Markov model by proposing a new algorithm and using probability that computes indirect trust using the level factor and confidence to measure the trust value. Where the level factor states that the more intermediate nodes in a path, the longer trust has to transfer, and so the weaker it becomes. And confidence is what a node believes about the truthfulness of information it receives from other nodes. The algorithm relies on the transmission history of peer nodes, but is not designed to work with cases where information between nodes is not common or not gathered.

In [125], to help a node protect itself from pollution attacks, each node in the P2P system computes a trust value for other nodes in the network based on direct and indirect trust. The trust values of peers in the network is stored in each peer locally. When calculating the trust value for a targeted peer, the peer collects recommenda-
tions from a group of other peers that had a direct interaction with the targeted peer and the enquiring peer. The peer then sums the values of the collected recommendations with its own direct trust with the targeted peer, it then uses the resulting weighted sum as the trust value for the targeted node, and based on the value, it decides whether it will trust it or not. The trust value is updated in relation to how recent a transaction is, new transactions gain more weight than older ones. This model deals with lots of parameters that change differently according to individual peers and their needs. In a real P2P network, a more standardized model might fit better.

To motivate peers to cooperate in P2P networks, a trust-based incentive system is proposed in [126] that aims in dividing a large network into smaller networks where nodes are allocated to different categories depending on the content a node is interested in. Each category has one super node and many regular nodes. In a category, the trust value of the super node is the sum of the ratings of each regular node in the category. In a category, the trust value of regular nodes in a category is computed from direct and indirect trust (recommendations from other peers in the same category). Super nodes communicate, keep, and manage records of the trust values of regular nodes in a category. If the trust value of a super node in a category decrease below the required threshold, the super node will be removed from the group, and another reliable super node takes the lead. To maintain a true trust value, recommendations from other nodes are revised before updating a node’s trust value. Inactive nodes are removed from the category they belong to.

The authors in [127] use swarm intelligence to help find trusted routes in ad hoc networks. The concept of swarm intelligence can be applied in ad hoc networks by collecting behaviours from nodes interacting with each other and with the network. In the proposed system, every node has an embedded trust agent that assigns a trust value for all the nodes it interacted with in the network. Nodes exchange this value between each other to help each other find a trusted path/route. A node that experiences an interaction with a malicious node assigns a low trust value to the malicious node. Nodes avoid interacting with the malicious node when receiving a low trust value for the malicious node. The level of trust in a route increases or decreases depending on the number and authenticity of packets that were transferred
The model in [128] aims on detecting a malicious node using cluster-based analysis. Using auto regression, nodes forecast the trust value of other nodes in the network where past experiences affect the current trust value of a node. Direct and indirect trust are then united by the cluster head to measure the trust value using a probabilistic model. Trust values are checked for their accuracy using the Proportional Integral Derivative (PID) controller. Nodes resulting with a low trust value will not be trusted, thus a secure route can be formed by avoiding interactions between legitimate and malicious nodes.

Using distributed trust based on public key cryptography, the model in [129] uses a probabilistic method to deal with problems involved with initial trust establishment in networks. At the initial stage of the network and with the help of a secret dealer (could be a service provider), nodes are supplied with adequate amount of trust enough to get them started in the network, nodes trust their secret dealer. After this stage, the network becomes ad hoc and any central tasks such as the secret dealer ends here. Nodes then create direct trust when meeting each other by viewing each other’s certificates and validating them by looking for a trusted route between each other. At this stage, each node is self-organized, and becomes responsible to issue public certificates for other nodes. Having the secret dealer simplifies the process of establishing trust in this model, however, in decentralized networks, implementing the centralized secret dealer at the start of the network might not be feasible.

The authors in [130] construct a model that simplifies, modifies and extends the TCP/IP protocol to better fit the features of ad hoc networks. Because UDP is connectionless and requires the least protocol mechanism, data transmission is carried through the UDP protocol. The trust model uses the protocol data unit (PDU) and service data unit (SDU) to transmit the data to another node. The trust system incorporates 4 modules: The metric, data, storage, and behaviour modules. The metric module is used to measure the trustworthiness of other nodes. The data module gathers feedback from nodes regarding their trust level towards other nodes. The storage module is used to record nodes data which may be shared with the metric module. The behaviour module uses the metric module to help nodes decide whether it is safe to interact with other nodes or not. These 4 modules are used to
enhance the functionality of the whole system.

The model in [131] uses context to enhance a Bayesian network trust model in ad hoc networks. To improve trust in a network, the authors specify that context is composed of a group of facilities that are publicly available to all nodes in the network. For a node to choose the most suitable utility, it uses past records with a given utility provider, and then makes its choice. Trust is defined in their model as the probability of a utility provider to satisfactorily provide a utility to the node that requested its utility. This trust is measured using direct and indirect experiences between a node and the chosen utility provider. A Bayesian network is created for each utility’s measured trust from direct and indirect trust. The authors use context information to make conditions that may affect the outcome of an interaction experience between 2 nodes, these conditions aim for improving the trust evaluation process. To recognise an experience, the model uses context information which includes: The node, the utility, the service provided by the utility, and the date of the interaction. The Bayesian network sorts these experiences based on their creation date.

A dynamic trust model in [132] uses trust to defend ad hoc networks from packet dropping attacks. At the initial stage of the network, a node trusts its surrounding nodes and updates the trust value according to their behaviour. Behaviour that decreases the trust value of a node includes: Dropping a packet, not forwarding a packet to the destination, not starting a route discovery phase. The trust value of a node increases when it forwards the packets in a route targeting the destination.

A clustering based model based on a honey bee mating algorithm is used in [133]. The proposed model is designed to address the energy consumed by nodes to increase the lifetime of a node. Nodes are grouped in clusters, and each cluster is assigned a cluster head. The cluster head is chosen based on the pre-requisites of having the highest energy, and has no records of malicious behaviour. The cluster head must provide services such as managing the cluster that was assigned to it, and forwarding data to other clusters and base stations. These extra services that the cluster head must provide over a normal node, increases the energy consumed by cluster heads over normal nodes. Other factors that increases the energy consumption of a cluster head is the distance of a cluster to a base station, and the size of the cluster. The
further the distance of the cluster from a base station, and the larger the cluster, the more energy a cluster head consumes. The proposed model puts these factors into consideration, and tries to balance the load of the cluster head in the network. To increase the lifetime of the network, when the energy level of a cluster head falls below the required threshold, a pre-elected cluster head takes the position of the previous cluster head. The trust value of nodes is calculated based on direct and indirect trust of nodes inside each cluster. The trust value of nodes increases with successful transactions, and decreases with unsuccessful ones.

Another cluster based approach in [134] is used in conjunction with fuzzy rules to make appropriate routing decisions in a WSN. To maintain trust in the network, nodes are grouped to form clusters and a key management design is used. The network starts with grouping nodes to form clusters. Each cluster in the network has an active cluster head that is in charge of forwarding data to other cluster heads or to a base station, the cluster head does these routing decisions based on applied fuzzy rules. Similar to the P2P technique in [126], if the cluster head’s services starts to drop below the required threshold, then it will be replaced with another potential cluster head. The potential cluster head should be active in the network and is chosen based on its ability to provide services in the network such as its rate in delivering, transmitting, and receiving packets. Each cluster head in each cluster is assigned a master key and is in charge of maintaining this key. The master key is public to nodes in the same cluster, the key is used to communicate with base stations and other cluster. The trust level of nodes in the network is monitored based on their ability to deliver data to base stations.

The authors in [135] present a novel trust management framework that builds trust among nodes using 3 levels of trust - subjective or direct trust, objective or indirect trust, and recommended trust for unfamiliar nodes. Each node in the network maintains a local trust list that records the 3 levels of trust for nodes in the network. Each node can establish direct trust towards another node using its direct past experiences with a node, if it never had a past experience with a node it can seek feedback from trusted neighbours towards their trust to a node, and past performance ability of a node to perform reliably in the network. Indirect trust is built from the reputation of a node as viewed by other nodes in the network that had previous interactions with
the node, and a node’s reputation in regards to its ability in preventing malicious behaviour. Nodes build recommended trust for unfamiliar nodes in the network by using both direct and indirect trust.

In [136], the authors present an algorithm that evaluates the trust value of a node based on the security, mobility, and reliability attributes of a node in a WSN network. The algorithm performs 2 steps, in step 1, it starts by calculating the initial trust value of a node which is also called the indirect trust. When node A wants to interact with node B, A calculates the trust value of B using its own past experiences with node B and using the feedback from other nodes in the network that had interactions with node B. If A finds that the calculated trust of node B is above threshold, it interacts with node B. If not, then step 2 of the algorithm is conducted, which involves calculating the direct trust. To calculate the direct trust value of a node, node A first evaluates node B’s first model, the security model. If the trust value of the security model is within the threshold range, it interacts with node B, if not, node A evaluates node B’s second model, the mobility model. If the trust value of the mobility model is within the threshold range, it interacts with node B, if not, node A evaluates node B’s third model, the reliability model. If the trust value of the reliability model is within the threshold range, it interacts with node B, if not, node A calculates the overall trust by adding the indirect trust it calculated in step 1, and the direct trust it calculated in step 2. If the result is above the required threshold, node A interacts with node B, if not, it disagrees to interact with node B. The algorithm has not been evaluated, as it has not been tested in a simulator.

### 2.7.2 Reputation-Based Trust Strategies

The reputation of the nodes can be used for building a system of trust. In [111], a robust reputation system for detecting the misbehaviour of nodes has been proposed using a modified Bayesian estimation approach. Each node on the network maintains a reputation rating, which represents the opinion of each node toward other nodes and a trust rating that represents the opinion of nodes about the honesty of other nodes. First-hand observations and second-hand reputation records from other nodes are used if other nodes have reliably been trustworthy or when nodes pass the deviation test. Nodes use their own rating to sporadically classify other nodes according to
two criteria, normal/misbehaving and trustworthy/untrustworthy. The Bayesian approach is used to accomplish both classifications, address lying nodes and detect false reports.

In CONFIDANT (Cooperation Of Nodes, Fairness in Dynamic Ad Hoc Networks) [112], a reputation based system is proposed. A watchdog mechanism is used to collect direct information where nodes direct, observe or monitor their neighbour and detects the misbehaviour of nodes, such as packet dropping, modification, fabrication, or timing misbehaviour. Nodes do this by comparing the message transmission of their neighbour with the message copied on its buffer. Nodes also gather second-hand information from other neighbour nodes and deal with false ratings. Nodes classify other nodes as misbehaving or normal using a Bayesian estimation probability. So, misbehaving nodes can be isolated from the network. Pathrater is used to select the best path according to direct and second hand information. CONFIDANT periodically reduces the ratings of nodes according to the observation. By doing this, nodes cannot exploit previous behaviour, and it is useful to allow redemption of isolated nodes no longer misbehaving.

CORE (Collaborative Reputation) [137] is a mechanism for stimulating node cooperation and can be resistant to a denial of service attack. CORE has a watchdog component for monitoring and a reputation mechanism that differentiates between subjective reputation (observations), indirect reputation (positive reports by others), and functional reputation (task-specific behaviour). These reputations are then weighted for a combined reputation value. According to this value, the neighbour node evaluates whether the node is cooperating, and if not, the node is gradually isolated from the network. This mechanism can be integrated with any network function such as route discovery, packet forwarding, location and packet management. Other nodes will not provide this service to a node with a reputation less than the threshold value.

NRMDM (Novel Reputation based Mechanism to Detect the Misbehaving) [138] is a reputation based mechanism for monitoring and detecting selfish nodes. It consists of four components, Monitoring, Reputation System, Path Manager and Witness Module. When a node send a message to a neighbour node it keeps a copy of this message in its cache, and monitors the behaviour of the neighbour node.
The node checks whether the neighbour forwards the message and compares the forwarded message with the cache copy. If the messages match, it means the packet was successfully sent and directly removed from the cache. If the message does not match, then it means the neighbour node modified the packet. The reputation system then starts calculating the new rating and exchanges the new rate with other neighbour nodes. According to this node’s reputation values, the path manager will choose the best path to send the message. The Witness module monitors accidental dropped packets and reports to the Monitoring Module before it is considered as malicious node.

In [113], a reputation based routing protocol technique is proposed for blackholes attacks where nodes maintain a local reputation for each node. In the next forwarding, the node selects the best neighbour according to the reputations of the nodes. The reputation mechanism depends on three elements; Acknowledgments, Node Lists and Aging. This protocol deals especially with sinkhole attacks, where a malicious node can send wrong routing information to attract messages and then drop all or some of them. When a node has a low reputation value the possibility it will be a malicious node is very high. Therefore, the likelihood it will be the chosen as the next hop will be very low. When the message reaches its destination, an acknowledgment message is sent to the sender and then the sender increases the reputation value of the forwarding nodes and so on. Reputation has age, so it periodically ages or its age decreases as some nodes may be selfish for a long time for reasons such as battery life or resources consumption.

SReD (Secure Reputation-Based Dynamic) [139] was designed mainly to mitigate three kinds of attacks; Black hole, Denial of Service (DoS) and Wormhole attacks. SReD is a linkstate-based and multi-path routing protocol using a dynamic window technique to switch between reputation and probabilistic routing generation modes. These models are: each node conserves two parameters in a dynamic window size ($w$) with a median value at the initial stage, and the trust of neighbouring nodes ($ta$). By comparing these two parameters, we can guess the security quality. If $ta > w$, the local network is secure and the sender will use the reputation-based routing mode. Otherwise, the local environment is insecure so the sender will use the probabilistic routing generation mode. Note the value of $w$ is always updated using the increase
and decrease functions. SReD depends on the trust evaluation of the neighbours of other nodes, which is based on trusted hardware in each node. The highest trusted node is chosen as the next hop.

MobiGame [140] is one of the first incentive protocols that uses a reputation based scheme in DTN to defend against selfish attacks. Game theory is used in MobiGame to design cost and reward parameters. Each node can maintain its own relay evidence in the local buffer so it will show its reputation on demand. There are four phases in MobiGame to generate system parameters. These are, System initialization phase, Bundle setup phase, Bundle forwarding phase and Reward clearance phase. The last two phases responsible for the calculations on the source node side to identify whether the message reaches its destination. However, MobiGame needs an Offline System Manager (OSM) for key distribution at system initialization and each node must register with the OSM so it can obtain a public key certificate and parameters before joining any networks.

In [141], the authors list the most popular ways to calculate trust and reputation:

- The weighted average method: Used by the most classical trust models Eigentrust and PeerTrust. The weight of the chosen factors such as direct trust and recommended trust are calculated using multiplication and summation.

- Probability methods: Trust and reputation are calculated using probability and statistics, such as the Bayesian method.

- The fuzzy reasoning method: Trust and reputation are defined at different levels starting from an unknown node to a very trustworthy node.

Eigentrust [142], one of the most classical trust models is a reputation system that uses transitivity of trust to assign a unique global trust value for each node based on its interactions with other nodes in the network. Before a node attempts to download content from another node, it checks the global trust value of the node. To calculate the global trust value for node A, ratings from all nodes that worked with node A are required. This process is demanding and time consuming, as updating the global trust value for a node requires lots of input from a large number of other nodes. Peertrust [143] is another classical model that uses reputation to build a trust framework. Trust is computed from a number of parameters: recommendations from
other nodes based on the services they provide, total number of interactions a node completes, credibility of recommendations from other nodes, context of the interactions, and the context of the community such as creating incentives to encourage recommendations. PeerTrust works with many parameters and metrics making it difficult to apply with a large number of nodes in the network.

The work in [144] uses game strategies to design a trust framework called GaTMo. The framework uses individual trust for each node, and the system’s trust (reputation) for each node in the network. The use of game theory is a strategy used to make a decision in a given situation. It involves the use of mathematical functions such as probability, statistics and algebra. There are a number of strategies to choose from when choosing candidates in a network: Tit for tat, self-trust, and dynamic trust. In tit for tat strategy, when peer A wants to deal with peer B, it uses the same previous action of peer B with peer A. If peer B was cooperative in their last meeting, then peer A implements the same action and becomes cooperative. If peer B was not cooperative, then peer A does not cooperate with peer B. In self-trust, peer A decides to cooperate with peer B based on its own personal experience. In dynamic trust, peer A uses its personal trust and the system’s overall trust for peer B, and then makes its choice. A monitoring node is also used in the GaTMo framework, the node monitors other peers in the network and records their behaviour. Peers with unacceptable behaviours are eliminated from the network.

A reputation aggregation method is used in [145] to solve issues with malicious peers who send malicious content at times, and honest content at other times. The method works by assigning a provisional trust value to peers in the network. When a peer receives malicious content from another peer it lowers the direct trust value of the peer that sent it malicious content. The proposed system then allows the receiving peer to spread the new updated trust value of the malicious peer to other peers which lowers the reputation of this malicious peer. This process happens every time a peer receives malicious content from another peer and this attempts to keep the reputation level of peers updated at all times. Before nodes decide to trust a peer, they rely on the reputation or global value of the peer.

The model in [146] minimizes the influence of malfunctioned or malicious nodes in a WSN using reputation. Reputation is built among nodes from the confidence level
that nodes have towards each other. A high confidence level, means high trust. Data flows between nodes that have a high confidence interim, as the level of confidence starts to fall, the system directs the data flow to other nodes that have a high confidence interim, this technique helps in reducing energy consumed by nodes in the network. Nodes collect direct and indirect experiences with other nodes to build a trust value and a confidence interim for them. An experience record of a node could be sensing data, forwarding or receiving data, or creating an experience record for another node. To collect direct experiences, nodes evaluate their neighbours by monitoring the communications performed by them and record these communications as experiences in an experience record. Nodes use these experience records to assign a trust value for other nodes. Nodes then weigh the experiences it collected for a node to calculate its confidence interim. To collect indirect experiences, a node first evaluates the accuracy of its next hop neighbour node in recording experiences. It starts by monitoring the experiences associated with its neighbour node, it then compares those experiences with its own experiences, if the differences are small, then it means its neighbour node is accurately recording its experiences, and can be trusted and weighs the trust value to calculate its neighbour’s confidence interim. The model addresses the trustworthiness of nodes in a WSN, but not the trustworthiness of the data that flows in them. The model doesn’t consider the broadcasting of trust values, which is useful in strengthening the reputation of a system.

A communal reputation and an individual trust based model in a WSN is presented in [147]. The model builds reputation from trust formed by feedback from nodes about each other. To build trust in the network the model uses voting and implements the watchdog mechanism [12] where each node monitors its neighbour. Each node issues a trust vote for other nodes, and records their trust vote in a trust table. The node issues a trust vote by first monitoring the node it transmitted the message to, it watches to see if it keeps the integrity of the message while forwarding it. It then compares the message upon successful transmission and checks if the message forwarded is an exact copy of the original. It records a positive vote for the node that forwarded the message without any changes to the message, otherwise, it records a negative vote for the node in its trust table. Each node in the network does this process, by monitoring nodes they forward messages to, and then voting positively
or negatively, and then updating and recording the trust value based on their vote in their trust table. Positive votes (result from successful message delivery) increase the trust value of a node, and negative votes decrease the trust value of a node. If a node’s trust value falls below the required threshold, it will be notified and reported as malicious to other nodes to bring awareness regarding this malicious node. Once this awareness reaches the cluster head through multiple nodes, it isolates the malicious node from the cluster by informing the nodes in the cluster to abandon any messages from the reported malicious node. In addition to the trust table that each node maintains in the network, each node also maintains a reputation table that includes the evaluated reputation values for all other nodes in the network. Each node builds the reputation table from its own trust table and other node’s trust tables which are broadcasted occasionally in a cluster. Nodes broadcast their reputation tables as well, when nodes receive other node’s reputation tables, they use the evaluated trust values of nodes to update their own reputation table by averaging the total values of each node’s reputation value.

A malicious node detection model using graph based iterative trust and reputation methods is introduced in [148]. Nodes in the network use past experiences to evaluate the trust value of other nodes. Nodes are either service providers where they provide a service, or are service consumers where they use the services provided by service providers. After an interaction, service consumers rate the service provider of whom they used a service from. Using the rate values and after examining them for their level of honesty, a reputation system is built for nodes in the network. These ratings are recorded in a table and are used by the iterative detection mechanism to keep the reputation of all nodes in the system updated regularly, where nodes with a low reputation are removed from the network. The gathered tables of each node in the network are then used to create a bipartite graph.

A reputation system is proposed in [113] where every node locally assigns a reputation value to nodes it interacts with and uses this reputation value in the future by choosing nodes with a high reputation to forward its message to. The integrity of messages are protected using digital signatures. Each node maintains the reputation of other nodes using acknowledgments from the destination, a nodes list, and aging. When a source sends a message to a certain destination through intermediate nodes,
the destination sends an acknowledgment back to the source once it receives the mes-
message. When the acknowledgment arrives at the source, it updates the reputation of
the intermediate nodes. Every message contains the list of nodes that contributed in
passing the message to the destination, and upon receiving the message, it updates
the reputation value of the nodes that forwarded the message. Also, the reputation
of a node is affected by aging, in which if it stops interacting with nodes, its repu-
tation value decreases. Because the reputation system is built locally among nodes,
this technique reduces the overhead cost of having to maintain a reputation system
globally.
A trust model based on ontologies is proposed in [101]. Using reputation, the ontol-
ogy classifies nodes in the network according to their trustworthiness in the network.
A node uses its past experiences with a target node to evaluate its direct reputation
value, and uses recommendations from other nodes towards a target node to evaluate
its indirect reputation value. Direct and indirect reputation are both combined and
then used as a decision parameter for nodes to look at before trusting other nodes.
The model does not consider trust changes according to a node’s location, and this
is a relevant feature in OppNets.
A trust-based framework in [78] evaluates a node’s capability to distribute data based
on a node’s reputation using watchdog monitoring nodes and the Positive Feedback
Message (PFM). The authors design a Watchdog mechanism that fits with OppNets
where there is no end to end path. When a source node wants to forward a message,
it searches for nodes with a high reputation then forwards the message and monitors
the behaviour of the node that got a forward of the message. When the node delivers
the message to the destination, the destination forms a positive feedback message
(PFM) which requests to increase the reputation of all the nodes that delivered the
message. When the source node receives the PFM from the destination, it updates
the reputation of the nodes listed in the PFM. The watchdog keeps track of the
number of PFM’s nodes as they encounter and forward messages with each other.
Nodes that don’t have a PFM returned to the sender for their attempt to forward a
message don’t prove their good forwarding behaviour. The collected PFMs are used
to create a reputation system of nodes in the network to help improve the future
data forwarding where nodes send their messages to nodes with a higher reputation.
Chapter 2 Literature Review

The source node receives the PFM from the destination through epidemic routing, and the advantage of doing this is to speed up the process of updating the reputation of nodes, but it also increases the overhead cost on the network.

In [149] the authors present a trust model that uses basic trust followed with application trust. Basic trust forms as initial trust is established, the model allows 2 nodes to exchange their credentials when they meet for the first time. The credentials of any node in the network contains symmetrically encrypted features about the node. These credentials are decrypted using a secret key that is exchanged between 2 nodes when they meet. Nodes can choose which features in their credentials to share with other nodes. Once basic trust is formed, application trust formation is followed. Application trust is measured based on the node’s context, roles are assigned to nodes based on their application trust which changes when a node’s context changes.

Another model in [114] uses certificates to evaluate the trustworthiness of a trust value, and the node’s ID. The model has two stages, offline and online. In the offline stage, the certificate, called Attribute Certificate (AC), is issued by a node to its neighbour, the certificate contains the neighbour’s evaluated trust, and the issued AC is also stored in the issuing node as well. Every node in the network issues an AC for their neighbours. In the online stage, when a node wants to send a message to a specific destination, it constructs paths to the destination. It then requests ACs from nodes that belong to the paths it constructed, it validates the ACs. Using the validated ACs it measures the trust value of each route, paths with fewer hop counts have higher trust, and then chooses the most trustworthy route to send its message.

In [150], a trust based Quality of Service (QoS) model is introduced. The model measures trust from direct trust and indirect trust. QoS ensures quality services in a route such as bandwidth, delay, and jitter. Usually the route is checked for its QoS before data is transmitted through it. The proposed model uses delay only to estimate a route’s QoS. The expected transmission count is used as a metric to measure the quality of a route. To calculate direct trust, each node in the network assigns a trust value to its neighbour according to the neighbour’s ability to authentically forward packets. To calculate indirect trust, feedback from neighbours about a node is used. Each node stores a trust table that contains the trust value of every neighbour’s direct and indirect trust. When a source wants to send a message to
the destination, using the stored routing table, it searches for the possible routes that the message can be transmitted through. When a route is identified for data transmission, its trust value is first calculated using direct and indirect trust of each node in the route. The route is also measured for its delay QoS level. If the accumulated trust value is high, and the delay QoS level is low, the route is chosen for data transmission. When a node is identified as malicious in a route, it is isolated from the network, thus making the route more secure.

To safeguard the QoS of data availability, a probability-based model in [151] adopts the Dempster-Shafer theory (reasoning with uncertainty) that competently collects recommendations from intermediate nodes and effectively discard malicious ones. Trust values are assigned and stored in the intermediate nodes of a path, the recommendations are prioritized based on the trust values of nodes in a path. Recommendations from nodes with a higher trust value are prioritized over recommendations from nodes with lower trust values. Recommendations from shorter distance nodes are given more priority over longer distance nodes. The model also enhances the trust values of nodes by measuring their ability to develop their trust, a node is given the choice to whether or not it wishes to trust another node regardless of its recommendation value.

To deal with uncertainty, a fuzzy recommendation based trust model is introduced in [115]. Each node in the network monitors its neighbour’s packet forwarding patterns. Nodes record the results of their neighbour monitoring patterns into a table that contains the data forwarding information. Every time a node interacts with another node in the network it rates the interaction as either a positive or a negative one. Using the information recorded in the table, fuzzy direct trust is computed. Latest interactions are more valid than past interactions, but both are used to measure the trust value of a node. To build a trusted path, direct trust and feedback from other nodes towards nodes they interacted with are both used to calculate the fuzzy indirect trust with fuzzy properties.

Another ontology based trust model is presented in [102]. A Semantic Web Framework is provided for nodes to help them make decisions regarding trusting other nodes in the network. The semantic service includes a list of all the possible services that may be needed by nodes. The model classifies nodes into 4 different groups
as either private unknown helpers, public unknown helpers, trusted known helpers, or OppNet reservists. The trust value is assigned to the whole group based on the direct and indirect trust value of the group.

In [152] a Trust Based Spreading (TBS) is proposed to allow nodes to collaborate with each other to filter spam messages by exchanging assessments to allow or block the spreading of the message between nodes in a network when they meet each other opportunistically. The system starts when a node receives a message, it classifies the content of the message as legitimate or spam. The node then places legitimate content in a whitelist, and spam content in a blacklist. When nodes meet each other, they exchange their white and black lists whether they exchange content or not. A threshold of required assessments must be met to confirm an assessment.

The authors in [116] took the activity of nodes in the network into consideration when designing an OppNet trust system. An active node in the network has more opportunities to meet with other nodes, hence has a higher chance of meeting the destination. Direct trust is calculated using the activity parameter, an active node is determined by the number of encounter rates with other nodes. To avoid having highly active malicious nodes with a high trust value, indirect trust is measured using the cooperation, honesty, and similarities parameters of nodes towards other nodes in the network.

A public key distribution without a centralised public key infrastructure (PKI) model based on trust is presented in [153]. To establish initial trust in a network, a Leverage of Common Friends system is also introduced. The model is decentralised and uses the Web of Trust principle. Nodes authenticate other nodes based on their confidence level that the node indeed owns its public key. The confidence level increases when nodes notice lots of occurrences of a node’s public key in the network as it would be difficult for a malicious node to tie itself with the identity of a node that is already known in the network. The more nodes a node meets with and the more friends a node has in the network, the higher its trust value becomes and the more confidence other nodes will have towards it.
2.7.3 History-Based Trust Strategies

These techniques use a nodes history to build a trust system as in [117], where a trust advisory framework is proposed to decrease the unreachability in the selfish network using the four trust advisory filters (Encounter Frequency, Encounter Duration, Encounter Location-Based Behaviour Vectors and Behaviour Matrix Filters). These filters are evaluated using a real-world data set where Epidemic protocol [24] is used. Using these filters, one can provide evidence of potential similarity between nodes and this can be used when we try to find the best path to the destination. Encounter frequency (when devices are on the same radio range) is used to develop trust between nodes and is based on the idea that nodes with similar interest frequently meet and interact. This means the more they meet the more trustworthy they are. The more time spent by the nodes, the greater the similarity between them and more trustworthy they are likely to be. The behaviour vector filter (BV) is based on the idea that similar people have a tendency to go to similar locations so capturing the location preference of nodes is used as a filter base. The BV preserves the vectors for the duration and frequency of capturing user behaviour. Some modification of the vector into a matrix allows maintaining a single entity to achieve a spatio-temporal representation of user behaviour. In this behaviour matrix, each column represents a location and each row represents a single day. A Random Trust filter (RT) is also used for comparison purposes. This randomly selects a T percent of encountered users and adds them to the trust list. However, in [117] misbehaviour of the nodes is not considered. The scheme introduced in [118] uses trusted authorities to measure the trust value of nodes, the scheme is modelled using game theory. Initially, nodes record their history interactions with other nodes, and later send their history records to the trusted authority (TA) which validates the trustworthiness of nodes in the network by observing their history records. The TA rewards nodes with good behaviour to encourage positive behaviour, and punishes misbehaving nodes to minimize negative behaviour. A probabilistic misbehaviour detection scheme is used where TAs could validate nodes or not, and where nodes could misbehave or not. A reputation system is then created for nodes in the network where positive behaviour increases a node’s reputation in the network, and a negative behaviour decreases a node’s reputation in the network.
2.8 Summary

In this chapter we provided a review of literature on routing, security and trust in OppNets. We have presented the available protocols on OppNets and their classifications. We demonstrated and evaluated OppNets routing protocols in terms of complexity, scalability, and the limitation of node types using the ONE simulator. Three different scenarios were designed to evaluate the routing performance of OppNets when network load increases, the number of nodes increases, and the type of nodes are limited. Based on our results, the network load, network area, the number of nodes, and the varying types of nodes does impact the performance levels of routing protocols in OppNets.

As seen in scenarios 1, as the load on the network increased, the performance of the 6 protocols decreased. Based on these results, the 6 protocols were given a medium rating in terms of dealing with complexity. From scenario 2, as the network enlarges in area and number of nodes, Epidemic and PRoPHET performed better than Spray and Wait in delivering messages to the destination, but at a very high cost. Epidemic and PRoPHET were rated with medium scalability. Although Spray and Wait did not achieve delivery rates as high as Epidemic or Prophet, the network cost was low. We rated it with medium scalability as well. First Contact and Direct Delivery achieved small delivery rates with high delivery delays, and a high cost for First Contact. We rated them with low scalability. From scenario 3, we concluded that trams carry and exchange information faster, and improve the connectivity in OppNets due to its features such as a high speed interface and mobility speed.

In this chapter we also investigate the available security approaches in OppNets and any techniques used to increase their security levels. In this chapter, we discussed secure routing and trust management systems and strategies to increase security levels in OppNets where social, reputation and history relationships play important roles in the implementation of these trust strategies. We also discussed secure routing techniques in OppNets and different defence mechanisms against various types of attacks like Blackhole, Wormhole, Dropping, and Sybil attacks. Anti-localization techniques of nodes have attracted the attention of researchers as many routing protocols use the location of nodes as a base for their routing decisions. Selfishness
was discussed in this paper where the node tries to obtain benefits from network facilities and resources but refuses to cooperate with other nodes for reasons such as limitation of resources. We have also discussed the impact of intrusion detection systems on OppNets and provided an overview of trust in OppNets.
Chapter 3

Catabolism Attack and Anabolism Defence

In this chapter, we present a novel attack and traceback mechanism against a special type of packet dropping attack where the malicious nodes selectively drop some packets (not all packets) and instead of them inject fake packets in order to maintain the original total number of packets originated from the sender node. We call this novel attack a Catabolism attack and we call our novel traceback mechanism against this attack, Anabolism defence. We have called our attack Catabolism attack as we have a “tear down” of the transmitted packets (dropping) and we have called our defence mechanism Anabolism defence as we have a “build up” process for detecting the attack and tracing back to the malicious nodes. Our detection and traceback mechanism at this stage relies on the packet creation time (PCT) of each packet and the contact time between nodes. There are two phases in our technique. The first phase is to detect the fake packets, and the second phase is to traceback to the malicious nodes. We have tested our detection and traceback mechanism using both a destination based aproach as well as a node by node based approach. In the first approach, destination node checks the received packets to detect the attack and then traceback to the malicious nodes. In the node by node approach, each legitimate node checks the received packets to detect the attack, prevent fake packets propagation and then traceback to the malicious nodes.
Chapter 3 Catabolism Attack and Anabolism Defence

3.1 Introduction

Packet dropping attack is one of the most difficult to detect DoS attacks in OppNets since neither source nodes nor destination nodes have the knowledge of where or when packets will be dropped. Selective packet dropping can significantly erode the quality of service for applications and impact network performance. Applications that require reliable data transfer would force increased retransmissions resulting in decreased throughput. Packet dropping attacks can also impact on efficient routing with malicious nodes forwarding incomplete or inaccurate routing information. Packet dropping can degrade the performance of the network and may obstruct the propagation of sensitive data. It is a significant challenge to deal with such an attack since the unreliable wireless communication and resource limitations can result in communication failure and result in the wrong prediction about the presence of a packet dropping attack. Moreover, a node’s resources, such as energy and bandwidth can be the real reasons behind packet dropping. A power shortage or communication failure such as physical damage can make a node unavailable. It may be difficult to recognize whether packets were dropped due to a security attack or for non security reasons. Dropping packets can lead to an increase in the number of packet retransmissions, transfer time, response time and network overhead. However, there is no doubt about the malicious behaviour if the node drops some legitimate packets and then injects fake packets to replace them. In this case the malicious node obviously has enough resources to do this. We call this novel attack a Catabolism attack where the malicious node can selectively drop some packets and instead of them inject new fake packets so it can maintain the original total number of packets originated from the sender node. We call our novel traceback mechanism against this attack an Anabolism defence. The existing packet dropping defence mechanism, such as the multipath routing-based mechanisms[6][154][81][7][8], reputation- based mechanism[9], data provenance-based mechanisms[10], acknowledgement-based mechanisms [80][11][1], are inefficient as in OppNets we have no end-to-end connections and usually have no alternative paths from the sender to the destination or vice versa. Network coding based mechanisms[2], are inefficient as well since the destination nodes should have a copy.
of all neighbours packets/messages so it can decode its message, which is difficult to achieve in OppNets. Watchdog and pathrater mechanism[12][13][84][83][14][85], are inefficient for detecting this type of attack as the detection idea is based on the calculation of the total number of transmitted/received packets. Encryption techniques [15][72][155] are inefficient as well, as we required the use of a secret key which is difficult to manage in OppNets since we have no centralized management. In this chapter, our Anabolism defence is time-based and relies on the packet creation time (PCT) and the nodes contact time.

**Contribution.** To the best of our knowledge, this is the first attempt to identify this type of attack and to propose a traceback mechanism. The main contributions of this work are:

1. To identify a Catabolism attack where malicious nodes selectively drop some packets (not all packets) and then inject fake packets instead.

2. To proposed time-based Anabolism defence where a destination node can check all of the received packets and then accurately distinguish between the legitimate and fake (injected) packets. It can also traceback and identify the malicious nodes that triggered this attack.

### 3.2 Overview on Catabolism Attack and Anabolism Defence

In a Catabolism Attack, a malicious nodes can drop one or more packets (not all the packets), and instead of them, inject fake packets. In Figure 3.1, we can see an example of a Catabolism Attack where malicious node (t15) drops one packet and then injects a new fake packet instead.

Our time-based traceback technique relies on the “Packet Creation Time” of each packet and the “Nodes Contact Time”. There are two parts in our technique:

1. Detect fake packets: When a message reaches a destination or any legitimate node, the legitimate node can compare the packet creation time of each received packet. The legitimate node can then detect the fake packet with a different
Figure 3.1: Catabolism attack where a malicious nodes (t15) drops and injects a new packet

creation time, as all packets in the same message should have the same creation time or be very close (with a difference of $\Delta t$).

2. Traceback malicious nodes: The destination or legitimate node can use fake packet creation time as a benchmark to compare it with the nodes contact time in the packets path to find the malicious node.

3.2.1 Assumptions

In our approach we make the following two assumptions:

1. The sender node should automatically include a packet creation time in the header of each packet sent.

2. The intermediate nodes automatically include the contact time with other nodes when they hand them the packets to be delivered to the destination node. Storage capacity in OppNets vary depending on the nodes. However, such logs file to save this information does not need high capacity. In our approaches this information is included as part of the packet headers and therefore there is a need to only store this information temporarily.
3. To save energy and process time, malicious nodes drop some legitimate packets and instead of them injecting fake packets without modifying the contents of the packet including the packet creation time and nodes contact time.

4. The sender and the destination nodes are legitimate.

Based on these assumptions, the destination or the legitimate nodes will be able to learn all the nodes crossed by the packets along with all the information, including the packet creation time and nodes contact time. Figure 3.2 shows the message path with the packet creation time and nodes contact times.

![Figure 3.2: Packets Path with Packet Creation Time](image)

### 3.2.2 Anabolism Defence (Time-based Approach)

In this approach of the Anabolism defence, destination or legitimate node will calculate and detect the fake packets, and then traceback to identify the malicious nodes. We have three phases in the approaches shown in algorithm 1. In phase one, we identify the packets with the lowest packet creation time. We have used the lowest PCT as a benchmark (correct PCT) for our calculations. However, any PCT different than the lowest one can be considered as a fake PCT and we can use it to compare with the nodes' contact time to traceback to the malicious node. When the destination or the legitimate node receives all the packets including Packet Creation Time ($PCT$), Nodes Contact Time ($NCT$), and the number of hops for each packet, it will attempt to detect any fake packets by sorting the packet creation time of each
received packet and then choosing the smallest value in the list. All packets should have the same creation time or a very slight difference ($\Delta t$). When the malicious node drops packets, it will inject fake packets instead of them at the current malicious node time. Therefore, the fake time will always be higher than the original packet creation time of legitimate packets. In phase two, we aim to detect any fake packets. The algorithm will continuously check all packets to distinguish and count all fake and true packets. We may find more than one fake packet creation time depending on the number of malicious nodes in the packet’s path. If all packets have the same creation time ($\pm \Delta t$) then there will be no fake packets and no malicious nodes. In phase three, we aim to detect any malicious nodes. Once we get the fake packet creation times, we can then use it to traceback and find malicious nodes. In this case, we will rely on the contact time between nodes as each node should keep a record of the contact time with other nodes and include within the header of each packet so the destination node will have a clear image about the packet’s path and the nodes in that path. Once we learn the contact time between nodes, we can then compare the fake packet creation time with the contact time of nodes and then accurately detect the malicious nodes.

As mentioned earlier, in this attack, malicious nodes drop some packets and instead of them, inject fake packets. Our algorithm is based on the assumption we have at least one legitimate packet at the destination or the legitimate node which has the lowest packet creation time. We can then rely on it and compare it with other packet creation times to find fake packets and also find malicious nodes. However, if “all received packets” are fake or the “same packet sequence ” is faked twice or more, then our algorithm will compare it according to the lowest packet creation time to find the fake packets in that message and then find the malicious nodes. However, in this case one or more fake packets will be missed and categorized as legitimate and we will miss the malicious nodes behind these fake packets. We can see this case in Figures 3.3 and 3.4. In Figure 3.3, we have two malicious nodes in the same path and each one drops/injects different packet sequences. Malicious node (t15) first drops two packets and then sends all four packets to another malicious node (w14). The second malicious node (w14) drops the last two packets and then sends all four packets to the destination (c9) so it will receive four fake packets. In Figure 3.4,
Algorithm 1: Detecting fake packets and malicious nodes

1: READ: packetsCreationTime, nodesContactTime, Nodes.

2: **Phase 1**: Select lowest packet creation time

3: For all packets

4: Sort packetsCreationTime[i]

5: lowestPacketCreationTime = packetsCreationTime[0]

6: packetsAreLegitimate = true

7: **Phase 2**: Detect fake packet(s)

8: For all packets

9: if packetsCreationTime[i] = (lowestPacketCreationTime ± Δt) then

10: legitimatePacketsCounter++

11: else

12: fakePacketsCounter++

13: packetsAreLegitimate = false

14: end if

15: if packetsAreLegitimate then

16: No fake packets and no malicious node, Exit

17: end if

18: **Phase 3**: Find malicious node(s)

19: For all Nodes


21: Malicious Node = Node[j + 1]

22: maliciousNodesCounter++

23: end if
malicious node (t15) drops/injects the first two packets, and then the malicious node (w14) drops/injects two packets in the same sequence (first two packets), and then sends out four packets to the destination (c9). Node (c9) will not be able to detect the fake packets of malicious node (t15) in either scenarios nor will it be able to detect malicious node (t15). It can only detect the fake packets of malicious node (w14), and it will recognize the fake packets of (t15) as legitimate packets. However, there is a high probability we can detect the missed fake packets of malicious node (t15) and then categorize (t15) as a malicious node in the next path, as we will see in the simulation results. The only way to detect all malicious nodes is to have one or more legitimate packets and one or more fake packet from each malicious node in the destination or the legitimate node side. When we transmit a high number of packets then the probability of having two or more malicious nodes dropping and injecting the same packet sequence will be low as all events are independent. However, in the case of colluding nodes this can occur with a high probability.

### 3.3 Mathematical Model

In order to calculate the probability of detecting the fake packets of all malicious nodes in the packets’ path, let us assume,

- $n$ be the total number of hops;
• $m$ be the number of malicious hops;

• $k$ be the number of packets;

• $p$ be the probability that a packet be changed at a malicious hop;

• $\alpha$ be the probability of at least one packet surviving all malicious hops (i.e., Probability of accurate detection).

• $A_j$ be the event that exactly $j$ packets survived all malicious hops, with $A = \bigcup_{j=1}^{k-1} A_j$, $A_j \cap A_\ell = \emptyset$, for $j \neq \ell$, $j, \ell \in \{1, \ldots, k-1\}$ (note that the index is only up to $k-1$ as we cannot have all packets survive all malicious hops);

• $B_j$ be the event that, given $j$ packets, at least one of them is attacked in each malicious hop; and

• $B$ at least one of the $k$ packets is attacked in each malicious hop.

### 3.3.1 Detection of Fake Packets

The probability of a packet unchanged at a malicious node is given by $(1 - p)$, for $p$ the probability of a packet changed at the malicious node. Now, the probability of a packet unchanged in a path with $m$ malicious node(s) is given by:

$$(1 - p)^m$$  \hspace{1cm} (3.1)
Therefore, the probability of a packet being changed in a path with \( m \) malicious nodes is given by:

\[
1 - (1 - p)^m
\]

(3.2)

Now, the probability of having \( j \) packets unchanged but the rest of the \( k \) packets changed in a path with \( m \) malicious nodes is given by:

\[
[(1 - p)^m]^j [1 - (1 - p)^m]^{k-j}
\]

(3.3)

Notice that there are \( \binom{k}{j} \) combinations of obtaining \( j \) objects out of \( k \). Hence the overall probability of exactly \( j \) out of the \( k \) packets unchanged in a path with \( m \) malicious nodes is given by:

\[
\binom{k}{j} [(1 - p)^m]^j [1 - (1 - p)^m]^{k-j}
\]

(3.4)

Now, let \( \alpha \) be the probability of at least one packet surviving all malicious hops, which is equal to 1– the probability of all packets changed. We have that:

\[
\alpha = 1 - [1 - (1 - p)^m]^k
\]

(3.5)

Observe that to increase \( \alpha \), whilst \( p \) and \( m \) are held constant, one can increase \( k \), the number of packets. In Figures 3.5, 3.6 and 3.7, we can see the probability of at least one packet surviving all malicious hops in a path with 2, 5 and 20 malicious hops. We can achieve a high probability of receiving legitimate packets \( \alpha \) when the number of malicious nodes is low in the packet’s path, then it starts to drop with an increase in the malicious nodes. We can observe the value of \( \alpha \) is affected by the number of malicious hops, because when we have two malicious hops the probability of receiving legitimate packets will be high even if the value of \( p \) is high. This is because we only have two hops in that path. The value of \( \alpha \) decreases when the number of malicious nodes increases.
Figure 3.5: Probability of at least one packet surviving all malicious hops in a path with 2 malicious nodes

Figure 3.6: Probability of at least one packet surviving all malicious hops in a path with 5 malicious nodes
Figure 3.7: Probability of at least one packet surviving all malicious hops in a path with 20 malicious nodes

3.3.1.1 Optimization of the Number of Packets

Our algorithm relies on the probability of receiving at least one unchanged packet at the destination/legitimate node (“α”) so we can use the packet creation time of that unchanged packet for detecting the fake packets. We need to keep this probability $\alpha$ as high as possible by optimizing the number of packets $k$ sent in each transaction with reference to the packet probability $p$. As in equation 3.5, the probability of having at least one packet surviving all malicious hops, i.e. $\alpha$ is given as:

$$\alpha = 1 - [1 - (1 - p)^m]^k$$

Hence,

$$1 - \alpha = [1 - (1 - p)^m]^k$$,

and therefore

$$\log(1 - \alpha) = k \log [1 - (1 - p)^m]$$,

which implies that

$$k = \frac{\log(1 - \alpha)}{\log [1 - (1 - p)^m]}.$$
Figures 3.8, 3.9 and 3.10 show the relationship between the number of packets and the probability of at least one packet surviving all malicious hops in different path lengths. When the number of malicious nodes increases, the probability of packet changes $p$ in a path with malicious nodes will increase as well. In this case, we will be required to send a large number of packets $k$ so we can achieve a high probability (0.85 - 0.95) of receiving at least one legitimate packet $\alpha$ in order to achieve a high accuracy of detecting fake packets.

![Graph showing the relationship between number of packets needed to achieve $\alpha$ in a path with 2 malicious nodes](image)

Figure 3.8: Number of packets needed to achieve $\alpha = (0.85, 0.90, 0.95)$ in a path with 2 malicious nodes

### 3.3.2 Malicious Node Detection

To detect malicious nodes, we require two conditions to be satisfied. The two conditions are:

1. At least one legitimate packet is received, and
2. At least one fake packet from each malicious node is received.

Given $p$, the probability of a packet surviving at a malicious hop is $q = 1 - p$. Now, let $\mathbb{P}$ be the probability that a packet will survive all malicious hops, i.e.

$$\mathbb{P} = q^m$$  \hfill (3.6)
Chapter 3 Catabolism Attack and Anabolism Defence

Figure 3.9: Number of packets needed to achieve \( \alpha = (0.85,0.90,0.95) \) in a path with 5 malicious nodes

Hence, the probability that a packet will be attacked at one or more of the hops, \( Q \), is given by:

\[
Q = 1 - P
\]  

(3.7)

The probability of \( A_j \), that is, exactly \( j \) out of \( k \) packets survived all malicious hops, is:

\[
Pr(A_j) = \binom{k}{j} P^j Q^{k-j}
\]

(3.8)

We now calculate \( Pr(B \cap A) \). As \( A_j \), for \( j \in \{1, \ldots, k-1\} \) are distinct sets, we have that:

\[
Pr(B \cap A) = Pr\left(B \cap \left( \bigcup_{j=1}^{k-1} A_j \right) \right) = \sum_{j=1}^{k-1} Pr(B \cap A_j)
\]

By Bayesian,

\[
Pr(B \cap A_j) = Pr(B|A_j) Pr(A_j)
\]

Note that when exactly \( j \) packets survived all malicious hops, it means that for the rest of the \( k - j \) packets, they must all be attacked by at least one of the malicious
Chapter 3 Catabolism Attack and Anabolism Defence

Figure 3.10: Number of packets needed to achieve $\alpha = (0.85,0.90,0.95)$ in a path with 20 malicious nodes

hops. Hence,

$$\Pr(B|A_j) = \Pr(B_{k-j}|A_j)$$

First, the probability that with $k - j$ packets, at least one of them will be attacked at each malicious hop is given by:

$$\Pr(B_{k-j}) = [1 - q^{k-j}]^m \quad (3.9)$$

However, this includes the event that at least one of the $k - j$ packets survived all malicious hops but not all $k - j$ packets. We use $C_{k-j}$ to denote such an event. Thus, we have:

$$\Pr(C_{k-j}) = \sum_{\ell=1}^{k-j-1} \binom{k-j}{\ell} \Pi^\ell \Omega^{k-j-\ell} \quad (3.10)$$

Therefore,

$$\Pr(B_{k-j}|A_j) = \Pr(B_{k-j}) [1 - \Pr(C_{k-j})] =$$

$$[1 - q^{k-j}]^m \left[ 1 - \sum_{\ell=1}^{k-j-1} \binom{k-j}{\ell} \Pi^\ell \Omega^{k-j-\ell} \right] \quad (3.11)$$
and we get:

\[ \Pr(B \cap A) = \sum_{j=1}^{k-1} \Pr(B_{k-j} | A_j) \Pr(A_j) = \]

\[ \sum_{j=1}^{k-1} \left\{ \left[ 1 - q^{k-j} \right]^m \left( 1 - \sum_{\ell=1}^{k-j-1} \binom{k-j-1}{\ell} P^\ell Q^{k-j-\ell} \right) \binom{k}{j} P^j Q^{k-j} \right\}. \] (3.12)

Notice that \( \Pr(B \cap A) \) includes the case when the same packet is corrupted in more than one malicious hop. However, if \( k >> m \), the probability of such an occurrence is negligible. Figures 3.11, 3.12 and 3.13 show the probability of detecting malicious nodes in paths with 2, 5 and 20 malicious nodes. When we have a small number of malicious nodes, the probability of detecting them will be high as we have a good chance of receiving legitimate packets as well as one or more fake packet from each malicious node. With the increasing number of malicious nodes, we will have less chance to receive legitimate packets and fake packets from each malicious node, which means the probability begins to decrease. The probability of detecting malicious nodes is also affected by the number of packets sent. When the number of transmitted packet is high (such as 1000), there is a higher probability of achieving the required sufficient condition of receiving at least one legitimate packet and one fake packet at a legitimate node. This leads to a very high (almost 1) detection probability being achieved.
Figure 3.11: Probability of detecting malicious nodes in a path with 2 malicious nodes

Figure 3.12: Probability of detecting malicious nodes in a path with 5 malicious nodes
Figures 3.14, 3.15, 3.16 and 3.17 show the best number of packets $k$ to be sent so we can achieve high probability of detecting malicious nodes “$Pr$” in different paths with 2, 4, 6 and 8 malicious nodes in each path. We consider the probability that a packet will be attacked $p$ and changing between 0.1 to 0.5, as we rely on the two events mentioned earlier in this section. When we increase the number of packets, the probability of detecting malicious nodes will be very close to 1.

### 3.4 Simulation Settings

To test our algorithm, we implemented a scenario in the ONE simulator [70]. The simulation was defined to last for 1 hour, with 0.5 seconds of update intervals. Bluetooth was chosen for connectivity with a transmit range of 10 meters, and transmit speeds of 1000 kbps. There were 35 active nodes composed of cars, trams and pedestrians. Pedestrians and cars have up to 50 MB of RAM for storage. Pedestrians move at random speeds between 1 and 1.5 m/s, cars drive only on roads and move at speeds between 10 – 50 km/h, with wait times of 0 – 120 seconds. MapBased-Movement is used for pedestrians and cars, with a network area of $4500m^2 \times 3400m^2$. Nodes move randomly on roads and walkways with a movement warm-up for 10 sec-
Figure 3.14: Probability of accurate detecting of malicious nodes with 2 malicious nodes in a path.

There are 3 groups of trams, with 2 trams in each group. MapRouteMovement is used for trams to follow a constructed tram line. Trams drive at speeds of $7 - 10$ m/s with a wait time of $10 - 30$ seconds at each configured stop. In addition to the Bluetooth interface, a group of trams used the high speed interface with a transmit range of 1000 meters and a transmit speed of 10mbps. Messages are generated every 1 to 5 seconds per node, with message sizes between 50kb and 100kb, and a message time to live of 5 hours. We used the simulator’s output as a dataset, and randomly corrupted the dataset based on the number of malicious nodes. We then fed the corrupted dataset to our algorithm. Two programs were written using C++. The first program (total of 353 lines) read the dataset and then corrupted it by making legitimate nodes malicious by changing the packet creation time for randomly chosen packets and nodes. The second program (total of 503 lines) implemented the algorithm, and began by taking as input the output dataset generated by program 1. The second program was run to get the algorithm results of the metrics calculations. We ran the simulator for an average of 30 times to represent each point on our graphs.
3.4.1 Simulation Results and Analysis

We have used four metrics for evaluating our algorithm,

1. Fake packet detection accuracy: The ratio of the total number of fake packets detected to the total number of actual fake packets.

2. False negative rate: The percentage of fake packets have been incorrectly classified as a legitimate packet.

3. Malicious node detection accuracy: The ratio of the total number of detected malicious nodes to the total number of actual malicious nodes.

4. Network traffic reduction: The ratio of the total number of fake packets detected in the destination nodes side to the total number of the fake packets detected in the node by node side.

3.4.1.1 Destination-based Approach

In this approach, our detection and traceback mechanism is destination based. Only destination node run the algorithm to detect the attack and then traceback the
malicious nodes. In this case nodes resources can be saved. We have assumed the source and destination nodes are not malicious as the source node always sends packets with the same creation time. The destination node should be legitimate so it can run the algorithm and do the calculations as in a Catabolism attack, the malicious node drops some of the packets, and instead of them injects fake packets with the current malicious node time. Dropping and injecting may take place in one or more nodes along packets path. In our calculations, we have assumed we have at least one legitimate packet (with lowest creation time) so we can use it as a benchmark comparison.

In Figure 3.18, we can observe the packet detection accuracy of our algorithm with different OppNets protocols (Epidemic [24], PRoPHET [59], Spray and Wait [25], MaxProp [66] and First Contact (FC) [36]). We see that we can achieve a detection accuracy of 96% for Epidemic, 95% for PRoPHET, 92% for Spray and Wait, 90% for MaxProp and 88% for First Contact when the percentage of malicious nodes is less than 5%. This is because there is a good chance for destinations to receive legitimate packets, especially when source nodes transmit large number of packets.
When the number of malicious nodes increases, the accuracy starts to decrease as fake packets may be missed if “all received packets” are fake or the “same packet sequence” is faked twice or more, as illustrated in Figure 3.3 and 3.4. However, our algorithm results show the packet detection accuracy does not drop below 70% (First Contact (FC) [36]), even when 100% of intermediate nodes act as malicious nodes. This is due to the low probability of having two or more malicious nodes dropping and injecting the same packet sequence along the same path. In addition, the probability of receiving all packets as fake is also low, especially when the sender sends a large number of packets.

Epidemic [24] and PRoPHET [59] achieved the highest detection accuracy, because both of them replicate multiple copies of a message in the network. In this case, nodes can have a good participation of receiving and transmitting the packets in the network and then our algorithm can have a good chance to detect the malicious nodes and achieve a good accuracy. Spray and Wait [25] achieved a moderate level of detection accuracy, as in this protocol the source node sprays a predefined number of copies to the network, and then the nodes perform direct delivery to the destination. MaxProp [66] is even lower, as MaxProp has a prioritization schema for both
the schedule of packets transmitted and dropped due to the assumption of limited storage and bandwidth and the use of broadcasted acknowledgments to update the encountered nodes about the delivered messages. First Contact algorithm (FC) [36], achieved the lowest detection accuracy as the message is forwarded to the next hop randomly. The message stays in the buffer and gets forwarded when the node carrying it encounters the first node it meets even when it has zero knowledge about it.

![Figure 3.18: Fake packet detection accuracy as the number of malicious nodes increases](image)

Next we studied the impact of the mobility on the detection accuracy. Figure 3.19, shows our detection accuracy based on the mobility types. We have used the Epidemic protocol with different mobility types (Trams, cars and pedestrians). Trams achieved very good accuracy (between 77% - 100%) due to their features such as a high speed interface and mobility speed. Cars achieve better accuracy (between 75% - 97%) than pedestrians (between 70% - 92%) because cars have high speed compared to the pedestrians. This enabled them to achieve a better message distribution.

As we can recall from Figure 3.3, we may categorize fake packets as legitimate when we have more than one malicious node in the same path where each malicious node fakes a different sequence of packets. This results in having all the received packets
Figure 3.19: Fake packet detection accuracy based on the mobility of nodes

at the destination as fake packets. In Figure 3.20, we can see a zero false negative rate for our algorithm when we have less than 2% malicious nodes. This is because of the small number of malicious nodes and the probability of having more than one malicious node sending to each other will be very low. The false negative rate starts to increase slightly until it reaches a maximum of 23%, when the number of malicious nodes increase, as the probability of having more than one malicious node in the same packet path will increase.

In order to compare our approach with other approaches, we have chosen the acknowledgement - based mechanisms [1] and the network coding-based mechanism [2]. As comparison metrics, we have measured the malicious node detection ratio. In Figure 3.21, we can see our algorithm achieved a node detection accuracy of 100% when the percentage of malicious nodes are less than 15%. This is because there is a good chance for destinations to receive legitimate packets and at least one fake packet from each malicious node. When the number of malicious nodes increases, the accuracy slightly starts to drop as a destination node may not receive legitimate packets from source node and/or fake packets from malicious nodes may be missed when two or more malicious nodes drop and inject packets into the same
packet sequence in the same path, as illustrated in Figures 3.3 and 3.4. However, our algorithm results show the nodes detection accuracy does not drop below 93%, even when 100% of intermediate nodes act as malicious nodes. This is due to the low probability of having two or more malicious nodes dropping and injecting in the same packet sequence in the same path. In addition, the probability of receiving all packets as fake is also low, especially when the sender sends a large number of packets. However, in Figure 3.21, we can see other approaches perform very poorly especially when the number of malicious nodes increase to 100% of the intermediate nodes. In acknowledgement [1] and network coding [2] based mechanisms, destination nodes need to send the acknowledgments value to the source node through a legitimate path which is difficult/impossible if the number of malicious nodes reach 100%. Malicious nodes can swap the destination acknowledgment values so it can go undetected, moreover this destination-based mechanism can only detect a malicious path (not the malicious nodes) and has high false positive rate.

To increase the security level we need to run the Anabolism defense algorithm in every node despite the fact that we may have increased overhead or delay. Furthermore, OppNets by design are delay tolerant networks. In the event that the network is severely resource constrained, we can revert to destination-based detection. However, this will decrease the security level of the network.
3.4.1.2 Node by Node-based Approach

In this approach, our detection and traceback mechanism is node-by-node-based. Any legitimate node can run the algorithm to detect the attack and then traceback to the malicious nodes. It is true that OppNets in some instances can be energy constrained. This is our motivation for proposing the destination node approaches as we do not required our algorithm to be run in each node, hence saving node resources. However, to achieve an increased security level we need to run the algorithms in-network (on all nodes) despite the fact that we may be resource constrained. In destination based method, the cost is fixed $O(1)$; however, in node based methods the cost is dependent on the number of nodes in the network and will be $O(n)$. We have also assumed the source nodes are not malicious as the source node always sends packets with the same creation time. Recall, in a Catabolism attack, the malicious node drops some of the packets, and instead of them injects fake packets with the current malicious node time. Dropping and injecting will occur on one or more nodes along the packet’s path. In our calculations, we have assumed we have at least one legitimate packet (with lowest creation time) so we can use it as a benchmark comparison.

In Figure 3.22, we can see the packet detection accuracy of our algorithm in different OppNets protocols (Epidemic [24], PRoPHET [59], Spray and Wait [25], MaxProp.
[66] and First Contact (FC) [36]) can achieve a detection accuracy of (97% for Epidemic, 96% for PRoPHET, 93% for Spray and Wait, 91% for MaxProp and 89% for First Contact) when the percentage of malicious nodes is less than 5%. This is because there is a good chance for the nodes to receive legitimate packets, especially when source nodes transmit large number of packets. When the number of malicious nodes increases, the accuracy slightly starts to drop as fake packets may be missed when “all received packets” are fake or the “same packet sequence ” is faked twice or more, as illustrated in Figures 3.3 and 3.4. However, our algorithm results show the packet detection accuracy does not drop below 70% (First Contact (FC) [36]), even when 100% of intermediate nodes act as malicious nodes. This is due to the low probability of having two or more malicious nodes dropping and injecting in the same packet sequence in the same path. In addition, the probability of receiving all packets as fake is also low, especially when the sender sends a large number of packets. In node by node defence we always achieved better packet detection accuracy than destination nodes defence as fake packets will not propagate through the network till it reaches the destination. Any legitimate node can stop fake packet propagation, in contrast, destination defence can detect fake packets only through destination node.

Epidemic [24] and PRoPHET [59] achieved highest detection accuracy, because both of them replicate multiple copies of a message in the network. In this case nodes will have good participation in the network and then our algorithm can achieve good accuracy. Spray and Wait [25] is lower in terms of detection accuracy as in this protocol the source node sprays a predefined number of copies to the network, and then nodes do direct delivery to the destination. MaxProp [66] is even lower, as MaxProb has a prioritization schema for both the schedule of packets transmitted and dropped due to the assumption of limited storage and bandwidth and the use of broadcasted acknowledgments to update the encountered nodes about the delivered messages. First Contact algorithm (FC) [36], achieved lowest detection accuracy as the message is forwarded to the next hop randomly. The message stays in the buffer and gets forwarded when the node carrying it encounters the first node it meets even when it has zero knowledge about it.
Figure 3.22: Fake packet detection accuracy as the number of malicious nodes increases

Figure 3.23, shows our detection accuracy based on the mobility types. We have used Epidemic protocol with different mobility types (Trams, Cars and Pedestrian). Trams achieved very good accuracy (between 79% - 100%) due to its features such as a high speed interface and mobility speed. Cars achieve better accuracy (between 77% - 97%) than pedestrians (between 71% - 92%) because, cars have high speed compared to the pedestrians. This results in achieving a better message distribution.
Figure 3.24 shows the networks traffic reduction in node by node based defence technique compare to the destination-based defence technique. Overall we have achieved good traffic reduction as each legitimate node can detect fake packets and malicious nodes and then stop fake packet propagation through the network.

As we can see in Figure 3.3, we may categorize fake packets as legitimate when we have more than one malicious node sending to each other in the same packets path where each malicious node fakes a different sequence of packets. This results in having all the received packets at the legitimate node as fake. In Figure 3.25, we can see a zero false negative rate for our algorithm when we have less than 2% malicious nodes. This is because of the small number of malicious nodes and the probability of receiving all packets as fake will be very low. The false negative rate starts to increase slightly till it reaches a maximum of 22%, and the number of malicious nodes also increase as the probability of receiving all packets as fake also rises.
Chapter 3 Catabolism Attack and Anabolism Defence

In order to compare our approach with other works focusing on the same subject, we have chosen the acknowledgement-based mechanisms [1] and the network coding-based mechanism [2]. As comparison metrics, we have measured the node detection ratio. In Figure 3.26, we can see our algorithm achieved a node detection accuracy of 100% when the percentage of malicious nodes are less than 16%. This is because there is a good chance for the legitimate node to receive legitimate packets and at least one fake packet from each malicious node. When the number of malicious nodes increases, the accuracy slightly starts to drop as the legitimate node may not receive legitimate packets from source node and/or fake packets from malicious nodes may be missed when two or more malicious nodes drop and inject packets into the same packet sequence in the same path, as illustrated in Figures 3.3 and 3.4. However, our algorithm results show the nodes detection accuracy does not drop below 92%, even when 100% of intermediate nodes act as malicious nodes. This is due to the low probability of having two or more malicious nodes dropping and injecting in the same packet sequence in the same path. In addition, the probability of receiving all packets as fake is also low, especially when the sender sends a large number of packets. In the same Figure 3.26, we can see other approaches perform very poor especially when the number of malicious nodes increase to 100% of the intermediate nodes. In acknowledgement [1] and network coding [2] based mechanisms, destination nodes need to send the acknowledgment values to the source node through a legitimate path which is difficult/impossible if the number of malicious nodes reach 100%. Malicious nodes can swap the destination acknowledgment values so it can go
undetected, moreover this destination-based mechanism can detect malicious path (not the malicious nodes) and has a high false positive rate.

Figure 3.26: Nodes detection accuracy, our approach with Epidemic vs ACK and network coding techniques

3.5 Summary

In this chapter, we presented a novel attack and traceback mechanism against a special type of packet dropping attack where the malicious nodes selectively drop some packets (not all the packets) and instead of them inject fake packets so it can maintain the original total number of packets that originated from the sender node. We call this novel attack a Catabolism attack and we call our novel traceback mechanism against this attack Anabolism defence. Our proposed detection and traceback mechanism is time-based and relies on the packet creation time (PCT) of each packet and the contact time between nodes. There are two phases in our technique. The first phase is to detect the fake packets, and the second phase is to find the malicious nodes. We have tested our detection and traceback mechanism using both a destination-based approach as well as a node-by-node-based approach. In the first approach, destination nodes check the received packets to detect the attack and then traceback the malicious nodes. In the node by node approach, each legitimate node
checks the received packets to detect the attack, prevent fake packets propagation and then traceback the malicious nodes. In terms of node detection accuracy, we have compared our approach with one of the acknowledgement-based approaches [1] and one of the network coding-based approaches [2] which are well known approaches in the literature. Simulation results show node-by-node detection and traceback mechanism achieve higher detection accuracy than the destination node with significant network traffic reduction. Other approaches perform very poorly especially when the number of malicious nodes increase to 100% of the intermediate nodes. However, the current approaches are based on the assumption that the malicious nodes have the ability of dropping legitimate packets and instead of them inject fake packets but has no ability to modify the packets contents including the packet creation time. This assumption is not always realistic as the malicious node may have the ability to write the time that it wants to be, so the injected packets will look like a legitimate packets. Smart malicious nodes can write other times so they can point to another legitimate nodes in the network to appear as malicious. It is possible for malicious nodes to collude with each other with one malicious node dropping packets only while another malicious node injects packets only. However, this kind of a “collusion” attack and the nodes involved will always be detected by both the Merkle based and Hash-chain based techniques. In the next chapter, Merkle tree techniques will be used to deal with this security lack. Any legitimate nodes in the packets path can detect the attack, stop fake packets propagation and then traceback the malicious nodes.
Chapter 4

Anabolism Defence - Merkle Tree Approach

In this chapter, we present a Merkle tree-based defence mechanism against Catabolism Attacks. We also present a node by node traceback mechanism. In this approach, each legitimate node can detect the attack and then traceback to the malicious nodes. Simulation results indicate that we can achieve a very high accuracy and mitigate the propagation of fake packets in the network. In this defence techniques we have two phases. The first phase is to detect the attack, and the second phase is to find the malicious nodes. We have compared our approach with one of the acknowledgement-based approaches [1] and one of the network coding-based approaches [2] which are well-known approaches in the literature. Simulation results show this robust mechanism achieves a very high accuracy and detection rate.

4.1 Introduction

As we mentioned in chapter 3, Catabolism attack is a combined packet dropping and packet injection attack where malicious nodes selectively drop some packets and instead of them inject fake packets in order to maintain the original total number of packets originated from the sender node. The approaches in chapter 3 rely on the packets creation time and nodes contact time with the assumption that the malicious node(s) has no ability to modify the packet’s contents, including the packet’s creation time and node’s contact times. This assumption is not always realistic as
the malicious node may have the ability to write the time that it wants to ensure that the injected packets appear to be legitimate packets. Smart malicious nodes can also write incorrect node contact times in order to make legitimate nodes in the network to appear malicious. One of the weaknesses of the PCT method is that it is only effective if the attacker is weak (i.e., does not have the ability to modify the packet contents). To address this weakness, we have proposed the use of the Merkle-based or hash chain-based methods. In this chapter, Merkle tree techniques will be used to deal with these security issue. Based on the proposed method, we will be able to detect the attack, stop fake packets propagation and then traceback to the malicious nodes.

**Contribution.** The main contributions of this work are:

1. To identify a solid and powerful node by node detection and defence mechanism using Merkle trees where the legitimate nodes can check all of the received packets and then accurately detect the attack even when the malicious nodes change the entire content of the packets.

2. To traceback and identify the malicious node(s) that triggered the Catabolism attack.

### 4.2 Overview on the Merkle Tree Approach Mechanism

In the Merkle tree - based approach each legitimate node can run the algorithm to detect the attack and traceback malicious nodes directly. Propagation of fake packets through the network is prevented as the legitimate nodes can detect and drop fake packets and directly classify a node as malicious.

#### 4.2.1 Assumptions

In our approach we make the following two assumptions:

1. The sender node constructs a Merkle (or hash) tree to calculate the merkle root value and then include it within each packet.
2. Malicious nodes have the ability of dropping some legitimate packets and then instead of them inject new fake packets with the ability of modifying entire contents of packets including the merkle root value.

3. The sender and the destination nodes are legitimate.

In our defence approach we aim to not only detect the attack but also to identify the malicious nodes that are the source(s) of the attack. In order to achieve this we construct and use a Merkle tree. We provide the details of our defence and detection method in the following section.

### 4.2.2 Merkle Based Defence and Traceback Method

As noted earlier, malicious nodes can drop one or more packets (but not all the packets), and inject fake packet’s with recalculated parameters such as packets creation times, node’s contact times and hash values. As an example, Figure 4.1 shows the packet’s path of message 1 (c8 → t16 → t15 → w14 → c9), where node t15 drops one packet at time T= 860.2 and then injects a fake packet instead. In our defence and traceback method our aim is to detect the packet injection and identify the source of the injection.

The basis for our method is the construction of a Merkle tree. A merkle tree can be
defined as a tree in which every leaf node is labelled with the hash of the labels of its child nodes. To construct a Merkle tree in OppNets we adopt the following steps. Given a set of packets to transmit, the source node calculates a secure hash for each packet and then builds a Merkle tree using the resulting hashes to get a Merkle root value for the message, as in Figure 4.2 where “H07” is the root of the message. The source node will then inject the Merkle root value in each packet as shown in the Figure 4.3.

Our detection method (algorithm 2), is based on a node by node defence techniques. It has three phases;

In phase one, the legitimate node calculates a secure hash for each received packet to generate tree leaves, where:

\[ H_0 = h(P_0), \]
\[ H_1 = h(P_1), \]
\[ H_2 = h(P_2), \]
\[ H_3 = h(P_3), \]

and so on.

In phase two, the legitimate node builds the tree by hashing the XOR value of the
first leaf \((H_0)\) and the second leaf \((H_1)\), and continues to hash every two leaves in their order from left to right until level one hashes are generated as follows:
\[
H_{01} = h(H_0 \oplus H_1),
\]
\[
H_{23} = h(H_2 \oplus H_3),
\]
\[
H_{45} = h(H_4 \oplus H_5),
\]
\[
H_{67} = h(H_6 \oplus H_7), \text{ and so on.}
\]
Level two will be built based on level one values as:
\[
H_{03} = h(H_{01} \oplus H_{23}),
\]
\[
H_{47} = h(H_{45} \oplus H_{67}).
\]
The root value will be calculated based on level two values, as:
\[
H_{07} = h(H_{03} \oplus H_{47}).
\]
At any level, if there is an odd number of hashes then we will hash the XOR value of the last hash with itself.

In phase three, our algorithm compares the equality of the root values. If the new recalculated root value equals the existing value that is already injected by the source node in each packet header, then there is no attack, hence no malicious node. If the two roots do not equal each other, then an attack has occurred and we can directly classify the previous node as malicious.

As mentioned earlier, malicious nodes can drop some packets and instead of them inject fake packets with new recalculated parameters or leaving some parameters unchanged. Malicious nodes can leave the root value unchanged or even recalculate a new root value and inject it with the new fake packets. For legitimate nodes to be able to detect the attack and then traceback to the malicious nodes, we need to achieve one of two sufficient conditions. These are:

i Legitimate nodes should receive at least one fake packet from any malicious node, and at least one legitimate packet (Figure 4.4(a)) or

ii At least one fake packet from two or more malicious nodes (Figure 4.4(b)).

Under both these conditions when a legitimate node builds a new Merkle tree to get the root value it will not match with the existing root values. In this case, a legitimate node will classify the previous node as malicious directly. In case we have two or more malicious nodes sending to each other as in Figure 4.5, then the
Algorithm 2 - Detecting fake packets and malicious nodes

1: READ: packets
2: \( n = \text{numberOfPackets} \)
3: **Phase 1: Hashing each packet**
4: For all packets
5: \( h_i = h(P_i) \)
6: **Phase 2: Build tree and compute Merkle root**
7: For all hashes in each level
8: if numberOfHashes = even then
9: \( h_{ij} = h(h_i \oplus h_{i+1}) \)
10: else
11: if !lastHash then
12: \( h_{ij} = h(h_i \oplus h_{i+1}) \)
13: else
14: \( h_{ij} = h(h_i \oplus h_i) \)
15: end if
16: if level = last then
17: rootValue = \( h_{ij} \)
18: end if
19: end if
20: **Phase 3: Compare Merkle root values**
21: For all packets
22: READ: \( rootValue'_{(i)} \)
23: if rootValue = \( rootValue'_{(i)} \) then
24: No attack, No malicious nodes, Exit
25: else
26: Attack detected
27: Previous node is malicious
28: maliciousNodesCounter++
29: end if
legitimate node will classify only one of them as malicious, that is the previous node (w14) and we will miss other malicious nodes (t15) as we do not have a clear trace from each malicious nodes and we cannot distinguish between the correct and the fake Merkle root value. However, as our algorithm is based on node by node detection, we will have a high probability to detect the malicious nodes in another path especially when the percentage of the malicious nodes is not high.

![Figure 4.4: Fake packets conditions that we need to achieve](image)

![Figure 4.5: Two malicious nodes sending to each other in one message path](image)

### 4.3 Mathematical Model

The aim of the mathematical model is to derive a formula for the probability of achieving the sufficient condition leading to malicious node detection. The sufficient condition is: receiving at least 1 fake packet from any malicious node, and receiving at least 1 legitimate packet or at least 1 fake packet from a different malicious node. We first introduce the notations used.

- \( n \) be total number of hops;
• $m$ be the number of malicious hops, where $m < n$;

• $k$ be the number of packets;

• $p$ the probability that a packet be attacked at a malicious hop; and

• $A_j$ be the event that exactly $j$ packets survived all malicious hops, where $j < k$.

Let $\beta$ be the probability that there exists at least one fake packet. Obviously,

$$\beta = 1 - \text{there exists no fake packets}$$

Given $p$, the probability of a packet surviving a malicious hop is $q = 1 - p$. Let $\mathbb{P}$ be the probability that a packet will survive all malicious hops, i.e.

$$\mathbb{P} = q^m.$$

The probability of having all $k$ packets not fake is

$$\mathbb{P}^k = q^{mk},$$

and thus:

$$\beta = 1 - q^{mk} \implies 1 - \beta = q^{mk}.$$

This means that

$$\log(1 - \beta) = mk \log q \implies k = \frac{\log(1 - \beta)}{m \log q} = \frac{\log(1 - \beta)}{m \log(1 - p)}.$$

Now, the probability that a packet be attacked at one or more of the hops, $\mathbb{Q}$, is given by:

$$\mathbb{Q} = 1 - \mathbb{P}.$$

The probability of $A_j$, that is, exactly $j$ out of $k$ packets survived all malicious hops, is:

$$\Pr(A_j) = \binom{k}{j} \mathbb{P}^j \mathbb{Q}^{k-j}.$$

As we wish to have at least one packet surviving all malicious hops, and at least one packet that does not, (i.e., at most $k - 1$ packets survives all malicious hops), the probability of such an event is:
\[
\sum_{j=1}^{k-1} \Pr(A_j) = \sum_{j=1}^{k-1} \binom{k}{j} I^j Q^{k-j}
\]

Now, the probability that none of the \( k \) packets survived all \( m \) malicious hops, i.e., all \( k \) packets are fake, is given as: \( Q^k \).

The probability of all \( k \) fake packets were attacked at one particular malicious hop is \((1/m)^k\). As there are \( m \) malicious hops, the probability that all \( k \) fake packets were attacked at the same malicious hop is \( \frac{1}{m^{k-1}} \). Hence, the probability that at least two of the fake packets were attacked at two different malicious hops is given by:

\[
1 - \frac{1}{m^{k-1}}.
\]

Therefore, the probability of none of the \( k \) packets survived all \( m \) malicious hops and that at least two of the fake packets were attacked at two different malicious hops is given by:

\[
Q^k \left( 1 - \frac{1}{m^{k-1}} \right).
\]

In summary, the probability that there is at least 1 fake packet from any malicious node, and that there are at least 1 legitimate packet or at least 1 fake packet from a different malicious node, is given by:

\[
\sum_{j=1}^{k-1} \Pr(A_j) = \sum_{j=1}^{k-1} \binom{k}{j} I^j Q^{k-j} + Q^k \left( 1 - \frac{1}{m^{k-1}} \right),
\]

which is really just one minus the probability of none survived, and that all \( k \) fake packets were attacked at the same malicious node.

Hence, the probability is given by:

\[
1 - \frac{Q^k}{m^{k-1}} - P^K.
\]

Notice that

\[
1 = \sum_{j=0}^{k} \binom{k}{j} I^j Q^{k-j} = I^0 Q^k + \sum_{j=1}^{k-1} \binom{k}{j} I^j Q^{k-j} + I^K Q^0.
\]

As

\[
Q^k = Q^k \left( 1 - \frac{1}{m^{k-1}} \right) + \frac{Q^k}{m^{k-1}},
\]
we have that

\[ 1 = Q^k \left( 1 - \frac{1}{m^{k-1}} \right) + \frac{Q^k}{m^{k-1}} + \sum_{j=1}^{k-1} \binom{k}{j} P^j Q^{k-j} + P^k. \]

The probability that there is at least 1 fake packet from any malicious node, and that there are at least 1 legitimate packet or at least 1 fake packet from a different malicious node, is given by:

\[ \sum_{j=1}^{k-1} \binom{k}{j} P^j Q^{k-j} + Q^k \left( 1 - \frac{1}{m^{k-1}} \right) = 1 - \frac{Q^k}{m^{k-1}} - P^K. \]

Figures 4.6, 4.7 and 4.8 show the probability of receiving at least one fake packet (\(\beta\)) in paths with 1, 2 and 5 malicious nodes. When we have a large number of malicious nodes, the probability of detecting the attack or the malicious nodes will be high as we have a good chance of receiving at least one fake packet and at least one legitimate packet or another fake packet from a different malicious node. The probability of detecting malicious nodes is also affected by the number of packets sent. When the number of packets sent increases, the probability of receiving fake packets will increase and then enable us to achieve a high detecting probability.

![Figure 4.6: Probability of receiving at least one fake packet (\(\beta\)) in a path with 1 malicious nodes](image)

Figures 4.9, 4.10 and 4.11 show the relationship between the number of packets and the probability of at least one packet being changed (\(\beta\)) across paths with different
number of malicious nodes. When the probability $p$ of a packet being changed by a malicious node is low, we need to send a large number of packets $k$ so we can achieve a high probability (0.85 - 0.95) of receiving at least one fake packet in order to achieve a high accuracy of detecting a malicious node. When the number of malicious nodes is high along the packet path, we do not need to send a high number of packets as we can have a good chance of receiving fake packets from the malicious nodes in the packet’s path.

### 4.4 Simulation Settings

To test our algorithm, we implemented a scenario in the ONE simulator [70] using the Epidemic, PRoPHET, Spray and Wait MaxProp and First Contact protocols. The simulation was defined to last for 1 hour, with 0.5 seconds of update intervals. Bluetooth was chosen for connectivity with a transmit range of 10 meters for node radio devices, and transmit speed’s of 1000 kbps. There are 35 active nodes composed of cars, trams and pedestrians. Pedestrians and cars have up to 50 MB of RAM for storage. Pedestrians move at random speeds between 1 and 1.5 m/s, cars drive only on roads and move at speeds between 10 – 50 km/h, with wait times of 0 – 120
seconds. MapBasedMovement is used for pedestrians and cars, with a network area of $4500m^2 \times 3400m^2$. Nodes move randomly on roads and walkways with a movement warm-up for 10 seconds. There are 3 groups of trams, with 2 trams in each group. MapRouteMovement is used for trams to follow a constructed tram line. Trams drive at speeds of $7 - 10$ m/s with a wait time of $10 - 30$ seconds at each configured stop. In addition to the Bluetooth interface, a group of trams use the high speed interface with a transmit range of 1000 meters and a transmit speed of 10mbps. Messages are generated every 1 to 5 seconds per node, with message sizes between 50kb and 100kb, and a message time to live of 5 hours. We used the simulator’s output as a dataset, and randomly corrupted the dataset based on the number of malicious nodes. We then fed the corrupted dataset to our algorithm. We also ran the simulator for an average of 30 times to represent each point on our graphs. Two programs were written using C++. The first program (total of 353 lines) read the dataset file and then corrupted it by making legitimate nodes malicious by changing the packet creation time for randomly chosen packets and nodes. The second program (total of 816 lines) implements the algorithm, and begins by taking as input the output dataset file generated by program 1. The second program is run to get the algorithm results of the metrics calculations.

Figure 4.8: Probability of receiving at least one fake packet ($\beta$) in a path with 5 malicious nodes
4.4.1 Simulation Results and Analysis

The metrics used for evaluating our algorithm were:

1. Malicious node detection accuracy: The ratio of the total number of detected malicious nodes to the total number of actual malicious nodes.

2. False negative rate: The percentage of malicious nodes that have been incorrectly classified as a legitimate nodes.

In our scenario, we have assumed the source nodes are not malicious as these nodes will generate the messages and then hand them to the neighbour nodes. Neighbour nodes can be malicious or legitimate. However, when the legitimate nodes run the algorithm, they should be able to accurately distinguish between a malicious and a legitimate sender. In the catabolism attack, the malicious node drops some of the packets, and instead of them injects fake packets. Dropping and injecting can occur over one or more nodes along packets’ path.

In Figure 4.12, we can observe that our algorithm achieves a node detection accuracy of 100% when the percentage of malicious nodes is less than 10%. This is because there is a good chance for a legitimate node to detect the difference between the existing Merkle root value and the new recalculated value as well as the probability...
of having more than one malicious node sending to each other in the same path will be low. When the number of malicious nodes increases, the accuracy slightly starts to drop as we may have two or more malicious nodes sending to each other as illustrated in Figures 4.5. However, our algorithm results show the malicious nodes detection accuracy does not drop below 91% for Epidemic protocol, 90% for PRoPHET protocol, 89% for Spray and Wait protocol, 88% for MaxProp protocol and 87% for First Contact protocol, even when 100% of intermediate nodes act as malicious nodes. This is due to the sufficient number of short paths where we can detect the malicious nodes directly without malicious nodes coming in to contact with other malicious nodes.

Epidemic [24] and PRoPHET [59] achieved the highest detection accuracy because both of them replicate multiple copies of a message in the network. In this case, nodes will have good participation in the network and our algorithm has more opportunity to detect them. Spray and Wait [25] is lower in terms of detection accuracy as in this protocol the source node sprays a predefined number of copies to the network, and then nodes do direct delivery to the destination. MaxProp [66] is even lower, as MaxProb has a prioritization schema for both the schedule of packets transmitted and dropped due to the assumption of limited storage and bandwidth and the use of

Figure 4.10: Number of packets needed to achieve $\beta = (0.85, 0.90, 0.95)$ in a path with 2 malicious nodes
Chapter 4 Anabolism Defence - Merkle Tree Approach

Figure 4.11: Number of packets needed to achieve $\beta = (0.85, 0.90, 0.95)$ in a path with 5 malicious nodes. Broadcasted acknowledgments to update the encountered nodes about the delivered messages. First Contact algorithm (FC) [36], achieves the lowest detection accuracy as the message is forwarded to the next hop randomly. The message stays in the buffer and gets forwarded when the node carrying it encounters the first node it meets even when it has zero knowledge about it.

Figure 4.12: Node detection accuracy for different OppNets protocols.
Figure 4.13, shows our detection accuracy based on the mobility types. We have used Epidemic protocol with different mobility types (Trams, Cars and Pedestrians). Trams achieved very good accuracy (between 93% - 100%) due to its features such as a high speed interface and mobility speed. Cars achieve better accuracy (between 88% - 99%) than pedestrians (between 70% - 98%), because cars have a higher mobility speed compared to pedestrians, this results in achieving a better message distribution.

![Figure 4.13: Node detection accuracy based on the mobility of nodes](image)

In order to compare our approach with other works focusing on the same subject, we have chosen the acknowledgement-based mechanisms [1] and the network coding-based mechanism [2]. As comparison metrics, we have measured the node detection accuracy.

In Figures 4.14, 4.15, 4.16, 4.17 and 4.18 we can see other approaches perform very poorly especially when the number of malicious nodes increases to 100% of the intermediate nodes. In acknowledgement [1] and network coding [2] based mechanisms destination nodes need to send an acknowledgment value to the source node through a legitimate path which is difficult/impossible if the number of malicious nodes reach 100%. Malicious nodes can modify the destination acknowledgments values so it can go undetected. Moreover this destination based mechanism can only detect a malicious path (not the malicious nodes) and has a high false positive rate.
Figure 4.14: Comparison of malicious node detection accuracy: Our approach (Epidemic) vs. ACK [1] vs. Network coding techniques [2]

Figure 4.15: Comparison of malicious node detection accuracy: Our approach (PRoPHET) vs. ACK [1] vs. Network coding techniques [2]
Figure 4.16: Comparison of malicious node detection accuracy: Our approach (Spray and Wait) vs. ACK [1] vs. Network coding techniques [2]

Figure 4.17: Comparison of malicious node detection accuracy: Our approach (MaxProp) vs. ACK [1] vs. Network coding techniques [2]
Figure 4.18: Comparison of malicious node detection accuracy: Our approach (First Contact) vs. ACK [1] vs. Network coding techniques [2]

As we can observe from Figure 4.5, we may miss some malicious nodes (t15) when we have more than one malicious node sending to each other as we have no clear trace from each malicious node and we cannot distinguish between the correct and the fake Merkle root value.

In Figure 4.19, we can see a zero false negative rate for our algorithm when we have less than 15% malicious nodes. This is because of the small number of malicious nodes and the probability of having more than one malicious nodes sending to each other will be very low. The false negative rate starts to increase slightly till it reaches a maximum of 8%, when the number of malicious nodes increase, as the probability of having more than one malicious node in the same packet’s path will increase.

As we mentioned in our simulation setting section, we run the stimulation for 1 hour. Figure 4.20 shows the nodes detection accuracy percentage during simulation time (1 hour) with different percentage of malicious nodes. We can achieve 100% detection accuracy at the end of simulation when the percentage of the malicious nodes is less than 10%. The detection accuracy slightly starts to drop when the percentage of malicious nodes increase in the networks. However, this accuracy does not drop below 91% even when 100% of the intermediate nodes act as malicious. The reason behind this is the assumption that the source and destination nodes are
Figure 4.19: False negative rate as the number of malicious nodes increases legitimate so they can run our algorithm. In any stage, when we have a two-hop path the legitimate node or destination node can clearly classify the intermediate node whether it is legitimate or malicious. This is why at the end of simulation we will have clear image about the legitimacy of the network nodes and then get a very high detection accuracy.

Figure 4.20: Nodes detection accuracy percentage vs. Simulation time
4.5 Summary

In this chapter, we present a Merkle tree-based defence mechanism against the Catabolism Attack. Our detection and traceback mechanism in this method is node by node based, where each legitimate node can detect the attack and then traceback to the malicious nodes. Fake packets propagation in the network is mitigated. This mechanism also has a very high accuracy and relies on the construction of the Merkle tree. There are two phases in our technique. The first phase is to detect the attack, and the second phase is to traceback the malicious nodes. We have compared our approach with one of the acknowledgement-based approaches and one of the network coding-based approaches which are well known approaches in the literature. Simulation results show that this mechanism achieves a very high accuracy and detection rate. However, in Merkle tree techniques, legitimate nodes need more process time to hash all the packets and then construct Merkle tree. For example, if we have a message with 4 packets we will need 7 hashes to build the tree in order to obtain the root value. In the next chapter, hash chain techniques will be used to deal with this issue. Legitimate nodes will need only one hash for each packet to build the chain. Any legitimate nodes in the packets’ path can detect the attack, stop fake packets’ propagation and then traceback the malicious nodes.
Chapter 5

Anabolism Defence - Hash Chain Approach

In this chapter, we present a hash chain based defence mechanism against Catabolism Attacks. We also present a node by node traceback mechanism. In this approach, each legitimate node can detect the attack and then traceback to the malicious nodes. Simulation results indicate that we can achieve a very high accuracy and mitigate the propagation of the fake packets in the network. In this defence technique we have two phases. The first phase is to detect the attack, and the second phase is to find the malicious nodes. Our hash chain-based technique addresses the processing overhead with Merkle trees by reducing the processing needed to build the hash chain. We have compared our approach with one of the acknowledgement-based approaches [1] and one of the network coding-based approaches [2] which are well known approaches in the literature. Simulation results show this robust mechanism achieves a very high accuracy and detection rate.

5.1 Introduction

As we mentioned in chapter 3, in a Catabolism attack, malicious nodes selectively drop some packets and instead of them inject new fake packets in order to maintain the original total number of packets originated from the sender node. In chapter 4, Merkle tree techniques are used to enable any legitimate node on the packets’ path to detect the attack, mitigate the propagation of the fake packets and
then traceback to the malicious nodes. However, in Merkle tree techniques, nodes need more processing time as they need to hash all the packets and then construct a Merkle tree. For example, if we have a message with 4 packets then we will need 7 hashes to build the tree and then get the root value. In this chapter hash chain techniques will be used to deal with this issue. Nodes will need less processing time as they will need only 4 hashes for the same message of 4 packets. Moreover any legitimate nodes on the packet’s path can detect the attack, mitigate the propagation of the fake packets and then traceback to the malicious nodes.

**Contribution.** The main contributions of this work are:

1. To identify an efficient and low cost node by node detection and defence mechanism using hash chain technique where the legitimate nodes can check all of the received packets and then accurately detect the attack even when the malicious nodes change the entire content of the packets.

2. To traceback and identify the malicious nodes that triggered the attack.

### 5.2 Overview on the Hash Chain Approach Mechanism

In the hash chain approach, each legitimate node can run the algorithm to detect the attack and traceback the malicious nodes. Propagation of fake packets through the network is prevented as the legitimate nodes can detect and drop fake packets and directly classify node as malicious. Our detection technique is very powerful and accurate. It is relies on the hash chain mechanism. There are two parts in our technique:

1. Detect the attack: When the packets reach any legitimate node, the node can recalculate the hash chain and compare it with the existing chain values that are already included by the sender in each packet.

2. Traceback malicious nodes: Based on the outcome of the first stage, the legitimate nodes can traceback and find the malicious node.
5.2.1 Assumptions

In our approach we make the following two assumptions:

1. The sender node should use the hash chain technique to calculate the hash chain values and then include them with each packet.

2. Malicious nodes have the ability of dropping some legitimate packets and then injecting new fake packets instead with the ability of modifying entire contents of packets including the hash chain values.

3. The sender and the destination nodes are legitimate.

5.2.2 Attack and Defence Scenario

In a Catabolism attack, malicious nodes can drop one or more packets (not all the packets), and instead of them, inject fake packets. Dropping and injecting can be done by one or more malicious nodes along the packets’ path. The legitimate nodes check to detect the attack, and then traceback to the malicious nodes. Figure 5.1 shows the packets’ path of message 1 (c8 → t16 → t15 → w14 → c9), where node t15 drops one packet at time T= 860.2 and then injects a fake packet instead.

![Figure 5.1: Packet Dropping/Injecting at Malicious Node (t15)](image-url)
In our defence technique, we require the sender node to calculate a hash chain based on the packet’s content and then inject the hash value in to the packet’s header. Figure 5.2 shows the construction of the packets with the hash chain values, where the hash chain values are calculated as below:

\[ h_1 = h(P_1 \oplus h_n \oplus P_n \oplus h_2 \oplus P_2) \]
\[ h_2 = h(P_2 \oplus h_1 \oplus P_1 \oplus h_3 \oplus P_3) \]
\[ h_3 = h(P_3 \oplus h_2 \oplus P_2 \oplus h_4 \oplus P_4) \]
\[ h_n = h(P_n \oplus h_{(n-1)} \oplus P_{(n-1)} \oplus h_1 \oplus P_1) \]

In general for \( i \neq 1, n \):
\[ h_i = h(P_i \oplus h_{(i-1)} \oplus P_{(i-1)} \oplus h_{(i+1)} \oplus P_{(i+1)}) \]

Malicious nodes can drop some packets and inject fake packets with existing or recalculated values including the hash chain values. However, when the legitimate nodes recalculate the hash chain again, it will not match the fake or existing one. Legitimate nodes have their own recalculation techniques. They will use the existing hash values rather than calculate totally new hashes.

For example, if a malicious node drops/injects packet two \((P_2)\) it will either leave the hash value \(h_2\) unchanged or recalculate and inject new \(h_2\).

In the case of leaving \(h_2\) unchanged, the legitimate node will calculate and verify the hash values as follows:

i. \( h_1 = \text{existing (legitimate)} P_1 \oplus \text{existing (unchanged)} h_n \oplus \text{existing (legitimate)} P_n \oplus \text{existing (unchanged)} h_2 \oplus \text{existing (fake)} P_2; \) this will result in a new \(h_1\) which does not equal the existing \(h_1\). Therefore, the legitimate node will classify \(P_1\) as the start of the chain break.
ii. \( h_2 = \text{existing (fake)} P_2 \oplus \text{existing (unchanged)} h_1 \oplus \text{existing (legitimate)} P_1 \oplus \text{existing (unchanged)} h_3 \oplus \text{existing (legitimate)} P_3 \); this will result in a new \( h_2 \), which is not equal to the existing \( h_2 \). Therefore, the legitimate node will classify \( P_2 \) as a part of chain break.

iii. \( h_3 = \text{existing (legitimate)} P_3 \oplus \text{existing (unchanged)} h_2 \oplus \text{existing (fake)} P_2 \oplus \text{existing (unchanged)} h_4 \oplus \text{existing (legitimate)} P_4 \); this will result in a new \( h_3 \), which is not equal to the existing \( h_3 \). Therefore, the legitimate node will classify \( P_3 \) as the end of the chain break.

iv. \( h_n = \text{existing (legitimate)} P_n \oplus \text{existing (unchanged)} h_{(n-1)} \oplus \text{existing (legitimate)} P_{(n-1)} \oplus \text{existing (unchanged)} h_1 \oplus \text{existing (legitimate)} P_1 \); this will result in a new \( h_n \), which is equal to the existing \( h_n \). Therefore, the legitimate node will classify \( P_n \) as a normal packet.

In the case of a new recalculated value of \( h_2 \), the legitimate node will calculate the hash values as follows:

i. \( h_1 = \text{existing (legitimate)} P_1 \oplus \text{existing (unchanged)} h_n \oplus \text{existing (legitimate)} P_n \oplus \text{existing (recalculated)} h_2 \oplus \text{existing (fake)} P_2 \); this will result in a new \( h_1 \), which is not equal to the existing \( h_1 \). Therefore, the legitimate node will classify \( P_1 \) as the start of the chain break.

ii. \( h_2 = \text{existing (fake)} P_2 \oplus \text{existing (unchanged)} h_1 \oplus \text{existing (legitimate)} P_1 \oplus \text{existing (unchanged)} h_3 \oplus \text{existing (legitimate)} P_3 \); this will result in a new \( h_2 \), which is equal to the existing \( h_2 \) as malicious node already recalculated \( h_2 \) using same manner. Therefore, the legitimate node will classify \( P_2 \) as a normal packet.

iii. \( h_3 = \text{existing (legitimate)} P_3 \oplus \text{existing (recalculated)} h_2 \oplus \text{existing (fake)} P_2 \oplus \text{existing (unchanged)} h_4 \oplus \text{existing (legitimate)} P_4 \); this will result in a new \( h_3 \), which is not equal to the existing \( h_3 \). Therefore, the legitimate node will classify \( P_3 \) as the end of the chain break.

iv. \( h_n = \text{existing (legitimate)} P_n \oplus \text{existing (unchanged)} h_{(n-1)} \oplus \text{existing (legitimate)} P_{(n-1)} \oplus \text{existing (unchanged)} h_1 \oplus \text{existing (legitimate)} P_1 \); this will result
in a new $h_n$, which is equal to the existing $h_n$. Therefore, the legitimate node will classify $P_n$ as a normal packet.

In our hash chain based defence technique, we can classify a previous node as malicious as long as we receive its packets with one or more chain breaks. However, we may miss some malicious nodes if we have two or more malicious nodes send to each other as in Figure 5.3.

![Figure 5.3: Path with two malicious nodes sending to each other](image)

In our defence, we used hash chain technique where the sender node should use a secure hash algorithm “SHA-1” for the hashing. Sender node includes the hash chain value in each packet as in Figure 5.2. Our technique (algorithm 3) is based on a node by node defence so when all the packets reach other legitimate nodes, these nodes start calculating and comparing the hash chain for each packet again based on the existing chain values. Legitimate node will then check whether this hash chain values are equal to the existing values in each packets. If all values are equal then, the sender node is legitimate and we confirm the absence of an attack. If the values are not equal, then we have a break chain and the sending node will be classified as a malicious node.

Once legitimate nodes detect differences in the hash chain values then it will classify the previous node as malicious directly. In case we have two or more malicious nodes sending to each other, then we will classify only one of them as malicious, i.e., the
Algorithm 3 - Anabolism defence

1: READ: numberOfPacket

2: numberOfPacket = n

3: For all packets

4: READ: $h_i$

5: if $i = 1$ then

6: \[ h'_1 = h(P_1 \oplus h_n \oplus P_n \oplus h_2 \oplus P_2) \]

7: Go to 14

8: end if

9: if $i = n$ then

10: \[ h'_{n} = h(P_n \oplus h_{(n-1)} \oplus P_{(n-1)} \oplus h_1 \oplus P_1) \]

11: Go to 14

12: end if

13: \[ h'_i = h(P_i \oplus h_{(i-1)} \oplus P_{(i-1)} \oplus h_{(i+1)} \oplus P_{(i+1)}) \]

14: if $h'_i = h_i$ then

15: No chain break

16: No malicious nodes, Exit

17: else

18: Chain break

19: Previous node is malicious

20: maliciousNodesCounter++

21: end if
previous node and we may miss other malicious nodes as we have no clear trace from each malicious node. However, as our algorithm is based on node by node detection, we will always have high probability to detect the malicious nodes in another path(s) especially when the percentage of the malicious nodes is not high. In our simulation, we always have enough paths to detect the malicious nodes.

5.3 Mathematical Model

The aim of the mathematical model is to derive a formula for the probability of achieving the sufficient condition leading to malicious node detection. The sufficient condition is: receiving at least 1 fake packet from any malicious node, and receiving at least 1 legitimate packet or at least 1 fake packet from a different malicious node. We first introduce the notation used.

- $n$ be total number of hops;
- $m$ be the number of malicious hops, where $m < n$;
- $k$ be the number of packets;
- $p$ the probability that a packet be attacked at a malicious hop; and
- $A_j$ be the event that exactly $j$ packets survived all malicious hops, where $j < k$.

Let $\beta$ be the probability that there exists at least one fake packet. Obviously,

$$\beta = 1 - \text{there exists no fake packets}$$

Given $p$, the probability of a packet surviving a malicious hop is $q = 1 - p$. Let $\mathbb{P}$ be the probability that a packet will survive all malicious hops, i.e.

$$\mathbb{P} = q^m.$$  

The probability of having all $k$ packets not fake is

$$\mathbb{P}^k = q^{mk},$$
and thus:
\[ \beta = 1 - q^{mk} \implies 1 - \beta = q^{mk}. \]

This means that
\[ \log(1 - \beta) = mk \log q \implies k = \frac{\log(1 - \beta)}{m \log q} = \frac{\log(1 - \beta)}{m \log(1 - p)}. \]

Now, the probability that a packet be attacked at one or more of the hops, \( Q \), is given by:
\[ Q = 1 - P \]

The probability of \( A_j \), that is, exactly \( j \) out of \( k \) packets survived all malicious hops, is:
\[ \Pr(A_j) = \binom{k}{j} P^j Q^{k-j} \]

As we wish to have at least one packet surviving all malicious hops, and at least one packet that does not, (i.e., at most \( k - 1 \) packets survives all malicious hops), the probability of such an event is:
\[ \sum_{j=1}^{k-1} \Pr(A_j) = \sum_{j=1}^{k-1} \binom{k}{j} P^j Q^{k-j} \]

Now, the probability that none of the \( k \) packets survived all \( m \) malicious hops, i.e., all \( k \) packets are fake, is given as: \( Q^k \).

The probability of all \( k \) fake packets were attacked at one particular malicious hop is \((1/m)^k\). As there are \( m \) malicious hops, the probability that all \( k \) fake packets were attacked at the same malicious hop is \( \frac{1}{m^{k-1}} \). Hence, the probability that at least two of the fake packets were attacked at two different malicious hops is given by:
\[ 1 - \frac{1}{m^{k-1}}. \]

Therefore, the probability of none of the \( k \) packets survived all \( m \) malicious hops and that at least two of the fake packets were attacked at two different malicious hops is given by:
\[ Q^k \left( 1 - \frac{1}{m^{k-1}} \right). \]

In summary, the probability that there is at least 1 fake packet from any malicious node, and that there are at least 1 legitimate packet or at least 1 fake packet from
a different malicious node, is given by:

$$\sum_{j=1}^{k-1} \Pr(A_j) = \sum_{j=1}^{k-1} \binom{k}{j} p^j q^{k-j} + q^k \left(1 - \frac{1}{m^{k-1}}\right),$$

which is really just one minus the probability of none survived, and that all $k$ fake packets were attacked at the same malicious node.

Hence, the probability is given by:

$$1 - \frac{q^k}{m^{k-1}} - p^K.$$

Notice that

$$1 = \sum_{j=0}^{k} \binom{k}{j} p^j q^{k-j} = p^0 q^k + \sum_{j=1}^{k-1} \binom{k}{j} p^j q^{k-j} + p^K q^0.$$

As

$$q^k = q^k \left(1 - \frac{1}{m^{k-1}}\right) + \frac{q^k}{m^{k-1}},$$

we have that

$$1 = q^k \left(1 - \frac{1}{m^{k-1}}\right) + \frac{q^k}{m^{k-1}} + \sum_{j=1}^{k-1} \binom{k}{j} p^j q^{k-j} + p^K.$$

The probability that there is at least 1 fake packet from any malicious node, and that there are at least 1 legitimate packet or at least 1 fake packet from a different malicious node, is given by:

$$\sum_{j=1}^{k-1} \binom{k}{j} p^j q^{k-j} + q^k \left(1 - \frac{1}{m^{k-1}}\right) = 1 - \frac{q^k}{m^{k-1}} - p^K.$$

Figures 5.4, 5.5 and 5.6 show the probability of receiving at least one fake packet ($\beta$) in paths with 1, 2 and 5 malicious nodes. When we have a large number of malicious nodes, the probability of detecting the attack or the malicious nodes will be high as we have a good chance of receiving at least one fake packet’s and at least one legitimate packet or another fake packet from a different malicious node. The probability of detecting malicious nodes is also affected by the number of packets sent. When the number of packets sent increases, the probability of receiving fake packets will increase and then enable us to achieve a high detecting probability.

Figures 5.7, 5.8 and 5.9 show the relationship between the number of packets and the probability of at least one packet being changed ($\beta$) across paths with different
Figure 5.4: Probability of receiving at least one fake packet ($\beta$) in a path with 1 malicious node.

Figure 5.5: Probability of receiving at least one fake packet ($\beta$) in a path with 2 malicious nodes.
number of malicious nodes. When the probability $p$ of a packet being changed by a malicious nodes is low, we need to send a large number of packets $k$ so we can achieve a high probability (0.85 - 0.95) of receiving at least one fake packet in order to achieve a high accuracy of detecting a malicious nodes. When the number of malicious nodes is high in the packet path, we do not need to send a high number of packets as we can have a good chance of receiving fake packets from any malicious nodes in the packets path.

Figure 5.6: Probability of receiving at least one fake packet ($\beta$) in a path with 5 malicious nodes
Figure 5.7: Number of packets needed to achieve $\beta = (0.85, 0.90, 0.95)$ in a path with 1 malicious node.

Figure 5.8: Number of packets needed to achieve $\beta = (0.85, 0.90, 0.95)$ in a path with 2 malicious nodes.
5.4 Simulation Settings

To test our algorithm, we implemented a scenario in the ONE simulator [70] using the Epidemic, PRoPHET, Spray and Wait MaxProp and First Contact protocols. The simulation was defined to last for 1 hour, with 0.5 seconds of update intervals. Bluetooth was chosen for connectivity with a transmit range of 10 meters for node radio devices, and transmit speeds of 1000 kbps. There are 35 active nodes composed of cars, trams and pedestrians. Pedestrians and cars have up to 50 MB of RAM for storage. Pedestrians move at random speeds between 1 and 1.5 m/s, cars drive only on roads and move at speeds between 10 – 50 km/h, with wait times of 0 – 120 seconds. MapBasedMovement is used for pedestrians and cars, with a network area of $4500m^2 \times 3400m^2$. Nodes move randomly on roads and walkways with a movement warm-up for 10 seconds. There are 3 groups of trams, with 2 trams in each group. MapRouteMovement is used for trams to follow a constructed tram line. Trams drive at speeds of 7 – 10 m/s with a wait time of 10 – 30 seconds at each configured stop. In addition to the Bluetooth interface, a group of trams use the high speed interface with a transmit range of 1000 meters and a transmit speed of 10mbps. Messages are generated every 1 to 5 seconds per node, with message sizes between 50kb and 100kb.
and a message time to live of 5 hours. We used the simulator’s output as a dataset, and randomly corrupted the dataset based on the number of malicious nodes. We then fed the corrupted dataset to our algorithm. We also ran the simulator for an average of 30 times to represent each point on our graphs. Two programs were written using C++. The first program (total of 353 lines) read the dataset file and then corrupted it by making legitimate nodes malicious by changing the packet creation time for randomly chosen packets and nodes. The second program (total of 816 lines) implements the algorithm, and begins by taking as input the output dataset file generated by program 1. The second program is run to get the algorithm results of the metrics calculations.

5.4.1 Simulation Results and Analysis

We have used two metrics for evaluating our algorithm,

1. Malicious node detection accuracy: The ratio of the total number of detected malicious nodes to the total number of actual malicious nodes.

2. False negative rate: The percentage of malicious nodes that have been incorrectly classified as legitimate nodes.

In our scenario, we have assumed the source nodes are not malicious as these nodes will generate the messages and then hand them to the neighbour nodes. Neighbour nodes can be malicious or legitimate. However, when the legitimate nodes run the algorithm, it will accurately distinguish between the malicious and the legitimate sender.

In Figure 5.10, we can see our algorithm achieved a node detection accuracy of 100% when the percentage of malicious nodes is less than 10%. This is because there is a good chance for a legitimate node to detect hash chain breaks as well as the probability of having more than one malicious node sending to each other in the same packets path will be low. When the number of malicious nodes increases, the accuracy slightly starts to drop as we may have two or more malicious nodes sending to each other as illustrated in Figures 5.3. However, our algorithm results show that the Node detection accuracy does not drop below 91% for Epidemic protocol, 90% for PRoPHET protocol, 89% for Spray and Wait protocol, 88% for MaxProp
protocol and 87% for First Contact protocol, even when 100% of intermediate nodes act as malicious nodes. This is due to the enough short paths where we can detect the malicious nodes directly.

Epidemic [24] and PRoPHET [59] achieved the highest detection accuracy, because both of them replicate multiple copies of a message in the network. In this case, nodes will have good participation in the network and then our algorithm can detect them. Spray and Wait [25] is the lower in terms of detection accuracy, as in this protocol the source node sprays a predefined number of copies to the network, and then nodes do direct delivery to the destination. MaxProp [66] is even lower, as MaxProp has a prioritization schema for both the schedule of packets transmitted and dropped due to the assumption of limited storage and bandwidth and the use of broadcasted acknowledgments to update the encountered nodes about the delivered messages. First Contact algorithm (FC) [36], achieved the lowest detection accuracy as the message is forwarded to the next hop randomly. The message stays in the buffer and gets forwarded when the node carrying it encounters the first node it meets even when it has zero knowledge about it.

Figure 5.10: Nodes detection accuracy for different OppNets protocols

Figure 5.11, shows our detection accuracy based on the mobility types. We have used Epidemic protocol with different mobility types (Trams, Cars and Pedestrians). Trams achieved very good accuracy (between 93% - 100%) due to their features
such as high speed interface and mobility speed. Cars achieved better accuracy (between 88% - 99%) than pedestrians (between 70% - 98%) because, cars have a good opportunity to distribute messages due to their speed feature.

![Figure 5.11: Nodes detection accuracy based on the mobility of nodes](image)

In order to compare our approach with other works focusing on the same subject, we have chosen the acknowledgement-based mechanisms [1] and the network coding-based mechanism [2]. As comparison metrics, we have measured the node detection accuracy. In Figures 5.12, 5.13, 5.14, 5.15 and 5.16, we can see other approaches perform very poorly especially when the number of malicious nodes increase to 100% of the intermediate nodes. In acknowledgement [1] and network coding [2] based mechanisms destination node needs to send acknowledgment values to the source node through a legitimate path which is difficult/impossible if the number of malicious nodes reach 100%. Also, malicious nodes can swap the destination acknowledgment values so it can go undetected. Moreover this destination-based mechanism can detect malicious paths (not the malicious nodes) and has a high false positive rate. As we can see in Figure 5.3, we may miss some malicious nodes (t15) when we have more than one malicious node sending to each other in the same packet’s path as we have no clear trace from each malicious node.

In Figure 5.17, we can see a zero false negative rate for our algorithm when we have less than 15% malicious nodes. This is because of the small number of malicious nodes and the probability of having more than one malicious nodes sending to each
Figure 5.12: Comparison of malicious node detection accuracy: Our approach (Epidemic) vs. ACK [1] vs. Network coding techniques [2]

Figure 5.13: Comparison of malicious node detection accuracy: Our approach (PRoPHET) vs. ACK [1] vs. Network coding techniques [2]
Figure 5.14: Comparison of malicious node detection accuracy: Our approach (Spray and Wait) vs. ACK [1] vs. Network coding techniques [2]

Figure 5.15: Comparison of malicious node detection accuracy: Our approach (MaxProp) vs. ACK [1] vs. Network coding techniques [2]
false negative rate starts to increase slightly till it reaches a maximum of 8%, when the number of malicious nodes increase, as the probability of having more than one malicious nodes in the same packets path will increase.

As we mentioned in our simulation setting section, we run the stimulation for 1 hour. Figure 5.18 shows the nodes detection accuracy percentage during simulation time (1 hour) with different percentage of malicious nodes. We can achieve 100% detection accuracy at the end of simulation when the percentage of the malicious

Figure 5.16: Comparison of malicious node detection accuracy: Our approach (First Contact) vs. ACK [1] vs. Network coding techniques [2]

Figure 5.17: False negative rate as the number of malicious nodes increases
nodes is less than 10%. The detection accuracy slightly starts to drop when the percentage of malicious nodes increase in the networks. However, this accuracy does not drop below 91% even when 100% of the intermediate nodes act as malicious. The reason behind this is the assumption of the source and destination nodes as legitimate so they can run our algorithm. In any stage, when we have a two hop path the legitimate node or destination node can clearly classify the intermediate node whether it is legitimate or malicious. This is why at the end of the simulation we will have a clear image about the legitimacy of the network nodes and observe a very high detection accuracy.

![Figure 5.18: Nodes detection accuracy percentage vs. Simulation time](image)

5.5 Summary

In this chapter, we present a hash chain-based defence mechanism against the Catabolism Attack. Our detection and traceback mechanism at this stage is node by node based, where each legitimate node can detect the attack and then traceback to the malicious nodes. The propagation of the fake packets in the network has been mitigated. This technique has a very high accuracy and relies on the construction of the hash chains. There are two phases in our technique. The first phase is to detect the attack, and the second phase is to traceback to the malicious nodes. Legitimate
nodes spend less processing time for the hash chain technique when compared to the processing time needed to build Merkle trees. We have compared our approach with one of the acknowledgement based approaches and one of the network coding based approaches which are well known approaches in the literature. Any legitimate nodes in the packet’s path can detect the attack, mitigate the fake packets propagation and then traceback to the malicious nodes. Simulation results show this mechanism achieves a very high accuracy and detection rate. In our proposed methods, the time-based approach has the lowest cost but cannot provide security guarantees with strong adversaries. Our Merkle based method will incur the highest cost due to cost of the construction of the Merkle tree making our Hash chain based approach the best suited for resource-constrained networks as it is able to achieve an equivalent accuracy level to the Merkle based method but with a reduced cost. We also have the option in severely resource-constrained networks of employing destination-based detection, which can further lower cost.
Chapter 6

Conclusion and Future Works

This chapter summarizes the main contributions of this thesis “Defence and Traceback Mechanisms in Opportunistic Wireless Networks”, provides an overall conclusion and points out the directions for further research.

6.1 Summary of Thesis

Opportunistic networks or OppNets refer to a number of wireless nodes opportunistically communicating with each other in a form of “Store-Carry-Forward”. This occurs when they come into contact with each other without proper network infrastructure. OppNets use wireless technologies, such as IEEE 802.11, WiMAX, Bluetooth and other short range radio communication. In OppNets, there is no end-to-end connection between the source and the destination nodes and the nodes usually have high mobility, low density, limited power, short radio range, and are often subject to different kinds of attacks by malicious nodes. Due to these characteristics and features, OppNets are subject to serious security challenges. OppNets strongly depend on human interaction, therefore the success of securing such networks is based on trust between people. OppNets aim to establish reliable networks where there is no end-to-end connection between the source and destination node. The nodes in OppNets usually have high mobility, low density, limited power, short radio range, and are often subject to different kinds of attacks by malicious nodes. Due to these characteristics, OppNets have gained significant research attention due to the security and privacy challenges that have emerged. OppNets have emerged from delay
tolerant networks (DTNs) where connectivity is intermittent. The nodes are often disconnected from each other and use Bluetooth, Wi-Fi, or any other wireless connectivity to exchange and forward data in an opportunistic hop by hop manner. In OppNets there is no end-to-end path between a sender and a destination, so the opportunity for forwarding of messages are usually limited, with possibly higher error rates and longer delays. Unlike DTNs, the routing algorithm in OppNets must be opportunistic. In DTNs, when a message is to be sent, an existing end to end route is first investigated; if none is found, the message is then sent opportunistically. However, in OppNets, the message is always sent opportunistically, and an existing end to end path is never required.

In this thesis, we commenced by providing a broad description about the early research and history of opportunistic networks and then an overview of the available OppNets routing protocols and their classification. An evaluation study on six routing protocols (Epidemic, PRoPHET, MaxProp, Spray and Wait, Direct Delivery and First Contact) was presented to understand their performance in terms of complexity, and scalability. Detailed simulation results show that as the load on the network increases, the performance of protocols decrease in terms of delivery delay and network overhead. As for scalability, simulation results show that Epidemic and PRoPHET achieved high delivery rates, but at a very high cost while Spray and Wait achieved lower delivery rates, but with a low cost. First Contact and Direct Delivery achieved low delivery rates with high delivery delays. Results also vary depending on the buffer size, contact times, and speed.

We discussed secure routing and trust management systems and strategies to increase security levels in OppNets where social, reputation and history relationships play important roles in the implementations of these trust strategies. We also discussed secure routing techniques in OppNets and different defence mechanisms against various types of attacks like Blackhole, Wormhole, Dropping, and Sybil attacks. Anti-localization techniques of nodes have attracted the attention of researchers as many routing protocols use the location of nodes as a base for their routing decisions. Selfishness was discussed in this thesis where the node tries to obtain benefits from network facilities and resources but refuses to cooperate with other nodes for reasons such as limitation of resources. We have also discussed the impact of intrusion
detection systems on OppNets and provided an overview of trust in OppNets. However, the main contribution of this thesis is to present a novel attack and traceback mechanism against a special type of packet dropping attack where the malicious nodes selectively drop some packets and instead of them inject fake packets so it can maintain the original total number of packets originated from the sender node. We call this novel attack a Catabolism attack and we call our traceback mechanisms against this attack the Anabolism defence.

In this thesis we have proposed and contributed three approaches

- Time-based approach: In this approach legitimate nodes check all the received packets to distinguish between the fake and legitimate and then traceback the malicious nodes. Our approach in this method relies on the packets creation time of each packet and the contact time between nodes. With this technique we can achieve a high accuracy and we can mitigate the fake packets’ propagation in the networks.

- Merkle tree approach: In the Merkle tree technique any legitimate node can check the received packets to detect the equality of the Merkle root values and then traceback to the malicious nodes. With this technique we can achieve a very high accuracy and we can mitigate the fake packets’ propagation in the networks.

- Hash chain approach: In the the hash chain technique any legitimate node can check the received packets to detect the hash chain break and then traceback to the malicious nodes. With this technique we can achieve a very high detection accuracy and we can mitigate the fake packets’ propagation in the networks. In addition we can reduce the processing overhead associated with the Merkle tree technique.

The PCT and nodes contact time approach achieve a very high accuracy, detection rate and good networks traffic reduction. However, this approach is based on the assumption that a malicious node has the ability of dropping legitimate packets and then injecting fake packets instead of them but has no ability to modify the packet’s contents including the packet’s creation time. This assumption is not always
realistic as the malicious node may have the ability to (re)write the time in order for the injected packets to look like a legitimate packet. Smart malicious nodes can also write times to point to legitimate nodes in the network in order to appear as malicious.

The Merkle tree approach achieves a very high accuracy and detection rate with no security lacks as the legitimate nodes can detect the attack even when one or more malicious nodes have changed the entire content of the packets. However, in this approach nodes need more processing time to construct the Merkle tree.

The hash chain approach addresses the processing time issue. It achieves a very high accuracy and detection rate with no security lacks as the legitimate nodes can detect the attack in few process time even when the malicious nodes change the entire content of the packets.

### 6.2 Discussion

Packet dropping attack is one of the major security threats in OppNets. It can be classified as a denial of service attack (DoS) where the malicious node drops all or some of the packets. This attack is one of the most difficult DoS attacks to detect since neither source node nor the destination node has the knowledge of where or when the packet will be dropped. Packet dropping can degrade the performance of the network and may obstruct the propagation of sensitive data. It is a significant challenge to deal with such an attack since the unreliable wireless communication and resource limitations can result in communication failure and result in the wrong prediction about the presence of a packet dropping attack. Moreover, a node’s resources, such as energy and bandwidth can be the real reasons behind packet dropping. A power shortage or communication failure such as physical damage can make a node unavailable. It may be difficult to recognize whether packets were dropped due to a security attack or for non-security reasons. Dropping packets can lead to an increase in the number of packet retransmissions, transfer time, response time and network overhead. However, there is no doubt about the malicious behaviour if the node drops some legitimate packets and then injects fake packets to replace them so it can maintain the total number of the packets originated from the sender. In this case
the malicious node obviously has enough resources to do this attack. Therefore, the objective of the research was to present this attack and to present our detection and traceback mechanism against it.

We have started with a basic capability of the malicious nodes where it can only drop some legitimate packets and instead of them inject new fake packets. In our defence techniques we have started with the destination node based techniques and then node by node based techniques. Node by node technique achieved better accuracy and detection rate with good network traffic reductions.

Smart malicious nodes not only can drop some legitimate packets and instead of them inject fake packets, they can also change the entire content of the packets, including packets creation time and nodes contact times. Our defence against this scenarios was the use of hash chain or Merkle tree techniques where we can detect any packets dropping or modification and then traceback to the malicious nodes.

### 6.3 Future Work

In this section, the possible research and experimental studies which could be carried out in the future are highlighted.

- Combining all the mechanisms (time, Merkle and hash chain-based) into one resilience mechanism will be one of the directions for future work.

- Developing a new secure mechanism for detecting attacks where all packets are dropped in OppNets is a real challenge, especially when there is no end to end connection, no centralized management and no clear line of defence in opportunistic networks.

- Building solid trust strategies in opportunistic networks where the direct and indirect (reputation) information can be used to detect packet dropping attacks and to provide secure routing in opportunistic networks.

- Designing a new secure routing protocol in opportunistic networks that combines and embeds all these defence techniques to cover the Catabolism attack as well as all packets dropping attack is one of the future works. This would
be an ultimate all in one solution where the protocol manages routing in opportunistic networks and provide security features during the routing process.
Bibliography


166


