Behaviour Enforcement in Ubiquitous Computing Environments

Oscar Fredy Gonzalez Duque

Submitted for the degree of
Doctor of Philosophy
from the
University of Surrey

Centre for Communication Systems Research
Faculty of Electronics and Physical Sciences
University of Surrey
Guildford, Surrey, GU2 7XH, UK

October 2009

Copyright © O.F. Gonzalez Duque 2009
Abstract

Ubiquitous computing environments (UCEs) are characterised by a myriad of heterogeneous devices fitted with computing and communication capabilities – many of them embedded in physical objects and usually imperceptible – whose main goal is to cooperate in a coordinated manner to supply users with a pool of services that facilitate their tasks. Communication among the participating devices is predominantly performed through the wireless medium. UCEs can also be partially or completely reliant on infrastructureless wireless multihop communication, which provides them with full network connectivity, great flexibility and a versatile dynamic network topology that can be desirable in many situations.

However, UCEs' wireless (often multihop) nature also makes them inherently susceptible to having their communication operation disrupted due to their dependence on the cooperative packet forwarding behaviour of each individual node. For example, misbehaving nodes can cause general network disruption by not forwarding any or only forwarding some packets on behalf of other nodes in the network. Consequently, protection of the data forwarding functionality against malicious or otherwise defective nodes is an important characteristic that UCEs must support.

This thesis proposes an adaptable protection scheme that can detect, accuse and penalise misbehaving nodes that disrupt the communication capabilities of a UCE by dropping data packets that they are expected to forward on behalf of their peer network nodes. This thesis presents different aspects of the design and implementation of the protection scheme. Among the most important aspects are: i) collection of information for behaviour evaluation, ii) accurate detection of misbehaving nodes, iii) accurate and effective accusation of nodes persistently misbehaving, iv) adaptability through network management policies, v) network clustering and a role-based organisational model, vi) resilience to nodes that report false metrics, and vii) resilience to colluding nodes. Concepts developed in this work are illustrated in the context of mobile ad hoc networks (MANETs) since they have emerged as an appropriate paradigm to enable the deployment of UCE technologies. The efficiency and effectiveness of each aspect of the protection scheme are evaluated and demonstrated through extensive simulations.
Dedicated to my Wife, my Mother, my Sister and the Living Memory of my Father.

Dedicada a mi Esposa, mi Madre, mi Hermana y a la Memoria Viva de mi Padre.
Acknowledgements

I would like to express my deepest gratitude to my supervisors Dr. Michael Howarth and Prof. George Pavlou for their inestimable supervision and guidance on every step of the journey towards this doctoral thesis. Their invaluable knowledge and experience in addition to their unconditional help have made possible the realisation of this work. Finally, I would like to thank them for their always available friendship and their willingness to support me when, due to difficulties in my personal life, my doctoral research was delayed.

Huge thanks to my wife Carolina, my mother Irma, and my sister Irma Rocio. I will never be able to thank enough all the love, support and encouragement they have offered me every day since the start of this doctorate. A special mention goes to my father Jose Oscar who originally encouraged me and supported me to come and study in England. Somehow he always knew I would end up doing a PhD.

I would like to thank my colleagues and friends Dr. Apostolos Malatras, Dr. Aimilios Chourouziadis and Dr. Antonis Hadjiantonis because through their input and cooperation they have made possible the completion of this thesis.

Last but not least, I would like to thank my friends Dr. Tasos Karousos, Dr. Symeon Chatzinotas, Dr. Stylianos Georgoulas, Dr. Marinos Charalambides, Paris Skoutaridis, Ioannis Papanagiotou, Adnan Nadeem and Yu Guo, who in addition to my three friends above were a fundamental part of the friendly and enjoyable environment that I had the pleasure to experience at University of Surrey.
# Table of Contents

Abstract ..................................................................................................................................... ii  
Acknowledgements .................................................................................................................. iv  
Table of Contents ...................................................................................................................... v  
List of Figures ....................................................................................................................... viii  
List of Tables ........................................................................................................................... xi  
Acronyms ................................................................................................................................ xii  
Publications ............................................................................................................................ xvi  

1 Introduction ....................................................................................................................... 1  
  1.1 Research Motivation ............................................................................................................... 2  
  1.2 Thesis Contributions ............................................................................................................... 3  
  1.3 Thesis Structure ..................................................................................................................... 5  

2 Background and Related Work ......................................................................................... 7  
  2.1 Ubiquitous Computing Environments (UCEs) ..................................................................... 7  
  2.1.1 UCE Challenges ................................................................................................................. 9  
  2.2 Autonomic Computing Systems (ACSs) ............................................................................. 14  
  2.2.1 Self-Management Properties of an Autonomic Computing System ........................... 17  
  2.2.2 Towards Fully Autonomic Computing Systems ........................................................... 18  
  2.3 Mobile Ad Hoc Networks (MANETs)................................................................................ 22  
  2.3.1 MANETs and the Proposed Protection Scheme............................................................ 23  
  2.3.2 Route Discovery and Packet Forwarding ...................................................................... 23  
  2.4 Network Security ................................................................................................................. 25  
  2.4.1 Basic Security Services ................................................................................................... 25  
  2.4.2 Security Attacks ............................................................................................................... 27  
  2.5 Misbehaviour in MANETs ................................................................................................... 28  
  2.5.1 Misbehaviour, Intrusion and Trust ................................................................................. 28
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5.2</td>
<td>Route Discovery and Packet Forwarding Protection</td>
<td>30</td>
</tr>
<tr>
<td>2.5.3</td>
<td>Anomalous Link Behaviour Detection</td>
<td>31</td>
</tr>
<tr>
<td>2.5.4</td>
<td>Intrusion and Node Misbehaviour Detection</td>
<td>34</td>
</tr>
<tr>
<td>3</td>
<td>Packet Forwarding Misbehaviour Detection and Node Accusation</td>
<td>38</td>
</tr>
<tr>
<td>3.1</td>
<td>Model Assumptions, Terminology &amp; Addressed Problems</td>
<td>39</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Model Assumptions</td>
<td>39</td>
</tr>
<tr>
<td>3.1.2</td>
<td>Terminology</td>
<td>40</td>
</tr>
<tr>
<td>3.1.3</td>
<td>Addressed Problems</td>
<td>41</td>
</tr>
<tr>
<td>3.2</td>
<td>Packet Forwarding Misbehaviour Detection</td>
<td>44</td>
</tr>
<tr>
<td>3.2.1</td>
<td>The Principle of Flow Conservation</td>
<td>44</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Adapting the Principle of Flow Conservation to Multihop Wireless UCEs</td>
<td>47</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Evaluation Results and Analysis</td>
<td>52</td>
</tr>
<tr>
<td>3.3</td>
<td>Accusation of Misbehaving Nodes</td>
<td>59</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Node Accusation</td>
<td>59</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Authenticity of Detection Alert and Accusation Packets</td>
<td>63</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Evaluation Results and Analysis</td>
<td>64</td>
</tr>
<tr>
<td>3.4</td>
<td>Summary</td>
<td>68</td>
</tr>
<tr>
<td>4</td>
<td>Policy-Based Adaptable Misbehaviour Detection and Node Accusation</td>
<td>70</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction to Policies for Network Management</td>
<td>71</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Policy Core Information Model (PCIM)</td>
<td>71</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Policy Types</td>
<td>73</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Mobile Ad Hoc Network Management – Related Work</td>
<td>75</td>
</tr>
<tr>
<td>4.2</td>
<td>Adaptable Misbehaviour Detection and Node Accusation</td>
<td>76</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Clustering Phase</td>
<td>77</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Detection Phase</td>
<td>84</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Accusation Phase</td>
<td>88</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Adaptability and Management Policies</td>
<td>93</td>
</tr>
<tr>
<td>4.3</td>
<td>Case Study, Evaluation Results and Analysis</td>
<td>96</td>
</tr>
</tbody>
</table>
Table of Contents

4.3.1 Case Study Scenario ................................................................. 96
4.3.2 Evaluation Results and Analysis .............................................. 99
4.4 Summary .................................................................................... 107

5 A Collusion-Resistant Misbehaviour Detection and Node Accusation Approach... 108
5.1 Introduction to Service Discovery Strategies ................................. 109
5.1.1 Mainstream Service Discovery Strategies ............................... 110
5.1.2 Service Discovery Protocols for MANETs ............................. 112
5.2 Home-Zone Based Service Discovery Mechanism ....................... 114
5.2.1 Clustering and Service Discovery ........................................... 114
5.2.2 Evaluation Results and Analysis .......................................... 119
5.3 Collusion and Liar Resistant Packet Forwarding Misbehaviour Detection ................................. 124
5.3.1 Improved Scalability: Clustering and Election of Cluster Heads 125
5.3.2 Dealing with liars: Discrepancy-based Detection and Accusation 129
5.3.3 Addressing Collusion .............................................................. 134
5.3.4 Network Nodes Authenticity and Metrics Integrity ................. 136
5.3.5 Evaluation Results and Analysis .......................................... 137
5.4 Summary .................................................................................... 145

6 Summary, Conclusions and Future Work ...................................... 147
6.1 Summary .................................................................................... 147
6.2 Conclusions .............................................................................. 150
6.3 Future Work and Open Issues .................................................. 151
References ...................................................................................... 153
List of Figures

Figure 2-1 Ubiquitous computing environments ................................................................. 9
Figure 2-2 Autonomic systems’ evolution [34] .................................................................... 19
Figure 2-3 Structure of an autonomic element [35] ............................................................ 20
Figure 3-1 Black-hole attack .............................................................................................. 41
Figure 3-2 Grey-hole attacks ............................................................................................. 42
Figure 3-3 Ambiguous collision problem (hidden terminal problem) ............................. 42
Figure 3-4 Receiver collision problem .............................................................................. 43
Figure 3-5 A node capable of controlling its transmission power ............................................. 43
Figure 3-6 Receiver collision in SCAN .............................................................................. 43
Figure 3-7 Rewarding of a misbehaving node .................................................................. 44
Figure 3-8 Pseudocode of our proposed detection algorithm ............................................. 48
Figure 3-9 Overhearing – nodes overhear node v7 as it changes position ......................... 49
Figure 3-10 Limited Broadcast .......................................................................................... 50
Figure 3-11 Percentage of detections vs. misbehaviour threshold (20 node network) ........... 53
Figure 3-12 Percentage of detections vs. misbehaviour threshold (60 node network) ........... 54
Figure 3-13 Percentage of Detections for non-misbehaving networks with 20 and 60 nodes. 55
Figure 3-14 Percentage of detections vs. mean node speed (20 node network) ................. 56
Figure 3-15 Percentage of detections vs. mean node speed (60 node network) ................. 57
Figure 3-16 Network overhead vs. mean node speed (20 node network) .......................... 58
Figure 3-17 Mean number of packets per check vs. mean node speed (20 node network) ... 58
Figure 3-18 Pseudocode of our proposed accusation algorithm ........................................ 60
Figure 3-19 A clever misbehaving node that avoids accusation ....................................... 61
Figure 3-20 Average network throughput vs. mean node speed (20 node network) ............ 65
Figure 3-21 Average network throughput vs. mean node speed (40 node network) ............ 65
Figure 3-22 Average network throughput vs. mean node speed (60 node network) ............ 66
List of Figures

Figure 3-23 Average network throughput vs. mean node speed (120 node network) ................. 67
Figure 3-24 Contribution to total network traffic vs. mean node speed (40 nodes).................... 68
Figure 4-1 Policy core information model (PCIM) architecture ................................................. 72
Figure 4-2 Clustering process – MANET before marking round .................................................. 79
Figure 4-3 Clustering process – MANET after marking round ..................................................... 79
Figure 4-4 Clustering process – MANET after optimisation rule 1 is applied to $G'$ ..................... 80
Figure 4-5 Clustering process – MANET after optimisation rule 2 is applied to $G'$ ................. 80
Figure 4-6 CDS size vs. number of nodes (node density = $6.25 \times 10^4$ nodes/m²) ....................... 82
Figure 4-7 CDS size vs. number of nodes (network area = 600m x 600m) .................................. 83
Figure 4-8 Probability of detection vs. misbehaviour threshold for different behaviours .......... 86
Figure 4-9 Probability of detection vs. misbehaviour threshold (30% packets drop) .................. 87
Figure 4-10 Probability of accusation ($P_A$) vs. behaviour checks ($ch$) with $P_D = 80\%$ ......... 89
Figure 4-11 Probability of accusation ($P_A$) vs. behaviour checks ($ch$) with $P_D = 60\%$ .......... 90
Figure 4-12 Probability of accusation ($P_A$) vs. behaviour checks ($ch$) with $P_D = 10\%$ ......... 91
Figure 4-13 Probability of detections vs. misbehaviour threshold (60 node network) ............. 94
Figure 4-14 Policy types for adaptable detection and accusation ............................................... 94
Figure 4-15 Average network throughput vs. simulation time (40 nodes) ............................... 100
Figure 4-16 Average network throughput vs. simulation time (60 nodes) ............................... 101
Figure 4-17 Average network throughput vs. simulation time (120 nodes) ............................. 101
Figure 4-18 Average network throughput vs. simulation time (60 nodes) ............................... 102
Figure 4-19 Misbehaviour threshold vs. simulation time (60 node network) ............................. 103
Figure 4-20 Percentage of misbehaving nodes being isolated vs. simulation time ................. 104
Figure 4-21 Percentage of misbehaving nodes isolated vs. simulation time (60 nodes) ........... 105
Figure 4-22 Contribution to total network traffic vs. mean node speed (120 nodes) ............... 106
Figure 5-1 Home-zone based service discovery approach .......................................................... 116
Figure 5-2 Average network control overhead vs. number of nodes in the network ............... 120
Figure 5-3 Average service discovery latency vs. number of nodes in the network .................. 121
Figure 5-4 Average network control overhead vs. mean node speed (300 node network) ...... 122
Figure 5-5  Average service discovery latency vs. mean node speed (300 node network)........ 122
Figure 5-6  Network control overhead vs. service providers percentage (100 node network).. 123
Figure 5-7  Service discovery latency vs. service providers percentage (100 node network)... 124
Figure 5-8  Clustering – Example of inefficient CHs selection........................................... 126
Figure 5-9  Virtual clustering with cluster head (CH) selection............................................ 127
Figure 5-10 Multihop packet transmission between two network nodes.............................. 130
Figure 5-11 Matrices corresponding to the multihop packet transmission of Figure 5-10 ....... 130
Figure 5-12 Multihop network transmission with a liar that drops packets............................ 132
Figure 5-13 Matrices at MNs when there is a liar dropping packets in the network ................. 132
Figure 5-14 Matrices at MNs when there is a non-lying node in the network.......................... 134
Figure 5-15 Colluding nodes lying in their metrics about node $v_2$........................................ 135
Figure 5-16 Colluding nodes in an attempt to not be detected............................................. 135
Figure 5-17 Average network throughput vs. simulation time (500 node network)............ 138
Figure 5-18 Instant Network Misbehaviour vs. simulation time (500 node network)........... 139
Figure 5-19 Accused nodes, throughput and network misbehaviour vs. simulation time ....... 140
Figure 5-20 Average network throughput vs. simulation time (320 node network)............. 142
Figure 5-21 Instant Network Misbehaviour vs. simulation time (320 node network)............. 142
Figure 5-22 Percentage of accused/isolated nodes vs. simulation time (320 node network).... 143
Figure 5-23 Average time that nodes spend as CHs vs. number of network nodes ............. 144
Figure 5-24 Number of times that network nodes are CHs vs. number of network nodes ...... 145
List of Tables

Table 4-1 Adequate Values for the Misbehaviour Threshold Offset ............................................. 88
Table 4-2 Management Policies for Adaptive Misbehaviour Detection - SRP Group ....................... 97
Table 4-3 Management Policies for Adaptive Misbehaviour Detection - PP Group ..................... 98
Table 4-4 Management Policies for Adaptive Misbehaviour Detection - EP Group ..................... 98
### Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK</td>
<td>Acknowledgement</td>
</tr>
<tr>
<td>ACS</td>
<td>Autonomic Computing System</td>
</tr>
<tr>
<td>AME</td>
<td>Application Manager Editor</td>
</tr>
<tr>
<td>AMS</td>
<td>Autonomic Middleware Service</td>
</tr>
<tr>
<td>ANMP</td>
<td>Ad Hoc Network Management Protocol</td>
</tr>
<tr>
<td>AODV</td>
<td>Ad Hoc On-Demand Distance Vector</td>
</tr>
<tr>
<td>AP</td>
<td>Accusation Packet</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ARAN</td>
<td>Authenticated Routing for Ad Hoc Networks</td>
</tr>
<tr>
<td>BBM</td>
<td>Backbone Management</td>
</tr>
<tr>
<td>CA</td>
<td>Certification Authority</td>
</tr>
<tr>
<td>CBR</td>
<td>Constant Bit Rate</td>
</tr>
<tr>
<td>CDR</td>
<td>Conflict Detection and Resolution</td>
</tr>
<tr>
<td>CDS</td>
<td>Connected Dominating Set</td>
</tr>
<tr>
<td>CH</td>
<td>Cluster Head</td>
</tr>
<tr>
<td>CN</td>
<td>Cluster Node</td>
</tr>
<tr>
<td>COPS</td>
<td>Common Open Policy Service</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CSCW</td>
<td>Computer Supported Cooperative Work</td>
</tr>
<tr>
<td>DAP</td>
<td>Detection Alert Packet</td>
</tr>
<tr>
<td>DCF</td>
<td>Distributed Coordination Function</td>
</tr>
<tr>
<td>DHCP</td>
<td>Dynamic Host Configuration Protocol</td>
</tr>
<tr>
<td>DMTF</td>
<td>Distributed Management Task Force</td>
</tr>
<tr>
<td>DoS</td>
<td>Denial of Service</td>
</tr>
<tr>
<td>DPA</td>
<td>Domain Policy Agent</td>
</tr>
<tr>
<td>Acronyms</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>DSD</td>
<td>Distributed Service Discovery</td>
</tr>
<tr>
<td>DSDV</td>
<td>Destination-Sequenced Distance-Vector</td>
</tr>
<tr>
<td>DSR</td>
<td>Dynamic Source Routing</td>
</tr>
<tr>
<td>ECA</td>
<td>Event-Condition-Action</td>
</tr>
<tr>
<td>GPA</td>
<td>Global Policy Agent</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>GSM</td>
<td>Global System for Mobile Communications</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HZGC</td>
<td>Home-Zone based Geo-Casting</td>
</tr>
<tr>
<td>HZSD</td>
<td>Home-Zone based Service Discovery</td>
</tr>
<tr>
<td>IDS</td>
<td>Intrusion Detection System</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>ITU</td>
<td>International Telecommunication Union</td>
</tr>
<tr>
<td>ITU-T</td>
<td>ITU – Telecommunication Standardisation Sector</td>
</tr>
<tr>
<td>LDAP</td>
<td>Lightweight Directory Access Protocol</td>
</tr>
<tr>
<td>LMR</td>
<td>Lightweight Mobile Routing</td>
</tr>
<tr>
<td>LPA</td>
<td>Local Policy Agent</td>
</tr>
<tr>
<td>LREP</td>
<td>Location Reply</td>
</tr>
<tr>
<td>LREQ</td>
<td>Location Request</td>
</tr>
<tr>
<td>MAC</td>
<td>Medium Access Control</td>
</tr>
<tr>
<td>MACA</td>
<td>Multiple Access with Collision Avoidance</td>
</tr>
<tr>
<td>MACAW</td>
<td>MACA for Wireless LANs</td>
</tr>
<tr>
<td>MANET</td>
<td>Mobile Ad Hoc Network</td>
</tr>
<tr>
<td>MARS</td>
<td>Multipath Routing Single Path Transmission</td>
</tr>
<tr>
<td>MAS</td>
<td>Mobile Agent System</td>
</tr>
<tr>
<td>MN</td>
<td>Manager Node</td>
</tr>
<tr>
<td>MREP</td>
<td>Metrics Reply</td>
</tr>
<tr>
<td>Acronyms</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>MREQ</td>
<td>Metrics Request</td>
</tr>
<tr>
<td>NWS</td>
<td>Neighbour Watch System</td>
</tr>
<tr>
<td>OLSR</td>
<td>Optimised Link State Routing</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>OSI</td>
<td>Open Systems Interconnection</td>
</tr>
<tr>
<td>OWL</td>
<td>Web Ontology Language</td>
</tr>
<tr>
<td>PAL</td>
<td>Personal Audio Loop</td>
</tr>
<tr>
<td>PBM</td>
<td>Policy Based Management</td>
</tr>
<tr>
<td>PBNM</td>
<td>Policy Based Network Management</td>
</tr>
<tr>
<td>PCIM</td>
<td>Policy Core Information Model</td>
</tr>
<tr>
<td>PDP</td>
<td>Policy Decision Point</td>
</tr>
<tr>
<td>PEP</td>
<td>Policy Enforcement Point</td>
</tr>
<tr>
<td>PG</td>
<td>Policy Group</td>
</tr>
<tr>
<td>PMT</td>
<td>Policy Management Tool</td>
</tr>
<tr>
<td>PR</td>
<td>Policy Repository</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>RAID</td>
<td>Redundant Array of Independent Disks</td>
</tr>
<tr>
<td>RFC</td>
<td>Request For Comments</td>
</tr>
<tr>
<td>RFID</td>
<td>Radio Frequency Identification</td>
</tr>
<tr>
<td>RREP</td>
<td>Route Reply</td>
</tr>
<tr>
<td>RREQ</td>
<td>Route Request</td>
</tr>
<tr>
<td>SAODV</td>
<td>Secure AODV</td>
</tr>
<tr>
<td>SCAN</td>
<td>Self-organised Network Security for Mobile Ad Hoc Networks</td>
</tr>
<tr>
<td>SDP</td>
<td>Service Discovery Protocol</td>
</tr>
<tr>
<td>SEAD</td>
<td>Secure Efficient Distance Vector</td>
</tr>
<tr>
<td>SLM</td>
<td>Salutation Manager</td>
</tr>
<tr>
<td>SLP</td>
<td>Service Location Protocol</td>
</tr>
<tr>
<td>SMT</td>
<td>Secure Message Transmission</td>
</tr>
<tr>
<td>Acronyms</td>
<td>Definition</td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
</tr>
<tr>
<td>SNMP</td>
<td>Simple Network Management Protocol</td>
</tr>
<tr>
<td>SOHO</td>
<td>Small Office Home Office</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>SRP</td>
<td>Secure Routing Protocol</td>
</tr>
<tr>
<td>SSP</td>
<td>Secure Single Path</td>
</tr>
<tr>
<td>SWRL</td>
<td>Semantic Web Rule Language</td>
</tr>
<tr>
<td>TCP</td>
<td>Transport Control Protocol</td>
</tr>
<tr>
<td>TESLA</td>
<td>Timed Efficient Stream Loss-Tolerant Authentication</td>
</tr>
<tr>
<td>TORA</td>
<td>Temporally-Ordered Routing Algorithm</td>
</tr>
<tr>
<td>TTL</td>
<td>Time To Live</td>
</tr>
<tr>
<td>UCE</td>
<td>Ubiquitous Computing Environment</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>UKCRC</td>
<td>United Kingdom Computing Research Committee</td>
</tr>
<tr>
<td>UMTS</td>
<td>Universal Mobile Telecommunications System</td>
</tr>
<tr>
<td>UOWN</td>
<td>User Oriented Wireless Network</td>
</tr>
<tr>
<td>UPnP</td>
<td>Universal Plug and Play</td>
</tr>
<tr>
<td>VANET</td>
<td>Vehicular Ad Hoc Network</td>
</tr>
<tr>
<td>VAP</td>
<td>Virtual Access Point</td>
</tr>
<tr>
<td>WMN</td>
<td>Wireless Mesh Network</td>
</tr>
<tr>
<td>WRP</td>
<td>Wireless Routing Protocol</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
Publications

Journal Papers


Conference Papers


Chapter 1

1 Introduction

Wireless communications have become a worldwide spread technology that has extended computing networks to environments that were previously unreachable or impractical to reach (e.g. satellite networks) through wired technologies. Wireless communications’ ubiquity has its roots in the advantages offered by the wireless medium. Mobility, spontaneity, flexibility, simplicity, cable-free environments and cost-effective deployment are a few of its desirable characteristics. Currently the average person makes use on a daily basis of many technologies that take advantage of wireless communications: mobile phone networks, the Wi-Fi networks of universities and coffee shops, television and radio media, satellite TV and telephony, GPS-based positioning systems, and Bluetooth peripherals that connect automatically to computers and mobile handsets are just a few examples. On the other hand, wireless communications have also facilitated the emergence of many other paradigms which capitalise on the strengths of the wireless medium, some examples are: wireless mesh networks (WMN) [1], vehicular ad hoc networks (VANETs) [2], mobile ad hoc networks (MANETs) [3] and ubiquitous computing environments [4]. These last two areas represent the research fields where the work described in this thesis finds its application.

Ubiquitous computing environments (UCEs) are networks consisting of a myriad of heterogeneous devices – often subtly camouflaged in their surroundings – fitted with computing and communication capabilities which cooperate in a coordinated fashion to deliver a suite of services to their users in an effort to facilitate the accomplishment of their tasks. Although communication between the participants of a UCE is predominantly performed through the wireless medium, which contributes towards its devices’ subtlety, UCEs make no presumptions regarding the communication technologies that can be employed for their deployment. Thus, wireless and wired technologies can both coexist in a UCE and can be employed to provide communication among participating nodes. However, in spite of the fact that a UCE can make use of existing infrastructure (e.g. base stations and wired backbones), an important characteristic of UCEs is that they can be partially or completely independent of such network infrastructure, and instead they can rely on wireless multihop communication such as MANETs for their communication process.
Chapter 1. Introduction

One significant drawback of UCEs is that their predominantly wireless nature renders them vulnerable to a wide variety of attacks by misbehaving nodes, especially in those areas where the network does not have any infrastructure on which to rely. Such attacks range from passive eavesdropping, where nodes try to obtain unauthorised access to information destined for another node, to active interference where malicious nodes attempt to hinder network performance by not obeying globally acceptable rules. For instance, UCEs whose communication process relies on intermediate nodes for packet forwarding are susceptible of having their performance disrupted by misbehaving nodes that fail to forward data packets on behalf of other participating nodes. Nevertheless, when a node exhibits such malicious behaviour is not always because it intends to do so. A node may also misbehave because it is broken, overloaded, compromised or congested, in addition to intentionally being selfish or malicious, as explained in Section 2.5.1.

1.1 Research Motivation

The motivation for the research presented in this thesis emanates from the increasing need to make ubiquitous computing environments reliable and safe as they gradually become a reality that permeates every aspect of our daily lives. In fact mobile phones and the countless services that they offer are a manifestation of the pervasiveness that UCEs can achieve. As more sensors, devices and services are made available in our surroundings the need for reliable communication grows evident. Users are generally impatient and do not like it when they have to resend a document to a wireless printer because their previous attempt failed due to communication or other problems (potentially due to misbehaving nodes). Similarly, when a super-service is composed out of a set of smaller services, it often assumes that communication between its parts is readily available, reliable and safe. An attack that compromises the communication capabilities of an environment could render its service(s) useless, especially if the network does not have in place the necessary security measures to mitigate the effects of the attack and prevent the attacker from disrupting the communication performance again. Protecting the communication process is especially important in UCEs where nodes can leave and enter the network in an ad hoc manner. Although the ad hoc characteristic of a network endows it with a great degree of flexibility, it also allows for the inclusion of untrustworthy nodes which seek to be part of the environment with the unique goal of disrupting its overall functionality.

It is the view of this thesis that previous work addressing the problem posed by misbehaving nodes that disrupt the functionality of routing protocols has failed to propose an approach that can effectively and efficiently protect a network's data packet forwarding capabilities. Many of the existing solutions offer protection strategies for the route discovery phase of routing protocols and ignore the importance of the data forwarding phase. Some other work has focused on identifying
link misbehaviour and proposes tactics to avoid sending packets over the anomalous links, or alternatively they propose sending redundant information over different and independent paths to improve the chances of a packet reaching its destination. An important drawback of this type of approaches is that they make no attempt to pinpoint and remove the source of the problem, which leaves other network participants open to its attacks. Finally, there are also approaches that aim at locating the source of misbehaviour and removing it from the network. However, their strategies to collect behaviour metrics rely on promiscuous listening, which affects their accuracy since nodes collecting the metrics are not the actual nodes sending and receiving the data packets and therefore can not establish with certainty what is happening in the transmission process.

In our opinion, all these existing shortcomings in previous approaches suggest that there is room for research and improvement in the field of multihop wireless communications and their protection against nodes capable of launching active attacks on their data transmission capabilities. In this context, the general objective of this thesis is to propose a novel and adaptable protection scheme that can detect, accuse and penalise misbehaving nodes aiming at disrupting the communication capabilities of a ubiquitous computing environment by dropping data packets that they are expected to forward on behalf of their peer nodes.

1.2 Thesis Contributions

The contribution of this thesis focuses on the proposal of an adaptable protection scheme that makes use of the principle of flow conservation (Chapter 3) to evaluate the behaviour of network nodes by comparing the number of packets they must forward with the number of packets they actually forward. This thesis contributes to different aspects of the design and implementation of the proposed protection scheme as outlined below:

a. **Collection of packet forwarding metrics for misbehaviour detection**

   The basic algorithm to collect packet forwarding metrics evolves through the chapters of this thesis. It is first based on a limited broadcast strategy that identifies nodes that have been in direct contact with the node whose behaviour is under evaluation. This strategy allows a local neighbourhood to detect and eventually accuse a node of misbehaviour. However, it also gives nodes a small chance to evade the security measures in place if they constantly change neighbourhoods in a clever manner (this is not easy to do). This limited broadcast strategy is later replaced by a hierarchical organisational model where nodes at the top tier are in charge of detecting and accusing misbehaving nodes. This strategy generates lower overhead and solves the problem posed by clever nodes that constantly change neighbourhoods.
b. **Accurate detection of misbehaving nodes**

In the proposed approach the metrics employed to evaluate a node's behaviour are collected from nodes which have actually sent and received packets from the evaluated node. This technique eliminates inaccuracies exhibited by previous approaches and allows the protection scheme to discriminate adequately misbehaving nodes from well-behaved nodes, as shown in our simulation results.

c. **Effective accusation and penalisation of nodes that persistently misbehave**

Accusing a node of misbehaviour based on a single misbehaviour detection can result in the probability of wrongly accusing well-behaved nodes to be very high. For this reason, the proposed protection scheme only accuses and penalises nodes that are persistently detected as misbehaving. This method provides a high probability of accusing misbehaving nodes while maintaining a low probability of wrongly accusing well-behaved ones.

d. **Adaptability**

Adaptability is brought to the protection scheme by means of a policy based management (PBM) framework which is implemented in the UCE through a hierarchical organisational model consisting of three tiers. Nodes at the top layer, i.e. Manager Nodes (MNs), offer an interface to network administrators to allow them to define the high-level policies that drive our protection scheme. Nodes in the middle layer, i.e. Cluster Heads (CHs), distribute and enforce the management policies on other network nodes and themselves. Finally, nodes at the bottom layer, i.e. Cluster Nodes (CNs), are typical network users who must respect the network policies in order to gain access to the services offered by the UCE.

e. **Clustering**

Selecting the role that a node should play in the network is an important part for the correct functioning of the protection scheme. Manager nodes and cluster heads should be nodes having certain properties, such as good processing power, good memory capacity and low mobility. Cluster nodes, on the other hand, can be devices of limited resources only capable of executing simple tasks. In this thesis we present and evaluate two different clustering techniques. The first method is a well-known clustering algorithm that allows us to verify the correctness of our developed concepts. The second method is an efficient clustering approach based on the geographical location of the network nodes that endows the proposed protection scheme with the ability to perform well in large-scale environments.
f. Tackling lying nodes

In the proposed approach there is the possibility that nodes may misbehave by reporting false metrics about other network nodes. We address this problem by allowing the evaluated nodes to tell “their own side of the story”, i.e. to report metrics about their own behaviour. Comparisons between the metrics reported by different network nodes allow for the detection and accusation of nodes that report false metrics in an attempt to incriminate other nodes or hide their own misbehaviour (i.e. how many packets they have failed to forward).

g. Tackling colluding nodes

A common problem in misbehaviour detection systems is how to tackle two or more colluding nodes that work in a cooperative manner to frame a well-behaved node for misbehaviour. In this thesis we develop the concept of discrepancies which allows two nodes to report metrics that contradict each other. Successive discrepancies between the same pair of nodes do not count towards their accusation. Thus, by varying the number of different discrepancies required to accuse a node of misbehaviour our system actually changes the number of colluding nodes that it can accommodate.

1.3 Thesis Structure

The rest of this thesis is organised as follows:

Chapter 2 provides a general introduction and literature review for various areas that have contributed ideas to develop this thesis work. In particular, the Chapter starts by offering an introduction to ubiquitous computing environments (UCEs) and guidelines for their deployment. It then focuses on Autonomic Computing Systems (ACSs) as a means to address the complexity of UCEs and discusses how adaptability can be achieved through its self-management features. Next, MANETs are introduced as a paradigm that can enable the development of UCEs and provides an insight into the benefits of designing and testing approaches in MANET-like environments. The Chapter ends with a general introduction to security systems and their characteristics, followed by an analysis of previous approaches addressing misbehaviour in wireless multihop environments.

Chapter 3 introduces our packet forwarding detection and accusation algorithm. It first presents the network model, assumptions and terminology used, and the specific issues addressed by our protection scheme. It continues with a formal definition of the principle of flow conservation and how it can be adapted to MANET-like environments in order to track down and detect misbehaving nodes. This Section finishes with the evaluation results and analysis for the misbehaviour detection scheme. The second half of this Chapter describes how nodes in a
neighbourhood cooperate to keep track of misbehaviour detections on a node and how they accuse in collaborative fashion nodes that persistently misbehave. This Section concludes with the evaluation results and analysis for the accusation phase of the protection scheme.

Chapter 4 presents a policy-based adaptable misbehaviour detection and node accusation approach. The Chapter starts by providing an introduction to policy-based management (PBM) and the policy core information model (PCIM). It then describes how the clustering phase of the proposed scheme works and explains its importance in the selection of “good enough” nodes which are assigned the role of cluster heads (CHs). Next, this Chapter presents the hierarchical organisational model used to distribute policies to nodes in the network and collect behaviour metrics for and from nodes actively communicating in the network. This Section also explains how the protection scheme parameters can be manipulated through high-level policies to achieve the goals set by network administrators. This Chapter is concluded by its respective evaluation results and analysis section.

Chapter 5 presents a collusion resistant misbehaviour detection and accusation approach. This Chapter begins with an introduction to service discovery strategies since the clustering approach introduced in this Chapter was originally designed to address service discovery in large MANETs. Then the role of the clustering strategy is presented within the context of service discovery in order to highlight its capabilities. This Section ends with its evaluation results and analysis. In the second half of this Chapter, a discrepancies concept is presented along with some of its desirable characteristics which make it resilient against lying and colluding nodes. This Section is concluded with a brief discussion on how data integrity and node authenticity can help address some weaknesses of the presented approach. Finally, a section on evaluation results and analysis completes this Chapter.

Chapter 6 provides a summary, some final conclusions and identifies open research issues for future work.
Chapter 2

2 Background and Related Work

It has been a little over 60 years since the first general-purpose electronic computer made its appearance in the world. Every since then the world has been undergoing a rapid transformation fuelled mainly by the capabilities of computer systems and information technology. Although it is difficult to predict with certainty how technology will alter our lives in a few years’ time, it is expected that the trend of a more computerised world will continue. These new computerised environments bring along new challenges and implications that are important to study and address in order to guarantee that they will effectively improve our quality of life and serve our needs hopefully in a seamlessly manner. In this Chapter we first review in Section 2.1 work that addresses the challenges posed by environments exhibiting a high degree of computer technology penetration, known to the scientific and engineering community as ubiquitous computer environments. We then move in Section 2.2 to consider autonomic computing systems and explain their usefulness in taming the complexity of managing environments with hundreds or thousands of heterogeneous computing devices that interact by making use of all sorts of communication technologies. Section 2.3 then reviews work on mobile ad hoc networks and gives some insights on how this technology can enable the deployment of ubiquitous computing environments. Finally, Sections 2.4 and 2.5 respectively introduce the subject of security and the difficulties faced by designers of security solutions tackling misbehaviour in multihop wireless networks.

2.1 Ubiquitous Computing Environments (UCEs)

Ubiquitous Computing Environments (UCEs) are characterised by a myriad of heterogeneous devices – often subtly camouflaged in their surroundings – with computing and communication capabilities that are embedded in almost every physical object surrounding their users. This fundamental idea of ubiquitous computing is considered to have been introduced first by Mark Weiser almost two decades ago in his paper “The Computer for the 21st Century” [4]. Weiser states in [4] and [5] that UCE users must be unaware of the existence of the computing devices embedded in their surroundings while those devices cooperate in a coordinated manner towards the fulfilment of the users’ goals. Thus, users can focus on their final goals rather than on how to
use the technologies that will enable them to achieve their objectives. Currently there is a commonly used electronic gadget that seems to be one of the closest devices to “disappear” from the users’ perception, i.e. the mobile phone, especially when phone calls are made using a wireless headset that connects to the handset by means of the Bluetooth communication protocol [6] [7]. By analysing this mobile phone example it is possible to see some of the desired characteristics in UCEs: 1) users walking on a street or buying items in a supermarket can talk on the phone unaware of the device thanks to the fact that their hands are free, instead 2) users focus on their main tasks, i.e. communicating with the person on the other side of the phone line and getting on a bus or selecting items in the supermarket. As this happens 3) the handset, the headset and the base station silently cooperate in a coordinated transparent manner to help the users to achieve their communication objective and to avoid interfering with other tasks performed by the users. And 4) the same task, or similar ones, can be performed simultaneously by many users all over a city or country without them losing their sense of individuality.

Figure 2-1 presents some typical ubiquitous computing environments that users may encounter on their daily routines. The figure depicts a scenario where the degree of computing pervasiveness is so high that a UCE is present almost everywhere. UCEs can permeate homes, offices, hospitals, libraries, shopping centres, restaurants and even in mobile environments such as automobiles, trains, airplanes and ships. Each UCE must be tailored to the needs of its particular set of users and the environments where they are. For instance, the needs of a doctor vary depending on whether they are at home, at the hospital or driving in between, and the devices present in each environment must act accordingly. Figure 2-1 also shows that all UCEs are connected through a ubiquitous network. This network is the result of the communication capabilities of the devices that constitute the different UCEs, which allow them to cooperate in a coordinated manner. Thus, UCEs can use the ubiquitous network to interact with each other and form a larger UCE, e.g. a country-wide UCE. However, current systems still have a long way to go before we can witness UCEs, as envisioned by Weiser, with hundreds of devices interacting in a dynamic environment in order to aid each user to perform several tasks. Advancing towards such UCEs requires that the scientific and engineering community continues its research efforts to create new ideas that complement and enhance the approaches proposed in the past two decades, some of which are presented in this Section.

Ubiquitous computing is a vast topic and it has received a lot of attention from the scientific and engineering community since its introduction by Weiser in 1991. Therefore, the amount of literature available can be overwhelming making it difficult to identify and analyse the main research themes [8]. Thus, work presented in this Section is by no means exhaustive or representative of all possible subareas of ubiquitous computing, but it offers instead an overview
of ubiquitous computing and some of its related subareas of study. For a detailed list of the main subjects of study within ubiquitous computing we refer the reader to [8].

Figure 2-1 Ubiquitous computing environments

2.1.1 UCE Challenges

Several researchers have focused their efforts on the study of ubiquitous computing environments and the challenges they face in the modern world. Such challenges have been considered from a diverse front of perspectives ranging from social and moral issues, passing through the users and their interaction with such systems, and going down to low level aspects of their implementation such as a device’s storage and computational capabilities.

In this line, the UK Computing Research Committee (UKCRC) has set up a Grand Challenge Manifesto [9]. Their view requires work on three fronts: 1) the experience perspective, which studies the interaction between users and UCEs, 2) the engineering perspective, which focuses on the architectural and network challenges posed by UCEs, and 3) the theoretical perspective, which focuses on concepts and rigorous models that capture the behaviour of UCEs. Authors claim that UCEs will be the result of these three perspectives merging together to create systems that are: fluid, purposive, autonomous, reflective, trustworthy, sustainable, efficient and scalable, among other qualities that may emerge at a later point. Work presented in [10] reviews and analyses previous and current work on UCEs, and sets a series of challenges that the authors consider need to be addressed in the future by the research community. Work is categorised according to the
authors' criteria in one of three possible application-driven interaction themes. The first theme is natural interfaces and includes approaches attempting to simplify interaction between users and ubiquitous environments with simple interfaces that account for users' mistakes. Context awareness is the second theme and contains work on applications that react depending on the "five Ws" rule for context aware designs: who uses the system, what they are doing, and where, when and why they are doing it. The last theme is capture and access, and deals with work on how to acquire and store information for its later retrieval by users in the UCE. Finally, the authors state the challenges that researches face on social issues and evaluation of ubiquitous computing. Similarly, the author of [13] lays some foundations for the design of UCEs based on his personal experience researching and designing such systems. In his work he presents a series of general design guidelines and argues that they are particularly valuable in the ubiquitous computing user experience world. The guidelines can be summarised as follows: do not include too many services in simple devices, focus on designing services not devices, do not overload users with information, do not reinvent the wheel, keep interfaces as consistent as possible between different devices offering the same services, provide feedback to users other than just visual (e.g. touch and audio), and only include extra information processing capabilities in your product if the cost justifies it.

2.1.1.1 UCE Limitations

Some work has focus on identifying the suitability of current technologies to develop UCEs. For example, the authors of [11] believe that nowadays, unlike in the past, hardware platforms are sufficiently advanced to make ubiquitous computing a reality. However, they also think that the software capabilities of systems have not developed at a pace that allows them to take full advantage of the new hardware infrastructure. Due to this, they state that new software is required to tackle the challenges related to power management, the limitations of wireless discovery and the user interface adaptation. In terms of power management, software must be able to control hardware in order to extend battery life, for instance selecting the most appropriate transmission interface, e.g. Wi-Fi, Bluetooth or Ethernet. The limitations of wireless discovery must be addressed with appropriate service discovery mechanisms that allow devices to know what other devices are in the network and the services they offer. Finally, user interface (UI) adaptation refers to the fact that UIs must be able to display properly in all types of devices regardless of their size, resolution and colour capabilities. The authors of [12] also consider that the current embedded technology is powerful enough to initiate the ubiquitous computing era. However, the paper states that networks do not have the fluidity necessary to deal with thousands of networked devices. Thus, they identify four challenges in communications networks: i) device interoperability in a network that consists of a myriad of heterogeneous devices - size, processing power and communication capabilities vary considerably between devices; ii) the logical network
topology can be very different from the geographical network topology, thus leading to virtual paths that can be very inefficient energy and resource wise; iii) it is currently very complex to manage short-lived and intermittent connectivity, typically devices exhibiting this behaviour are very unreliable; and iv) managing the evolution of large long-lived systems can be overwhelmingly difficult, for instance updating the software and hardware of a system with thousands of devices requires state of the art managing techniques.

2.1.1.2 User-Centred UCEs

There is user-centred research that demonstrates the efficiency of employing time-use studies and prototypes before implementing a ubiquitous computing system. Although these pieces of work do not introduce any specific guidelines, they can be considered guidelines themselves since they are an optional and sensible step in the design of UCEs. To start with, work in [16] demonstrates that time-use studies conducted by governments and commercial institutions can be used to provide data such as classifiers and baseline metrics for activity inference applications in ubiquitous environments. These time-use studies are useful for ubiquitous computing design because they provide large and detailed amounts of data for long periods of time in unbiased environments as opposed to data obtained in laboratories where the results can be biased by the artificial setting. Large time-use studies can include tens or hundreds of thousands of participants, span from days to years, and cost millions of dollars, yet the collected data by some of these studies are open to the public. The authors believe that using this information is an inexpensive yet precise approach to the design of ubiquitous computing services, which permits researchers to better tailor their work towards the satisfaction of the UCE users’ needs. Another study conducting user-centred research is presented in [17]. The techniques proposed in this study focus on environments in which common techniques may be too intrusive or might fail to produce helpful results, for example in hospital or home environments. The authors use two techniques to determine the adequacy of a product for certain users and their common surroundings in the prototype phase. The first technique is known as compound prototypes and provides a User Interface (UI) that controls a faithful implementation of an application in a computer system different to the device where the UI is, e.g. a mobile phone UI that controls an application running in a desktop computer. The second technique is referred to in this work as paratypes – situated experience prototypes. Paratypes permit researchers to evaluate users’ experiences by observing a technology’s likely use in real-world situations, i.e. not in a laboratory. The authors claim that both techniques were successfully tested in the design and implementation of their Personal Audio Loop (PAL) project, which aims at providing people with the ability of listening to anything that they have said and heard in the last hour by means of non-intrusive technologies.
User-centred research has also attracted the attention of many institutions around the world which have created testing environments capable of adapting themselves to make their users' experience as comfortable as possible. Such environments allow researches not only to see how an environment responds to users, but also to analyse how users interact with and perceive their "intelligent" surroundings. Among the most popular and well known projects are: The Aware Home [18] [19] from the Georgia Institute of Technology, House_n [20] [21] from the Massachusetts Institute of Technology, and the EasyLiving Home Project [22] from Microsoft Research.

2.1.1.3 Context-Aware UCEs

Work in [23] proposes an approach to provide context-aware applications with an infrastructure to process context information generated in devices of general access such as servers, and devices of personal use such as mobile phones. The proposed solution consists of user and ambient side agents (i.e. software modules) depending on which device they are installed. Additionally, context-based reasoning can be performed locally on each device or cooperatively if privacy issues arise that prevent one of the parts from accessing directly a needed variable value. Finally, reasoning on context information, which is described using the Web Ontology Language (OWL) [24], is achieved by using a set of rules defined in the Semantic Web Rule Language (SWRL) [25]. Another approach is proposed in [26], which bases its functionality in an architecture consisting of three layers. The basic or management layer, which is lightweight and is available in all devices, is in charge of managing the services – loading, starting, stopping and unloading them. The second or local services layer is the place where all the local services are. Local services are services that only use resources – and services – located in the same local device. The third layer consists of services that require remote resources to do their job. Local services, remote services and their interactions are all supervised and controlled by the management layer.

2.1.1.4 Middleware for UCEs

Proposing a middleware approach but addressing ubiquitous computing environments from an infrastructural software point of view is the work presented in [27]. The authors propose a middleware layer consisting of six modules each addressing different infrastructural software concerns in UCEs: roaming, discovery, ad-hoc networking (routing), limited connectivity, quality of service and system context adaptation. The authors then focus on addressing limited connectivity, service discovery and system context adaptation, providing examples on how to use modularisation to solve these specific problems. The final part of this work suggests an approach to evaluate the system's performance through a set of quantifiable high-level indicators of software quality that from the authors' point of view demonstrate how their proposed six modules improve the overall performance and usability of UCEs. Similarly, work in [28] offers a...
software-based solution for UCEs. This work aims at providing a framework that allows any common generic application to run in a UCE and take advantage of its features without needing to recode the application or implementing a wrapper/proxy for the application to work. Their framework, which they have called Plethora, runs on top of the Gaia Operating System [29], a distributed meta-operating system that runs on top of existing operating systems. The Gaia OS offers “Active Spaces,” each of which can be thought of as the software abstraction of a ubiquitous computing environment, thus providing access to all sorts of devices and services available in the UCE. On the other hand, the Plethora framework has the task of adequately translating common requests made by generic applications so that they are properly adapted to the Active Space, i.e. the UCE. Thus, if a user is doing a presentation, Plethora can automatically show the presentation in the available projectors and displays available in the current active space – for example a conference room – according to the user’s preferences. However, one issue that the Plethora framework does not deal with is how to allow users to switch from one device to another without having to close and open again the application in use, i.e. allowing for hot-switching or hot-swapping.

In [30] the author argues that truly ubiquitous computing environments must let processes migrate with users. The author then identifies a set of challenges to migrate a program that is in execution from one device to another. The first of these challenges is environment mobility, which is concerned with bindings between threads and the external environment (e.g. a program using a printer). The second challenge deals with channel mobility, which has to do with the re-routing of any communications to the new device. Then follows code migration, the main issue here is to ensure that the application runs properly regardless of the operating system it migrates to. State migration is next in the list; it deals with issues on how to migrate an application’s dynamic state which can include hundreds of variables. Finally, the paper analyses how to migrate views with minimal loss of consistency due to differences between the display capabilities of those devices involved in the application migration.

2.1.1.5 Currently Developed UCEs

In addition to research proposing solutions to the challenges posed by ubiquitous computing, it is possible to find in the literature research work that analyses computing environments currently available in search of possible solutions that have silently emerged. Such an approach to UCEs represents an alternative option to the mainstream ideas based on the futuristic vision stated by Weiser. For instance, authors of [31] challenge Weiser’s view of ubiquitous computing by claiming that it is invalid because it was conceived almost 20 years ago, and since then the technological panorama has changed significantly. In fact the authors do not propose a solution but instead they claim that somewhat the ubiquitous computing era has already arrived to some
countries. They analyse and present the technological advances of Singapore and Korea which they consider are already living the ubiquitous computing era. They also explain how the new ubiquitous technologies have benefited both people and governments in these countries. Finally, they conclude the paper stating that the ubiquitous computing era has arrived, but it is not as we had envisioned it 20 years ago. The ubiquitous technology is there but it is not evenly distributed, on the contrary it is very messy, heterogeneous and it has not “disappeared” completely from the users’ perception, perhaps because it is not meant to.

On the other hand, work presented in [14] examines two case studies of fully operational Radio Frequency Identification-based (RFID-based) systems. The two case studies analysed are the Oyster Card ticketing system used in public transport in London, UK, and the retail applications deployed at the Mitsukoshi department stores in Tokyo, Japan. The authors assert that the analysis of both projects demonstrates that current research is not dealing with real-world issues such as: the lengthy and costly preparation to upgrade existing infrastructures, the training of employees and users in the new ways of working, and the gradual and controlled introduction of new functionality with minimum disruption for the users, among others. Finally, this work concludes that the challenge for researchers is not only to address the technical issues of ubiquitous computing, but also how to translate the principles, guidelines and models created through research to large-scale real world environments. Work presented in [15] also examines real world case studies. However, in this case the authors themselves were part of the team that implemented the systems, which allowed them to come up with some concepts based on their own experience. The systems deployed consisted mainly of visual aids provided by public displays and projectors, which were installed in two different venues and an underground bus station. In spite of the systems being significantly smaller than those evaluated in [14] some guidelines are similar as it can be inferred from of their main conclusions: deployments are costly, environmental challenges can be significant, after deployment comes maintenance, following government regulations is important, system monitoring should be done from a user's perspective, providing feedback on the state of the system to the user is good practice, creating content is expensive, and managing content is complex.

2.2 Autonomic Computing Systems (ACSs)

The term “autonomic computing systems (ACSs)” refers to systems composed of heterogeneous devices capable of individual and collective self-management which can achieve their goals with minimum human intervention. Due to their characteristics autonomic computing systems are considered to be one of the building blocks of ubiquitous computing environments. Although the protection scheme proposed in this thesis, which we introduce from Chapter 3 onwards, does not
attempt to offer a complete autonomic solution or self-management property for UCEs, it has been endowed with adaptability (as presented in Chapter 4), which is one of the first and essential steps towards a fully autonomic approach. This Section provides a short introduction to ACSs and the self-management properties that typify them.

From the appearance of the first computer systems, complexity has been an issue that scientists, engineers, designers, programmers and even users have had to deal with [32]. What is more, as new sophisticated technologies and techniques are developed, complexity tends to increase and it can be argued that in many cases it hinders the advancements achieved by such technologies owing to the fact that they cannot be easily adopted, require a long time to be implemented or necessitate experts to manage them. An example of an emerging paradigm that has brought an enormous amount of complexity along with it is ubiquitous computing environments, and some of the issues that researches have identified in these environments have been illustrated in Section 2.1. The intricacy of these systems lies mainly on the large number of heterogeneous devices that can be present at one time in an environment, thus making management-related tasks very convoluted. This argument is easily demonstrated with a simple question that illustrates an example scenario: how can a network administrator maintain up to date hundreds or even thousands of devices that exhibit heterogeneity in size, memory, computational power and software among many other features? Such a task can be overwhelmingly difficult even for a team of network experts. Unfortunately, things can only get worse as other issues such as security and users' privileges are taken into account. From this it becomes clear that as the complexity of systems continues to increase there will be a point at which no team of experts will be able to handle the problem [33]. Therefore, it is imperative to find a solution that eases the task of managing systems by making it less complex for system administrators. Autonomic Computing Systems (ACSs), which are inspired in self-regulated biological systems, are a very attractive solution that takes away the management burden from the system administrators and places it on the system itself, as explained in the following paragraphs.

The idea of using autonomic computing systems to cope with the systems' own complexity was first introduced by Paul Horn from IBM on 8 March 2001 in a presentation to the National Academy of Engineering at Harvard University [34]. Such an idea – IBM's vision – is also presented and explained in detail in IBM's Autonomic Computing Manifesto [33]. In this document Horn states that they found inspiration in the nervous autonomic system of the human body. This system controls the heart beat, body temperature, blood sugar levels and pupils to regulate the amount of light reaching the eyes, among many other body functions. But what is really important is that we – the nervous system users – are completely unaware of all these processes while we run, eat, read, etc. In other words, we focus on what we want to do (i.e. our
goal) and let our nervous system to take care of everything else. Likewise ACSs should allow us to concentrate on our tasks while they take care of their tedious own management.

As can be appreciated ACSs share some common goals with UCEs, the most important of them is perhaps that they aim at allowing users to focus on their foreground tasks whilst the system takes care of supporting background activities. The difference is that UCEs are a more abstract paradigm that implies, but does not specify how, devices and their tasks are to disappear from the users’ perception, i.e. different technologies and solutions can be employed to achieve this goal. ACSs on the other hand are a specific solution that proposes self-management as a means of taking certain tasks off the system users and placing them on the system itself, which results in user unawareness of the background activities carried out by the system. This is the reason why ACSs are considered a solution that can enable the deployment of UCEs.

In [33] Horn also proposes a minimum set of “eight key elements or characteristics” that are essential for a system to be autonomic and that have been widely accepted by the scientific community, as it can be observed from existing literature on autonomic computing. These key elements are as follows:

- An ACS needs to “know itself”. It requires detailed knowledge of its individual components, current status, capacity and connections to other systems. It must also know the resources available directly to it, and those that can borrow, lend, share or reserve.

- An ACS must be able to automatically find an appropriate configuration and dynamically adjust it to best tackle changes in its surroundings.

- An ACS always seeks to optimise its performance. It must constantly fine-tune its constituent parts in order to achieve predetermined, yet bound to change, system goals.

- An ACS must execute a process similar to healing. It must do both detection of potential problems and recovery from component failures. In either case the system needs to find an appropriate way to use the untroubled resources and reconfigure its components to continue working as smoothly as possible.

- An ACS must be capable of protecting itself. It must remain alert, anticipate threats, resist attacks and take the necessary actions to maintain the overall system security and integrity.

- An ACS must know its environment and the implicit context of its activity in order to take actions that satisfy its overall goal within its current circumstances.

- An ACS must work in a heterogeneous world and therefore it must adhere to open standards. Proprietary solutions do not offer the versatility required for an ACS to communicate and collaborate with other systems of its kind.
An ACS must anticipate the optimised resources required by any user, while maintaining its complexity hidden from them. This point is of crucial importance for the users as they usually want a system ready to act and simple to understand.

### 2.2.1 Self-Management Properties of an Autonomic Computing System

As research on autonomic computing systems advances it has become clear that such systems are required to implement a very distinctive set of attributes which in spite of being incredibly common to biological systems are rare and difficult to accomplish in computing systems. These attributes are known as the self-management or self-* attributes or properties of ACSs and some of the most popular are: self-configuration, self-optimisation, self-healing, self-protection, self-awareness, self-maintenance and self-adaptation. It is important to realise that this list is by no means exhaustive and that the attributes are not mutually exclusive, i.e. they can overlap or embrace one or more of the other listed self-management properties. In the following paragraphs five of these properties are concisely analysed due to the vital role they play in ACSs.

The final goal of autonomic computing systems is self-management as a means to free administrators from the burden of handling complex systems and to provide users with an always-available system that offers its best possible performance at all times [35]. Achieving self-management requires the incorporation of the eight key characteristics suggested by Horn [33], and IBM has once more taken the lead in this area by proposing that ACSs should be endowed with four fundamental features or self-properties [34][35]: self-configuration, self-optimisation, self-healing and self-protection.

- **Self-configuration** addresses the need for new components to incorporate themselves seamlessly to a new system, while the system adapts itself to the presence of the new components. Thus, components and systems must react to changes in their surroundings according to high level policies that specify the desired objectives for the overall system (or enterprise) rather than for individual components.

- **Self-optimisation** tackles the very common problem of fine-tuning a system to obtain the most efficient and cost effective configuration. The problem is that large systems can have hundreds and even thousands of parameters that can be tuned to modify their performance. To make matters worse, these systems are usually connected to equally large systems and modifying a parameter in one of them may have unexpected repercussions on the entire system. ACSs will never set for the status quo; they will always strive for new configurations leading the system to more efficient and cost effective states.

- **Self-healing** addresses the common and costly problem of system failures and service disruption. ACSs must be able to isolate failing components and either replace them with
backup modules of similar capabilities or redistribute the computational load among other
components while keeping service disruption to a minimum. Then if the failed component
cannot be brought back online a notification should be passed to the human counterpart for
them to take care of it.

- **Self-protection** is the feature that enables ACSs to anticipate, detect, identify and protect the
  system against all type of threats including intentional attacks, malicious behaviour, intrusion
  and random component failure. If self-protection fails to anticipate and prevent a problem from
  happening – whether unintentional or not – then self-healing is required.

### 2.2.2 Towards Fully Autonomic Computing Systems

Similarly to many other disciplines ACSs have attracted the scientific community's interest in
developing guidelines, techniques and frameworks that can be followed and employed to reach
the ACSs' ultimate goal, i.e. self-management. Much of this effort is based – sometimes
implicitly – on the original idea proposed by IBM described in Section 2.2. In this Section we
have a look at some of this work, especially efforts aiming at providing a set of guidelines or
designing a holistic approach to build ACSs. For a more general and broader review of work on
ACSs we refer the reader to [36] where several industry and academia oriented projects are
described.

To start with, IBM researchers argue in [34] that autonomic computing systems will arrive as the
result of an evolutionary process as opposed to a revolutionary one. The authors envision five
levels, each of which indicates how autonomic a system is. A system that is currently at level 1
(basic) has the possibility of gradually work its way up to level 5 (fully autonomic). At the basic
level each element of an IT system requires an administrator to configure it, monitor it, optimise
it, update it, fix it when damaged, and eventually to replace it. At the managed level (level 2) the
system gathers information from disparate elements and collects it into fewer system points
making it easier for the administrator to analyse and synthesise such information. At this level
system administrators can cope with an increased degree of complexity in the system. At the
predictive level (level 3) there are technologies capable of working out the existent correlation
between different elements of the system. At this point the system is able to recognise patterns,
calculate the most advantageous configuration, and suggest the best course of action for the
system administrators. The adaptive level (level 4) is the result of an improvement in the
predictive technologies and an increased administrators' confidence in them. Systems can make
decisions and take actions based on the information available to them and the knowledge of the
state of each of their components. Finally, at the fully autonomic level the system's operation is
managed by business policies and objectives; system administrators monitor the system at
business level and intervene only to change the business goals. The steps that a system undergoes to evolve from a basic level to a fully autonomic level are illustrated in Figure 2-2.

![Figure 2-2 Autonomic systems' evolution](image)

In [35] the authors (also from IBM) present some architectural considerations along with engineering and scientific challenges to build ACSs. They propose that an autonomic element consists of one or more managed elements controlled by an autonomic manager. The autonomic manager monitors the managed element, analyses the collected information by comparing it against its stored knowledge database, plans the necessary actions to maintain the managed element working as expected, and then it executes the plan by modifying the managed element as required. This approach also assumes that autonomic elements constantly interact among them and can modify the goals of elements under their authority depending on the internally set goals of each element. This architecture is depicted in Figure 2-3. Typically managed elements can be hardware devices such as a raid of hard drives, a printer or a CPU, while autonomic elements are the representation of managed elements at the highest level, for example a safe-backup application, a prepaid photo printer, or an e-utility. After introducing their structure the authors analyse engineering challenges from four different points of view: life cycle of an autonomic element, relationships among autonomic elements, system wide issues, and goal specification. On the other hand, in the scientific challenges five main points are identified: behavioural abstractions and models, robustness, learning and optimisation, negotiation, and automated statistical modelling. Finally, IBM’s contributions to autonomic computing include a white paper specifying an architectural blueprint for autonomic computing [37] that develops in-depth the areas covered by papers [34] and [35]. The latest version of IBM’s white paper, i.e. their fourth edition, has been modified to reflect the up-to-date trends and current IT solutions developed by IBM and can be found in [38].
The authors of [39] propose a simple architecture which incorporates many of the ideas available in the year 2003. The authors claim that their idea can be used as a general template for the design of ACSs. Their architecture has a main module called the Self Monitor which has direct communication with every module of their approach. An Internal Monitor module observes the current state of the managed element and passes it to the Self Monitor for its analysis. The state is then compared against an expected state maintained in a System Knowledge module. Unwanted deviations are then reported by the Self Monitor to the Self Adjuster module for it to take action changing the managed element if necessary. Similarly, there is an External Monitor module which reports changes in the environment which can also lead to alterations to the state of the managed element depending on the system's objectives. The final module is a Heartbeat Monitor which periodically reports the current state of the autonomic element to other autonomic elements for them to keep track of their environment. A different approach is taken by the authors of [40] who have developed a three level theory of human behaviour which they claim can help with the design and implementation of ACSs. The three suggested levels of behaviour are: reaction, routine and reflection. The reaction level is the lowest level; it constantly monitors the organism/system and the environment and reacts immediately when it detects problematic or dangerous situations by interrupting ongoing higher level activities. In humans, for instance, this level automatically makes us drop a cigarette from our hands and stop our ongoing reading activity to analyse what just happened. In autonomic systems it may detect a very high electric current and shut down an appliance to prevent any damage even if it was being used at that
instant. The routine level is home to skilled and well-learnt behaviours. In humans this level corresponds to motor skills such as language generation, walking or chewing food. In autonomic systems this system can be very complex since it requires learning from past experiences and accepting inputs from the reaction level below and the reflection level above in the form of control signals. For example an autonomic storage system that has a two disk RAID may learn from past experiences that once a disk fails there is an increased probability of an overall system failure, thus it can decide to put in place a process that creates a remote backup every time a disk fails. At the reflection level the mind performs operations based on its own experiences, current behaviour, and the current environment, yielding as output planning, reasoning and problem solving. This level in ACSs could lead to cautionary behaviour when executing a task or even avoidance and termination of the task if it detects potential harm, excess of resource utilisation or a deadlock state.

ACSs are tackled from an engineering point of view in [41]. In the first part of this work the authors suggest a set of seven engineering principles to guide the planning of ACSs. These principles can be summarised as follows: i) provide needed features only, unnecessary features add to the complexity of the system, ii) regardless of how clever the system is users must always have the option of switching off the autonomic functionality of any element, iii) many users do not use available features in many applications because they do not know they exist, so autonomic features should be enabled by default so that user can take advantage of them, iv) if the developers are not able to make a decision at the design or implementation phases such responsibility should not be passed to the user since the most likely thing to happen is that they will be confused, v) assessing autonomic solutions in benchmarking systems is inappropriate because these systems do not represent the real world effectively, vi) autonomic functionality should never contradict instructions given by the human counterpart since, generally speaking, humans have a better understanding of the big picture and are better at judging it, and vii) at the highest level policy definition should be minimal and as close as possible to natural human language. In the second part of this work the authors present eight existent mathematical approaches which they consider suitable for the construction of ACSs. These eight foundational techniques are: dependency management, expert systems, trade-off elimination, static optimisation, online optimisation, feedback control loop, correlation modelling and security; a brief description of each of them is offered in [41]. Also following an engineering approach is work presented in [42] which introduces an autonomic computing environment framework called AUTONOMIA. The main goal of AUTONOMIA is to provide application developers with the tools required for them to implement networked autonomic applications and services. This work uses mobile agents to carry out the core self-management tasks: self-configuration, self-optimisation, self-healing and self-protection. The framework consists of a set of modules with
two main purposes: managing the mobile agents and providing an application programming interface so that the application developers can interact with the autonomic framework. The first module is the Mobile Agent System (MAS), which provides the mobile agents with a uniform execution environment independent of the underlying architecture and operating system (OS) of the platform where the agent is. It also provides functions to receive agents, start the execution of new agents, monitor their execution and transfer agents from host to host. Platform independency is achieved using Sun Microsystems tools such as java [43] [44], java remote method invocation [44] and Jini [45]. The second module is the Application Manager Editor (AME) and allows a user to develop applications using existing autonomic components through well-defined libraries. Additionally, it enables developers to specify the management requirements for each deployed component. The last module, the Autonomic Middleware Service (AMS) module, consists of the mobile agents themselves and it endows the system with the services required to achieve autonomic management.

2.3 Mobile Ad Hoc Networks (MANETs)

A mobile ad hoc network consists of a group of devices that relies solely on the wireless communication medium and themselves for data transmission. Thus, MANETs unlike other wireless networks – e.g. Global System for Mobile Communications (GSM) [46] or Universal Mobile Telecommunications System (UMTS) [46] – do not require the existence of any centralised administration or pre-deployed network infrastructure, e.g. base stations; instead the management responsibilities are distributed among the network nodes, which also cooperate by forwarding packets on behalf of each other when destinations are out of their direct wireless transmission range. These characteristics make MANETs’ deployment quick and inexpensive, and the nodes’ ability to move freely endows them with a flexible and versatile dynamic network topology that can be desirable in many situations. Mobile ad hoc networks are ideal in environments where installing an infrastructure is not appropriate for reasons such as cost, quality, or vulnerability, or where the network is too transient, or the infrastructure has been destroyed. Examples of MANET applications vary from emergency disaster relief (destroyed infrastructure), military operations over a battlefield (vulnerable infrastructure), and wilderness expeditions (transient networks), to community networking and interaction between students during a lecture. However, one of the most promising applications for mobile ad hoc network technologies is their deployment in ubiquitous computing environments. Although UCEs can be supported by a network infrastructure, it is not likely that all mobile devices (or sensors) can always be within a base station’s area of coverage, thus reliance on intermediate nodes for packet forwarding becomes necessary. Furthermore, due to the great number of devices present in UCEs it is
unfeasible to have them permanently connected to the network (this could lead to bandwidth exhaustion), it is a better approach to have them join and leave the network (ad hoc behaviour) as required. This point is further enhanced in MANETs thanks to reactive routing protocols, which make use only of those nodes essential for the route discovery and data forwarding processes while allowing the other network nodes to save their battery power by remaining inactive, as is explained later in this Section.

2.3.1 MANETs and the Proposed Protection Scheme

As mentioned above node mobility combined with a lack of network infrastructure make a mobile ad hoc network's topology extremely dynamic. Consequently, conceiving solutions capable of managing the dynamicity of MANETs can be very challenging since they must cope, among others, with constant link breakages, node unavailability, variable paths between source and destination, and the unreliable wireless channel. However, MANET solutions have a welcome benefit that springs from the complexity they deal with, when they are migrated to other technologies or paradigms migration can be done with relatively ease. Since MANET solutions address so many and difficult issues (close to the worst case scenario) once they work well in a MANET environment they normally require minimum or no modifications to perform well in other environments. For instance, if a MANET approach is migrated to a ubiquitous computing environment the available infrastructure and static devices most certainly will only make things easier and better. Another advantage of designing solutions for MANETs is that they can be tested over a choice of well known network simulators before deploying them for evaluation in the real world, which can be costly, time consuming or even unachievable as in the case of MANETs with hundreds of nodes. Some well known networks simulators with support for MANETs are: NS-2 [47] and GloMoSim [48] [49] which are intended for academic and research use, and OPNET [50] and QualNet [51] (a GloMoSim based product) which are commercial options. For these reasons we have decided to test and evaluate the suitability of the proposed misbehaviour detection and node accusation approach using mobile ad hoc networks, even though it is easily seen that some of our assumptions described in Section 4.2 would fit better in a ubiquitous computing environment.

2.3.2 Route Discovery and Packet Forwarding

Routing protocols in MANETs have two important tasks to perform: route acquisition and data packet forwarding. The main task in the route acquisition phase is to find an available path from a source node to a given destination node. The task of the data forwarding phase is to forward any data packets sent from the source towards the destination (and vice versa) using the path that was
found in the route acquisition phase. Both these tasks are essential for a MANET to fulfil its purpose and all nodes in the network should abide by the rules of the routing protocol in place. In this respect the work described in this thesis aims firstly at ensuring the detection of any nodes that misbehave during the data forwarding phase and secondly to punish nodes that persistently misbehave, as explained in Chapter 3.

Depending on the methodology employed by routing protocols to acquire an available path to a destination they can be classified as either proactive or reactive [52] [53]. Proactive protocols, also known as table-driven protocols, periodically exchange routing information so that all network nodes can keep an up-to-date view of the network topology. A common characteristic of these protocols is that routes to any network destination are available even before any packet transmission commences [54]. On the other hand, the various proactive protocols differ in the number of tables maintained by each node and the strategy used to preserve a consistent network view in all network nodes [52]. Examples of proactive routing protocols include the Destination-Sequenced Distance-Vector (DSDV) routing protocol [55], the Wireless Routing Protocol (WRP) [56], and the Optimised Link State Routing (OLSR) protocol [57] [58]. By contrast, reactive routing protocols, also known as on-demand protocols, do not exchange any packets until a source node desires to send a packet to a destination node and a route discovery procedure is initiated [54]. In this scheme, source nodes are required to keep the packet(s) in their buffer until a route to the destination is found. Although reactive protocols incur a delay before sending the first packet(s), they are very popular because they do not produce any network overhead when the network is idle – also saving battery power – and they adapt very well to highly dynamic environments [52]. Examples of reactive routing protocols include the Ad Hoc On-Demand Distance Vector (AODV) routing protocol [59] [60], the Dynamic Source Routing (DSR) protocol [61] [62], the Lightweight Mobile Routing (LMR) protocol [63], and the Temporally-Ordered Routing Algorithm (TORA) [64].

Finally, once a source node knows of an available path to its desired destination the data packet forwarding phase is initiated. In this phase the packets are trusted to intermediate nodes which forward them along the route towards the intended destination. Depending on the routing strategy adopted the path followed by each packet can be determined at intermediate nodes by choosing the next hop from their routing tables, as in AODV, or at the source node and then appended to each forwarded packet so that nodes in the path are able to read what the next hop towards the destination is, as in DSR. Our approach to enforce behaviour has been design to protect the data forwarding phase of routing protocols and one of its main advantages is that it can be used regardless of the routing strategy adopted; it can even be used in environments that dynamically change routing strategies on the go in order to adapt themselves to an environment’s current
conditions. The details on how routing independence is achieved in our approach are provided from Chapter 3 onwards.

2.4 Network Security

Most people have the notion that security is something good, but very few are capable of defining security in an exact manner [65]. This is partly due to the fact that the definition of security is subject to the context of each particular situation in which security is considered. For instance, security for a person as a citizen could mean to be out and about in their city without risking being hurt, but when their role changes to a car owner security could mean that their car is fitted with the latest technology to prevent it from being stolen. Likewise in the communications world the definition varies depending on the field for which security is required.

Network security or internet security (internet with lowercase, as opposed to the Internet, refers to a collection of interconnected networks) is concerned with the measures needed to protect data during its transmission between two or more entities, i.e. users, devices or systems [66]. The ITU-T (International Telecommunication Union – Telecommunication Standardisation Sector) recommendation X.800 [67] describes a systematic approach to provide security for the Open Systems Interconnection (OSI) Reference Model defined in the ITU-T recommendation X.200 [68]. Due to the fact that this recommendation was designed as an international standard, it has been widely adopted by computer and communication vendors all over the world [66]. For this reason this approach is now used in most communications systems even if they do not strictly follow the OSI reference model.

2.4.1 Basic Security Services

According to recommendation X.800 and the Internet Engineering Task Force (IETF) informational RFC 4949 (Internet Security Glossary – Version 2) [69] a security service is a processing or communications service provided by a protocol layer that ensures adequate protection of system resources and data transfers. Generally speaking the following are the security services that can be provided in a communications system [67]:

2.4.1.1 Authentication

The authentication service ensures that the parties involved in a communication have genuine identifications and remain as such from start to finish. Firstly, at the connection initiation stage it ensures that the parties are who they claim to be, and then during the communication stage it prevents a third party from masquerading as one of the legitimate parties so as to transmit or receive information in an unauthorised manner.
2.4.1.2 Access Control

Access control specifies through access rights which modules, equipment or functions a remote entity is allowed to use. However, before granting access to a system resource the identity of the remote entity must be verified through an authentication process.

2.4.1.3 Data Confidentiality

The data confidentiality service aims at protecting the information transmitted over a communication link from being released to or read by unauthorised entities, even if those parties directly manipulate the transmitted packets, e.g. for forwarding purposes. A second feature of the confidentiality service is the protection of traffic flow from analysis. Here the source, destination, session length, frequency and other traffic characteristics must be hidden from parties that are not directly involved in the communication process.

2.4.1.4 Data Integrity

In its most general form the data integrity service assures that the transmitted information is received as sent. The X.800 recommendation differentiates between connectionless and connection-oriented integrity service. The connectionless service provides for the integrity of a single message by determining whether a received message has been modified. The connection-oriented service guarantees received messages have not been modified, duplicated, inserted, reordered or replayed. It can also assure that no messages have been destroyed on their way towards their destination.

2.4.1.5 Non-repudiation

The non-repudiation service prevents a sender from denying sending a message that it has actually sent and a receiver from denying receiving a message that it has actually received. Alternatively, this service can be viewed as a way for a sender to prove that an alleged receiver has in fact received a message or for a receiver to prove that an alleged sender has in fact sent a message.

2.4.1.6 Availability – Property/Service

Although both the recommendation X.800 and the RFC 2828 treat availability as a property of a system that is associated with various security services, in [66] Stallings argues that it makes sense to explicitly specify an availability service that protects the system, especially against security threats raised by denial of service attacks, to ensure its services and resources are available when required.
2.4.2 Security Attacks

A security attack is any deliberate attempt to evade security services that compromises the security of information, system resources or services [66] [69]. Security attacks are commonly classified as passive attacks and active attacks.

2.4.2.1 Passive Attacks

Passive attacks try to gain unauthorised access to information from a system or from in-transit messages over a communication link, but it does not compromise system resources or services [69]. Regarding communications systems, passive attacks can be of two types [66]:

- **Eavesdropping** aims at gaining access to confidential information by secretly tapping into a communication link (wiretapping), i.e. without the knowledge of the data packets’ source and destination.

- **Traffic analysis** aims at obtaining useful information, such as the nature of a communication taking place, by means of determining the location and identity of the communication parties along with the length and frequency of the exchanged messages. This type of attack can still be used when eavesdropping is not possible.

Due to the nature of passive attacks they are very difficult to detect. Nevertheless, it is possible to deal with them by preventing their success usually making use of encryption techniques.

2.4.2.2 Active Attacks

Active attacks try to modify system resources or hinder their operation [69]. In communication systems they attempt to modify or destroy packets in transit, or create false packets that can diminish the efficiency of the system in any possible way. There are four common categories of active attacks [66]:

- **A masquerade attack** is launched by an entity that pretends to be another entity. It usually involves the creation of packets and the use of other active attacks to break the security measures in place.

- **A replay attack** involves the passive capture of a message and its later retransmission to accomplish an unauthorised and usually harmful effect.

- **Modification of messages** implies that a message is captured, altered and retransmitted without the consent of its source and destination. This type of attack also includes capturing packets and reordering them to manage an unauthorised effect.

- **Denial-of-service (DoS) attacks** hinder or stop completely the normal use or management of a communication system. DoS attacks can affect individual entities (e.g. impeding messages
from reaching a destination), various entities (e.g. by dropping messages directed to more than
one destination, or bringing down a network service such as an e-mail server), or the entire
network (e.g. flooding packets all over the network to overload it and degrade its overall
performance) – Section 2.5 offers a review of work addressing DoS attacks targeted at the
network layer of MANETs and aimed at disrupting its route discovery and packet forwarding
functionality.

Contrary to passive attacks, active attacks are easier to detect than prevent. Thus the course of
action is to detect them, tackle them and recover from any disruption that they may have caused.

2.5 Misbehaviour in MANETs

As was stated in Section 2.3 the intrinsic characteristics of mobile ad hoc networks, such as node
mobility and the lack of infrastructure, endow them with a flexible and versatile dynamic network
topology that can be desirable in many situations. Unfortunately, those same characteristics in
addition to the broadcast nature of the wireless channel and the fact that nodes rely on other nodes
for packet transmission and service accessibility make MANETs vulnerable to a wide variety of
attacks by misbehaving nodes. Such attacks range from passive eavesdropping, where a node tries
to obtain unauthorised access to data destined for another node, to active interference where
malicious nodes hinder network performance by not obeying globally acceptable rules (passive
and active attacks were introduced in Section 2.4.2). The rest of this Section is organised as
follows: in Section 2.5.1 we clarify the differences between the terms misbehaviour, intrusion and
trust. Then in Section 2.5.2 we introduce schemes for the protection of the routing discovery and
packet forwarding services of the network layer. Section 2.5.3 reviews work by other authors on
anomalous link behaviour detection and Section 2.5.4 reviews work on intrusion and node
misbehaviour detection.

2.5.1 Misbehaviour, Intrusion and Trust

In the network security literature the terms misbehaviour, intrusion and trust are commonly used
with vague and sometimes interchangeable meanings. Although the literature review offered in
this Section is likely to have the same shortcomings as we prefer to use the same terms employed
by the authors of each work, we believe that it is important to clarify to the reader the difference
between these terms as it helps to understand why we have favoured the use of the word
misbehaviour in our work instead of intrusion or trust.

Trust can generally be thought of as the level of confidence that an entity can have on a system
that it relies on. The higher the level of confidence an entity has on a system the higher the
probability that the system meets the specifications, i.e. that the system does what it claims to do and it does not carry out any harmful or unwanted functions [69]. Typically the level of confidence of an entity on a system increases every time it uses the system with a successful outcome and decreases otherwise. Additionally, trust on a system can be affected by the opinion that an already "trusted" third party has on it. For instance, the X.509 ITU-T recommendation [70] specifies how a trusted Certification Authority (CA) can create certificates for trustworthy systems and revoke them for untrustworthy systems. Thus, when an entity wants to use a system it first checks with the CA that the system’s certificate is valid, in which case it assumes that the system can be trusted, i.e. confidence in the system starts at a high level.

According to RFC 4949 [69] an intrusion occurs when an entity, i.e. the intruder, gains or tries to gain access to a system or system resource without having been granted authorisation to do so. However, such definition is somewhat narrow in the context of mobile ad hoc networks since it does not account for attacks that degrade the communication performance of a MANET by using solely the intruder’s own resources and without requiring any authorisations or violating any access rights. A more general definition is offered by the authors of [71] and [72]. In their work, which is tailored to MANETs, an intruder is considered to be any node (i.e. user or device) that subverts the functioning of the network by intentionally causing undesirable events known as attacks or intrusions.

On the other hand, misbehaviour – or malicious behaviour – in MANETs has a more general definition than that of intrusion. While in the definition of intrusion the node hindering the network performance causes the undesirable events in an intentional manner, in the definition of misbehaviour such events can also be caused unintentionally. For instance, if a node behaves maliciously by not forwarding packets on behalf of other peer nodes, the node does not always exhibits such a malicious behaviour because it intends to do so. The node may also misbehave because it is overloaded, broken, compromised or congested as well as intentionally being selfish or malicious [73] [74]. An overloaded node lacks the CPU cycles to attend its local and/or network tasks, which leads it to not comply with its network duties owing to its inability to process them. A broken node has a software or hardware fault that prevents it from performing its network tasks properly. A compromised node may be victim of an attack that degrades its network capabilities. A congested node receives more packets than the bandwidth available to it allows it to send, its buffer fills and eventually it has to drop incoming packets. A selfish node is unwilling to use its resources such as battery life, bandwidth or processing power to forward packets on behalf of other nodes. A malicious node drops packets or generates additional packets solely to disrupt the network performance and prevent other nodes from accessing network services (a denial of service attack). Thus, misbehaviour accounts for both intentional and unintentional network-harmful events.
2.5.2 Route Discovery and Packet Forwarding Protection

In MANETs misbehaviour at the network layer can involve the routing protocol and it can be divided into two categories [73]: route discovery misbehaviour and packet forwarding misbehaviour. Route discovery misbehaviour refers to the failure by a node to correctly execute the route discovery procedure acting in accordance to a set of rules defined by a routing protocol. Packet forwarding misbehaviour instead refers to the failure by a node to properly forward packets towards their destination on behalf of other network nodes. In this Section we review work on security measures that protect the route discovery and packet forwarding services from the disruption caused by intruders or misbehaving nodes. However, unlike proposals made later in this thesis, the work described in this Section makes no attempt to pinpoint the links or nodes causing the network service disruption, it rather prevents the disruption from happening or mitigates it as much as possible.

Secure routing protocols have been proposed based on existing ad hoc routing protocols. These protocols add several security features but also eliminate some of the optimisations introduced in the original routing protocols because these optimisations can be exploited to launch different types of attacks. Although available work in this area mostly seems to address reactive routing approaches, there is also work that addresses proactive routing strategies. For example, the secure efficient distance vector (SEAD) routing [75] is based on the destination sequenced distance vector (DSDV) [55] but makes use of inexpensive one-way hash chains cryptography to create a routing protocol that offers robustness against multiple uncoordinated attackers by authenticating hop counts and sequence numbers. The secure ad-hoc on-demand distance vector (SAODV) routing protocol [76] [77], which is based on AODV [59] [60], also uses one-way hash chains cryptography in addition to digital signatures to protect Route Request (RREQ) and Route Reply (RREP) messages and prevent malicious nodes from advertising false routing information on behalf of other nodes in the network. The secure on-demand routing protocol for ad hoc networks (Ariadne) [78] is based on the dynamic source routing (DSR) protocol [61] [62] and the timed efficient stream loss-tolerant authentication (TESLA) broadcast authentication protocol proposed in [79] which is used to authenticate routing messages. A common characteristic of SEAD, SAODV and Ariadne is their need for a mechanism to set up secret keys, shared secret keys or to distribute authentic public keys, but these mechanisms are not always readily available in transient MANETs. Moreover, these approaches only secure the path discovery and establishment functionality of routing protocols; as we will see in Chapter 3, our approach complements them by securing the data forwarding functionality.

Some research effort has also been focused on the development of new routing protocols whose objective is to protect the network from security threats that were not addressed by earlier work.
Chapter 2. Background and Related Work

The Secure Routing Protocol (SRP) [80] assumes the existence of an a priori security association between communicating nodes, i.e. source and destination, and it specifically guarantees that false, compromised or replayed route replies do not reach back the querying node or are rejected if they do. Authenticated Routing for Ad hoc Networks (ARAN) [81] on the other hand assumes that network nodes can exchange keys a priori with a trusted network certificate server – possibly using an out of band method. Interaction among nodes is based on their certificates, and erratic nodes (i.e. misbehaving nodes) can be removed from the network by revoking their certificates. Although ARAN includes a mechanism to remove misbehaving nodes, it does not address how to detect them. As with SEAD, SAODV, and Ariadne these protocols can be coupled with our approach, which is not routing protocol dependent, to offer an improved security solution.

There is also previous work that has been focused on ensuring that the data forwarding functionality of routing protocols can withstand the attacks launch by malicious network nodes while maintaining the level of network disruption as low as possible. Nodes dropping packets that are expected to be forwarded is an example of a security attack seeking to disrupt packet forwarding; this attack and similar ones along with their effects on MANETs are meticulously studied in [82]. Offering limited resilience against such attacks the Secure Message Transmission (SMT) and Secure Single Path (SSP) protocols are both introduced in [83]. In SMT a message that is to be sent towards a destination is first divided in N parts and then sent by N independent paths. Each part carries a limited amount of redundancy in such a way that only M parts, where M<N, are needed at the destination to recover the whole message. This means that (N-M) paths can fail to deliver the packet to the destination and the original message will still be received. If fewer than M parts are received at the destination the source resends the missing parts using the previously successful paths. SSP is a specific case of SMT where only one path is used at a time and the source tries a different path each time an acknowledgment is not received. However, SMT is very bandwidth-intensive and these protocols do not attempt to find the source of the packet loss. Our proposed approach, on the contrary, identifies any source(s) that appear(s) to be causing packet losses, allowing for their penalisation at a later stage through an accusation phase.

2.5.3 Anomalous Link Behaviour Detection

In this Section, work proposing approaches to detect anomalous link behaviour, possibly due to a node’s misbehaviour, is reviewed. The common feature of these schemes is that they can pinpoint anomalous behaviour occurring in a link between a node \( u \) and a node \( v \), but they cannot confidently discern which node the offender is since the most likely situation to occur is that node \( u \) blames node \( v \) and vice versa.
A similar system to that in [83] (introduced in the previous section) is proposed in [84], but instead of using a multipath approach all the way between source and destination, it uses a Neighbour Watch System (NWS) which only uses multipath if there is evidence of misbehaviour. In this approach a node has a list of its neighbours as well as of its neighbours' neighbours. In this manner if a node \( u \) forwards a packet to node \( v \) there is a group of sub-watch nodes, i.e. neighbours common to \( u \) and \( v \), which save the packet on their buffers until they overhear node \( v \) forwarding the packet to the next hop. If a timeout period expires, then each sub-watch node forwards the packet to their corresponding next hop gracefully starting a multipath transmission based on evidence of a link's anomalous behaviour. Node \( u \), the primary-watch node, keeps track of node \( v \) and the sub-watch nodes and if none of them forward the packet it selects a next hop different to node \( v \) and starts the process again. This procedure is repeated at each node in the path towards the destination to guarantee the successful delivery of each packet. Although this scheme significantly reduces the network overhead compared to [83], it still fails to identify the source of the packet loss.

The authors of [85] propose a secure traceroute protocol for fixed networks enabling end hosts or routers to locate a link compromised by packet forwarding misbehaviour. In the secure traceroute protocol the traceroute source (i.e. the node performing the behaviour check on routers in the path) sends traceroute packets to each router in the path starting by the closest router and orderly continuing with next hop. Each router receiving the traceroute packet responds including the next hop in its reply so that the traceroute source always knows the expected next hop. To prevent intermediate nodes from distinguishing traceroute packets from normal traffic and tampering with them, the packets are encrypted with a different pair of secret keys by the traceroute source and each router in the path. Routers respond to traceroute packets with some agreed-upon secret information so that the traceroute source can confirm the packet has not been forged or tampered with. Finally, traceroute responses contain a message authentication code to guarantee its origin. This process is then applied iteratively over a path until a compromised link is found or a complete route is cleared of misbehaviour. Although this work has been targeted at fixed networks its principle can be adapted to MANETs.

An acknowledgement-based approach to detect anomalous behaviour in links is proposed in [86]. The authors propose a scheme called 2ACK where nodes send two-hop acknowledgement packets in the opposite direction of the traffic flow. Thus, in a route \( a-b-c-d-e \) node \( c \) sends acknowledgements to \( a \), node \( d \) to \( b \), and node \( e \) to \( c \). However, acknowledgements are not sent for every packet since it would produce a high network overhead; instead nodes \( c,d \) and \( e \) perform a selective acknowledgement and include the ID of the acknowledged packet in the 2ACK packet so that the receiver can identify the arriving packet. The receiver can then maintain a ratio of unacknowledged packets while observing the link over a period of time. If during that
period the ratio goes over a pre-determined threshold then the node can determine that the link between the next two hops is behaving anomalously. For instance, node $b$ could determine that the link $c-d$ is anomalous either because $c$ is not forwarding the packets to $d$, or $d$ is not sending back the two-hop acknowledgements to try to incriminate $c$. A report is then sent to the source for it to start a new route discovery process avoiding including the anomalous link in the new path.

The routing protocol proposed in [87] offers resilience to Byzantine failures (any disruption or degradation of the data forwarding service) by an algorithm that allows the detection of an anomalous link after $\log n$ faults have occurred on a path, where $n$ is the hop length of the path. In this routing strategy nodes maintain a link weight list in which links exhibiting Byzantine failures have larger weights. Then a source intending to transmit data to a destination selects the path which has the overall lowest weight in an attempt to ensure that anomalous links do not constitute part of its path. In [88] the authors propose an approach to detect anomalous link behaviour in a clustered MANET under the control of a central Access Point (AP). In this scheme the destination must send an acknowledgement (ACK) back to the source signed using the TESLA broadcast authentication protocol [79]. Intermediate nodes check the ACK on its way back to the source and keep track of those packets which are not acknowledged. Then in a periodically manner all nodes report to the AP the number packets they have forwarded (PF) and the number of packets left unacknowledged (PU). If all nodes over a path contain the same PF and the same PU it means that the entire path is well behaved. Otherwise, an anomalous link is found between any two adjacent nodes differing in the reported PF and/or PU. The AP can then decide not include anomalous links in future forwarding paths. The authors of this work state that their approach can benefit by using existing reputation systems in order detect misbehaving nodes rather than anomalous links. However, they do not specify how this can be achieved.

A different approach is followed by MARS (Multipath Routing Single Path Transmission) [89] where misbehaviour is detected over a path, not a link. MARS can only work over routing protocols capable of obtaining at least two disjoint paths from source to destination. MARS selects a pair of disjoint paths: the shortest one to send the information and the second to send control packets at the start and end of a communication. When the communication starts the source sends two identical information packets over the two paths with information such as the expected data rate for this communication session. If the destination receives both packets with a time difference equal to or less than a predefined amount of time then the destination assumes that both paths are well behaved. From this point onwards the destination keeps track of the current packet data rate and periodically compares it against the expected packet data rate. If there is a substantial fall in the packet rate, the destination uses the second route to send a control packet to the source notifying about the possible misbehaviour detected over the first path. On receipt the source selects another pair of disjoint paths, or initiates a new route discovery if there are none
available, and starts the whole process again. A negative aspect of this scheme is that in MANETs it is not always possible to guarantee the existence of two disjoint paths from a source to a destination.

The main drawback of the above approaches is that routing survivability is guaranteed by finding problematic links rather than pinpointing problematic nodes, i.e. these strategies do not aim at detecting or penalising misbehaving nodes and consequently fail to tackle the root of the problem. This drawback is addressed in our proposed solution since it detects and effectively penalises misbehaving nodes.

### 2.5.4 Intrusion and Node Misbehaviour Detection

Work presented in this Section also aims at protecting mobile ad hoc networks against the disruption caused by intentionally or unintentionally misbehaving nodes. However, unlike work in the previous section, the main goal here is to detect, and in some cases penalise, any nodes generating the security breach.

In [90] when a node has broken the security mechanisms of a network it is regarded as an intruder. In this scheme an intrusion is detected when a node’s behaviour presents a substantial deviation from the normal behaviour profile displayed by all network nodes. Each node is equipped with an intrusion detection system (IDS) module allowing them to detect intrusion locally or cooperate with neighbouring nodes to further investigate ambiguous intrusion cases. In this approach a node may use its own data to identify another node as an intruder. By contrast, in our approach a node detects anomalies in packet forwarding based on data acquired by other nodes in the network as well as on its own data. IDS modules have also been the object of research in order to detect attack patterns and identify known attacks through abnormal packets.

In [71] and [72] the authors propose a framework for misuse detection which divides the nodes in a network into two categories: insiders and outsiders. Insiders are always well-behaved nodes that belong to trusted users and run the intrusion detection system (IDS) module to detect attacks launched by outsiders through packets with abnormal contents. Outsiders on the other hand can be thought of as users of the network since they are not trusted to carry out route discovery or packet forwarding tasks, thus limiting the number of possible attacks that they can perform. Unfortunately, this framework fails to make use of well-behaved outsiders that could contribute to relevant tasks and rewards misbehaving outsiders by allowing them to use the network. In this regard, our protocol punishes misbehaving nodes by denying them access to the network and its services.

In [91] the authors propose a grammatical evolution approach – an artificial intelligence based learning technique inspired by natural evolution – to detect intruders. In this work an intrusion is
defined as one of a flooding attack, a route discovery disruption attack or a packet dropping attack. The grammatical evolution algorithm receives as an input a sample of network data marked as malicious or non-malicious. Then during the training process the original sample is mutated to produce new samples and the resulting samples go through a crossover procedure to produce yet more samples. After each mutation and crossover iteration the fitness of the resulting samples to address the intrusion detection problem is measured and those that exhibit better results – i.e. samples with greater fitness – are selected to continue the evolution process. Ultimately, the solution with the best fitness to solve the intrusion detection problem is distributed to the network nodes and used to detect any possible intruders. Although this approach always tends towards an optimal solution, its accuracy strongly depends on the initial data set and how well it maps malicious and non-malicious behaviour and the environment where the MANET is deployed. This information is not easy to obtain for unplanned transient networks created on the fly.

There has been some work that aims specifically at protecting data packet forwarding against malicious attacks in order to provide reliable network connectivity. The final part of this Section describes some approaches that detect malicious behaviour in the data forwarding phase. CONFIDANT (Cooperation of Nodes: Fairness in Dynamic Ad-hoc Networks) [92] is a routing protocol that, like Ariadne [78], is based on DSR [61] [62]. In CONFIDANT neighbouring nodes monitor the behaviour of each other, i.e. neighbourhood watch, as they forward packets towards the destination. Any deviation from normal behaviour, e.g. packets not being forwarded, is registered in their reputation system by modifying the rating of the observed node. If the deviation is considered to be persistent then an ALARM is raised and sent to the neighbourhood to be aware of the misbehaving node and to the source for it to start a fresh route discovery process. Reputation knowledge is then used to take action against misbehaving nodes in the route discovery phase either by not including them in the paths created in the network or by not accepting to be in the path of a communication where the source or destination have a very “low” reputation rate. In [93] the authors look at traffic transmission patterns between any two communicating nodes in order to facilitate verification by a receiver. Such traffic patterns can be analysed if they are used in concert with suboptimal techniques at the medium access control (MAC) layer that preserve the statistical regularity from hop to hop. In this scheme nodes can distinguish between a misbehaving node and a congested node, knowing the traffic transmission rates from other nodes to the target node. This work, however, has a very narrow scope of application due to its MAC layer assumptions to preserve statistical regularity, and thus it is very unlikely for it to be useful in scenarios other than military applications.

WATCHERS (Watching for Anomalies in Transit Conservation: a Heuristic for Ensuring Router Security) [94] is a protocol designed to detect disruptive routers in fixed networks through the
analysis of the number of packets entering and exiting a router – i.e. the principle of flow conservation. In this approach each router executes the WATCHERS protocol at regular intervals in order to identify neighbouring routers that misroute traffic and avoid them. To work well the WATCHERS protocol requires the existence of at least one path not affected by disruptive routers between any two well-behaved routers in the network. Although WATCHERS is based on the principle of flow conservation in the same way as will be seen in Section 3.2.1 for our proposed algorithm, its design focuses only on fixed networks and is not applicable to mobile ad hoc networks. Additionally, in our approach the broadcasting nature of the wireless medium allows for multiple possible paths between any two well-behaved nodes.

SCAN (self-organised network layer security in mobile ad hoc networks) [73] focuses on securing packet delivery. It uses AODV [59] [60], but argues that the same ideas are applicable to other routing protocols. SCAN assumes a network with sufficient node density that nodes can overhear packets being received by a neighbour, in addition to packets being sent by the neighbour. SCAN nodes monitor their neighbours by listening to packets that are forwarded to them. The SCAN node maintains a copy of the neighbour’s routing table and determines the next-hop node to which the neighbour should forward the packet; if the packet is not overheard as being forwarded, it is considered to have been dropped. In contrast, in our algorithm nodes do not need to overhear transmissions to and from any neighbour in order to detect misbehaviour. In SCAN each node must possess a valid token to be able to interact with the network and though nodes monitor their neighbours independently, all nodes in a local neighbourhood collaborate with each other to eventually convict a suspicious node by revoking its token. The tokens’ lifetime is determined by a credit strategy that helps reducing the total network overhead. Token renewal and revocation is done through threshold secret sharing and secret share updates. SCAN develops these ideas from [95] where they are used to give a valid key to a new node entering the network and from then onwards to renew its key periodically. Similar techniques have also been studied in various papers such as [96], where they are used to achieve distribution of trust throughout a network. SCAN is similar to our approach in the sense that it does not only detect the source of misbehaviour, but it also punishes any misbehaving nodes. However, SCAN makes use of cryptographic techniques that may prove too resource demanding for devices with limited capabilities.

In [97] a system that can mitigate the effects of packet dropping is proposed. This is composed of two mechanisms that are kept in all network nodes: a watchdog and a pathrater. The watchdog mechanism identifies any misbehaving nodes by promiscuously listening to the next node in the packet’s path. If such a node drops more than a predefined threshold of packets the source of the communication is notified. The pathrater mechanism keeps a rate for every other node in the network it knows about. A node’s rate is decreased each time a notification of its misbehaviour is received. Then, nodes’ rates are used to determine the most reliable path towards a destination,
thus reducing the chance of finding a misbehaving node along the selected path. This work uses DSR but its authors claim that it can be easily adapted to other source routing protocols. However, its applicability has not yet been addressed for distance-vector based routing protocols. Moreover, the watchdog might not detect a misbehaving node in the presence of ambiguous collisions, receiver collisions or nodes capable of controlling their transmission power (how these problems affect the detection of packet forwarding misbehaviour is explained in Section 3.1.3). Such weaknesses are the result of using promiscuous listening to determine whether a node has forwarded a packet or not. Also, using pathrater can be considered a reward for selfish nodes since the flow is diverted towards other nodes in the network while selfish nodes preserve their resources. In order to address these weaknesses the authors of [98] propose a solution that further develops watchdog and pathrater. The main difference is the way in which a node’s rate is calculated. While in watchdog there is a unique rate that is modified based on a node’s misbehaviour, in [98] there are two rate tables: a local rate table and a global rate table. In the local rate table a node calculates the rate for its neighbours based not only on their misbehaviour, but also on their good behaviour – i.e. the rate is in fact a reputation value. The global rate table requires that neighbouring nodes exchange their local rate tables; it maintains an average for each node’s reputation as reported by neighbouring nodes. Later during the routing process an action module, which is the counterpart of pathrater, avoids routing packets through nodes whose reputation indicates that they are misbehaving. Furthermore, the action module avoids forwarding packets on behalf of misbehaving nodes to prevent the system from rewarding them for their misbehaviour. This approach does not suffer from the same limitations of the original watchdog module, such as ambiguous and receiver collisions, since it evaluates nodes over a period of time and only penalise them if their reputation rate crosses a specified threshold. In this respect our approach is also immune to these limitations since it is based on metrics obtained from nodes that are actually sending and receiving packets to and from the node whose behaviour is under evaluation. Additionally, the authors of [98] claim that due to the fact that a node’s reputation is also affected by its good behaviour there is an opportunity for penalised misbehaving nodes to redeem themselves. While our approach denies access to the network to any node that has been accused of misbehaviour – thus discouraging them from dropping packets, whether they are or not admitted back in a network depends solely on the network policies defined by the network administrator (as described in Chapter 4).
Chapter 3

3 Packet Forwarding Misbehaviour Detection and Node Accusation

Ubiquitous computing environments (UCEs) rely on the ability of their nodes (users, devices, sensors or systems) to communicate with each other and cooperate in a coordinated manner to create a pool of services geared towards satisfying their users’ needs. In these environments communication plays an essential role and any malicious behaviour compromising its efficiency or reliability can also jeopardise partially or entirely the overall UCE’s functionality. As stated in Chapter 2, UCEs are especially vulnerable to malicious attacks in areas where their communication is partially or completely based on the mobile ad hoc network (MANET) paradigm. This statement is in line with existing literature which regards mobile ad hoc networking as a paradigm rather than as a specific technology [99] [100]. In MANET-supported areas – which can potentially be the entire network – network nodes rely on each other to forward packets to nodes out of their transmission range, i.e. to perform multihop communication. Forwarding nodes are not necessarily trusted entities and therefore it is important for the network as a whole to possess a means to check that they do not represent a threat capable of hindering its performance or blocking essential network services.

In this Chapter we present our proposed approach which consists of an algorithm that performs two tasks: i) it enables packet forwarding misbehaviour detection through the principle of flow conservation [94], and ii) it enables the accusation of nodes that are consistently detected exhibiting packet forwarding misbehaviour. A node that is accused of misbehaviour is denied access to the network by its peers, which ignore any of its transmission attempts. Thus, misbehaving nodes are isolated from the rest of the network (i.e. isolation is the penalisation for misbehaving nodes). Our scheme is not coupled to any specific routing protocol and, therefore, it can operate regardless of the routing strategy adopted. Our criterion for judging detections on a node is the estimated percentage of packets dropped, which is compared against a pre-established misbehaviour threshold. Any node that drops packets in excess of this threshold is deemed a misbehaving node while those below the threshold are considered to be correctly behaving.
Chapter 3. Packet Forwarding Misbehaviour Detection and Node Accusation

The proposed scheme allows for the detection and accusation of misbehaving nodes capable of launching two known attacks: the black-hole attack and the grey-hole attack, both of which are described in Section 3.1.3. In this Chapter we present our proposed framework, algorithm and protocol to deal with these attacks. We also demonstrate through simulations that an appropriate selection of the misbehaviour threshold allows for a good discrimination between misbehaving and well-behaved nodes. Consequently our approach correctly isolates misbehaving nodes and helps to improve the average network throughput. Finally we show that the proposed solution provides robustness against different degrees of node mobility in a network that is affected by black-hole and/or grey-hole attacks.

This Chapter is organised as follows: Section 3.1 introduces the network model, some terminology used throughout this thesis, and the problems that we seek to address with the proposed approach. Then Section 3.2 presents the packet forwarding misbehaviour detection approach. It starts by formally defining the principle of flow conservation and afterwards it explains how it can be adapted to mobile ad hoc networks by means of some reasonable assumptions. This Section ends with our simulation results and their corresponding analysis. The proposed node accusation strategy is described in Section 3.3 along with a brief explanation of how this strategy can benefit from security techniques such as encryption and digital signatures. Simulation results and their analysis are offered at the end of this Section. Finally, Section 3.4 summarises this Chapter. The misbehaviour detection algorithm (Section 3.2) and the node accusation algorithm (Section 3.3) were originally published in [101] and [102] respectively.

3.1 Model Assumptions, Terminology & Addressed Problems

3.1.1 Model Assumptions

The physical layer of a wireless network is often vulnerable to attacks such as frequency interference. Spread spectrum and frequency hopping are examples of techniques that have been studied as means of preventing this type of attacks [103] [104]. The link layer is also subject to attacks where nodes gain unfair access to the medium by ignoring the back-off time, or where they disrupt communications by dropping packets related to typical handshake processes. Approaches have been proposed that tackle link layer misbehaviour in wireless networks [105] [106]. Attacks aimed at the physical and link layers are out of the scope of this work.

We assume bidirectional communication symmetry in every direct link between a pair of nodes. This means that if a node \( v_2 \) receives a packet from node \( v_1 \), \( v_1 \) can also receive a packet from \( v_2 \). This is a sensible assumption since our approach needs MAC (Medium Access Control) protocols with collision avoidance mechanisms to work properly, such as the extensively deployed IEEE
802.11 [107], MACA (Multiple Access with Collision Avoidance) [108] and MACAW (MACA for Wireless LANs) [109], which require bidirectional communication for packet transmission.

We assume that the MAC layer protocol makes use of handshake techniques similar to those available in the IEEE 802.11 standard, where successfully transmitted packets are acknowledged. This is required to provide confidence that a data packet has been successfully transmitted to the next-hop node, and enables us to apply the principle of flow conservation (Section 3.2.1).

We also assume that all nodes in the network are adapted with wireless interfaces that support promiscuous mode operation. This operational mode allows a node to process all transmissions from nodes within hearing range. This is required in order to determine active nodes in a node’s neighbourhood and schedule events to check their behaviour at a later stage. However, the data used to determine a node’s behaviour are not collected by this means.

At the network layer we assume that nodes misbehave by dropping packets despite having agreed to forward them during route discovery. Other types of misbehaviour are not taken into account in this Chapter, including attacks by colluding nodes which are considered in Chapter 5.

This thesis does not address security measures for the proposed approach since it focuses on the basic mechanisms to detect and accuse misbehaving nodes. However, cryptographic techniques such as threshold secret sharing and secret share updates used in SCAN [73] could be used as viable ways of protecting the detection and accusation protocols (Section 3.2 and Section 3.3).

3.1.2 Terminology

We use the term neighbour to refer to a node that is within the direct wireless transmission range of another node. From this, it follows that both nodes are able to establish a bidirectional communication. Likewise, the term neighbourhood refers to all nodes that are neighbours of a particular node. A node is not a neighbour of itself and, therefore, a node does not belong to its own neighbourhood.

We use the term detection to mean that our algorithm has identified that a node appears to be misbehaving. A detection is based on a single check of a node’s behaviour. An accusation, on the other hand, occurs when a node reports another node as misbehaving. It is our view that an accusation should be based on more than a single detection to increase confidence in the assessment, as we discuss in Section 3.3. Additionally, in Section 3.3.3 it is shown how the number of detections needed to raise an accusation affects the percentage of nodes correctly accused of misbehaviour.

A misbehaving node is a network node – regardless of whether they are overloaded, broken, compromised, congested, selfish or malicious, as explained in Section 2.5.1 – that agrees to
forward packets on behalf of other network nodes in the route discovery stage but that instead launches a black-hole or gray-hole attack as described in Section 3.1.3

A misbehaving node is represented by a given drop characteristic. In our simulations, misbehaving nodes drop packets in a statistical manner with a predefined dropping probability, e.g. dropping packets with 30% probability.

### 3.1.3 Addressed Problems

Although the main goal of our approach is to detect black-hole and grey-hole attacks aimed at disrupting the packet forwarding functionality of a routing protocol, we also seek to address various weaknesses exhibited by some previously proposed misbehaviour detection strategies. In this Section we describe the attacks and weaknesses that the proposed protection scheme addresses.

To start with, Figure 3-1 depicts a typical black-hole attack. In this attack a misbehaving node drops all the packets that it receives instead of normally forwarding them. In the figure the nodes $v_1$, $v_2$, $v_3$, $v_4$ and $v_5$ are part of a path through which UDP and TCP packets are being forwarded. For simplicity the figure depicts a path that forwards packets in only one direction, from $v_1$ to $v_5$. In this path the malicious node $v_3$ fails to forward any packets to node $v_4$, thus bringing the path throughput to zero. Moreover, despite the fact that link $v_3$-$v_4$ behaves as a broken link, node $v_1$ is left unaware of the situation since no route errors are generated. Thus, this attack can continue indefinitely if a misbehaviour detection strategy is not in place.

![Figure 3-1 Black-hole attack](image)

Figure 3-2 shows two types of grey-hole attacks. A grey-hole attack is a "clever" variation on the black-hole attack; it can essentially be of two types: selective dropping of packets or probabilistic dropping of packets (or a combination of both). In the selective dropping of packets attack a malicious node targets a specific protocol and drops all its packages while it continues forwarding packets for other protocols, e.g. dropping all UDP packets while forwarding TCP packets as malicious node $v_3$ does in Figure 3-2-a, perhaps because it knows that UDP transmissions do not have flow control. In the probabilistic dropping of packets attack a misbehaving node drops packets in a statistical manner, e.g. dropping packets with 50% probability or dropping them according to a probabilistic distribution. Figure 3-2-b depicts an attack that combines selective and probabilistic dropping, in this attack the malicious node $v_3$ is more likely to drop UDP packets.
than TCP packets. The main goal of both types of grey-hole attacks is to disrupt the network without being detected by the security measures in place.

The proposed packet forwarding misbehaviour detection and node accusation approach also addresses several problems that have been identified in previously proposed schemes due to their reliance on overhearing to collect an evaluated node’s essential behaviour data. For instance the work proposed in [97] suffers from ambiguous collisions and receiver collisions problems, and is vulnerable to nodes capable of controlling their transmission power. Work presented in [98] also suffers from the same problems but the authors claim that their improved strategy accommodates for such occurrences by evaluating a node’s behaviour over a long enough period of time. An ambiguous collision (Figure 3-3, also known in the literature as the hidden terminal problem) occurs when a node $v_2$ is trying to determine if another node $v_3$ is properly forwarding a packet. It may happen that node $v_3$ forwards the packet to a further node $v_4$, which is out of the transmission range of $v_2$, while a second transmission initiated by node $v_5$ prevents $v_2$ from overhearing the forwarded packet, thus $v_2$ will not know if the packet was forwarded.

On the other hand, in the receiver collision problem a node $v_2$ forwards the packet to $v_3$ at which point a collision occurs with $v_4$, as in Figure 3-4. Node $v_5$, which is trying to determine if $v_2$ is properly forwarding packets, is unaware of such a collision and assumes that the packet was forwarded even if $v_2$ does not attempt a retransmission.

Another common problem is caused by nodes capable of controlling their transmission power (Figure 3-5). Thus, if a node $v_2$ is trying to determine whether a node $v_5$ is correctly forwarding packets, node $v_2$ can transmit with enough power for $v_5$ to overhear but not enough power for $v_7$ to receive the packets, leaving $v_7$ unaware of the situation.
The SCAN strategy [73] described in Section 2.5.4 can also suffer from the same problems described above. However, in highly dense networks SCAN has a better chance of withstanding some of the attacks since more than one neighbour can monitor an under-evaluation node, but this is not always possible to guarantee. Figure 3-6 depicts a scenario in which monitoring neighbours $v_5$ and $v_6$ are left unaware of a receiver collision occurring at node $v_3$ (because $v_5$ and $v_6$ are out of range of $v_3$), thus assuming that node $v_2$ has forwarded the packet on behalf of node $v_3$ even if $v_2$ does not try retransmitting the packet.

The above misbehaviour detection weaknesses, which can be used by malicious nodes to disrupt the network, are due to the fact that overhearing is used by nodes to determine misbehaviour in other nodes. In Section 3.2 we present our strategy to detect nodes exhibiting misbehaviour and we demonstrate how by coupling the proposed scheme with a MAC layer that acknowledges successfully transmitted packets immunity against the above problems is provided.

Another common problem in approaches such as watchdog and pathrater [97] or the schemes reviewed in Section 2.5.3 (Anomalous Link Behaviour Detection) is that they reward misbehaving nodes by diverting the network flow towards other nodes and fail to punish their
misbehaviour. Continuing to forward packets on behalf of misbehaving nodes can encourage them to misbehave since it results in less forwarding work and no penalties at all. Figure 3-7 illustrates this situation in a network where well-behaved nodes reward misbehaving node $v_i$ first by diverting network traffic through nodes $v_7$, $v_8$, and $v_9$, which permits $v_i$ to save its resources (e.g. battery life); and secondly by allowing nodes $v_2$ and $v_3$ to continue forwarding packets on behalf of $v_i$ in spite of its already detected misbehaviour.

The proposed approach penalises misbehaving nodes by isolating them from the rest of the network. In this way nodes are discouraged from misbehaving since they may lose access to the network services as explained in Section 3.3.

### 3.2 Packet Forwarding Misbehaviour Detection

Our work provides a novel methodology to secure the data forwarding functionality in multihop wireless UCEs. We propose an approach that takes advantage of the principle of flow conservation in a network. This states that all bytes/packets sent to a node, and not destined for that node, are expected to exit the node. In this Section we first present, from a theoretical point of view, how this principle works assuming it is implemented in an ideal network, and then we demonstrate that by making some reasonable assumptions and adaptations, our algorithm can cope with the practical problems that are encountered in real MANET-like environments.

#### 3.2.1 The Principle of Flow Conservation

We now formally introduce the principle of flow conservation over an ideal static network model:

- Let $v_j$ be a node such that $v_j \in V$, where $V = \{v_I, v_2, v_3 \ldots v_N\}$ is the set of all nodes in the network, $N$ is the total number of nodes in the network, and $j = 1, 2, 3 \ldots N$.

- Let $U_j$ be the subset of nodes in the network which are neighbours of $v_j$, i.e. $U_j$ is the neighbourhood of $v_j$. It follows that $v_j \notin U_j$ and also $U_j \subseteq V$.

- Let $\Delta t$ be the period of time elapsed between two points in time $t_0$ and $t_1$ such that $\Delta t = t_1 - t_0$.
Let $T_y$ be the number of packets that node $v_i$ has successfully sent to node $v_j$ for $v_i$ to forward to a further node; $v_i \in U_j$, $v_j \in U_i$, $i \neq j$ and $T_y(t_0) = 0$. $T_y$ can be read as “packets transmitted from $v_i$ to $v_j$ for $v_i$ to forward”.

Let $F_{jj}$ be the number of packets not originated at node $v_i$ that node $v_j$ has successfully forwarded to node $v_i$; $v_i \in U_j$, $v_j \in U_i$, $i \neq j$ and $F_{jj}(t_0) = 0$. $F_{jj}$ can be read as “packets forwarded from $v_j$ to $v_i$”.

If all nodes $v_j \in V$ remain static for a period of time $\Delta t$ during which no collisions occur in any of the transmissions over an ideal (noiseless) wireless channel, and provided that all packet transmissions are executed within $\Delta t$, then for a given intermediate node $v_j$: 

$$\sum_{\forall v_i \in U_j} F_{jj}(\Delta t) = \sum_{\forall v_i \in U_j} T_y(\Delta t) \quad (3-1)$$

Equation 3-1 states the fundamental premise of the principle of flow conservation in an ideal static network applied to packets rather than raw bytes. It states that if all neighbours of a node $v_j$ are queried for i) the amount of packets sent to $v_j$ to forward and ii) the amount of packets forwarded by $v_j$ to them, the total amount of packets sent to and received from $v_j$ must be equal.

In practice networks exhibit conditions that are far from ideal. First of all, the wireless channel is error prone and packets get lost while in transit. Secondly, collisions happen when the network uses protocols where nodes have to compete for the medium, such as when the link layer protocol is based on the distributed coordination function (contention period) of the IEEE 802.11 a/b standard. In order to allow Equation 3-1 to hold we assume a MAC protocol such as IEEE 802.11, MACA or MACAW as described in Section 3.1.1.

If the MAC protocol at the link layer acknowledges each successfully transmitted packet, then the transmitter and receiver can maintain synchronised values of their metrics $T_y$ and $F_{jj}$. For instance, when node $v_1$ needs to forward a packet to $v_2$, $v_1$ sends an RTS frame and $v_2$ replies with a CTS frame. Following the reception of the CTS, $v_1$ sends the data which may collide at the receiver with the transmission of some other node $v_3$ that heard neither the RTS nor the CTS frame for example because $v_3$ has just moved into range. In this case $v_2$ does not increase its $F_{12}$ (forwarded from $v_1$ to $v_2$) metric because it did not receive the data, and $v_1$ does not increase its $T_{12}$ (transmitted from $v_1$ to $v_2$) because the packet was never acknowledged. Even in the eventuality that an ACK frame gets lost the nodes would realise the error when $v_1$ retransmits the data. In this case, $v_2$ increases its $F_{12}$ metric the first time it receives the data packet and sends back the respective ACK frame. Node $v_1$ does not increase its $T_{12}$ metric since it does not receive the ACK frame and instead it retransmits the packet, sending an RTS frame and waiting for a CTS frame. The second time that $v_2$ receives the packet it will notice that the packet has been already received.
by checking the sequence control field in the MAC header, so it does not increase its $F_{12}$ metric and it acknowledges again the packet as stipulated in the 802.11 standard. When $v_j$ receives the ACK it will increase its $T_{12}$ metric and Equation 3-1 holds again.

The use of a MAC protocol that acknowledges successfully transmitted packets in conjunction with the conservation of flow principle means that we are not susceptible to problems that arise when overhearing other nodes' transmissions. Thus, problems such as ambiguous collisions, receiver collisions, and the ability of a node to control its transmission power do not exist in our approach. In our algorithm the nodes keeping the data used to determine whether the forwarding was properly made are the nodes actively involved in the transmission process, i.e. the transmitter and the receiver of each transmission.

However, a node may exhibit malicious behaviour even if it is not purposefully doing so. For example, an overloaded node may temporarily lack the CPU cycles, buffer space or bandwidth to forward packets. In addition, some reactive routing protocols, e.g. AODV, cause buffered packets to be dropped if they go through a path that is even just temporarily unavailable. For these reasons Equation 3-1 cannot be applied in a rigorous manner and a threshold needs to be established to account for packets dropped by a node through no fault of its own. Equation 3-2 reflects this change:

$$\sum_{\forall t, y \in U_j} F_{12}(\Delta t) \geq (1 - \alpha_{\text{threshold}}) \sum_{\forall t, y \in U_j} T_{12}(\Delta t)$$

Equation 3-2 implies that well-behaved nodes are required to forward at least some fraction of the total packets transmitted to them to forward. Any nodes unable to fulfill the criteria are deemed to be misbehaving. The factor $(1 - \alpha_{\text{threshold}})$ sets the minimum fraction of packets that a node must forward in order to avoid detection. However, we prefer to analyse a node's behaviour in terms of the $\alpha_{\text{threshold}}$ factor since it represent a more tangible concept, i.e. $\alpha_{\text{threshold}}$ represents the network misbehaviour threshold which is the maximum fraction of packets that a node can drop without misbehaving. The misbehaviour threshold can take values between 0 and 1 and as we shall see plays an important role in the detection power of our proposed algorithm, i.e. the capability of the algorithm to detect misbehaving nodes. The lower $\alpha_{\text{threshold}}$ is the more likely it is that our algorithm detects any malicious behaviour. However, it also means that the probability of a wrong detection increases, as will be observed in our simulations (Section 3.2.3). A wrong detection occurs when the result of a single evaluation of a node mistakenly determines that the node appears to be misbehaving. Therefore, fine tuning is required to reach a fair point in this trade-off.
In MANET-like UCEs the neighbourhood $U_j$ of a node $v_j$ changes dynamically over time, making it difficult to determine those nodes that have transmitted or received packets to or from a node $v_j$. The proposed scheme overcomes this problem by means of a limited broadcast that tracks down nodes that have been in contact with node $v_j$ as explained later in this Section. Every node in the network is required to keep a table – the overheard nodes table – with the IDs of those nodes that have been overheard recently through promiscuous listening. Entries are removed once they go stale (e.g. if a node in the table has not been overheard in the last $t$ seconds). This process helps a node $v_j$ to keep track of nodes that have become part of its neighbourhood $U_j$ while they were actively intervening in the network.

The core parts of our algorithm are detailed in the pseudocode shown in Figure 3-8. A node $v_i$ maintains a table with two metrics $T_{ij}$ and $F_{ji}$, which contains an entry for each node $v_j$ to which $v_i$ has respectively transmitted packets to or received packets from. Node $v_i$ increments $T_{ij}$ on successful transmission of a packet to $v_j$ for $v_j$ to forward to another node, and increments $F_{ji}$ on successful receipt of a packet forwarded by $v_j$ that did not originate at $v_j$ (Figure 3-8-a). All nodes in the network continuously monitor their neighbours and update the list of those they have heard recently (Figure 3-8-b). If the ID of an overheard node is not included in the table of overheard nodes a new entry is created. Otherwise, the existing entry is updated with a timestamp corresponding to the time the node was last overheard. Upon the creation of a new entry, a node schedules a task (or event) to check the behaviour of the node whose address or ID has been saved in the new entry. Nodes randomly select a period of time between $T_{\text{min}}$ and $T_{\text{max}}$ to schedule a behaviour check task. This random selection seeks to reduce the possibility of two or more nodes starting a behaviour check on the same node at the same time, wasting network bandwidth, battery energy and other network resources.

When a scheduled task is triggered in node $v_k$ to check $v_j$'s behaviour (Figure 3-8-c), node $v_k$ broadcasts a metrics request packet (MREQ) with TTL = 1 in the IP header. An MREQ includes the ID of the node originating the request (SRC_ID), the ID of the node whose behaviour is to be checked (CHK_ID), an MREQ_ID and a timestamp indicating the time at which the task was triggered. The MREQ_ID is used in the same way as in some routing protocols which base their route discovery phase on broadcasting. If a node sees an MREQ that has the same MREQ_ID and SRC_ID of a packet seen before, the MREQ is dropped. This technique prevents flooding packets from traversing a zone of the network more than once.
a. MONITORING
if node $v_i$ successfully transmits a packet to node $v_j$ for $v_j$ to forward
  . increase $T_y$ by one
endif
if node $v_i$ receives a packet successfully forwarded by node $v_j$
  . increase $F_{jj}$ by one
endif

b. OVERHEARING
if node $v_i$ overhears a node $v_j \in U_k$
  . if node $v_j$ is not in $v_i$'s table of overheard nodes
    . add node $v_j$ to $v_i$'s table of overheard nodes
    . schedule an event to check $v_j$'s behaviour
  . else
    . update last time node $v_j$ was heard
  . endif
endif

c. INITIATE BEHAVIOR CHECK
if in node $v_i$ an event to check node $v_j$'s behaviour is triggered
  . send a metrics request packet (MREQ) with node $v_j$'s ID
  . schedule another event to check $v_j$'s behaviour again at $(t + T_{safe})$
endif

d. REQUEST HANDLING
if node $v_k$ receives a metrics request for node $v_j$
  . if node $v_k$ has node $v_j$ in its table of overheard nodes
    . rebroadcast metrics request packet (MREQ)
    . reschedule any event to check $v_j$'s behaviour
    . if node $v_k$ has metrics for node $v_j$
      . send a metrics reply (MREP) back to the requesting node
    . . endif
    . else
      . ignore request
    . . endif
  . else
    . ignore request
  . endif
endif

e. REPLY HANDLING
if a request was sent out
  . while there are more replies to be received for node $v_j$
    . receive reply
    . acknowledge reply reception (send MACK)
    . add received metrics to totals
    . endwhile
  . if $\sum_{\forall v_i, v_j \in U_j} F_{ji} \geq (1 - \alpha_{threshold}) \sum_{\forall v_i, v_j \in U_j} T_{ji}$
    . node $v_j$ is not misbehaving (non-detection)
  . else
    . node $v_j$ is misbehaving (detection)
  . endif
endif

Figure 3-8 Pseudocode of our proposed detection algorithm
Chapter 3. Packet Forwarding Misbehaviour Detection and Node Accusation

The timestamp of an MREQ packet, on the other hand, is used to resolve conflicts when two nodes start a behaviour check on the same node at almost the same time. In such cases, nodes can see which packet corresponds to the earliest triggered task and disregard the other. This does not require accurate synchronisation of the nodes’ clocks; approximate synchronisation is all that need be assumed. Finally, after the MREQ packet is broadcast, a task is scheduled to be triggered a period of time $T_{\text{max}}$ (maximum elapsed period of time without checking an active node’s behaviour) later. This means it is highly unlikely that the same node will originate two successive behaviour checks for another node, and gives other nodes a chance to perform the behaviour check.

The handling of requests (Figure 3-8-d) is the heart of the limited broadcast algorithm. When a node receives an MREQ it first checks if the CHK_ID is in its table of overheard nodes; if it is not the node ignores the MREQ and discards the check. However, if the CHK_ID appears in its table then it rebroadcasts the MREQ with TTL = 1 in the IP header. Setting the TTL to one allows the algorithm to control how far the broadcast of the MREQ is to go, instead of leaving this task to the IP protocol. Thus, every MREQ travels only one hop at a time, and is then analysed and rebroadcast if the protocol so determines. By following this algorithm, our protocol is capable of tracking down nodes that have been in contact with the checked node, as illustrated in Figures 3-9 and 3-10.

![Figure 3-9 Overhearing – nodes overhear node $v_7$ as it changes position](image-url)
Chapter 3. Packet Forwarding Misbehaviour Detection and Node Accusation

Node performing behaviour check

Nodes that have overheard \( v_7 \)

Limited MREQ broadcast

Figure 3-10 Limited Broadcast - node \( v_8 \) starts a behaviour check on node \( v_7 \) and tracks down nodes that have overheard it

We assume that in Figures 3-9 and 3-10 transmissions can be overheard by vertically, horizontally and diagonally adjacent nodes. In Figure 3-9, node \( v_7 \) is first in position \( a \) where it can be overheard by nodes \( v_1, v_2, v_3, v_6, v_{11}, v_{12} \) and \( v_{15} \). Each of these nodes makes an entry in their table of overheard nodes when \( v_7 \) first transmits and each of them schedules a task to check its behaviour. At some point in time, \( v_7 \) decides to move, following the path depicted in Figure 3-9 and coming in contact with nodes \( v_7, v_{17}, v_{18}, v_{19}, v_{20}, v_{23}, v_{24} \) and \( v_{25} \). It finally stops in position \( b \).

In this example the scheduled behaviour check initiation task (Figure 3-8-c) in node \( v_8 \) is the first to be triggered and the limited broadcast commences, as shown in Figure 3-10. All nodes that have overheard node \( v_7 \) re-broadcast the MREQ, whereas nodes such as \( v_4, v_9 \) and \( v_{15} \) also receive the MREQ but ignore it because they have not overheard node \( v_7 \).

It may also happen that node \( v_7 \) stops transmitting and receiving packets before it moves to a different network area. Then, after moving, \( v_7 \) may become active again forming a new neighbourhood. In this case, the old and new neighbourhoods are not connected by nodes that have overheard \( v_7 \) and, therefore, a limited broadcast triggered in one neighbourhood will not reach the other. In spite of this, our algorithm still works properly because two independent behaviour checks will be performed on \( v_7 \): one at its old neighbourhood and another at its new neighbourhood. The outcome of each of these behaviour checks depends on the behaviour exhibited by \( v_7 \) at each neighbourhood.

50
Once a node has decided whether to continue or not broadcasting a MREQ, it reschedules any pending task to check the behaviour of the checked node specified in the CHK_ID field of the MREQ. The new behaviour checking task is scheduled in the same way as when a new entry is made in the table of overheard nodes, i.e. a period of time is randomly selected between $T_{\min}$ and $T_{\max}$. In this way if the random selection is uniformly distributed, the average frequency with which an active node's behaviour is checked is:

$$\text{avg freq} = \frac{1}{(T_{\min} + T_{\max})/2} = \frac{2}{(T_{\min} + T_{\max})}$$  \hspace{1cm} (3-3)

The last thing a node does when it receives a MREQ is to check if it has any metrics $P_\ell$ or $T_y$ relating to the node being checked. If any of the metrics has a value other than zero the node returns a metrics reply packet (MREP) (Figure 3-8-d) containing the metrics, but if the value of both metrics is zero then the node does not send back any response. A metrics reply packet is returned to the node that originated the MREQ following the reverse of the MREQ's path. This requires nodes to set a backward pointer when they receive a previously unseen MREQ. We have adopted this approach to return MREP packets since it creates much less network overhead than an approach where MREPs are trusted to a reactive network protocol. The same reason does not necessarily apply to proactive routing protocols but reactive protocols need to be accounted for since the approach aims at being routing protocol independent. Thus, if every MREP packet were to be trusted to a reactive routing protocol, e.g. AODV, this would result in several new broadcast-based route discovery processes being executed per each behaviour check. In a network with a few active nodes forwarding packets on behalf of various sources these simultaneous route requests would increase the network load substantially.

Reply handling is executed in the node that initiated the MREQ. This node, $v_8$ in Figure 3-10, waits for a period of time in order to give all nodes with metrics about the checked node the opportunity of replying. When the time expires, the node checks the behaviour of the analysed node by verifying that Equation 3-2 holds (Figure 3-8-e). If it does not, it flags the checked node as a misbehaving one; this is a detection. Using a single detection to accuse a node is not sufficient since such an algorithm may lead to wrong accusations against correctly behaving nodes. A scheme in which multiple detections by different nodes are necessary to accuse a node is fairer to well-behaved nodes, while keeping a high probability of correctly accusing misbehaving nodes. The proposed method to accuse misbehaving nodes is presented in Section 3.3.

Due to the nature of the algorithm nodes are not perfectly synchronised with each other. A MREQ will reach close nodes faster than nodes placed a few hops away. The last nodes to receive the MREQ have enough time to send or receive some extra packets to and from the analysed node, thus unbalancing the values of the $T_y$ and $P_\ell$ metrics. This discrepancy, in which some packets
may have been sent to the node being analysed but not yet forwarded by it, is accommodated by $\alpha_{\text{threshold}}$.

A problem that has been detected in our simulations has its roots in the dynamic nature of multihop wireless UCEs. Nodes receiving a metrics request (MREQ) send a reply if they have non-zero metrics for the checked node. However, such replies sometimes get lost due to collisions, noise in the wireless channel or link/path breakages due to the mobility of the nodes. If the value of the metrics contained in the lost reply is small compared to the total obtained after adding up the replies that do not get lost, $\alpha_{\text{threshold}}$ can accommodate them and Equation 3-2 holds. Unfortunately, this is not always the case and some of those replies that get lost contain key information for the calculations and the checked node is then wrongly detected as misbehaving. This is one of the reasons why an accusation should not be made based on a single detection. Using a single detection to accuse a node is not sufficient since such an approach may lead to wrong accusations against correctly behaving nodes. To circumvent the lost replies problem we propose an optional module to our algorithm. A node receiving an MREP as it is forwarded towards its destination (i.e. towards the node performing the behaviour checking task) will also send back a metrics acknowledgement packet (MACK). Thus, nodes sending/forwarding an MREP wait for an MACK from the next hop in the route. If the confirmation does not arrive then they retransmit the MREP. The process is repeated up to $\text{MAX}_{\text{Retx}}$ retransmission retries before giving up. The results obtained in our simulations have demonstrated that this technique can significantly improve the results in MANETs with a high degree of mobility. Simulations have also shown that the most significant improvement can be seen when comparing the results for $\text{MAX}_{\text{Retx}} = 0$ (without retransmitting any reply) and $\text{MAX}_{\text{Retx}} = 1$. Subsequent increases to $\text{MAX}_{\text{Retx}}$ improve the results further but not significantly.

### 3.2.3 Evaluation Results and Analysis

We perform our simulations using the GloMoSim simulation package [48] [49]. The results presented for each value are the average of 10 simulation runs. Tests with a larger number of simulations (e.g. 20) give results that vary typically no more than 1% from those presented here. Unless explicitly stated otherwise our simulation parameters take the following values: i) nodes move according to the random waypoint mobility model with a speed randomly chosen with uniform distribution between 0m/sec and 10m/sec, this yields a mean node speed of 5m/sec and a speed standard deviation of 2.89m/sec, ii) the pause time takes a value that is exponentially distributed with mean 30 seconds, iii) the wireless transmission range of every node is 100 metres, iv) the link capacity is 2 Mbps, v) the MAC layer protocol is the IEEE 802.11 DCF, vi) the underlying routing protocol is AODV, and vii) the total simulation time for each scenario is 1800 seconds.
Chapter 3. Packet Forwarding Misbehaviour Detection and Node Accusation

An important parameter to evaluate the effectiveness of an approach that detects and accuses misbehaving nodes is its detection power. In this Section we present results that demonstrate that our approach has a high probability of detecting truly misbehaving nodes while maintaining a low probability of performing wrong detections, i.e. wrongly detecting a well-behaved node as a misbehaving one. For this set of results the network was set-up with 40% of its total nodes misbehaving by not forwarding all packets. This value corresponds to a reasonable worse case scenario. If more than half the network nodes were to misbehave, then their misbehaviour would be considered the normal network behaviour as they would represent the majority of the network participants. Nodes check the behaviour of active nodes within a period chosen uniformly between 40 and 60 seconds, and keep any overheard node in their tables for 120 seconds after the last time they are heard. The principal metric in our tests is the percentage of detections and it is assessed in terms of misbehaviour threshold and node speed.

We initially consider our misbehaviour detection algorithm in terms of the misbehaviour threshold, which is the parameter $\alpha_{\text{threshold}}$ in Equation 3-2, i.e. the maximum percentage of packets that a node is allowed to drop without being detected as a misbehaving node. In order to see the effect of the misbehaviour threshold on nodes, simulations were carried out with networks containing 20 and 60 nodes, and areas of $40000m^2$ ($200m \times 200m$) and $120000m^2$ ($346.41m \times 346.41m$) respectively. These values ensure that node density is the same in both scenarios. We varied both the packet drop probability of misbehaving nodes and the misbehaviour threshold between 0% and 100%.

![Figure 3-11 Percentage of detections vs. misbehaviour threshold (20 node network)](image)
Figures 3-11 and 3-12 show the percentage of detections as a function of the misbehaviour threshold for nodes exhibiting different probabilities of misbehaviour for networks with 20 and 60 nodes respectively. It can be inferred from both graphs that the criterion to select an adequate misbehaviour threshold $\alpha_{\text{threshold}}$ depends on the level of trust required in the network as well as on network characteristics such as network size. The lower the threshold is the greater the fraction of packets that nodes need to forward to be considered well behaved. However, since characteristics inherent to MANETs such as mobility and the noisy wireless medium can cause some packets to be lost (including packets of our own protocol), a lower value of $\alpha_{\text{threshold}}$ also means that an increasing number of correctly behaving nodes can be wrongly detected as misbehaving ones. A similar problem occurs with misbehaving nodes that drop a small percentage of packages, e.g. less than 10% of packets. By comparing the 10% Misbehaviour Drop curve with the No Misbehaviour curve we can see how the less misbehaviour a node exhibits the more its curve resembles that of a well-behaved node (i.e. $d = 0$), making distinguishing between them a complex task.

Finally, it can also be seen from Figures 3-11 and 3-12 that selecting a misbehaviour threshold equal to a node’s misbehaving probability prevents our approach from identifying misbehaving nodes with certainty, i.e. the probability of detection is approximately 50%. These occurrences are seen for each curve at points for which the misbehaviour threshold has a value close to a node’s packet drop probability. Selecting an acceptable or tolerable level of misbehaviour $x$ in a network is a policy decision. This policy then allows a value of $\alpha_{\text{threshold}}$ to be set depending on the desired
detection probability. For example, for a detection probability of 90% or greater our results suggest that $\alpha_{\text{threshold}}$ should then be set at approximately 10% lower than the nodes’ packet dropping probability for the 20 node network and 20% lower than the nodes’ packet dropping probability for the 60 node network. This aspect will be considered further in Section 4.2.2.

Figure 3-13 is a subset of the data shown in the above figures and corresponds to the curves for well-behaved nodes in 20 and 60 node networks. In spite of the fact that the curves belong to networks with the same node density, the figure clearly shows that the number of nodes in a network has an effect on the precision of our approach. There is a greater likelihood of wrongly detecting well-behaved nodes as misbehaving in a 60 node network than in a 20 node network. This occurs because in larger networks it is more probable that the average distance between source and destination is greater, thus requiring more nodes to be involved in the packet forwarding process in order to reach the destination. This consequently leads to lower path stability, i.e. there are more nodes likely to move out of the path at any instance causing link breakages and forcing new route discovery processes, which in AODV are performed through broadcasting methods. Additionally, more active nodes mean more behaviour checks, more limited broadcast procedures, and potentially more key metric reply packets (MREP) being lost to packet collisions. This problem is exacerbated because at the MAC sub-layer broadcasting procedures do not employ handshaking techniques.

![Figure 3-13 Percentage of Detections for non-misbehaving networks with 20 and 60 nodes](image)

Our second set of results assesses the performance of our misbehaviour detection algorithm in terms of the degree of mobility of the nodes in the network. This time the misbehaving nodes drop
packets with 50% probability while the misbehaviour threshold is kept at 40% ($\alpha_{\text{threshold}} = 40\%$). The mean node speed varies between 0m/sec (a static network) and 20m/sec with nodes randomly selecting a speed in the range given by mean node speed $\pm 1m/sec$, which results in a speed standard deviation of 0.58m/sec for all measurements. Whereas Figure 3-14 is plotted for misbehaving and well-behaved nodes in a 20 node network, Figure 3-15 is plotted for misbehaving and well-behaved nodes in a 60 node network.

![Figure 3-14 Percentage of detections vs. mean node speed (20 node network)](image)

It can be seen from Figure 3-14 that the misbehaviour detection protocol is not significantly affected by the speed of the nodes in small networks. Our approach robustly keeps a gap between misbehaving nodes and well-behaved nodes making it easy to spot nodes that are purposefully violating the principle of flow conservation. The fluctuations seen in both curves are likely to have occurred due to the sporadic losses of metrics reply packets (MREP) rather than the node speed. However, what has been said for small networks does not apply for large scale networks, as can be appreciated from Figure 3-15. As the mean node speed increases the gap between misbehaving and correctly behaving nodes grows smaller. This can be due to a higher network overhead introduced during the route discovery (i.e. expanding ring broadcast by AODV) and metrics request phase (i.e. limited broadcast by our approach), as they have larger areas and number of nodes to cover. As the speed increases, link breakages occur more often and new broadcast-based requests are generated. This in turn leads to a higher probability of metrics replies (MREP) being lost to packet collisions, thus degrading the accuracy of the proposed protection scheme. Nevertheless, a good level of discrimination is maintained. These results support our
hypothesis that using a single detection to accuse a node is not sufficient since such an algorithm may lead to wrong accusations against correctly behaving nodes.

![Figure 3-15 Percentage of detections vs. mean node speed (60 node network)](image)

Finally we assess the network overhead generated by our misbehaviour detection algorithm in a 20 node network. In this set of simulations misbehaving nodes drop packets with 50% probability, the misbehaviour threshold \( \alpha_{\text{threshold}} \) is 40% and the node speed varies between 0m/sec (a static network) and 20m/sec. Figure 3-16 shows the total network resources used, measured by the number of packets sent over each link during the entire simulation. Figure 3-17 displays the mean number of packets sent over each link per behaviour check. In both figures, the total network resources are calculated by adding one each time a packet crosses a different link: thus a MREQ packet broadcast that traverses three hops (links) contributes three packet-links to the total.

The total overhead in Figure 3-16 is the sum of the overhead produced by the MREQ, MREP and MACK packets. It is least in a stationary network and increases with the mean node speed. This increase is due to the fact that in highly dynamic networks more nodes actively intervene in the communication process because new paths are constantly being formed. Subsequently, behaviour checks have to be performed on a greater number of nodes. This observation is confirmed by the data in Figure 3-17. It shows that the average number of packets per behaviour check in dynamic networks is about 65 packets and it seems to be approximately independent of the mean node speed. Thus, if the network overhead increases but the number of packets per behaviour check remains the same then the network overhead growth must be due to a greater number of checks being performed in the network.
Chapter 3. Packet Forwarding Misbehaviour Detection and Node Accusation

It can also be seen from Figures 3-16 and 3-17 that the MREQ overhead represents the main contribution to the total algorithm overhead. This is expected since the dissemination of MREQ packets is based on broadcast, albeit limited to the area in which the node has been heard. Finally, the figures show that MREP and MACK packets produce more or less the same overhead. Again
this is expected since each node receiving a MREP packet must acknowledge it to the previous nodes. The small difference observed between the curves corresponds to transmitted MREP packets which are lost to channel noise, link breakages and packet collisions.

3.3 Accusation of Misbehaving Nodes

In order to enable the accusation of misbehaving nodes every node in the network is required to keep two tables in addition to the overheard nodes table introduced in Section 3.2.2: a detection table and an accusation table. The detection table contains the IDs of those nodes that have been detected as misbehaving and the number of times their misbehaviour has been reported. The accusation table keeps the IDs of those nodes that have been accused of misbehaviour. Nodes are typically accused of misbehaviour because they have reached within a predefined period of evaluation \( T_{\text{evaluation}} \) the number of misbehaviour detections \( (m_d) \) required to be accused.

We believe that nodes should not be accused based on a single detection since it can lead to mistakenly accuse well-behaved nodes of misbehaving. This idea is strongly supported by the evidence provided by the simulation results obtained in Section 3.2.3. In this Section we introduce a scheme in which multiple detections by different nodes are necessary to accuse a node. This approach is fairer to well-behaved nodes, while keeping a high probability of correctly accusing misbehaving nodes as it can be seen from our simulations results in Section 3.3.3.

3.3.1 Node Accusation

Accusing misbehaving nodes requires the detection algorithm, introduced in Section 3.2.2, to be further developed to provide it with the features needed to keep track of any misbehaving nodes and the number of times they have been detected. Additionally, nodes should be able to communicate this information to other network nodes and thus reach a consensus on whether a node should or should not be accused of misbehaviour. Figure 3-18 shows the pseudocode to accuse misbehaving nodes. Subroutines a, b, c and d are not shown in this figure as they remain exactly as they are in Figure 3-8. Subroutine e has a small modification and replaces our previous subroutine e (Figure 3-8-e). Finally the new subroutines f, g and h are to be appended at the end of the pseudocode of Figure 3-8.

As seen in Figure 3-18-e the only change made to this subroutine instructs a node performing a behaviour check to send out a detection alert packet (DAP) if it detects a misbehaving node. Detection alert packets contain similar fields to MREQ packets: the detected node ID (DN_ID), the ID of the node that realised the detection (SRC_ID), and a packet ID (DAP_ID); the IP header’s TTL field is also set to 1. As with MREQ the SRC_ID and DAP_ID fields are used to
Chapter 3. Packet Forwarding Misbehaviour Detection and Node Accusation

prevent a DAP packet from being broadcast twice by a same node; the basic idea is that a node drops a packet whose SRC_ID and DAP_ID have already been seen.

e. **REPLY HANDLING**
   if a request was sent out
   . while there are more replies to be received for node \( v_j \)
   . . receive reply
   . . acknowledge reply reception (send MACK)
   . . add received metrics to totals
   . endwhile
   
   if \( \sum_{F_{ji} \neq 0} \) \( \geq (1 - \alpha_{\text{threshold}}) \sum T_{ji} \)
   . node \( v_j \) is not misbehaving (non-detection)
   . else
   . node \( v_j \) is misbehaving (detection)
   . send a detection alert packet (DAP) with node \( v_j \)'s ID
   . endif
   endif

f. **DETECTION ALERT HANDLING**
   if node \( v_i \) receives a detection alert for node \( v_j \)
   . if node \( v_i \) has node \( v_j \) in its table of overheard nodes
   . . rebroadcast detection alert packet (DAP)
   . . if node \( v_j \) has been reached md misbehaviour detections
   . . . broadcast an accusation packet (AP) with node \( v_j \)'s ID
   . . endif
   . else
   . . ignore detection alert
   . endif
   endif

g. **ACCUSATION HANDLING**
   if node \( v_i \) receives an accusation packet for node \( v_j \)
   . if node \( v_i \) has node \( v_j \) in its accusation table
   . . ignore accusation packet
   . else
   . . add node \( v_j \) to \( v_i \)'s accusation table
   . . rebroadcast accusation packet (AP)
   . endif
   endif

h. **PUNISHING ACCUSED NODES**
   if node \( v_i \) receives a packet from node \( v_j \)
   . if node \( v_j \) is in node \( v_i \)'s accusation table
   . . ignore packet
   . else
   . . handle and process the packet
   . endif
   endif

Figure 3-18 Pseudocode of our proposed accusation algorithm
The handling of detection alerts (Figure 3-18-f) generates a limited broadcast which is conducted in the same fashion as the limited broadcast of MREQs. This means that the information to accuse a node of misbehaviour is collected locally rather than globally. When a node receives a detection alert packet (DAP) it first checks if the reported node ID contained in the received packet is present in its table of overheard nodes; if it is not the node stops broadcasting the DAP. However, if the ID appears in its table then it rebroadcasts the DAP with TTL = 1 in the IP header. Thus, nodes can control how far a DAP travels in the same manner that they control a MREQ. For instance, assuming that node $v_9$ in Figure 3-10 detects that $v_7$ is misbehaving, it sends out a DAP that follows the same path as that depicted in the Figure 3-10 for a MREQ packet, i.e. DAP are transmitted using the limited broadcast algorithm, which was introduced in Section 3.2.2.

Although using the limited broadcast method prevents nodes from generating excessive network overhead, it may also permit malicious nodes constantly changing their geographical position in a clever manner (without going back to previously visited areas within a certain period of time $T_{evaluation}$) to avoid being accused; this situation is depicted in Figure 3-19.

Figure 3-19 A clever misbehaving node that avoids accusation by constantly changing its geographical location

Figure 3-19 shows a 46 node network in which a misbehaving node $v_9$ is first at position ‘a’ ($v_{36}$ in the figure) and agrees to forward packets on behalf of some nodes of its neighbourhood but it actually launches a black-hole or grey-hole attack. Node $v_9$ knows that if it stays for too long a
time in the same neighbourhood enough number of detections will be collected to accuse it of misbehaviour. It therefore stops transmitting packets and quietly moves to position ‘b’ ($v_{9b}$ in the figure). Once there it agrees to forward packets on behalf of its new neighbourhood and launches its attack again. In Figure 3-19 node $v_9$ travels to position ‘c’ and then ‘d’ ($v_{9c}$ and $v_{9d}$ in the figure) in order to avoid accusation, and at each position it strictly follows the same process. By the time node $v_9$ returns to position ‘a’ it knows that it can be detected again without risking being accused of misbehaviour because the number of detections required for an accusation will not be reached within the predetermined time, i.e. $v_9$ will not reach $md$ misbehaviour detections within the period $T_{evaluation}$. However, the attacker has to follow a somewhat complex and cumbersome procedure to avoid being accused, and even in such a scenario it can not guarantee that it will not be accused. For instance, in Figure 3-19 node $v_9$ cannot impede other nodes from moving around the network too. Thus as it travels to different neighbourhoods it may bump into a node that happens to know of its previous detections which still are fresh enough. If this old acquaintance becomes aware of some new detections made on node $v_9$ it can potentially have enough information to accuse $v_9$ of misbehaviour. Alternatively a node may move and become a bridge between an old and a new neighbourhood. For example, let us assume that node $v_{16}$ in Figure 3-19 moves to where node $v_{22}$ is. It will form a bridge between the neighbourhood of position ‘a’ and that of position ‘b’. As a result detection alert packets (DAPs) regarding node $v_9$ can now travel between both neighbourhoods (i.e. the limited broadcast covers both neighbourhoods) since node $v_{16}$ has overhead $v_9$ and will rebroadcast the DAPs, thus allowing for the accusation of $v_9$. These examples demonstrate that a clever misbehaving node that tries to avoid accusation by constantly changing its geographical location is unlikely to be successful all the time.

In Figure 3-18-f after a node has decided whether continue broadcasting a DAP or not, it checks if an entry for the reported node ID has been already created in its detections table. If it has not, a new entry is created with its field number of detections equal to one. If the entry is already present its number of detections is increased by one and then compared against the misbehaviour detections ($md$) required to accuse a node of misbehaviour. When the number of detections reaches the $md$ value in less than a predefined period of evaluation $T_{evaluation}$ there is enough evidence to accuse the reported node of misbehaviour, and an accusation packet (AP) is broadcast in a network-wide fashion to inform all network nodes about the misbehaving node.

As stated above, whether a node is accused or not of misbehaviour depends on the number of times that it has been detected as misbehaving, the period of evaluation $T_{evaluation}$ and the number of misbehaviour detections $md$ required for an accusation. On the other hand, the number of behaviour checks ($ch$) performed on an evaluated node is not taken into account to accuse the node of misbehaviour since this could potentially lead to unfairness on some network nodes. The reason for this is that the misbehaviour detection approach of Section 3.2.2 cannot guarantee how
often or when a node’s behaviour will be checked. Instead, neighbouring nodes randomly select a period between $T_{\text{min}}$ and $T_{\text{max}}$ to check an overheard node’s behaviour, which in addition to the possible loss of synchronisation between neighbours could result in the premature accusation of a node. By contrast, the misbehaviour detection approach that we introduce in Section 4.2.3 establishes a regular period of time to check the network nodes’ behaviour, and in this case a misbehaving node’s probability of accusation depends on its exhibited number of detections, the misbehaviour detections $md$ required for an accusation and the number of misbehaviour checks $ch$ performed on the evaluated node.

Nodes that receive an accusation packet (Figure 3-18-g) examine their accusation tables to see whether the reported node has been accused previously. When an AP with a newly accused node is received a new entry is created in the accusation table and the broadcast of the AP continues. Otherwise, the packet is ignored and dropped to prevent unnecessary network traffic. Finally, all nodes in the UCE are responsible to ensure that packets coming from an accused node (a node present in their accusation table) are immediately dropped (Figure 3-18-h). This approach denies misbehaving nodes any chance to have their packets transmitted in the network as well as to participate in route discovery, thus preventing them from causing further disruption in the communication process at the network layer. Isolating a node by preventing it from using the network services is an effective incentive for nodes not to misbehave, and at the same time it avoids rewarding nodes by permitting them use the network in spite of their misbehaviour.

### 3.3.2 Authenticity of Detection Alert and Accusation Packets

A problem of our detection and accusation approach has its roots in the inability of a node to verify the authenticity of a detection alert packet (DAP) and an accusation packet (AP). This problem can be exploited by malicious nodes to accused well-behaved nodes of misbehaviour and persuade other network nodes of isolating them from the network. A workaround to this problem is to employ digital signatures and encryption (using either private or public keys) to protect our protocol packets. In this way a node receiving a packet can be sure of the packet’s authenticity and the originality of its contents, i.e. be sure that the source of the packet is who it claims to be and that the packet has not been tampered with along the way.

A possible solution using this strategy is to have nodes that receive a DAP to keep in their detection table the fields ID of the detected node and ID of the node that originated the DAP. Later when a node has to decide whether to accuse or not another node of misbehaviour not only will it check that the number of detections for the evaluated node has reached the required $md$ value, but also that the nodes generating each DAP are all different. Thus a node guarantees that there are $md$ independent detection alerts in order to accuse a node of misbehaviour. In other
words, the node guarantees that there is a consensus in the neighbourhood with respect to the misbehaviour exhibited by the evaluated node. A similar approach can be followed by nodes receiving an AP. Instead of denying access to the network to a node accused by a single AP (i.e. accused by a single source), a node can wait to receive a predefined number of accusation packets from different sources before starting to drop all packets transmitted by the accused node.

3.3.3 Evaluation Results and Analysis

In this Section, as in Section 3.2.3, we perform our simulations using the GloMoSim simulation package [48] [49]. The simulation parameters, which remain as in Section 3.2.3, are repeated here for completeness: i) nodes move according to the random waypoint mobility model with a speed randomly chosen with uniform distribution between 0m/sec and 10m/sec, this yields a mean node speed of 5m/sec and a speed standard deviation of 2.89m/sec, ii) the pause time takes a value that is exponentially distributed with mean 30 seconds, iii) the wireless transmission range of every node is 100 metres, iv) the link capacity is 2 Mbps, v) the MAC layer protocol is the IEEE 802.11 DCF, vi) the underlying routing protocol is AODV, and vii) the total simulation time for each scenario is 1800 seconds. Neighbouring nodes randomly select a period of time between 40 and 60 seconds to check an overheard node’s behaviour, and a node is accused of misbehaviour if it exhibits 3 or more misbehaviour detections (i.e. \(m_d = 3\)) in 300 seconds (i.e. \(T_{\text{evaluation}} = 300\text{sec}\)).

This Section presents first an evaluation of the average throughput gain offered by our proposed algorithm to networks affected by nodes that drop packets in a probabilistic manner. Results are shown for networks with 20, 40, 60 and 120 nodes. Then, the final set of results analyses the network overhead created by our approach and how it compares against the total traffic produced in the network. Networks simulated in this Section were set up with 20% of its total nodes misbehaving by dropping packets with 60% probability. The misbehaviour threshold was 40%.

In order to see the improvement that our approach can bring to networks affected by packet forwarding misbehaviour we consider the average network throughput in terms of the mean node speed. Our graphs present results for networks without misbehaving nodes, networks with misbehaving nodes but without mechanisms to defend them against misbehaviour, and networks that use our algorithm to deny access to misbehaving nodes. Results are displayed for networks containing 20, 40 and 60 nodes, which are distributed over areas of 40 000m\(^2\) (200m*200m), 80 000m\(^2\) (282.84m*282.84m), and 120 000m\(^2\) (346.41m*346.41m) respectively in order to maintain a constant node density.

Figures 3-20, 3-21 and 3-22 show curves for 20, 40 and 60 node networks respectively. Each graph displays the average network throughput as a function of the increasing mean node speed for i) networks without misbehaving nodes (No Misbehaviour), ii) networks making use of our
detection and accusation approach (Detection & Accusation), and iii) networks with misbehaving nodes but with no means of defending themselves from any type of attack (Misbehaviour Alone).

As can be seen from the figures our approach improves the network throughput when it is used in networks exhibiting packet forwarding misbehaviour. However, the average throughput cannot
reach that of a network where there is not any misbehaviour present. This is due to the fact that our algorithm requires a certain amount of time to collect the necessary data to detect and accuse misbehaving nodes. Thus, during this initial phase (data collection) misbehaving nodes can drop packets before their behaviour is evaluated long enough ($T_{evaluation}$) to accuse them and isolate them from the network. Then the network throughput is expected to improve gradually as the simulation time elapses and misbehaving nodes are effectively accused. Nevertheless, the average network throughput shown in the figures is unavoidably affected by the initial bad performance of the network. Moreover, the resulting network is a network with a lower node density than the original network, which can have some influence on the network connectivity and the number of packets lost to reasons other than network misbehaviour.

![Figure 3-22 Average network throughput vs. mean node speed (60 node network)](image)

With the purpose of seeing how our approach reacts to changes in a network’s node density our next set of results has been carried out in a network that preserves the same area as the previous 60 node network ($120,000 \text{ m}^2 = 346.41\text{m}*346.41\text{m}$), but has double its number of nodes (120 nodes) so as to double its density. Figure 3-23 presents results for networks without misbehaving nodes, networks with misbehaving nodes but without defence mechanisms in place, and networks that use the detection and accusation algorithms to deny access to misbehaving nodes.

From Figure 3-23 it can be seen that our approach still works in networks with relative high node density. The network throughput of networks using the detection and accusation approach improves when compared against networks containing misbehaving nodes that are neither avoided
nor penalised. However, networks that do not present node misbehaviour at all still exhibit the best performance in terms of network throughput.

![Graph showing average network throughput vs. mean node speed](image)

**Figure 3-23** Average network throughput vs. mean node speed (120 node network)

A network that implements our detection and accusation algorithm looks, when it first starts operating, like a network without a protection scheme. However, as time elapses the algorithm starts detecting and accusing those nodes that drop a fraction of packets above a preset misbehaviour threshold $\alpha_{\text{threshold}}$. Consequently, in such a network most misbehaving nodes will eventually be detected, accused and denied network access, allowing the network to obtain an overall performance close to a network where nodes do not misbehave. This point is studied in detail in the evaluation results of Section 4.3.2, where it is possible to observe how the throughput evolves through the simulation time.

The final set of results assesses the network overhead generated by our misbehaviour detection and accusation algorithm as a function of the increasing mean node speed, and compares it against the network traffic produced by four constant bit rate (CBR) connections present in the network. Although CBR traffic is generated at the application layer, it is accounted for at the TCP/IP layer in the form of UDP packets. In this set of simulations, misbehaving nodes drop packets with 60% probability, the misbehaviour threshold $\alpha_{\text{threshold}}$ is 40%, the node speed varies between 0m/sec (a static network) and 20m/sec, and the speed standard deviation is set at 0.58m/sec. The CBR traffic and the overhead of the proposed approach are evaluated in terms of their contribution to the total network traffic. Results are displayed for a network containing 40 nodes distributed over an area of 80 000m$^2$ (282.84m*282.84m).
Chapter 3. Packet Forwarding Misbehaviour Detection and Node Accusation

3.4 Summary

In this Chapter we have presented an approach that uses the principle of flow conservation to effectively detect and accuse nodes that exhibit packet forwarding misbehaviour. It has been shown that by using data gathered by nodes actively intervening in the communication process, as opposed to data gathered by nodes overhearing a communication, our approach is immune to problems such as ambiguous collisions, receiver collisions and nodes capable of controlling their transmission power.
Although the principle of flow conservation was originally envisaged to detect misbehaviour in fixed networks, we have demonstrated that by making some reasonable assumptions this principle can be adapted to detect misbehaviour in mobile ad hoc networks. We have also shown that by pairing our algorithm with a MAC layer protocol such as the IEEE 802.11 standard, our approach can effectively detect malicious nodes launching black-hole and grey-hole attacks regardless of the routing protocol employed by the network.

Finally, through simulations we have established that our limited broadcast solution effectively tracks nodes that have behaviour metrics about an evaluated node. Additionally, our results verify that our approach efficiently isolates nodes that persistently exhibit packet forwarded misbehaviour. As a result our scheme significantly improves the average network throughput of multihop wireless networks affected by nodes that maliciously fail to forward packets on behalf of other network nodes.
Chapter 4

4 Policy-Based Adaptable Misbehaviour Detection and Node Accusation

In this Chapter we extend the work of Chapter 3. First of all, we introduce a new approach to collect the metrics for behaviour evaluation and node accusation, which creates some new parameters in addition to those used in the algorithm of Chapter 3. Secondly, we aim at making automatic adjustments – by using polices – to the various parameters that control the proposed protection scheme in an attempt to optimise the network performance.

The protection scheme that we propose in this Chapter consists of two modules in each node: one that works in the security plane and another that works in the management plane. The security plane module consists of an algorithm that performs three tasks: i) collects and aggregates metrics for behaviour evaluation and node accusation, ii) detects misbehaving nodes that maliciously drop packets above a predefined limit, and iii) accuses and punishes nodes that are persistently detected as misbehaving. The management plane module uses high-level policies to control and adjust the security plane module by varying the algorithm’s parameters. The interworking of both modules drives network adaptation based on current conditions and the objectives of high-level entities, such as a network administrator.

The rest of this Chapter is organised as follows: Section 4.1 offers an introduction to polices for network management. This introduction provides an overview of the policy core information model (PCIM), it then examines the types of policies that can be used within the network management context, and it ends with a brief literature review of policies in the context of mobile ad hoc networks (MANETs). Section 4.2 presents our adaptable misbehaviour detection and accusation scheme. This has three parts: the first is a selected clustering approach which is the driving mechanism for assigning roles to network nodes based on their connectivity and their computational capabilities. The second part is an improved version of the detection and accusation scheme presented in Chapter 3. The final part of Section 4.2 demonstrates how adaptability can be achieved with the use of network management policies. Next, Section 4.3 presents a case study scenario to evaluate our proposed protection scheme followed by an extensive set of simulation results and their corresponding analysis. Finally, Section 4.4
summarises this Chapter by highlighting its main achievements. The work presented in this Chapter was originally published in [110].

4.1 Introduction to Policies for Network Management

As stated in Chapter 2 ubiquitous computing environments (UCEs) consist of a myriad of heterogeneous devices that communicate with each other through a hybrid set of network protocols and technologies. For example, a UCE can be composed of areas where the network is wired, or wireless with infrastructure support, or pure mobile ad hoc network (MANET) based. Chapter 2 also discussed how the complexity associated with the management of these systems can outweigh their benefits due to the fact that their administration can be extremely costly and sometimes technically close to impossible. Additionally, we presented autonomic computing systems and showed how this plausible solution to the complexity of system management places the burden of administration on the system itself and allows network managers to concentrate on the design of high level goals or objectives. In this Section we now introduce policies for network management, which are considered a key tool to enable the deployment of truly autonomous systems, and we discuss how they can help network administrators to automate a system’s response to a series of network or external events.

As is common in the computing world the term policy has received different definitions according to the field and context where it is used and sometimes depending on how experts interpret the word in order to make it fit their goals. For example, the IETF gives two perspectives for the definition of policy within the network management context; both these perspectives are provided in complementary Request for Comments (RFC) documents produced by the Policy Framework Working Group. RFC 3060 [111] defines policies as a set of rules to administer, manage, and control access to network resources; alternatively RFC 3198 [112] defines them as a definite goal, course or method of action to guide and determine present and future decisions. Also, many members of the scientific community have proposed their own definitions, for instance the authors of [113] define a policy as a mechanism to configure the behaviour of a system in ways that are not predictable at system initialisation time because the configuration depends on the dynamic state of the system and its surroundings. Fortunately, in spite of the variety of policy definitions in the area of network management, the policy information model presented next has been generally accepted as a core template and most new proposals are based on it or extend it.

4.1.1 Policy Core Information Model (PCIM)

Policy-based management (PBM) has been studied for over two decades but it gained relevance thanks to the joint efforts of the Distributed Management Task Force (DMTF) and the Internet
Engineering Task Force (IETF) on producing a standard architecture and object-oriented information model for policy-based management in the late 1990s [113]. The results of this joint effort are available in the RFC 3060 (Policy Core Information Model (PCIM) – Version 1 Specification) [111], update RFC 3460 [114], and some complementary RFCs that propose concrete implementations for specific parts of this policy information model. The core PBM architecture described in [111] and [114] is shown in Figure 4-1.

The components depicted in Figure 4-1 and its corresponding functions are [113]:

- **Policy Management Tool (PTM):** The policy management tool is a user interface that allows for the reading, creation, deletion and modification of policies. It can be as simple as a graphical user interface (GUI) that facilitates access to policies stored in a policy repository, or it can be a complex application that takes a high-level input by a network administrator or operator, translates it into polices of the adequate format, checks for possible policy conflicts, and stores the output in a policy repository.

- **Policy Repository (PR):** A policy repository can be a physical data store (such as a database or directory) or a logical container (such as a “Quality of Service” domain) that holds policy rules, their conditions and actions, and any related policy data [112]. RFC 3703 [115] defines a mapping for the PCIM shown in Figure 4-1 that can be implemented in a service repository that uses Lightweight Directory Access Protocol (LDAP) [116] as its access protocol.
However, the IETF standard does not make compulsory the use of an LDAP directory, and other repository solutions are also accepted, e.g. a relational database.

- **Policy Decision Point (PDP):** A policy decision point is a logical entity where the intelligence of a policy system is contained. It is in charge of evaluating a policy rule’s conditions and if they are true it triggers the necessary actions to ensure the appropriate policy enforcement.

- **Policy Enforcement Point (PEP):** A policy enforcement point is a logical entity – often contained in the physical element where the policy is enforced – that executes the actions indicated by policy decisions made by a PDP.

The above components communicate by means of the interfaces shown in Figure 4-1 as follows [113]:

- **PTM-PR Interface:** The policy management tool interacts with the policy repository using an adequate access protocol to retrieve and store policies. The access protocol used depends directly on the implementation of the policy repository. For instance, if the repository is an LDAP directory then the PMT uses LDAP to access the PR, but if the repository is a relational database then the PMT uses SQL (Structured Query Language) queries.

- **PDP-PR Interface:** The policy decision point accesses the policy repository in the same way that a PMT does, i.e. it depends on the specific implementation of the PR.

- **PDP-PEP Interface:** This interface allows a PDP to control the behaviour of one or more PEPs based on the policies specified by a PMT that have been stored on a PR. The IETF standard does not command the use of any particular protocol over this interface. COPS (Common Open Policy Service) [117], SNMP (Simple Network Management Protocol) [118], and even proprietary solutions are suitable to be used.

- **PEP-PR Interface:** It is not common for PEPs to access directly a PR since they receive their instructions directly from a PDP. However, there are circumstances in which it may be appropriate for a PEP to retrieve directly its instructions from a PR, e.g. when a network administrator in order to bypass a PDP defines some policies in a low enough level for a PEP to be able to interpret them directly. In this case the PEP accesses directly the PR using an adequate protocol so as to retrieve the necessary policies.

### 4.1.2 Policy Types

In this Section we review the types of policies that can be used within the network management context, which are three: Event-Condition-Action policies, Access Control Policies and Configuration Policies.
4.1.2.1 Obligation or Event-Condition-Action Policies

Event-Condition-Action (ECA) policies describe network management actions that must be taken when certain network or external event occurs, provided that certain given conditions are satisfied at the moment the event is triggered. Typically these policies have the following format: *on* <Event(s)> *if* <Condition(s)> *then* <Action(s)>. A brief description of the elements of an ECA policy is provided below [113]:

- **Events**: Events represent network activities of special interest in a relevant context. These activities are used to trigger policies, such that the policies react to the activities happening. For example, an activity of special interest in the context of security is an intrusion or misbehaviour detection. Events are an optional component of an obligation policy.

- **Conditions**: Conditions check that certain network states are satisfied, i.e. evaluate to true, when an event occurs and before executing an action. They are very useful preventing harmful, unfair, unnecessary or insensible actions when a service or network element is in inadequate states. For example, when a misbehaviour detection event happens a condition may check that it has been detected at least three times before executing an action on the offender.

- **Actions**: Actions are a type of management network operations that are useful to monitor and configure a service or network element. Actions can also act as triggers to other network components or services. For example, if a node reaches three or more misbehaviour detections an action can bring up an accusation module which has embedded the intelligence necessary to deal best with the offender.

4.1.2.2 Access Control Policies

Access control policies are rules that determine whether grant or not permission to an entity to perform certain actions provided that certain conditions are satisfied. They specify what actions are permitted by which users on which network resources. Typically the general form of an access control policy has the following components [113]:

- **Subject**: The entity on which a decision is to be made (e.g. a user with username “Oscar-Gonzalez”, or a computer called Poseidon).

- **Request**: Parameters that define the activity to be performed by the subject (e.g. authorisation to configure a router called Zeus).

- **Target**: Network resource that the subject is to access (e.g. router Zeus).

- **Conditions (optional)**: Conditions that restrict the circumstances under which access is granted or denied (e.g. Zeus CPU load is lower than 75% and Zeus is not in standby mode).

- **Permit/Deny**: Indicates whether a request is allowed or rejected (e.g. accept).
4.1.2.3 Configuration Policies

Configuration policies specify the parameters for network elements, services, protocols and software components. They can be categorised in two groups [113]:

- **Administration Policy**: Administrative policies represent high level goals or objectives to be achieved. These policies can express optimisation rules for the overall system. As these policies are refined to low level configuration they impose constraints to the components enforcing them. An example of such a policy is: keep as many web servers in standby mode as possible such that the users' queue mean time is never more than 2 seconds.

- **Configuration Policies for network elements, services and protocols**: These policies contain configuration information, typically low level instructions, for network hardware or software. For example, a policy indicating a firewall the types of network traffic that it must or must not let traverse it.

4.1.3 Mobile Ad Hoc Network Management – Related Work

Related literature in mobile ad hoc network (MANET) management is relatively limited and existing approaches do not address all available problems. A first effort to address this field of research is presented in [119]. The proposed Ad Hoc Network Management Protocol (ANMP) is based on a hierarchical clustering approach with three tiers and a unique manager. ANMP uses two methods to cluster a network: graph-based clustering, which is performed by the network nodes, and geographical clustering, which is performed by the network manager. Additionally, the authors claim that ANMP is designed to be SNMPv3 [118] compatible because SNMP is already widely used for network management in today's networks and MANETs are just extensions to cover areas lacking network infrastructure, thus there is no reason for MANETs to use a different management protocol. A different approach is adopted by authors of [120] who propose an architecture called Guerrilla which makes use of mobile code techniques to provide nodes with capabilities to manage or be managed. In this scheme the network is clustered in two tiers according to what code is deployed on what node: nomadic managers, which are deployed in managing nodes (i.e. cluster heads), correspond to the higher tier, while active probes, which are installed in managed nodes, belong to the lower tier.

The authors of [121] proposed a policy-based network management (PBNM) system that uses intelligent agents responsible for managing and enforcing the policies of a policy domain. This approach organises the network in three tiers, each with different type of agents: global policy agents (GPA), domain policy agents (DPA), and local policy agents (LPA). Although the agents have the same management capabilities their scope is not the same, as suggested by their names. Also, the policy definitions of this work follow the principles of the IETF, but the implementation
uses several proprietary protocols, which restricts its wider adoption. Another PBNM approach was proposed in [122]. The authors propose a solution suite composed of four schemes: a k-hop clustering algorithm, dynamic service redundancy, policy negotiation, and automated service discovery. This approach also makes some extensions to the standard Common Open Policy Service (COPS) [117] to add policy server delegation and redirection capabilities. The main drawback of this solution is that it is relatively heavyweight, which may limit its applicability to MANETs. Finally, the policy based paradigm introduced in [123] and [124] offers a promising solution the management of MANETs since it allows dynamic alteration and controlled programmability of the behaviour of automated managers without having to re-implement them, thus allowing for the reuse of managers in different environments.

4.2 Adaptable Misbehaviour Detection and Node Accusation

Our proposed mechanism introduces a novel adaptable method, which builds on the approach and techniques introduced in Chapter 3, to detect packet forwarding misbehaviour based on the use of policy-based management (PBM) [125] in addition to the principle of flow conservation [94]. Such adaptability allows the system to better judge the behaviour of nodes and decide whether they should, or not, be accused of misbehaviour and penalised according to current network management policies. The approach is deployed over a role-based wireless network, organised in a tiered manner [126] [128]. Nodes are assigned a role that defines the tasks they are responsible for as well as the policies that apply to them. For example, depending on their role, nodes may hold behaviour information only about their neighbours, or a localised network area or the entire network.

As in Chapter 3, the approach presented in this Chapter is distinguished from existing schemes because it does not use promiscuous listening to detect nodes that fail to forward packets in the network. However, whereas the work presented in Chapter 3 uses promiscuous listening to determine the active neighbourhood of a node (i.e. neighbours that are actively transmitting and receiving packets), the work proposed here does not use promiscuous listening at any stage of the packet forwarding misbehaviour detection and node accusation scheme. All relevant calculations and subsequent decisions are based on metrics directly acquired by nodes actively sending and receiving packets. Likewise, the problem posed by nodes that constantly change their geographical position in a clever manner to avoid the security measures in place, as explained in Section 3.3.1 and depicted in Figure 3-19, is solved in this new scheme due to the fact that nodes making the accusation decision have a holistic instant network misbehaviour view. Also, to the best of our knowledge, our proposal is the first attempting to connect misbehaviour detection and accusation with the use of policies at the management plane. By allowing policies to manipulate
key features of our algorithm, such as the misbehaviour detection threshold and the maximum expected percentage of wrong accusations, the network manager or administrator can fulfil the requirements stipulated by high level management goals, e.g. the desired security level.

This Section is divided in 4 subsections: Section 4.2.1 presents a clustering algorithm and demonstrates how by making use of a capability function it selects a suitable connected dominating set (CDS) of nodes to carry out essential management tasks and metrics collection for the behaviour evaluation of nodes in the network. Section 4.2.2 introduces an improved and more efficient version of the detection algorithm of Section 3.2.2 and investigates in detail how the selection of the misbehaviour threshold $\alpha_{\text{threshold}}$ affects the probability of wrongly detecting a well-behaved node as misbehaving. Section 4.2.3 describes an also improved version of the accusation strategy of Section 3.3.1. It also provides an important analysis on how the desired probability of accusation for misbehaving and well-behaved nodes affects the selection of an adequate misbehaviour threshold $\alpha_{\text{threshold}}$. Finally, Section 4.2.4 explains how adaptability can be achieved by using management policies and proposes a method so that the management plane can gain control over our protection scheme and adjust it as required to accomplish the overall system objectives.

4.2.1 Clustering Phase

Later in this Chapter we propose a metrics collection procedure (Section 4.2.2) and a management policies distribution process (Section 4.2.3 and Section 4.2.4) which are based on the clustered organisation of the underlying multihop wireless UCE. We assume that in our network model nodes are in one of three possible roles: Manager Node (MN), Cluster Head (CH) or Cluster Node (CN). We also assume, as explained in Section 4.2.2, that MNs are statically pre-assigned while CHs are selected through a dynamic algorithmic selection driven by a clustering strategy. Clustering is an effective method to reduce traffic overhead in wireless networks, both for routing (e.g. the selection of multipoint relay nodes in OLSR [57] [58]) and management purposes [126]. By selecting cluster heads and forming clusters, scalability can be increased and locality can be preserved.

4.2.1.1 Cluster Heads Selection Algorithm

In this Chapter, we employ a clustering algorithm originally developed by Wu [127] and modified by the authors of [126] and [128] to make it suitable for network management. In this context, they propose the use of a capability function to select the most suitable nodes as CHs, as explained later in this Section. By exploiting the fully distributed algorithm proposed in [127] to achieve cluster creation and maintenance, the framework dynamically decides what nodes become cluster heads (CHs), taking into consideration their capabilities and mobility attributes, as
Policy-Based Adaptable Misbehaviour Detection and Node Accusation

proposed in [126] and [128]. To aid the reader, below we provide a brief description of the clustering algorithm and its cluster head selection process. For a detailed description of the algorithm we refer the reader to [127]. The main reason behind the selection of this particular clustering approach is that, in spite of its inefficiency, it is a simple and effective method to obtain a Connected Dominating Set (CDS) of nodes in a network in a fully distributed and decentralised manner. A CDS is a subset of network nodes such that they fulfill two conditions: i) a network node that does not belong to the subset has a direct link with at least one member of the subset, and ii) a node that is part of the CDS also has a direct link with a least another member of the subset (i.e. nodes in the subset are connected). Wu’s algorithm generates a CDS that is scattered throughout the network, which facilitates the collection of metrics for behaviour evaluation and the distribution of network management policies to the network nodes. In Chapter 5 we propose the use of a more efficient clustering approach designed specifically to deal with large-scale networked environments.

The algorithm is executed in two stages: the marking round and the optimisation round. The marking round generates a redundant CDS and the optimisation round reduces the set to make it closer to a Minimum Connected Dominating Set (MCDS). Since the problem of calculating a MCDS is known to be NP-complete [127], heuristics are used to achieve a suboptimal solution. We now formally define the graph model [127]:

- Let $G = (V, E)$ be a connected, undirected and non-weighted MANET (or MANET-like UCE).
- Let $V = \{v_1, v_2, v_3 ... v_N\}$ be the set of all MANET nodes. $N$ is the total number of nodes in the network.
- Let $E$ be the set of links between the MANET nodes. Thus if $v_i \in V, v_j \in V, i = 1, 2, 3 ... N, j = 1, 2, 3 ... N, i \neq j$, and there is a direct link between nodes $v_i$ and $v_j$, then the link $\{v_i, v_j\} \in E$.
- A node $v_i$ is part of the open neighbour set $N(v_i)$ of a node $v_i$ if and only if there is a direct link between nodes $v_i$ and $v_j$, i.e. $v_j \in N(v_i)$ if and only if $\{v_i, v_j\} \in E$ and $i \neq j$.
- Let the closed neighbour set $N[v_i]$ of node $v_i$ be the set $N[v_i] = \{N(v_i), v_i\}$, i.e. the set formed by the open neighbour set of $v_i$ and node $v_i$ itself.
- Each node $v_i$ has a marker $m(v_i)$ to indicate whether it belongs to the CDS, $m(v_i) = T$ (true) if $v_i \in CDS$, and $m(v_i) = F$ (false) if $v_i \notin CDS$.
- Each node has an arithmetic identifier $id(v_i)$.

The marking process is as follows [127]:

1. Initially all nodes $v_i \in V$ assign their marker $m(v_i) = F$. 78
2. Every node $v_i$ exchanges its open neighbour set $N(v_i)$ with all its neighbours.

3. Every node $v_i$ assigns its marker $m(v_i) = T$ if there exists two unconnected neighbours, i.e. $m(v_i) = T$ if node $v_i$ can directly reach a neighbour that is not directly reachable by at least one of its neighbours.

Figure 4-2 shows an example of a MANET and Figure 4-3 shows the same network after the marking process. In Figure 4-3 node $v_2$ is marked ($m(v_2) = T$) because $v_1$ and $v_6$ can not reach directly node $v_3$ and vice versa, but $v_2$ can. Node $v_3$ is marked because $v_2$ can not reach directly $v_8$ and vice versa, but $v_3$ can. Similar reasoning can be applied to explain the marking round for the rest of the marked nodes $v_4, v_7, v_8$ and $v_9$.

The resulting reduced set composed of the nodes $v_i$ with marker $m(v_i) = T$ is denoted $G'$. In Figure 4-3 $G' = \{v_2, v_3, v_4, v_5, v_6, v_9\}$. Optimisation rules (heuristics) are applied to $G'$ in order to eliminate unnecessary redundant vertices in the connected set. The optimisation rules are [127]:

1. Consider two nodes $v_i$ and $v_j$ in $G'$:
   
   $N[v_i] \subseteq N[v_j]$ in $G$ and $id(v_i) < id(v_j)$, then $m(v_i) = F$

2. Assume $v_i$ and $v_k$ are two marked neighbours of marked node $v_l$ in $G'$:

   $N[v_i] \subseteq (N[v_l] \cup N[v_k])$ in $G$ and $id(v_i) = \min\{id(v_l), id(v_k), id(v_j)\}$, then $m(v_i) = F$.

Figure 4-4 shows the result of applying optimisation rule 1 to the set $G'$ in Figure 4-3. From the figure $N[v_4] = \{v_4, v_5, v_6, v_9, v_{10}\}$ and $N[v_9] = \{v_4, v_5, v_6, v_9, v_{10}\}$, that is $N[v_4] \subseteq N[v_9]$. If we assume that $id(v_4) = i$, then $id(v_5) = 4$ and $id(v_9) = 9$, which means that $id(v_5) < id(v_9)$ and therefore $m(v_5) = F$, i.e. node $v_5$ is unmarked. Node $v_9$ can not be unmarked in spite of the fact that $N[v_9] \subseteq N[v_4]$ because $id(v_9) > id(v_4)$. 
The result of applying optimisation rule 2 is depicted in Figure 4-5. From the figure it can be seen that \( N[v_2] \subseteq (N[v_3] \cup N[v_7]) \) because \( N[v_2] = \{v_1, v_2, v_3, v_6, v_7\} \) and \( N[v_3] \cup N[v_7] = \{v_2, v_3, v_6\} \cup \{v_1, v_2, v_3, v_6, v_7\} \), and \( \min\{id(v_2), id(v_3), id(v_7)\} = id(v_2) = 2 \) therefore node \( v_2 \) is unmarked \( (m(v_2) = F) \). The figure also shows that \( N[v_3] \not\subseteq (N[v_2] \cup N[v_8]) \) but \( \min\{id(v_2), id(v_3), id(v_8)\} = id(v_2) \neq id(v_3) \), therefore node \( v_3 \) remains marked \( (m(v_3) = T) \). A similar situation arises when node \( v_7 \) is analysed. Thus, Figure 4-5 holds our solution CDS.

Another attractive feature of this clustering algorithm is its ability to adapt the CDS to topological changes in the network. The authors of [127] identify and provide solutions for three types of dynamic changes: a node is switched on, a node is switched off, and a node is moving. In order to address mobility the authors propose two ideas: i) each individual node updates its marked status, and ii) the whole network recalculates the CDS. The former option is efficient when few nodes are moving about; the latter is a better approach when many nodes are in movement. Since cluster formation in our approach is very important in those instants when the network behaviour is to be evaluated, we periodically recalculate the CDS of the whole network just before behaviour evaluation is about to take place.

Having adopted the described approach for cluster formation it was necessary to modify it to allow its integration with the proposed policy-based and role-based misbehaviour detection and node accusation approach. The first modification has already been described above and concerns the periodic CDS recalculation of the whole network. The second modification is regarding its arbitrary arithmetic function \( id(v) \). Instead of using such a function we have substituted it with a scalar Capability Function \( CF(v) \) as originally proposed by authors of [126] and [128]. The higher the capability function of a node is the higher the probability that it stays marked (i.e. it stays part of the CDS) after the optimisation rounds. \( CF(v) \) in our approach expresses three different facets.
of a node's capabilities: its processing power \(PP(v_j)\), its memory \(MEM(v_j)\) and its mobility ratio \(MR(v_j)\) as seen in Equation 4-1.

\[
CF(v_j) = \frac{w_1 PP(v_j) + w_2 MEM(v_j)}{MR(v_j)} \quad (4-1)
\]

With the intention of making them comparable, each variable in Equation 4-1 is normalised to a value range of \((0, 1]\) by dividing each one with its maximum value, e.g. we consider the maximum processing power to be \(PP_{\text{max}}(v_j) = 3.2 \text{ GHz}\). Weights \(w_1\) and \(w_2\) are assigned according to the significance of their respective variables, the only restricting condition being that \(w_1 + w_2 = 1\). Finally, it is important that the value of the mobility ratio \(MR(v_j)\) effectively reflects the topology changes occurring around node \(v_j\). In this respect, a node's speed and movement frequency do not necessarily mean that a node is unfit to be part of the final CDS. For example, a group of nodes may be moving in the same direction at an equal speed while the network topology remains almost static. Conversely, a static node may not be suitable to be part of the CDS if its neighbours are all moving away from it. For this reason the mobility ratio \(MR(v_j)\) of node \(v_j\) in our scheme is associated with its average frequency of link breaks with its neighbours.

In the proposed protection scheme, the connected dominating set obtained after applying Wu's clustering algorithm and its respective optimisation rules [127], and its modified capability function [126] [128], represents the set of network nodes that are assigned the cluster head (CH) role in the network. Nodes that do not belong to the CDS are assigned the cluster node (CN) role.

### 4.2.1.2 CDS Population Size Evaluation and Analysis

As usual, we perform the simulations in the Glomosim network simulation package [51] [52]. The objective of this Section is to evaluate the efficiency of Wu’s clustering algorithm [127]. To this end, we analyse how the CDS population size is affected by the number of nodes in the network and the nodes’ transmission range. The results presented for each value are the average of 20 simulation runs. No packets are transmitted in the network except for the essential packets to create the CDS; as a result no routing protocols are employed. Nodes move according to the random waypoint mobility model with a constant speed chosen uniformly between 0 and 10m/sec, this yields an average node speed of 5m/sec and a standard deviation of 2.89m/sec. The link capacity is 2 Mbps.

For the first set of results, we simulated networks of 25, 50, 75, 100, 225 and 400 nodes with areas of 40 000m\(^2\), 80 000m\(^2\), 120 000m\(^2\), 160 000m\(^2\), 360 000m\(^2\) and 400 000m\(^2\) respectively. These values yield a constant node density of 6.25x10\(^{-4}\) nodes/m\(^2\). Figure 4-6 shows the results for the CDS population size (i.e. the number of nodes that are part of the CDS).
Figure 4-6 illustrates how the CDS population size is affected by the total number of nodes in networks with node transmission ranges of 100m, 150m, 200m and 250m. It can be appreciated from the figure that as the transmission range increases the CDS population decreases. This is due to the fact that nodes have more overlapping neighbours within each other’s transmission range. Therefore fewer nodes become part of the CDS as their neighbours are already covered by other network nodes. Additionally, Figure 4-6 also shows that the main factor influencing the number of nodes in the CDS is the number of nodes in the network. In fact, the CDS population grows sharply as the number of nodes in the network rises. These results demonstrate the inefficiency of employing Wu’s clustering algorithm in large networks, where the CDS population is typically more than half the total number of nodes in the network. For example, for a 225 node network with a node transmission range of 100m the CDS population size is about 155 nodes, and for a 400 node network with the same transmission range the CDS population size is about 300 nodes.

These evaluation results are not in agreement with the results obtained by the authors of [126]. Generally speaking, their simulated networks generate a CDS whose population is about a third of the population shown in Figure 4-6 for the curve “Tx Range = 100m”. Regarding this issue, in discussions with Dr. Hadjiantonis, who is one of the authors of [126], we have found a possible inadvertent error in their interpretation of Wu’s clustering algorithm [127], which could explain the difference in the obtained results. In our opinion, Hadjiantonis et al have calculated a minimum connected dominating set (MCDS) in their simulated networks instead of a CDS corresponding to Wu’s optimisation rules. Unfortunately, this Section results and those obtained
Policy-Based Adaptable Misbehaviour Detection and Node Accusation

in [126] are not comparable to those presented in [127] since Wu et al present evaluated networks consisting of fewer than 100 nodes with node transmission ranges of less than 100m. However, in Section 5.3.1 we demonstrate through an example that it is possible for this clustering algorithm to select more than 50% of the network nodes to be part of the CDS.

For the second set of results, we simulated a set of networks deployed over an area of 360 000m² (600mx600m) with different node densities and node transmission range of 100m. The total number of network nodes was gradually increased from 40 nodes to 800 nodes. Consequently, the node density increased from 1.11x10⁻² nodes/m² to 2.22x10⁻¹ nodes/m². Figure 4-7 shows the results obtained for the CDS population size.

![Figure 4-7](image)

Figure 4-7 corroborates that the CDS population size increases linearly but steeply as the total number of nodes in the network grows. It can also be seen from the figure that the CDS population size is more than half the total number of nodes for large networks. For example, for the 800 node network the CDS population size is about 640 nodes, i.e. about 80% of the nodes are part of the connected dominating set. Such a large CDS population in a network as dense as the 800 node network (nodes have about 60 neighbours in average) means that many of the CDS participants are redundant nodes.

The analysis of the evaluation results has revealed the inefficiency of Wu’s clustering algorithm [127] to select a non-redundant connected dominating set of nodes in a network. This means that it is possible that for CHs in the proposed protection scheme not to have any subordinated nodes.
Policy-Based Adaptable Misbehaviour Detection and Node Accusation

(i.e. CNs) in their managed cluster – this point is further investigated in Section 5.3.1, where various issues are pointed out and a more efficient clustering approach is presented. However, as stated at the beginning of Section 4.2.1.1, Wu’s clustering algorithm is a simple and effective method to obtain a CDS in a fully distributed and decentralised manner, which makes it appropriate for the distribution of policy management policies and the collection of behaviour-related metrics, as discussed in the next Section.

4.2.2 Detection Phase

We now describe how the algorithm of Chapter 3 is modified to take advantage of a clustered network (i.e. where the CHs have been selected) during the metrics collection process. Our approach can be divided into two main phases: gathering of behaviour information for misbehaviour detection, and node misbehaviour accusation with penalty enforcement. The former involves collecting and aggregating behaviour information in the low levels of the tiered network and presenting it to the top level for analysis in order to detect misbehaving nodes. The latter phase on the other hand starts at the top level with the decision making of which nodes to accuse of misbehaviour and how to penalise them, and continues with the enforcement of the respective penalties from the top to the bottom tiers.

Three roles are defined, namely manager node, cluster head and cluster node. For modularity, a MN encapsulates the functionality of a CH and in turn a CH encapsulates that of a CN. In brief, by using the clustering approach described in Section 4.2.1 the most suitable nodes are selected as cluster heads, while remaining nodes become CNs and register with their nearest CH. A CH uses a policy decision point (PDP) to locally manage the policy enforcement point (PEP) of CNs that belong to its cluster and communicates with other CHs to exchange management information, including policies. Network deployment issues have an impact in the selection of MNs and CHs, e.g. in a ubiquitous computing environment (UCE) with infrastructure support they can be statically assigned and controlled by an administrator. In MANET-like scenarios, the assignment of roles can be fully dynamic. Management policies are defined at MNs and are distributed to CHs for enforcement on CNs. Management policies are enforced on MNs and CHs as well. These policies can be specific for each role, as described later, and transparently prescribe the participation of the network nodes during the metrics collection, misbehaviour detection and node accusation phases of the proposed protection scheme. For the rest of this Section, we will assume a clustered multihop wireless UCE with a pre-assigned number of MNs and CHs selected as described in Section 4.2.1, i.e. the selection of the connected dominating set (CDS) corresponds to the selection of the CHs in the network. Nodes that are not selected as part of the CDS are assigned the role of CNs.
Misbehaviour detection continues to be based on the principle of flow conservation in the same way it was in Chapter 3. For convenience, Equation 3-2 is repeated below as Equation 4-2 since it can help the reader understanding the concepts described in this Chapter. However, the metrics collection process is not based on the limited broadcast strategy introduced in Section 3.2.2, instead a more efficient and effective process reliant on the role-based clustered UCE is employed.

\[
\sum_{\forall i \in U_j} F'_{yi} (\Delta t) \geq (1 - \alpha_{\text{threshold}}) \sum_{\forall i \in U_j} T_y (\Delta t)
\]  

Equation 4-2 needs to be applied to all nodes in the network, which requires the gathering and aggregation of the \(T_y\) and \(F'_{yi}\) metrics for each active node in the network. Our scheme achieves this by requiring all CNs to report their collected \(T_y\) and \(F'_{yi}\) metrics to their respective CHs in a periodic fashion. The period that elapses between two consecutive reports (\(T_{\text{report}}\)) depends on the current network management policies. CHs aggregate the reported metrics, and any metrics proactively collected. This aggregation at CHs provides them, to a certain extent, with a local view of the behaviour of nodes in their cluster. This view is more accurate for clusters with relatively infrequent changes in nodes participation. To anticipate node movements, aggregated cluster information is passed on from CHs to MNs.

Finally, MNs exchange behaviour information between them and perform a new metrics aggregation. At this point, MNs have managed to acquire information on the overall network behaviour as well as on the individual behaviour of each active node in the network. This allows for the detection of misbehaving nodes by applying Equation 4-2 on the collected metrics, as described later. Since MNs obtain a holistic view of all active nodes in the network, in this new approach the problem posed by nodes that cleverly change their geographical location, as explained in Section 3.3.1, does not exist. Regardless of where on the network nodes are, their metrics are reported up the role-based hierarchy to MNs.

As we stated in Section 3.2.3 and showed in Figures 3-11 and 3-12, selecting an appropriate misbehaviour threshold \(\alpha_{\text{threshold}}\) is very important to avoid wrong detections, i.e. a state where a well-behaved node is mistaken for a misbehaving one. We now develop this point in detail, illustrating it with Figures 4-8 and 4-9. All values shown on our graphs are the result of averaging 20 simulation runs. More details on the value of the parameters and the tools used for our simulations can be found in Section 4.3.2. Figure 4-8 shows three curves depicting the percentage of detections as a function of the increasing misbehaviour threshold for three different 60 node networks. By comparing this Figure to Figure 3-12, it is possible to see the improvement offered by the new strategy to collect behaviour-related metrics. The network can now discriminate better between misbehaving and well-behaved nodes. This is represented by curves with greater
detection probabilities for misbehaving nodes (i.e. nodes that drop more packets than those allowed by $a_{\text{threshold}}$), and lower detection probabilities for well-behaved nodes (i.e. nodes that drop packets below the amount permitted by $a_{\text{threshold}}$).

In Figure 4-8 all nodes drop packets with the same probability. This means that in the figure the curves correspond to networks with nodes that drop packets with probabilities of 70%, 30% and 0% respectively. For these three particular cases, it is assumed that this average behaviour corresponds to the normal behaviour of a well-behaved node. For the networks dropping 70% and 30% of the packets it is presumed that such node behaviour is perhaps due to increased noise or mobility issues and therefore nodes are not expected to be detected as misbehaving. Consequently the misbehaviour threshold $a_{\text{threshold}}$ should at least be set to a value equal to the packet dropping average plus an offset which helps preventing the algorithm from wrongly detecting well-behaved nodes. For example, it can be observed from the figure that for the network where nodes drop about 70% of their packets it is required that $a_{\text{threshold}} \geq 70\% + 10\%$ (i.e. offset = 10%) in order to have less than 20% chance of detecting a node as misbehaving.

Simulations have also been carried out for 20, 40 and 120 node networks. The curves obtained exhibit similar behaviour to the 60 node network of Figure 4-8. However, our results show that the offset to keep a desired detection probability suffers small changes between different networks. Figure 4-9 shows a subset of curves corresponding to networks of 20, 40, 60 and 120 nodes that drop packets with 30% probability. The range of the misbehaviour threshold axis has
been modified so as to offer a better view of our area of interest, i.e. around $\alpha_{\text{threshold}} = 30\%$. We can now see the effect that the number of nodes in a network has on the selection of an adequate misbehaviour threshold. For instance, if we desire to set the misbehaviour threshold to a value that yields less than 10% change of wrongly detecting a well-behaved node in the networks of Figure 4-9, then it is required that $\alpha_{\text{threshold}} \geq 30\% + 9\%$ (offset = 9%) for the 20 node network, $\alpha_{\text{threshold}} \geq 30\% + 12\%$ (offset = 12%) for the 40 node network, $\alpha_{\text{threshold}} \geq 30\% + 14\%$ (offset = 14%) for the 60 node network, and $\alpha_{\text{threshold}} \geq 30\% + 16\%$ (offset = 16%) for the 120 node network. Thus, it is evident that there is a relation between the misbehaviour threshold and the number of nodes in a network. Our results also imply that using the same misbehaviour threshold value in networks with different nodes can produce different likelihoods of wrongly detecting well-behaved nodes. This is true even if the average number of dropped packets is the same between networks. Therefore in order to not exceed a desired maximum probability of wrong detections, the offset value has to be adjusted to account for the higher number of nodes in a network. Table 4-1 introduces a set of suggested values that can be given to the misbehaviour threshold offset depending on the number of nodes in the network and the maximum probability of wrongly detecting well-behaved nodes as misbehaving.

![Figure 4-9 Probability of detection $P_D$ vs. misbehaviour threshold $\alpha_{\text{threshold}}$ for networks dropping 30% of packets](image)

The misbehaviour threshold parameter $\alpha_{\text{threshold}}$ can be linked to factors or requirements which influence the way it is calculated. For instance, $\alpha_{\text{threshold}}$ can be set to specific values or levels by network management policies that seek to guarantee certain security level or quality of service.
required by high-level entities such as network administrators or application layer software modules. A second option is to set $a_{\text{threshold}}$ to the average network misbehaviour, considered to be the normal behaviour of a well-behaved node, plus an estimated offset. In this case, the average network behaviour can be obtained thanks to the overall behaviour view that MNs have for the network, after aggregating the behaviour metrics. Then management policies could set the offset to a value that satisfies a given maximum probability of wrongly accusing a node as it will be seen in the next Section.

<table>
<thead>
<tr>
<th>Number of Network Nodes</th>
<th>10%</th>
<th>20%</th>
<th>30%</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>Offset = 9%</td>
<td>Offset = 5%</td>
<td>Offset = 3%</td>
</tr>
<tr>
<td>40</td>
<td>Offset = 12%</td>
<td>Offset = 7%</td>
<td>Offset = 4%</td>
</tr>
<tr>
<td>60</td>
<td>Offset = 14%</td>
<td>Offset = 8%</td>
<td>Offset = 5%</td>
</tr>
<tr>
<td>120</td>
<td>Offset = 16%</td>
<td>Offset = 9%</td>
<td>Offset = 6%</td>
</tr>
</tbody>
</table>

### 4.2.3 Accusation Phase

As described in the previous section the network behaviour information is collected, aggregated and sent to MNs for them to analyse. Thus, MNs have access not only to the overall network behaviour but also to each node’s individual behaviour data. This makes them responsible for controlling and adjusting the procedures carried out to detect and accuse misbehaving nodes. Therefore the accusation concept developed in this Section is implemented by MNs within our proposed protection scheme.

A single detection is not sufficient to accuse a node of misbehaviour since this would result in the probability of wrongly accusing well-behaved nodes being very high. If the punishment of accused nodes is network isolation, then in such a case the overall network performance is likely to be worse than that of a network with misbehaving nodes, due to the lack of enough nodes to guarantee network connectivity. For this reason, we propose that a node in our scheme be accused of misbehaviour only if in a number of behaviour checks $ch$ the node is detected $md$ (misbehaviour detections) times as misbehaving. Since the order of those $md$ detections does not matter, combinatorics and probability theory can be used to derive Equation 4-3. We assume that detections are independent from each other and that the probability of detection $P_D$ remains constant during the evaluation period.
Policy-Based Adaptable Misbehaviour Detection and Node Accusation

\[
P_A = P_{md, ch} = \sum_{i=md}^{ch} \binom{i}{i-1} \times P_D^{md} \times (1 - P_D)^{(i-md)}
\]  

(4-3)

Where,

- \( P_A \rightarrow \) Probability of accusation,
- \( P_{md, ch} \rightarrow \) Probability of \( md \) misbehaviour detections in \( ch \) behaviour checks,
- \( P_D \rightarrow \) Probability of a single detection,
- \( \overline{P_D} = 1 - P_D \rightarrow \) Probability of a single no-detection, and
- \( \binom{ch}{md} = \frac{ch!}{md!(ch-md)!} \).

Equation 4-3 states that if, for instance, \( ch = 4 \) and \( md = 2 \) the probability of a node being accused of misbehaviour is the probability that the two detections occur in the first two behaviour checks, plus the probability that the second detection occurs in the third check, plus the probability that the second detection occurs in the last behaviour check, i.e. the fourth behaviour check. It follows that the probability of accusation \( P_A \) depends not only on the detection probability \( P_D \) but also on the number of behaviour checks \( ch \), which defines our sliding window size (memory buffer), and on the number of detections \( md \) required for a node to be accused of misbehaviour. Figures 4-10, 4-11 and 4-12 show how Equation 4-3 varies with different values given to its factors.

Figure 4-10  Probability of accusation \((P_A)\) vs. behaviour checks \((ch)\) for a number of required detections \((md)\) with \( P_D = 80\% \)
Policy-Based Adaptable Misbehaviour Detection and Node Accusation

Figure 4-10 presents a set of curves displaying the probability of accusation $P_A$ for different combinations of misbehaviour detections $md$ and behaviour checks $ch$ when the probability of a single detection is $P_D = 80\%$. As an example, if we analyse the curve corresponding to 3 detections ($md = 3$) it can be seen that the curve starts at $ch = 3$, since lower values of $ch$ are not possible for $md = 3$. Then for $ch = 3$ we have that the probability of accusation is $P_A = 0.8 \times 0.8 \times 0.8 = 0.512 = 51.2\%$, which is the probability that all three behaviour checks yield positive misbehaviour detections. When $ch = 4$, three misbehaviour detections out of the 4 possible behaviour checks are required in order to accuse the evaluated node; this gives a higher probability of accusation ($P_A \approx 81\%$). To sum up, the fewer misbehaviour detections $md$ are required and the more behaviour checks $ch$ are taken into account, the higher the probability of accusation $P_A$ is.

In Figure 4-10 the probability of accusation $P_A$ increases to almost 100\% for all curves as the performed behaviour checks $ch$ get closer to 10. Figure 4-11 shows how the probability of accusation $P_A$ reacts when the probability of a single detection $P_D$ is 60\%. In this figure not all curves climb quick enough to reach $P_A \approx 100\%$, which is an indication that the probability of detection $P_D$ is not good enough (not high enough). Finally, Figure 4-12 illustrates what happens when the probability of detection is too low ($P_D = 10\%$). Here the maximum probability of accusation $P_A$ reached is 65\% and corresponds to the case when at least 1 detection out of 10 behaviour checks is needed to accuse an evaluated node.
These figures enable us to select adequate values for the number of behaviour checks $ch$ and the minimum number of detections $md$ needed to accuse a node. In order to illustrate this notion, let us assume a 60 node network consisting of two types of nodes: well-behaved nodes that do not
drop packets on purpose, and misbehaving nodes that drop about 70% of the packets they are supposed to forward. Figure 4-13 shows separately the probability of detection ($P_D$) as a function of the increasing misbehaviour threshold ($\alpha_{\text{threshold}}$) for well-behaved nodes, misbehaving nodes, and the average network behaviour for a network in two different states. In state 1 – the initial state – the network has 50% of well-behaved nodes and 50% of misbehaving nodes. In state 2 the network has detected and isolated half the misbehaving nodes, i.e. 25% of the total number of nodes in the network. If the curves for states 1 and 2 are compared, it is simple to notice that the state 2 curve resembles more the curve for well-behaved nodes. In general the more misbehaving nodes are isolated the more the average network behaviour resembles that of a network with well-behaved nodes only.

If in our example network, regardless of its state, we set $\alpha_{\text{threshold}} = 5\%$ we see from Figure 4-13 that $P_D \approx 10\%$ for well-behaved nodes ($d = 0$). Then if we want to have a probability of wrongly accusing well-behaved nodes no greater than 1%, from Figure 4-12 we observe that setting the number of behaviour checks $ch = 4$ and the minimum number of required detections $md = 2$ would not satisfy our requirement since in such case the probability of accusation $P_A$ is about 5%. Instead if $ch = 5$ and $md = 3$ we obtain that $P_A$ is less than 1% (actually from Equation 4-3 we know $P_A = 0.856\%$). Furthermore, with $\alpha_{\text{threshold}} = 5\%$ the probability of detecting misbehaving nodes is $P_D \approx 98\%$ (from Figure 4-13, 70% Drop) and by using Equation 4-3 (with $ch = 5$ and $md = 3$) we obtain that the probability of accusing misbehaving nodes in our example network is virtually 100%. This example verifies an important property of the protection scheme proposed in this Chapter, i.e. the property of exhibiting a low probability of wrongly accusing well-behaved nodes while maintaining a high likelihood of accusing misbehaving ones.

As has been seen, our protection scheme has several configurable parameters. Furthermore, knowing the desired value of one or more parameters allows our scheme to calculate and adjust the other parameters in order to satisfy the given requirements. We propose therefore that these parameters be the interface between the protection scheme and the management plane. The algorithm’s parameters can be manipulated through low-level policies in order to adapt the system to the current network conditions and at the same time fulfil the goals specified by high-level entities. On the other hand, our scheme also provides the management plane with information related to the current performance of the network. For example, event-driven policies could be triggered when the overall misbehaviour of the network exceeds a pre-established limit. These policies would execute a set of procedures and parameter configurations in order to lead the overall system to a new desired state.

It was briefly mentioned that the number of behaviour checks ($ch$) defines our sliding window size. In fact, a sliding window is kept at MNs for each active node in the network. Every time a
node’s behaviour is evaluated, the new result is inserted into the sliding window and the oldest value is removed from it. This ensures that MNs keep track of the latest \( ch \) behaviour check results. If in those behaviour checks, \( md \) misbehaviour detections or more are found, MNs accuse the respective node of misbehaviour. Then, MNs inform cluster heads (CHs) of the accusation and in turn CHs enforce the respective policies on cluster nodes (CNs). The enforced policies apply to MNs and CHs too. However, the punishment that a node receives due to its misbehaviour depends on the set of policies implemented by the PBM system, which may distinguish between different node types and apply different punishment policies, as explained in the following section.

4.2.4 Adaptability and Management Policies

The adaptability of the detection and accusation processes is achieved in two ways. First, a new method is proposed for the calculation of the detection threshold \( \text{threshold} \). This configurable method uses a weighted algorithm of recent metrics and inherently adapts to the dynamic forwarding behaviour of participant nodes. The second aspect of adaptability is achieved with the use of policies, based on the role-based network organisation of Section 4.2.1. Special policies are introduced to manage the proposed protection scheme, taking into account management objectives and the changing network conditions.

Policies can express the high-level goals of a managing entity and can be interpreted into low-level policies that dynamically control the operation of participating wireless devices. Although policy refinement is currently out of the scope of our work, we propose a simplified translation of high-level policies (e.g. the level of detection rigidity) and variables (e.g. the probability of wrong accusation) to low-level algorithm configuration (e.g. values for \( ch \) and \( md \)).

The main benefit from the proposed use of policies with the aforementioned protection scheme is the facilitation of dynamic management functionality in a controlled manner. This functionality enables a management entity to redefine the expected behaviour and performance of the network through policies. The event-driven evaluation of policy conditions and the automated enforcement of policy actions are two important properties of PBM systems that have been integrated with the proposed adaptive detection and accusation scheme.

We have adopted the established ECA policy notation, where a policy can be expressed as a statement in the form of: on \( \text{Event(s)} \) if \( \text{Condition(s)} \) then \( \text{Action(s)} \), in the same fashion that is done in [126]. In spite of its simplicity, this notation is very effective because it provides the building blocks for complex management logic. This is achieved by creating groups of policies that can be assigned to management or organisational roles. For example, different policies can apply to devices in the CH and CN role. The use of events and conditional expressions in policies provides a dynamic element to the management system, making it able to
adaptable to network conditions. For example, policies can be triggered if the node density in an area has become lower than a dynamically configurable threshold. A series of policy types for adaptable detection and accusation of packet forwarding misbehaviour have been defined, as seen in Figure 4-14.

Figure 4-14 Policy types for adaptable detection and accusation of packet forwarding misbehaviour

We classify management policies in two sets, the protection scheme set and the network deployment set. Policies in the protection scheme set are organised in the security requirements and penalisation policy groups. The network deployment policy set includes an exceptions policy group. Each policy group contains a number of policy rules that express the low-level policies to be enforced on network devices. The modelling of policies in sets, groups and rules follows the recommended practices of IETF, as described in RFC 3460 [114]. The aforementioned policy groups are only a sample of possible policy extensions and form the basis for the use of policies to achieve an adaptable protection scheme. The use of multiple policies and dynamic conditions to affect the values of the same managed objects, introduces the risk of policy conflicts and the need to address conflict detection and resolution (CDR) arises. However, policy conflict resolution is out of the scope of the work described in this thesis.

The protection scheme policy set aims to facilitate the interaction of a managing entity with the underlying methods and algorithms that implement the misbehaviour detection and node accusation. In a way, this set provides an abstraction to the manager, hiding the complex
implementation details and offering two high-level parameters that can be dynamically manipulated: The *detection rigidness* expresses the level of tolerance the network should exhibit to misbehaviour, while the *accusation accuracy* expresses the expected confidence in accusing and punishing nodes in the network. These parameters offer a mechanism to encapsulate the high-level management objectives as enforceable policies, by affecting the accusation probability $P_A$ and determining the detection threshold offset.

The *security requirements group* can be further divided in packet forwarding misbehaviour policies and routing misbehaviour policies. The latter can be used to control an ad hoc routing protection scheme like SAODV [76] [77]. The former policies are the ones that manipulate the parameters of our protection scheme and provide the desired versatility to change according to management objectives. It should be emphasised that the proposed policy set interaction not only provides the means to influence the protection scheme, but also provides the increased adaptability and parameterisation offered from PBM systems by taking in mind network events and other related policy sets (e.g. penalisation or network deployment groups).

The *penalisation group* includes policies to control the node accusation, punishment, and revocation of punishment. Their aim is to refine the actions against those nodes detected as misbehaving by deciding when and how to penalise them. The importance of this group lays in the fact that it can offer a differentiation in the treatment of misbehaviour, thus allowing a manager to introduce different policies for each user class. For example, assuming free vs. paid network access, paying customers can be offered a lower probability of being wrongly accused. Likewise, in ubiquitous computing environments, where there is the possibility that some pre-deployed trusted infrastructure exists, when a trusted cluster head (CH) is accused of misbehaviour an action can trigger other policies to check the CH's current working condition and enforce appropriate actions instead of penalising the trusted device, e.g. the CH may be a base station that needs to be restarted. Accusation policies can provide additional conditions to expedite or delay node accusation while punishment policies control the type of actions against accused nodes. Revocation policies define if and when an accused node can be cleared of its current punishment, e.g. stop node isolation in 30 minutes.

Finally, the *network deployment set* can be used to express policies that prescribe the usage and purpose of the network and are not directly related to the proposed protection scheme. This set can encapsulate a priori management knowledge like the characteristics of the deployment area (e.g. city, stadium, rural), the type of network (infrastructure-based, pure MANET, hybrid), the expected user mobility (e.g. pedestrian, vehicular, mixed) and the types of supported applications (e.g. data, voice, video). Based on the above parameters, an *exceptions group* was created that includes similar policies to the penalisation group. The purpose of this group is to differentiate the penalisation of nodes depending on the deployment conditions that affect their normal behaviour.
For instance, areas with high mobility are expected to have a reduced packet delivery ratio, making mobile nodes more prone to misbehaviour detection.

4.3 Case Study, Evaluation Results and Analysis

In order to evaluate the applicability and implications from the use of the proposed detection and accusation scheme, we now outline a case study scenario of a ubiquitous computing environment based on the wireless ad hoc paradigm. We then present the evaluation results regarding key aspects of our scheme and use the case study network to illustrate its effect regarding performance and management objectives.

4.3.1 Case Study Scenario

Ubiquitous computing environments (UCEs) (Section 2.1) are a prominent technology that can optionally adopt and benefit from the versatility offered by MANETs (Section 2.3). A self-organising UCE integrates a number of privileged nodes (base stations or routers) that act as gateways for the rest of the participating wireless devices. In order to provide connectivity in areas out of the range of any gateway, such networks are reliant on the packet forwarding behaviour of individual nodes along the created multihop paths. We assume the deployment of an UCE as a case study scenario and consider a number of fixed nodes assigned the MN role, i.e. the devices under the direct control of a managing entity. A number of base stations are also deployed in the examined area and these nodes are assigned the CH role. For the purpose of this scenario, a number of user devices can be dynamically assigned the CH role, based on the adopted role-based framework and distributed algorithm (Section 4.2). The rest of network nodes assume the CN role. All of the participating devices are expected to forward packets to facilitate the required multihop communication. In order to safeguard the operation of the UCE’s wireless communication, the managing entity integrates the proposed protection scheme and introduces appropriate policies to control and automate the detection and isolation of misbehaving users.

As a proof of concept, some example policies are included in Tables 4-2, 4-3 and 4-4. The effect of our scheme on network performance is demonstrated in the simulations of Section 4.3.2. For this case study, policies can be used to achieve the following example goals:

1. Control the calculation method of the detection threshold based on management objectives (e.g. accusation accuracy, such as security requirement policy 1 – SRP1) or on network conditions (e.g. network congestion, such as SRP2 and SRP3).

2. Control the offsetting of the detection threshold based on the desired maximum probability of wrongly accusing a well-behaved node (e.g. policies from SRP1 to SRP5).
3. Differentiate the accusation and punishment enforcement based on the device role, the application type and the business model (e.g. penalisation policies PP1 and PP2, and exception policies EP1 and EP2).

<table>
<thead>
<tr>
<th>Policy ID</th>
<th>Security Requirements Policy Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRP1</td>
<td>on { startup } if { true } then { set_accusation_accuracy(low), set_detection_rigidness(medium), set_threshold_method(fixed_value) }</td>
</tr>
<tr>
<td>SRP2</td>
<td>on { congestion } if { accused_nodes_count &lt;= 40% } then { set_accusation_accuracy(medium), set_detection_rigidness(medium) }</td>
</tr>
<tr>
<td>SRP3</td>
<td>on { congestion } if { accused_nodes_count &gt; 40% } then { set_accusation_accuracy(high), set_detection_rigidness(low) }</td>
</tr>
<tr>
<td>SRP4</td>
<td>on { low_real_time_traffic_throughput } if { accused_nodes_count &lt;= 40% } ^ { congestion == false } then { set_accusation_accuracy(high), set_threshold_method(weighted), set_detection_rigidness(high) }</td>
</tr>
<tr>
<td>SRP5</td>
<td>on { low_real_time_traffic_throughput } if { accused_nodes_count &gt; 40% } then { set_accusation_accuracy(high), set_threshold_method(fixed_value), set_detection_rigidness(low) }</td>
</tr>
</tbody>
</table>

Table 4-2 contains example policies for the security requirements policy group. These policies are somewhat high-level policies in the sense that they specify levels of accuracy and rigidness (low, medium and high) rather than exact values for our scheme parameters such as the misbehaviour threshold $a_{threshold}$, its offset, the number of behaviour checks $ch$, or the number of misbehaviour detections $md$. Thus, this set of policies needs to go through a refinement process to map them into...
actual algorithmic parameters before they can be distributed to the policy enforcement points (PEPs). It is the task of the UCE administrator to decide the parameter values corresponding to low, medium and high levels of accuracy and rigidness. In Table 4-2 SRP1 represents a typical policy to set the initial parameters of a system. Notice that the condition is always true and the triggering event alone is enough for this policy to be enforced. Every time the system starts up the accuracy of our approach is set to low, its rigidness to medium and the threshold is set to a fixed value, which is derived from the rigidness level. SRP2 and SRP3 represent examples of how our scheme can be adapted on the fly to deal with network congestion. In these policies the accuracy and rigidness depend on the percentage of nodes that have already been accused of misbehaviour. Similarly, policies SRP4 and SRP5 are triggered when the real time traffic throughput in the network is low (this network event is considered to be the consequence of the presence of active misbehaving nodes when no congestion exists, see SRP4). Thus if the percentage of accused nodes is less than 40%, the threshold is set to be calculated using a weighted average algorithm applied to the overall network behaviour plus an offset which depends on the rigidness level indicated (high on the example).

![Table 4-3](image)

**Table 4-3  Management Policies for Adaptive Misbehaviour Detection, Penalisation Policy Group**

<table>
<thead>
<tr>
<th>Policy ID</th>
<th>Penalisation Policy Group</th>
</tr>
</thead>
</table>
| PP1       | on { startup } if{ true } then {} 
{ set_accusation_parameters(ch:=5, md:=3) } |
| PP2       | on { startup } if{ true } then {} 
{ set_punishment(isolation) } |

![Table 4-4](image)

**Table 4-4  Management Policies for Adaptive Misbehaviour Detection, Exceptions Policy Group**

<table>
<thead>
<tr>
<th>Policy ID</th>
<th>Exceptions Policy Group</th>
</tr>
</thead>
</table>
| EP1       | on { node_check } if { Node.role == CN } \{ Node.owner == paying_user } then {} 
{ set_accusation_parameters(ch:=5, md:=4) } |
| EP2       | on { node_check } if { Node.role == CH } \{ previous_warning == false } then {} 
{ set_punishment(warning) }, 
{ set_accusation_parameters(ch:=ch+2, md:=md+2) } |
Table 4-3 shows example penalisation policies to configure the parameters behaviour checks $ch$ and misbehaviour detections $md$ on startup (PP1) in addition to set network isolation as the default punishment for nodes accused of misbehaviour (PP2). Table 4-4, on the other hand, shows example exception policies that modify the type of punishment to be applied to an accused node depending on the role it plays in the network. EP1 changes the number of detections required to accuse of misbehaviour a paying network user. EP2 sets the punishment for cluster heads (CHs) to an initial warning and allows for other two misbehaviour detections before punishing the node.

### 4.3.2 Evaluation Results and Analysis

As usual, we perform our simulations using the GloMoSim simulation package [48] [49]. The results presented for each value are the average of 20 simulation runs. Widely deployed standard protocols are used to provide traffic and communication between nodes. The simulation parameters have been selected in order to obtain results that are meaningful when coupled with our study case scenario. Unless explicitly stated, our simulation parameters take the following values: i) nodes move according to the random waypoint mobility model with a speed randomly chosen with uniform distribution between 4m/sec and 6m/sec, yielding a mean node speed of 5m/sec (which is about the average speed of a car in a city centre), and a standard deviation of 0.58m/sec, ii) the wireless transmission range of every node is 100 metres, iii) the node density is $5 \times 10^4$ nodes/m$^2$, iv) the link capacity is 2 Mbps, v) the MAC layer protocol is the IEEE 802.11 DCF, vi) the underlying routing protocol is AODV, vii) the total simulation time for each scenario is 1800 seconds, and viii) network isolation is the punishment enforced for any nodes accused of misbehaviour.

Roles are assigned to each node in the network following the adopted role-based framework (Section 4.2). The number of MN is predefined as 3 and remains constant for all simulations, while the number of CHs depends on the number of network nodes. Before every behaviour metrics collection for misbehaviour detection, the simulated networks are re-clustered using Wu's approach [127] as described in Section 4.2.1. Having in mind our case study, MN nodes can be operator-controlled fixed routers, while a number of user devices are selected as CHs to assist distributed management. In this Section we first present a set of results showing how our protection scheme effectively improves the network performance in the presence of misbehaving nodes. Then the reaction time of our approach is shown for different network sizes and number of nodes. Finally, we demonstrate that the overhead introduced by our proposed scheme allows it to scale well.

An important parameter to evaluate the effectiveness of our approach is the network throughput gain that it can offer to networks affected by misbehaving nodes that drop packets in a
Policy-Based Adaptable Misbehaviour Detection and Node Accusation

In this Section we present results that demonstrate that our protection scheme effectively improves the average network throughput as it detects and accuses misbehaving nodes in the network. For our first set of results, simulations for networks of 40, 60 and 120 nodes were run in areas of 80 000m$^2$ (282.8m x 282.8m), 120 000m$^2$ (346.4m x 346.4m) and 240 000m$^2$ (489.9m x 489.9m) respectively. The networks were set-up with 40% of their nodes misbehaving by dropping 80% packets. The misbehaviour threshold $\alpha_{\text{threshold}}$, i.e. the maximum amount of packets that nodes are allowed to drop without being detected, is 60%. The sliding window size or number of behaviour checks is $c_h = 5$, the minimum number of detections required for an accusation is $m_d = 3$, the reporting period is $T_{\text{report}} = 60$ sec.

![Figure 4-15 Average network throughput vs. simulation time (40 nodes)](image)

Figures 4-15 (40 node network), 4-16 (60 node network) and 4-17 (120 node network) depict the value of the average network throughput as the simulation time progresses from 180 to 1800 seconds. Each throughput value corresponds to the average network throughput in the previous 180 seconds. For this reason, all curves start at 180 seconds instead of 0 seconds. Each graph displays a) networks without misbehaving nodes (No Misbehaviour), b) networks with misbehaving nodes using our proposed protection scheme (Detection & Accusation), and c) networks with misbehaving nodes but with no means of defending themselves from any type of attack (Misbehaviour Alone). As can be seen, our protection scheme effectively mitigates the effects of packet forwarding misbehaviour in a network and offers a substantial improvement on the average network throughput.
At the start of the simulations the throughput in networks implementing our protection scheme is similar to that of a network with misbehaviour alone. This is due to the fact that our algorithm requires the acquisition of behaviour metrics before it can detect and accuse any misbehaving nodes. However, as the simulation time elapses our approach rapidly detects, accuses and isolates misbehaving nodes leading to a steep increase of the network throughput until it closely
Policy-Based Adaptable Misbehaviour Detection and Node Accusation

approaches that of a network with well-behaved nodes only. It is interesting to notice that the 120 node network (Figure 4-17) seems to have a sharper network throughput increase than that of the 40 (Figure 4-15) or 60 (Figure 4-16) node networks. This suggests that in larger networks nodes exhibiting packet forwarding misbehaviour are detected quicker than in smaller networks. The reason for this is that in larger networks the mean hop distance between source and destination is greater and more nodes are involved in the packet forwarding process. This increases the probability of misbehaving nodes actively intervening on the path between source and destination, which in turn leads to an earlier evaluation of their behaviour. Thus, misbehaving nodes can be detected and accused sooner in large networks than in small networks.

We next consider a 60 node network (Figure 4-18) over a terrain of 120 000m² (346.41m x 346.41m). Four groups of misbehaving nodes are set that drop 80%, 60%, 40% and 20% of packets respectively. Each of the groups consists of 10% of the total number of nodes in the network, i.e. 6 nodes per misbehaving group. The remaining network and algorithm parameters stay unchanged. For this set of results we assume a relaxed security policy with $\alpha_{\text{threshold}}$ fixed at 70% for the first half of the simulation. Half way through the simulation policies change the misbehaviour threshold from a fixed value to be equal to the weighted average network misbehaviour plus a pre-established offset (8%).

![Figure 4-18 Average network throughput vs. simulation time (60 nodes)](image)

Figure 4-18 shows how the fixed threshold $\alpha_{\text{threshold}} = 70\%$ policy is ineffective since the network throughput does not improve significantly during the first half of the simulation. In fact, $\alpha_{\text{threshold}} = 70$ fails to detect nodes dropping less than 70% of the packets (60%, 40% and 20%). In
the middle of the simulation (900 seconds) the misbehaviour threshold is changed by policies to be the weighted average network misbehaviour plus an offset of 8%. This yields a value much lower than 70% which permits the algorithm to detect and accuse previously undetected misbehaving nodes. As these nodes are isolated from the network, the average network misbehaviour is reduced which leads to a reduced misbehaviour threshold thus allowing for the detection of nodes dropping lower amounts of packets. By the end of the simulation when all misbehaving nodes are isolated, the network throughput becomes close to that of a network with well-behaved nodes only.

Figure 4-19  Misbehaviour threshold vs. simulation time (60 node network)

Figure 4-19 shows the misbehaviour threshold ($\alpha_{\text{threshold}}$) corresponding to the “Adaptable Detection & Accusation” curve of Figure 4-18. On this figure it is possible to observe the significant drop in the value of $\alpha_{\text{threshold}}$ as is changed from a fixed value (70%) to the average network misbehaviour plus an offset of 8% (at 900 seconds). Then $\alpha_{\text{threshold}}$ continues to decrease gradually as the remaining misbehaving nodes are isolated from the network. By the end of the simulation the misbehaviour threshold value is close to 10%, which indicates that the average network misbehaviour is of about 2%. This particular example illustrates the importance that policies can play in ubiquitous computing environments implementing our proposed protection scheme. A single network event can trigger a set of policies capable of driving the network into a new state where previously undetected misbehaving nodes are isolated from the network. To accomplish such a task, pre-defined policies must interface with our protection scheme through
the configurable parameters of our algorithm (e.g. \( ch, md \) and \( offset \)) and the selected method to calculate/set the network misbehaviour threshold \( \alpha_{thres} \) as seen in Section 4.2.3.

Additionally, policies can help in the identification of critical nodes whose removal can result in a lower network performance, e.g. a lower network throughput. By constantly monitoring variables such as the network throughput and the average network misbehaviour, it is possible to identify whether the isolation of a node causes more harm than good to the network. For instance, if after a node is isolated from the network the average network misbehaviour and the network throughput decrease, then this is an indication that a misbehaving node was effectively isolated but also that the node was of critical importance for the network. Therefore, policies can be triggered that let the node back in the network despite its misbehaviour.

A different angle to observe and analyse our simulation results is offered by the reaction time of our proposed approach. Figures 4-20 and 4-21 allow us to observe how quick our protection scheme responds to the presence of misbehaving nodes in networks of 40, 60 and 120 nodes with areas of 80 000\( \text{m}^2 \), 120 000\( \text{m}^2 \) and 240 000\( \text{m}^2 \) respectively. Additionally, Figure 4-21 demonstrates that policies can effectively help to improve the detection and accusation of misbehaving nodes in networks with nodes exhibiting different levels of misbehaviour.

![Figure 4-20 Percentage of misbehaving nodes being isolated vs. simulation time](image)

Figure 4-20 shows the percentage of misbehaving nodes isolated as the simulation time elapses. This graph corresponds to the same networks shown in Figures 4-15, 4-16 and 4-17. However, Figure 4-20, unlike the previous ones, allows us to see the reaction speed of our protection...
scheme in accusing and isolating misbehaving nodes. It shows that the 120 node network responds the quickest to the presence of misbehaving nodes and effectively isolates them. It is followed by the 60 node network and then by the 40 node network which exhibits the slowest reaction time. This confirms that larger networks exhibit smaller reaction time, i.e. they take less time to detect misbehaving nodes as previously explained in this Section.

Figure 4-21 illustrates the reaction times for the networks of Figures 4-16 and 4-18. It clearly shows the importance of selecting an adequate misbehaviour threshold in order to effectively accuse and isolate misbehaving nodes from the network. The curve corresponding to the network of Figure 4-18 (60 Node Network – 2 Thresholds) shows that initially with $\alpha_{\text{threshold}} = 70\%$ only 25% of the misbehaving nodes are isolated from the network, i.e. only nodes dropping 80% of the packets are isolated. However, at simulation time = 900 seconds the misbehaviour threshold is set to the weighted average network misbehaviour plus an 8% offset, at which point the amount of misbehaving nodes isolated rapidly increases to reach 100%, i.e. now our approach detects nodes dropping 60%, 40% and 20% of the packets.

![Figure 4-21 Percentage of misbehaving nodes isolated vs. simulation time (60 nodes)](image)

Our final set of results presents the overhead produced by our protection scheme compared against the overhead produced by the underlying routing protocol (AODV), and the CBR traffic load (the application layer traffic), in terms of their contribution to the total network traffic. The information displayed in Figure 4-22 corresponds to the 120 node network of Figure 4-17. The mean node speed is increased from 0 to 20m/sec in order to see its effect on our approach.
Figure 4-22 shows the contribution to the total network traffic of three types of protocols: CBR traffic, AODV overhead and the proposed protection scheme overhead. The total network traffic takes into account contributions from generated and forwarded packets. This means that if a packet travels over a three hop network path, it is accounted three times since every time it is transmitted it consumes network resources. The traffic percentage is calculated with respect to packets transmitted instead of bytes, since in the IEEE 802.11 wireless networks standard each packet is associated with the competition to gain access to the wireless medium regardless of its size. As can be seen from the figure, our approach contributes with a small overhead fraction to the total network traffic, and such a fraction is independent of node speed. This is expected since the traffic produced by our approach depends on how often behaviour metrics are reported rather than on mobility issues. The reporting of information is controlled by the adopted policy-based framework (Section 4.2) and the reporting period $T_{report}$, which is 60 seconds for the networks of Figure 4-22. Based on our results, we conclude that with the correct tuning of reporting parameters and its immunity to node speed, our protection scheme can scale well to medium mobile wireless environments. On the other hand, the figure also shows how the contribution of the AODV overhead to the total network traffic increases from about 5% in a static network to above 50% when the mean node speed is 20m/sec. This increase is due to more frequent link breakages, which leads to more route request (RREQ) broadcasts. Finally, the CBR traffic contribution to the network traffic decreases as a result of the increase of the AODV overhead.
4.4 Summary

In this Chapter we have presented an adaptable protection scheme that is capable of effectively detecting, accusing and punishing nodes that exhibit packet forwarding misbehaviour in accordance with the changing network conditions and the management policies set by high-level entities. The effectiveness and efficiency of our approach was verified through an extensive set of simulations.

We have also identified an interface between the management plane and our protection scheme. Such an interface is provided by parameters such as the misbehaviour threshold $\alpha_{threshold}$, the offset applied to misbehaviour threshold, the number of behaviour checks $ch$ performed on each node, the misbehaviour detections $md$ required to accuse a node of misbehaviour, and the method to calculate/set the misbehaviour threshold, e.g. a moving weighted average or a fixed value. At the same time we have identified the overall network behaviour, which is calculated at manager nodes (MNs), and the partial and local behaviour view obtained by cluster heads (CHs) as feedback mechanisms that can be used by the management plane to fine tune the level of security desired on a ubiquitous computing environment.

Finally, we have shown that by making use of policies at the management plane to control our scheme's detection and accusation parameters, the rigidness and accuracy of our approach can be customised in order to punish nodes that exhibit different levels of misbehaviour in the network. Furthermore, different types of punishment can be established for nodes that execute different tasks or have different roles in the network. Also, we have demonstrated through simulations that policies can effectively improve the detection and accusation power of our proposed algorithm while maintaining a low probability of accusing well-behaved nodes, i.e. nodes whose behaviour does not exhibit significant and consistent variances above the average network behaviour.
Chapter 5

5 A Collusion-Resistant Misbehaviour Detection and Node Accusation Approach

As discussed in Chapter 2 ubiquitous computing environments (UCEs) typically consist of large numbers of heterogeneous devices that can be distributed over large physical areas. For this reason it is desirable that approaches designed to work in UCEs address any scalability issues.

Our proposed approach in Chapter 4, for instance, requires a more efficient clustering algorithm that allows it to scale better. Additionally, such a clustering algorithm should provide a means to keep track of manager nodes (MNs) in the network (even if they are on the move) and select only “good enough” nodes to be cluster heads (CHs). In the previous chapter “good enough” nodes were identified through the capability function $CF(V_i)$ introduced in Section 4.2.1. However, its use is somewhat limited by the prior fulfilment of a set of rules regarding an evaluated node’s neighbourhood. One of the objectives of the work in this Chapter is to propose the use of a clustering algorithm that allows for better scalability, manager nodes traceability, and appropriate cluster heads selection. Our proposed strategy, along with some other ideas, is a modified version of a clustering algorithm that we have developed and used in some of our previous work on service discovery mechanisms [129]. Therefore, in this Chapter we introduce part of such work in order to make the concepts presented easier to understand and offer a logical development of ideas.

Chapters 3 and 4 also present approaches in which it is assumed that nodes in the network do not collude in order to bypass the security measures in place. Additionally, it is supposed that nodes reporting metrics do not lie, i.e. if node $V_i$ reports that it sent $x$ packets to node $V_j$ for $V_j$ to forward it is presumed that the transmission of the $x$ packets actually took place, which is not necessarily true in a real network. In this Chapter we introduce a new concept based on discrepancies between reported metrics that, as we demonstrate, offers a predefined degree of resilience against colluding nodes and prevents the accusation of well-behaved nodes due to metrics reported by lying nodes (referred to as liars).

This Chapter is organised as follows: Section 5.1 reviews service discovery strategies with emphasis on mainstream service discovery solutions and service discovery strategies for
MANETs. Section 5.2 introduces an efficient network clustering algorithm within the context of service discovery and evaluates it through simulations. This algorithm is a key component for the selection of cluster heads in our network model. Then, Section 5.3 describes how our approach can achieve resilience against misbehaving, lying and colluding nodes by allowing nodes to report metrics on their own behaviour. To this end, a new concept based on discrepancies between nodes is also introduced. Evaluation results and analysis for our discrepancy-based protection scheme are also presented in this Section. This Chapter is finally concluded in Section 5.4 with a brief summary.

5.1 Introduction to Service Discovery Strategies

Service discovery and service composition play a fundamental role in ubiquitous computing environments where applications and services are not necessarily deployed onto a pre-existing network, but instead the network itself grows out of the applications and services the users want. This approach enables users to view the network in the manner most appropriate to them and their requirements [129]. An expected result of this approach is the ease with which larger, more complex services can be composed from smaller ones. However, in order to interact with each other, services and applications need first to know of the existence of other services in the environment as well as the tasks that they are capable of performing and the information they need to perform them. A solution to this problem is offered by service discovery mechanisms which typically are a set of software components with network and storage capabilities that allow applications and services in a system to find and make use of other network services that are useful to them.

A service can be either software as a web or e-mail server, or hardware as a printer, camera, scanner, etc. A client application wishing to use a service needs to know two things: the location of the service, i.e. the address of the machine running the service, and how to use the service, i.e. the interface. Both problems may be dealt with through a Service Discovery Protocol (SDP). However, some approaches prefer to follow the model of a well known interface for every service. This approach assumes that a client trying to contact a service knows the way to use it beforehand while the service location stills a responsibility of the SDP.

Service discovery strategies can be categorised depending on the methodology they follow to advertise services available in an environment. The three main categories are [130]:

- **Centralised Architecture**: A group of devices is selected to act as service repositories – also known as service coordinators, directory agents or lookup services. When a device has a service to offer it must inform service repositories (at least one of them) about the type of service, attributes and location where the service is running. In this approach the device that
registers the service is not necessarily the device offering the service. Later, when a client needs to use a specific service it consults the service repositories and if it finds one or more matches for its query it selects the service it wants to use and contacts it.

- **Distributed Architecture**: A client needing a service multicasts or broadcasts a query requesting the service. In the multicast approach the query reaches those devices belonging to the multicast group. The multicast group consists of the set of all devices that have a service available (self-advertising). In the broadcast approach, on the other hand, flooding is limited using techniques such as expanding ring search to prevent queries from circulating indefinitely or traversing unnecessarily the whole network. In both approaches each service provider having a matching service responds back to the query originator with a unicast reply.

- **Hybrid Architecture**: This architecture combines the centralised and distributed architectures in an attempt to blend their strengths while filtering out their weaknesses. Different approaches are possible each of them resulting in substantial improvements under certain circumstances.

All service discovery strategies can be placed in at least one of the categories above. What approach is followed by an SDP depends a lot on the environment that it is targeting. For instance, service discovery strategies addressing current computing systems tend to use more centralised architectures, whereas those targeting UCEs or MANETs seem to prefer more distributed approaches.

### 5.1.1 Mainstream Service Discovery Strategies

Many of the most well known service discovery strategies currently available have been designed as a response to the arising needs of existent computing systems (i.e. computing systems of the last decade). Additionally, these protocols have been developed by large companies, consortiums and well established standardisation bodies as part of larger comprehensive solutions for networked systems. In this subsection we review some of these well known service discovery protocols.

To start with, Jini [45] is Sun Microsystems' solution to the service discovery problem. Jini is purely implemented in Java [43] [44], which provides its components with independence from the operating system (OS) on which they are running. Jini consists of four essential component types: lookup services, Jini services, Jini clients and services' proxies. Lookup services are service repositories where the services available in the network are registered. Lookup services can typically be found by polling a well known multicast address. Jini services refer to services that are registered with one or more lookup services. Jini clients are the users, applications and services that make use of a Jini service. Finally, a service's proxy is a software component that a Jini service stores in a lookup service to be later retrieved by a Jini client. A service's proxy
contains all the intelligence on how to contact and use its Jini service. Jini, however, has not been widely adopted and Sun Microsystems has transferred responsibility for this project to The Apache Software Foundation under the project name Apache River [131].

The Service Location Protocol (SLP) [132] [133] has been developed by the Internet Engineering Task Force (IETF), and unlike Jini it is a vendor independent standard and has been designed to work specifically with TCP/IP networks. SLP consists of three main processes or agents [132]: user agents, service agents and directory agents. A user agent is a process working on behalf of a user or application to acquire a service’s attributes and configuration. A service agent is a process working on behalf of one or more services to advertise their attributes and configuration. A directory agent receives information from service agents and stores it in a centralised fashion for its later retrieval by user agents in an efficient manner. SLP has been designed to work in a distributed or centralised manner. In small networked environments service agents can be configured to respond to the queries of each user agent (distributed approach). However, in larger networks, where a distributed approach would be inefficient, service agents register services they are responsible for with a directory agent which can be later contacted by user agents requiring a service (centralised approach).

The Salutation Protocol [134] is a cooperation architecture developed by the Salutation Consortium (which has formally disbanded). It follows a centralised architecture only but it supports operating system and communication protocol independence. The Salutation Architecture defines an entity called the Salutation Manager (SLM) that works as a service broker between service providers and service users. SLMs play two important roles in the Salutation Architecture: i) it acts as service repository, and ii) it provides for operating system and communication protocol independence. To achieve the latter SLMs offer a standard Salutation Manager Application Programming Interface (SLM-API) to service providers and service users. Thus, a service user and a service provider, which can be in different networks with different communication protocols, do not establish direct contact, but instead they contact their intermediate SLM through its SLM-API and let it take care of translating information from one network to the other if required.

The Universal Plug and Play (UPnP) protocol [135] is developed by an industry UPnP Forum [136] led by Microsoft Corporation. Its initiative is similar to that of the Plug and Play (PnP) protocol developed some years ago, which allows for the installation and configuration of a computer’s peripherals in a simple manner. PnP’s basic idea is that a user plugs a peripheral to a computer and it will automatically be ready to play almost instantly. UPnP is an extension of this concept applied to networked environments, i.e. a device can be plug to a network and it should be installed and configured without intervention of the human counterpart. To make this possible UPnP uses proven techniques and well known standards such as TCP/IP, DHCP [137] and XML.
These standards can be implemented in any computing language thus generating the concept of universality. Additionally, UPnP defines two system components: device points which offer services, and control points which use and control the device points. Control points broadcast their requests while device points periodically advertise its presence to the network. Due to its broadcasting nature UPnP is mainly used in small networked environments such as small office home office (SOHO) networks.

Targeting short range wireless transmission environments (typically one hop networks) is the Bluetooth protocol [6] [7] [139] designed by the Bluetooth Special Interest Group which is a privately held, not-for-profit trade association whose member companies are leaders in the telecommunications, computing and network industries. The main service discovery element in the Bluetooth technology is the service discovery protocol (SDP) component. Each Bluetooth network device holds a SDP module. On the server side SDP maintains information about the services available for remote use in its service records database. On the client side a local application uses the SDP component to discover services available in other machines. Thus, when an application requires a service it contacts its local SDP which sends out a request packet to be received by the SDPs of one-hop-away devices. Remote SDPs check their service records database and send back a response if they find any matches. Local SDPs receiving more than one response to their request pass a list of available services to the application or user for them to decide what service to employ.

5.1.2 Service Discovery Protocols for MANETs

Service discovery protocols for mobile ad hoc networks and ubiquitous computing environments have been extensively studied recently due to the importance that the scientific community and the industry have given to user-oriented environments, where the network is principally seen a collection of nodes working in a collective and cooperative manner to provide services that make users’ tasks and goals simpler to execute and accomplish. However, this subsection reviews only three approaches which represent just a small subset of the large amount of service discovery schemes available in the literature.

Due to the dynamic nature of the topology of MANETs centralised approaches are generally not used in these environments. However, fully distributed approaches tend to generate a lot of network overhead, which does not make them ideal to scale to large networks. Thus, a common approach in MANETs is to use hybrid service discovery strategies where a subset of the network nodes – generally known as the dominating set – is selected to act as an array of service repositories where service providers can keep information about their offered services. One such approach is the fully distributed mediator based service location protocol presented in [140]. The
A Collusion-Resistant Misbehaviour Detection and Node Accusation Approach

The idea of the mediator based technique is to form clusters by scattering mediators (i.e. service repositories) in those network areas where service providers are present while leaving other network areas untouched. In this approach, when a service provider has to transmit information for the first time it checks if it has heard of any mediators in its neighbourhood. If it has, it stays as service provider and registers its services with any mediators it is aware of. If it has not, it becomes itself a mediator and periodically advertises its mediator services to its local neighbourhood. Then, when a client needs a service it can query local mediators for the required service. However, this approach can lead to an unconnected set of virtual clusters and it is not clear how clients can contact services out of their local clusters.

Another distributed architecture based on the selection of a dominating set is that proposed in [141]. This solution proposes to tackle the problem in two phases: a backbone management (BBM) phase and a distributed service discovery (DSD) phase. The BBM phase aims at creating a small size (not necessarily optimum) and relatively stable backbone. After this phase is completed network nodes are either virtual access points (VAPs) or routers between VAPs. The selection algorithm is such that any two VAPs are separated by two or three hops i.e. one or two routers are present between any two VAPs. Additionally, VAP nodes are required to maintain a directory to keep a record of registered services. Once the VAPs have been selected the DSD phase starts. In this phase service providers register their services with at least one VAP. Then, when a client requires a service it sends a request to a VAP (which is never more than a hop away thanks to the VAP selection process). If the VAP has a service matching the query it sends back a response with the service location, otherwise it passes on the request to other VAPs which repeat the process.

Unlike the previous two approaches, Konark [142] is a fully distributed service discovery protocol in which every network node behaves as a peer, i.e. it behaves as both a service provider and a client. The dual role can be assumed by any of the nodes at any time depending on the demanding operations given by the different applications spread throughout the network. Each device has a Konark Service Discovery Protocol Manager (SDPM) which performs three essential tasks: i) discovers services on behalf of Konark applications, ii) advertises a device’s own services, and iii) keeps a service registry with information about remote (in other devices) and local (in the same device) services available in the network. The service registry is organised in a tree-like structure where higher levels represent generic service categories while the lower levels are closer to concrete services. In fact, the lowest level in the tree structure consists of those components that represent actual services.

Konark uses two methods to carry out service discovery. The first one is passive push. This model is followed by peers that wish to offer services to others. They advertise their services periodically to a multicast address that is listened to by all peers having a Konark SDPM. The second method,
active pull, is performed by peers that are in need of a service. Peers looking for a particular service send out a discovery message on a fixed multicast group. Then, nodes with services matching the query reply including the necessary service information. When a peer receives a reply it updates its own service registry and passes to the application layer the list of services matching the service request. Finally, aiming at reducing network overhead, nodes can overhear each other’s replies. Thus, a node avoids responding with redundant service information that has been included in other nodes’ replies to the same service request [142].

5.2 Home-Zone Based Service Discovery Mechanism

We now introduce a location-based clustering approach (virtual clusters) and a home-zone based service discovery mechanism, which were originally published in [129]. We then show in this Section how the virtual cluster and home-zone concepts can also be applied to node location discovery for routing purposes, as originally proposed in [143]. In Section 5.3 we will show how these concepts, in turn, can be integrated with our proposed protection scheme to improve its scalability. Although the virtual clustering approach [143] and the home-zone strategy [144] were previously proposed, to the best of our knowledge, work in [129] and in this Section is the first to employ such concepts in a area different to routing protocols.

The basic idea of our scheme is to dynamically assign service repository functionality to certain nodes located in certain pre-defined network locations (i.e. home-zones) so that they become responsible for keeping information on a particular service or set of services. This strategy enables nodes providing a service $S_q$ to update their details and those of the provided service – address, location, service description, etc – with those nodes that have been specifically selected to be service repositories for service $S_q$, regardless of the service provider’s location or degree of mobility. Also, any clients requiring service $S_q$ can contact the same selected service repositories and get the details of those nodes providing service $S_q$. This strategy improves scalability in a number of ways. First, it minimises superfluous flooding, secondly it prevents control messages from traversing unnecessary parts of the network, and thirdly it minimises the latency involved in the service discovery process.

5.2.1 Clustering and Service Discovery

It has been shown that routing protocols that use approximate location information scale better than topology-based routing protocols [143]. This motivates us to use geographic location information in our service discovery mechanism for scalability reasons. Thus, our proposed approach assumes that a node is capable of acquiring its network location through either a GPS or a GPS-free positioning system, and that such a node advertises its position to its neighbours
through periodical HELLO packets. This approach allows network nodes to know not only their own position but also the current position of its neighbours. When a source node desires to send a packet to a destination node it must first obtain the destination’s position. This could be done, for example, by broadcasting a location request (LREQ) to which the destination can respond with a location reply (LREP). However, our approach uses a more effective method to locate destination nodes which is also based on the concept of home-zones, which we introduce later. Once the source knows the destination’s location it appends it to the data packet to be sent, calculates which of its neighbours is the closest to the destination (since it knows its neighbours’ positions), and transmits the data packet to that neighbour. The node receiving the packet follows the same steps, i.e. finds the next closest hop to the destination and forwards the packet to it, and the process is repeated until the destination is reached. This is a conceptual illustration of how data forwarding is achieved in our network model.

Our approach also makes use of the virtual cluster principles proposed in [143]. The idea is that a geographical area is divided into equal regions of squared shape in a systematic way, so that each mobile node can determine the square it resides in if its location information is available. For this purpose we assume that i) each node has a clear picture of the network dimensions and its virtual clusters at the instant the network starts operating, ii) each virtual cluster has a unique identifier \( VC_id \), iii) nodes get their location information through either a GPS or GPS-free positioning system, iv) each network service has a unique service identifier \( S_{id} \), and v) there exists a universal hash-function that maps every service to a specific virtual cluster based on its service identifier \( S/d \). Nodes that have a service to offer (or service providers) are responsible to register their service with the corresponding virtual cluster and according to the network hash-function as explained later in this Section. Also, service providers may move about the network possibly changing virtual clusters on a constant basis, but the virtual cluster they are associated with remains the same. By contrast, nodes located in a particular virtual cluster act as service repositories for one or more services. However, when a service repository changes virtual clusters, it stops being repository for the set of services associated with its old virtual cluster, and becomes repository for a new set of services associated with its new virtual cluster. This means that the service repository functionality of a node varies depending on its location. For example, a node \( v_i \) in virtual cluster \( VC_a \) may work as service repository for the service \( S_q \) (regardless of the location of \( S_q \) and the virtual cluster \( S_q \) is in); while if it moves to virtual cluster \( VC_b \) it may function as service repository for service \( S_r \).

Figure 5-1 illustrates how the assumptions above work. All network nodes are aware of the network dimensions and the coordinates defining each virtual cluster. Consequently, if a node is capable of acquiring its position with a GPS or GPS-free mechanism, it is also able to work out in which virtual cluster it is physically located. Then when a service provider has a service to offer
and wants to make it available to other network nodes, it uses the service identifier ($S_{ID}$) to calculate the virtual cluster ($VC_{ID}$) in which it has to register and update its service. In Figure 5-1 node $v_{45}$ offers service $S_{1}$ and it has calculated that $VC_{3}$ is the virtual cluster where this service must be registered/updated (we explain later how this calculation can be done). The figure also shows that nodes in $VC_{3}$ ($v_{6}, v_{7}, v_{12}, v_{13}$ and $v_{19}$) are service repositories for service $S_{1}$. Later when node $v_{9}$ wishes to use service $S_{1}$ it can also calculate that virtual cluster $VC_{3}$ is responsible for keeping information on this service by using the standard hash-function. Thus, node $v_{9}$ contacts a node in $VC_{3}$ to obtain the location of a node providing service $S_{1}$.

In our scheme each node has its own service repository. The service repository functionality for a given service $S_{ID}$ is assigned to nodes located in a virtual cluster that has a unique identifier $VC_{ID}$. Accordingly, every node residing in that virtual cluster is responsible for maintaining the details of service $S_{ID}$ in its service repository. This means that each virtual cluster becomes a home-zone for a particular collection of services. Since our mechanism requires all nodes to store service information about some other service providers, it can be classified as an all-for-some approach, which can scale well [143].

![Figure 5-1 Home-zone based service discovery approach](image)

Our home-zone based service discovery strategy employs a hash-function to map a service identifier to a corresponding virtual cluster. The selected hash-function must ensure an equally
A Collusion-Resistant Misbehaviour Detection and Node Accusation Approach

distributed load on all virtual clusters; otherwise it may produce increased traffic in some virtual clusters and cause congestion. Equation 5-1 illustrates the hash-function concept.

\[
\text{hash-function}(S_{ID}) = hf(S_{ID}) = VC_{ID}
\]  

(5-1)

An example of a suitable hash-function is the modulo operation (a mod b), which finds the remainder of a division between two given numbers – modulo is the hash-function used in the examples and simulations of this Chapter. Equation 5-2 shows an example where the modulo operation is used along with the service identifier \(S_{ID}\) and the number of clusters in the network \(VC_{#}\) to find the virtual cluster \(VC_{ID}\) where the service \(S_{ID}\) details should be registered and updated, i.e. the virtual cluster that is the home-zone of service \(S_{ID}\).

\[
hf(S_{ID}) = (S_{ID} \mod VC_{#}) + 1 = VC_{ID}
\]  

(5-2)

In our approach each service provider first needs to identify the corresponding set of service repositories (i.e. virtual cluster) for each service it provides by applying the hash-function of Equation 5-1 to each service identifier \(S_{ID}\), and then it registers and updates the service location with nodes in that cluster. In our scheme the service location update mechanism is mobility driven as well as time-driven. A static node sends service updates in a periodic manner to its home-zone while a node on the move sends its updates depending on its speed – the faster it moves the more often it updates its location. This mechanism aims at ensuring that the service location information maintained in service repositories is fresh and correct.

Another characteristic of our service discovery strategy is that service updates (which are used to register and update the location of a service) and service requests are unicast towards the centre of their corresponding virtual cluster. Service replies, on the other hand, are unicast towards the location of the node that sent the service request. This unicast approach prevents control messages from traversing unnecessary parts of the network. Once a service update reaches its intended home-zone it is not unicast anymore towards its centre, instead the first node receiving the service update stores the new information in its service repository and, through cluster-wide flooding, informs other nodes within the same cluster so that they can update their repositories.

Whenever a client node is interested in a particular service it makes use of the same hash function used by service providers to determine the home-zone associated with the needed service. Then it sends a service request towards the calculated virtual cluster in order to contact any of its service repositories (i.e. nodes in the virtual cluster). In our scheme a service reply can be initiated by service repositories in the home-zone, the service provider itself (if the service request happens to pass through it) or any intermediate node as long as it contains “fresh” information about the service being requested. In the worst case, when a querying node has not received any response within a pre-specified period of time after having tried for a pre-specified number of times, it will start gradually flooding its service request in the network following an expanding search
approach. This may happen, for example, due to the fact that a given service’s home-zone is currently empty.

Let us apply this algorithm to Figure 5-1. In the figure $v_{45}$ is a service provider offering service $S_{11}$ (e.g. a colour printer). If we assume that $S_{ID} = ID$ then we have that for service $S_{11} = 11$. Also, for Figure 5-1 $VC_3 = 9$ since the network consists of 9 virtual clusters. Consequently, when node $v_{45}$ wants to advertise its service $S_{11}$ it uses our example hash-function as follows:

$$hf(S_{11}) = hf(11) = (11 \mod 9) + 1 = 2 + 1 = 3 \rightarrow VC_3$$

In this way node $v_{45}$ determines that it must send its service update packets to virtual cluster $VC_3$ (it actually sends the updates to the geographical centre of $VC_3$), which is the home-zone of service $S_{11}$, as shown in Figure 5-1. It can also be seen that in $VC_3$ the first node to receive the service update is $v_{19}$, which takes care of letting other cluster nodes know of the updated location of node $v_{45}$. Also, in the figure node $v_8$, which is a service client, needs to use service $S_{11}$. Subsequently, it utilises the hash-function in exactly the same way the service provider $v_{45}$ did, and it therefore finds that the home-zone of service $S_{11}$ is virtual cluster $VC_3$. Node $v_8$ then sends a service request to $VC_3$ (directed to its geographical centre) which is promptly replied to by the first node in $VC_3$ to receive the request, i.e. node $v_{12}$. The reply travels back towards node $v_8$ containing a list of network nodes offering service $S_{11}$. Finally, node $v_8$ chooses the closest service to itself, or optionally the decision can be left to the user counterpart for them to decide which service is more convenient, e.g. the printer closest to the exit as they are about to go out to a lunch meeting.

The process described above can be extended to be used in other scenarios besides service discovery. For example, in this Chapter we use the home-zone concept to assist our selected location-based routing protocol in finding a destination’s location, in the same way that is done in [143]. In this case every network node applies a hash-function to its own network ID (e.g. IP address) to discover which virtual cluster is to become its home-zone. Consequently every virtual cluster is selected as the home-zone for a subset of network nodes in the same way that it happens for services, and the subset of network nodes a node is responsible for depends solely on which virtual cluster it is. In our approach network nodes must periodically send location updates to their corresponding home-zones to ensure that possible enquiring nodes can obtain their current location. On the other hand, source nodes wishing to find a destination apply the hash-function to the destination’s ID to work out the destination’s home-zone. Then, source nodes send a Location Request (LREQ) to the destination’s home-zone and obtain back a Location Response (LRES) with the destination’s location. After this, data packet forwarding occurs as described in the first paragraph of this Section (5.2.1).
5.2.2 Evaluation Results and Analysis

We perform our simulations in the GloMoSim network simulator [48] [49]. For comparison purposes we have implemented a fully distributed service discovery mechanism that works in the same fashion as AODV. In this approach when a node needs a service it broadcasts a service request packet with a very low TTL (time to live) so that the packet can not travel more than a few hops. Nodes having a matching service may reply to the request – provided they are willing to offer their services. If after a pre-defined time no service replies are received, the requesting node broadcasts a new request with a larger TTL in order to cover a greater area. In the absence of replies this process is repeated a couple of times (each time with a larger TTL) and as last resort the requesting node executes a network wide broadcast in an attempt to find the needed service. In our simulations we use TTL = 1, 3 and 5 before attempting a network wide broadcast. The fully distributed approach described in this paragraph is evaluated with two different underlying routing protocols: AODV [59] [60] and home-zone based geo-casting (HZGC) [143]. Our reason to evaluate the fully distributed approach with HZGC as its routing protocol is that HZGC is the underlying routing protocol used by our home-zone based service discovery (HZSD) strategy, thus this decision brings fairness to our evaluation.

Unless explicitly stated otherwise, our simulation parameters takes the following values: i) Nodes move according to the random waypoint mobility model with a constant speed chosen uniformly between 0 and 10 m/sec, this yields an average node speed of 5 m/sec and a standard deviation of 2.89 m/sec, ii) the wireless transmission range of every node is 100 metres iii) the node density is $5 \times 10^4$ nodes/m$^2$, iv) the link capacity is 2 Mbps, v) the MAC layer protocol is IEEE 802.11 DCF, vi) the total simulation time for each scenario is 300 seconds, and vii) the hash-function utilised is that of Equation 5-2.. Additionally, at simulation start time nodes are selected as service providers with a 50% probability, thus about half the nodes offer services. A service provider randomly selects only one of the twelve possible services to offer, and all services have an equal chance of being selected by a service provider. All network nodes can become clients at some point during the simulation. Nodes remain inactive during a uniformly selected period of time between 10 and 15 seconds. After this period they decide to make a service request with 70% probability. Then, regardless of whether they decide or not to request a service, they go inactive for another randomly selected period between 10 to 15 seconds. This process is repeated through the simulation time and provider nodes are not permitted to request the same service that they are offering.

The principal metrics of interest to evaluate our scheme are network control overhead and service discovery latency. Control overhead is defined here as the control traffic generated as part of the
service discovery processes only, i.e. control overhead associated with the routing algorithms AODV and HZGC are not included.

The objective of our first set of simulations is to assess the scalability of our service discovery approach in terms of the increasing node-count. In order to properly see the effect of increasing the number of nodes on the simulated service discovery schemes, the terrain-area is also proportionally increased so that the average node density is kept constant. The number of nodes for this simulation set is 20, 80, 180, 320, 500 and 720 for terrain-area sizes of 200x200 m², 400x400 m², 600x600 m², 800x800 m², 1000x1000 m² and 1200x1200 m² respectively. Our simulation figures show the results for the fully distributed approach with AODV and HZGC routing, and for our home zone service discovery strategy.

Figure 5-2 Average network control overhead vs. number of nodes in the network

Figure 5-2 depicts the average control overhead incurred by the service discovery schemes as a function of increasing number of nodes in the network. It can be observed that the two fully distributed approaches performed similarly. However, the approach employing home-zone based geo-casting (HZGC) routing offers a slight improvement. The fully distributed approach does not benefit greatly from HZGC since its service discovery phase is based on broadcasting, which is the main source of control overhead. Additionally, the inherent working mechanism of the underlying MAC layer protocol, i.e. IEEE 802.11 distributed coordination function (DCF), exacerbates the problem. In IEEE 802.11 DCF when a collision occurs the nodes try to retransmit creating even more traffic which congests the network rapidly. As a result, the control overhead of the fully distributed approaches display a sharp increase when the number of network nodes
increases beyond 100. On the other hand, our home-zone based service discovery mechanism offers a significant reduction in network overhead when compared to the fully distributed strategy. As already stated, our home-zone service discovery mechanism employs HZGC routing and broadcasting is only used when strictly necessary, e.g. when a service cannot be found by querying the corresponding home-zone. For this reason, our proposed approach introduces very little control overhead in the network.

**Figure 5-3** Average service discovery latency vs. number of nodes in the network

Figure 5-3 shows the average latency involved to successfully discovery and contact a service as a function of the increasing node count. The latency for our simulations is defined as the time elapsed between the instant a service client sends a service request up to the moment the service client receives an acknowledgement from a service provider confirming that it was successfully contacted. Additionally, we only display information for services that are successfully contacted. Any failures to contact a service, e.g. due to a lost service request or service reply, are not included in our calculations. In this respect, the figure does not represent the success rate of the service discovery strategies, but it indicates how long in average a service client takes to successfully contact a service provider. Figure 5-3 shows that our home-zone based service discovery approach outperforms the fully distributed strategy. However, a gradual increase is seen on our scheme's curve when the number of network nodes goes over 300. This behaviour can be linked to collisions, and their consequent retransmissions, of some service requests and replies due to increased network traffic, which delays the final acknowledgement from the service provider.
The second set of results show how network control overhead and service discovery latency react to changes in the mean node speed. For these simulations we consider a network consisting of 300 nodes spread over an area of 1000x1000 m², which yields a node density of $3 \times 10^{-4}$ nodes/m². The mean node speed is then gradually increased from 0 m/sec up to 20 m/sec in steps of 2 m/sec. This time the minimum and maximum speeds are set such that the standard deviation is always 0.58 m/sec.

![Figure 5-4](image)

**Figure 5-4** Average network control overhead vs. mean node speed (300 node network)

![Figure 5-5](image)

**Figure 5-5** Average service discovery latency vs. mean node speed (300 node network)
Figure 5-4 shows that the network overhead is relatively unaffected by the nodes' mean speed. Also, unlike Figure 5-2, the fully distributed approach using HZGC routing introduces much less control overhead in the network than its AODV routing counterpart. Because the fully distributed approach and AODV rely both on broadcasting there is a higher chance that they saturate the network quicker, which leads to increased collisions. In this situation, as already mentioned, the IEEE 802.11 DCF protocol reacts by trying to inject even more packets in the network, thus generating an abrupt increase in the network overhead. This same explanation applies to Figure 5-5 since more collisions result in more service requests and replies being lost, subsequently delaying the communication between service clients and service providers.

Our final set of simulations aims at illustrating the effect that the population of service providers can have on network control overhead and service discovery latency. The simulated network consists of 100 nodes confined to an area of 600x600 m² while the mean speed remains constant a 5 m/sec with a standard deviation of 2.89 m/sec. The percentage of service providers is increased from 10% to 80% of the total number of network nodes.

![Network control overhead vs. service providers percentage](image)

**Figure 5-6** Network control overhead vs. service providers percentage (100 node network)

Figures 5-6 and 5-7 show that the only approach affected by the number of service providers in the network is the fully distributed approach employing AODV routing. Again the broadcasting nature of the fully distributed approach and AODV routing together generates enough traffic to rapidly saturate the network and lead it to a state where nodes constantly try to retransmit their packets unsuccessfully. This yields a lot of network control overhead and the subsequent collisions produce a large average latency since contact between service clients and service
providers is delayed. The other two curves in both figures seem to be fairly independent from the number of service providers available in the network as only a tiny upwards trend can be observed. Finally, the figures demonstrate that the lowest network control overhead and average latency is exhibited by our proposed home-zone service discovery strategy.

![Figure 5-7]

> Figure 5-7  Average service discovery latency vs. service providers percentage

(100 node network)

This Section has demonstrated the advantages of using the virtual clustering and home-zone based strategies to reduce network overhead and improve service discovery latency, especially when compared to approaches making use of AODV and other broadcast-based techniques to provide service discovery. Additionally, these results offer a general idea of the benefits that virtual clustering can bring to the protection scheme proposed in this thesis in terms of its scalability, while the home-zone based strategy can provide an effective and efficient way to keep track of the location of manager nodes. Section 5.3.1 describes in detail how these concepts are integrated with the misbehaviour detection and node accusation algorithms.

### 5.3 Collusion and Liar Resistant Packet Forwarding

#### Misbehaviour Detection and Node Accusation

This Section introduces two modifications targeting two different aspects of the protection scheme described in Chapter 4. It is our view that these modifications provide for a strong misbehaviour detection and accusation scheme that can cope with issues that have not been addressed in the previous chapters of this thesis. The first of our modifications (Section 5.3.1)
A Collusion-Resistant Misbehaviour Detection and Node Accusation Approach

seeks to improve the scalability of our approach by employing the position-based clustering algorithm presented in Section 5.2. We then extend this idea to propose an improved method to select cluster heads in the network and obtain a more efficient solution in comparison to the clustering approach described in Section 4.2.1. Additionally, by using the home-zone concept presented in Section 5.2 the network can keep track of all manager nodes in a well-organised manner. The second modification affects our misbehaviour detection algorithm. We introduce the concept of discrepancies which allows our scheme to prevent wrong accusations due to nodes that lie about their metrics (i.e. nodes that are liars, Section 5.3.2) and offers a predefined degree of resilience to colluding nodes (Section 5.3.3).

5.3.1 Improved Scalability: Clustering and Election of Cluster Heads

Adaptability is brought to our protection scheme by means of a hierarchical management plane that has cluster nodes (CNs) at its bottom tier, cluster heads (CHs) in its middle tier, and manager nodes (MNs) at its top tier. As stated in Section 4.2.2, in ubiquitous computing environments (UCEs) MNs and CHs can be statically assigned by a network administrator to satisfy network deployment issues. However, this is not always possible and the assignment of these roles can also be partially or completely dynamic. Our approach’s hierarchy has two main objectives: efficiently distribute and enforce network management policies defined at MNs, and accurately and effectively collect the behaviour metrics of nodes actively intervening in the communication process. Consequently, a well organised selection of MNs and CHs can result in a reduction of control overhead associated to our scheme and in a more effective manner to distribute policies and gather behaviour metrics. In our strategy we assume that there exists a small number of MNs (static or mobile) that have been assigned by a network administrator and are therefore trusted entities; this assumption remains as in Chapter 4. These MNs serve as network management policies definition points, which are used by a high level entity (i.e. an application or a network administrator) to define or modify policies to be applied to a UCE. The selection of CHs, on the other hand, is left to our clustering algorithm.

The clustering algorithm introduced in Section 4.2.1 is not very efficient in terms of the potential number of nodes that may be elected CHs in the network. For example, if we consider the network topology depicted in Figure 5-8, it can be observed that the number of CHs, which are the nodes belonging to the connected dominating set, can be considerably higher than the number of CNs. In the figure the relation of CHs to CNs is 3:1, i.e. 75% of the nodes have been selected as CHs even after applying the optimisation rules of Section 4.2.1. Additionally there are two subtle but important problems that can be highlighted by analysing Figure 5-8. The first is that due to their neighbourhood conditions, nodes in the figure do not have a chance to apply their capability function $CF(v)$ in order to select the most capable nodes as CHs. This proves that the

125
selection of capable ("good enough") CHs can be limited by neighbourhood conditions, as stated in the introduction of this Chapter. The second problem is that 8 out of the 12 selected CHs do not have any CNs associated to them. This is very inefficient, especially if a reactive routing protocol such as AODV is used. In this case, after collecting their local metrics all 12 nodes will need to broadcast a route request to find a MN to which report their collected metrics, and after all this effort there are 8 CHs that will only report metrics about themselves since they do not have any associated CNs, i.e. they did not collect any metrics apart from their own. A similar situation occurs with the distribution of network management policies. MNs distribute new policies to CHs to be enforced on themselves and their associated CNs. However, in Figure 5-8 half the nodes (i.e. the CHs without associated CNs) only have to enforce the policies on themselves, which is a clear indication that some of them could be CNs instead.

The problems exhibited by the example of Figure 5-8 have motivated us to look for alternative options. A more efficient clustering approach has been introduced in Section 5.2 as part of our work on service discovery mechanisms, and in this Section we illustrate how it can be modified to account for the requirements of our protection scheme. Additionally, its location-based routing protocol paired with our home-zone concept applied to the discovery of a destination's location permits our approach to scale better than when using topology-based routing algorithms such as AODV, which is used in the evaluation results of previous chapters.

The idea applied to our new clustering approach is similar to that of Section 5.2. We assume that every network node has a clear picture of the distribution and area covered by each virtual cluster in addition to the network dimensions. Each virtual cluster has a unique identifier $VC_d$ and nodes get their location information through either a GPS or GPS-free positioning system. This last assumption is very reasonable since nowadays devices capable of calculating their geographic position are very common. For example, many mobile phones offer map services highlighting the user's current position. Some of them use GPS systems and others do it by calculating their distance from near by Wi-Fi hotspots or cellular towers. Once a node has acquired its position it can work out in which virtual cluster it currently is physically located. Figure 5-9 illustrates a network that has been clustered employing the virtual clusters notion. However, unlike virtual clusters of Section 5.2 (see Figure 5-1), in every virtual cluster of Figure 5-9 a leader or cluster...
A Collusion-Resistant Misbehaviour Detection and Node Accusation Approach

head (CH) has been elected: \(v_8\) in \(VC_1\), \(v_4\) in \(VC_2\), \(v_{13}\) in \(VC_3\), \(v_{15}\) in \(VC_4\), \(v_{24}\) in \(VC_5\), \(v_{26}\) in \(VC_6\), \(v_{47}\) in \(VC_7\), \(v_{37}\) in \(VC_8\) and \(v_{45}\) in \(VC_9\). CHs are then responsible for collecting metrics and enforcing network management policies in CNs within their own virtual cluster.

The cluster head selection process is as follows: new nodes in a virtual cluster (because they just moved in or started up) query their neighbours for the ID and location of the current CH; if they receive no reply they send a virtual cluster wide broadcast query. After a node sends these two queries there are two possible situations: i) it receives a reply with the ID and location of the CH, or ii) it receives no replies. In the first case the new node registers as a CN with the CH. In the second case the node triggers a cluster head election process by broadcasting within its virtual cluster a Cluster Election Packet (CEP). Nodes receiving a CEP calculate their capability function \(CF(v)\) and inform other virtual cluster nodes of its value. Then the node \(v_i\) with the higher capability function \(CF(v_j)\) is elected as CH while the rest become CNs and register as such with the CH. Nodes include in their registration packets their location and the value of their capability function. In this way when a CH changes virtual cluster it can determine from its stored information the most capable node in its old virtual cluster. Next, it informs about the CH change to nodes in its old virtual cluster by sending them a Changeover Packet (COP). The COP packet is initially unicast towards the old virtual cluster and then broadcast within it, thus informing every node in the cluster of the CH change. The new CH assumes its new responsibilities and the CNs
associate with it. Meanwhile the old CH marks its own role as CN and starts the process that all new nodes follow when they arrive to a new cluster.

As implied above a node’s capability function $CF(v_i)$ plays a major part in electing the most appropriate nodes to be CHs in their respective virtual clusters. Unlike the approach presented in Chapter 4, where the capability function has limited influence in the selection of CHs because it is used as part of an optimisation rule, in our new clustering approach this function is the only driving force to elect cluster heads, which emphasises its importance. The capability function employed to elect appropriate cluster heads is presented in Equation 5-3 below.

$$CF(v_i) = \frac{w_1PP(v_i) + w_2MEM(v_i) + w_3PVCC(v_i)}{w_4MR(v_i) + w_5AL(v_i)}$$ (5-3)

Equation 5-3 shows that our capability function depends on five node properties: processing power ($PP(v_i)$), memory capacity ($MEM(v_i)$), mobility ratio ($MR(v_i)$), proximity to the virtual cluster centre ($PVCC(v_i)$), and accusation likelihood ($AL(v_i)$). Properties $PP(v_i)$ and $MEM(v_i)$ were first introduced in Chapter 4 and their definition remains exactly as in Section 4.2.1. Property $MR(v_i)$ was also introduced in the same section and it was associated with a node $v_i$’s average frequency of link breaks with its neighbours. However, in this Section $MR(v_i)$ is associated instead with the average frequency with which node $v_i$ changes virtual clusters. As can be deduced from Equation 5-3 the more often a node changes virtual clusters the less ideal it is to become a CH. By contrast, a node that resides in a virtual cluster for a long time is a good cluster head choice since its election would result in fewer changeovers and CH elections, thus reducing network overhead and providing cluster stability.

Property $PVCC(v_i)$ in the context of Equation 5-3 expresses the convenience that represents to have the elected node as close as possible to the virtual cluster centre so that it can be easily reached by all nodes in the cluster. While this property is not essential, we include it so that between two nodes of similar capabilities and characteristics the one closest to the virtual cluster centre is favoured to be elected as CH. The final property $AL(v_i)$ is closely related to the concept of discrepancies introduced later in Section 5.3.2. The basic idea is that the more discrepancies a node $v_i$ exhibits the higher its accusation likelihood is. High likelihoods of accusation are one of the characteristics of misbehaving nodes, which are targeted by our approach to isolate them from the network. Consequently, property $AL(v_i)$ in Equation 5-3 expresses our approach’s unwillingness to accept a probably misbehaving node as a virtual cluster’s CH. After all, the node will soon be isolated from the network and a new cluster head election would be necessary.

Just as in Section 4.2.1 each variable in Equation 5-3 is normalised to a value range of $[0, 1]$ by dividing them by their maximum values. Weights $w_1$, $w_2$, $w_3$, $w_4$ and $w_5$ are assigned values according to the importance of their pairing variables in the definition of the capability function.
Nevertheless, the weight value assignment should be done such that the sum of $w_1$, $w_2$ and $w_3$ is equal to 1, and the same applies for weights $w_4$ and $w_5$.

In our approach metrics are reported in a periodic manner in a similar fashion to the one described in Chapter 4 (as explained later in Section 5.3.2). This implies that CHs contact MNs to report their local information and they must have a way to acquire the MNs’ location. We propose that MNs have a home-zone just a service does in Section 5.2. However, instead of applying a hash-function MNs can have one or more predefined and well-known home-zones. Thus, MNs can update their home-zones on their locations in the same fashion that services do in Section 5.2 (i.e. time and mobility driven). This concept is illustrated in Figure 5-9 where node $v_{j_2}$, which for simplicity is the only MN in the network, is updating its location with virtual cluster $VC_5$, which in this network is the unique well-known home-zone for all manager nodes. On the other hand, CHs can contact those well-known home-zones to obtain the list of MNs in the network and their respective locations. Then, each CH chooses its closest MN and sends to it its local metrics report.

5.3.2 Dealing with liars: Discrepancy-based Detection and Accusation

In Chapter 3 and Chapter 4 misbehaviour detection was based on a modified version of the principle of conservation of flow (Equation 3-2). This approach gathers the metrics provided by the neighbourhood $U_j$ of a node $v_j$ in order to evaluate its behaviour. However, node $v_j$ does not have a say in its own behaviour. This can be a problem because a node $v_p$ which is a neighbour of $v_j$ may be a liar, i.e. $v_p$ may be a node providing false metrics about node $v_j$ in an attempt to get $v_j$ accused and isolated from the network. For example, $v_p$ may claim to have transmitted $x$ packets to node $v_j$ for $v_j$ to forward, when in fact $v_p$ did not transmit any packets to $v_j$. Consequently, node $v_j$ may be persistently detected as a misbehaving node and eventually accused and isolated from the network without giving $v_j$ a chance to “tell its own side of the story”. In this Section we propose an approach that provides for a fairer detection and accusation system that also offers a predefined degree of resilience against colluding nodes.

We propose that all network nodes maintain a table with four metrics: $T_{ij}$, $T_{ji}^*$, $F_{ji}$ and $F_{ji}^*$. Metric $T_{ij}$ is the number of packets that node $v_i$ claims to have sent to node $v_j$ for $v_j$ to forward. Metric $T_{ji}^*$ is the number of packets that node $v_j$ claims that node $v_i$ has actually sent to it for it to forward. Metric $F_{ji}$ is the number of packets that node $v_i$ claims that node $v_j$ has forwarded to it. Metric $F_{ji}^*$ is the number of packets that node $v_j$ claims that it has actually forwarded to node $v_i$. In other words, $T_{ij}^*$ and $F_{ji}^*$ are the metrics reported by the node under evaluation, i.e. “its own side of the story”. Cluster nodes report these four metrics in a periodic manner to their CHs, which store this information, wait for all registered CNs to report their metrics, and forward it to the network MNs. A clear difference with our approach of Chapter 4 is that CHs and MNs do not aggregate
any metrics; instead they have four matrices – one for each type of metric – in which they store individual values. In every matrix rows and columns represent network nodes related to the values stored in each cell \([\text{row}, \text{column}]\). Rows represent claiming nodes and columns represent nodes involved in the claim. This concept is illustrated in Figure 5-11 that shows the matrices that an MN would keep in its memory for the forwarding multihop network of Figure 5-10 where node \(v_1\) has sent 100 UDP packets to node \(v_2\), and it has also received 200 UDP packets sent by node \(v_3\).

![Figure 5-10 Multihop packet transmission between two network nodes](image)

![Figure 5-11 Matrices corresponding to the multihop packet transmission of Figure 5-10](image)

From Figure 5-11 we see that in matrix \(T\) the metric \(T_{42}\), which is stored in the cell \([4, 2]\), is equal to 100 and it is read as follows: “node \(v_4\) claims to have sent 100 packets to node \(v_2\) for \(v_2\) to forward”. Likewise, in matrix \(T^*\) metric \(T_{24}^*\), which is stored in the cell \([2, 4]\), tell us that “node \(v_2\) claims that \(v_4\) has sent to it 100 packets for it to forward” or alternatively “node \(v_2\) claims to have received 100 packets from \(v_4\) for \(v_2\) to forward”. Similar analyses apply to metrics in matrices \(F\) and \(F^*\).
A characteristic of our model is that in an ideal network, such as that of Figure 5-10, the following equations hold (the superscript $^T$ stands for the transpose of a matrix):

$$T^T = T^* \text{ and } F^p = F^*$$

Or alternatively:

$$(T^*)^T = T \text{ and } (F^*)^T = F$$

However, in real environments packet losses occur in an unintentional manner and it is necessary to account for such losses. This notion is depicted by Equations 5-4 and 5-5:

$$(1 + \alpha_{\text{threshold}}) T_{ij} \geq T_{ij}^* \geq (1 - \alpha_{\text{threshold}}) T_{ij}$$

$$\text{(5-4)}$$

$$(1 + \alpha_{\text{threshold}}) F_{ji} \geq F_{ji}^* \geq (1 - \alpha_{\text{threshold}}) F_{ji}$$

$$\text{(5-5)}$$

It is the task of MNs at behaviour evaluation time to go through the metric matrices checking that Equations 5-4 and 5-5 hold. If any of the two equations does not hold the MN performing the behaviour check records a behaviour discrepancy between nodes $v_i$ and $v_j$. A discrepancy indicates that two nodes differ in their claims with regard to the number of packets transmitted to forward and forwarded by each other. A discrepancy may be due to one or more of several factors: packet loss owing to reasons other than a node’s own fault (e.g. a problem in the communication channel, a node is overloaded, broken, compromised, etc), a node is misbehaving by intentionally dropping packets, a node is lying in its reported metrics about another node’s behaviour which is also perceived as misbehaviour, or a combination of these factors. Figures 5-12 and 5-13 illustrate how misbehaviour is perceived at a MN when a misbehaving node drops 50% of the packets and then lies in its reported metrics (i.e. the node is a liar) in an attempt to incriminate neighbouring nodes. In Figure 5-12 node $v_2$ forwards only 50 packets towards $v_5$ and 100 packets towards $v_4$ (values shown in black colour), but it reports having forwarded 100 and 200 packets respectively (values shown in red colour). In Figure 5-13 discrepancies are highlighted with red and blue colours. Red corresponds to the lying node while blue correspond to its well-behaved neighbours. For this example, by following the process described below and assuming that $\alpha_{\text{threshold}} < 50\%$, a MN finds two discrepancies for node $v_2$, one for $v_4$ and another one for $v_5$.

Discrepancies have a valid time which for simplicity we express in values multiple of the report period ($T_{\text{report}}$). Consequently, if behaviour checks are performed every 60 seconds ($T_{\text{report}} = 60\sec$) the discrepancy valid time can take values such as 60, 120, 180 and 240 seconds, or any other value multiple of $T_{\text{report}}$. Manager nodes maintain a discrepancies table where each entry consists of 3 parameters: first node involved in the discrepancy, second node involved in the discrepancy, and the discrepancy expiry time. Thus, the process of recording a new discrepancy between two nodes $v_i$ and $v_j$ requires a MN to create an entry in its discrepancies table including $v_i$, $v_j$ and the discrepancy expiry time, which is equal to the current time plus the discrepancy valid
time. However, before creating a new entry a MN checks whether a discrepancy for the same pair of nodes already exists. If it does not, a new entry is created. If it does, the expiry time of the already existing entry is updated. This ensures that discrepancies between a same pair of nodes are counted only once regardless of how many times the discrepancy is detected.

Figure 5-12 Multihop network transmission with a liar that drops packets

![Multihop network transmission with a liar that drops packets](image)

After removing any stale entries (i.e. entries that have expired) from its discrepancies table, a MN goes through the table counting the number of behaviour discrepancies exhibited by every node. If a node exceeds the discrepancies limit \( L_{\text{discrepancies}} \) established for its network, it is accused of misbehaviour and isolated from the network. This process is carried out in a similar fashion to that described in Section 4.2.3, i.e. MNs inform CHs of the accusation and in turn CHs enforce the respective policies on themselves and their registered CNs. However, if the accused node is a CH, then the MN sends a special accusation packet towards the CH’s virtual cluster. Cluster nodes within the target cluster receive the special accusation packet and broadcast it in a cluster-wide
manner. This leads to the effective isolation of the current CH and CNs have to trigger a new CH election process. Other virtual clusters are informed of the accused CH node through the conventional medium, i.e. CHs are informed first. Next, the MN removes entries from its discrepancies table that are associated to the accused node. In this way nodes that were previously recorded because of their discrepancies with misbehaving nodes get their discrepancy count reduced by one.

This new discrepancy-based accusation scheme gives rise to a special case. A node that misbehaves by dropping packets but does not lie about its behaviour in its reported metrics is not detected by Equations 5-4 and 5-5. The network corresponding to this particular case is also depicted by Figure 5-12. However, this time we assume that node \( V_2 \) reports its real metrics.

Figure 5-14 shows the matrices obtained at MNs after the metrics collection procedure is carried out. Metrics in blue correspond to metrics somehow related to the misbehaving node, and as it can be seen from the figure, Equations 5-4 and 5-5 hold even for \( \alpha_{\text{threshold}} = 0 \). In fact, for this example \( T^* = T^\ast \) and \( F^* = F^\ast \), i.e. the network exhibits the characteristics of an ideal network. For this reason it is necessary to check for one more characteristic that identifies a well-behaved node: the number of packets that it forwards should be equal (within a factor of \( \alpha_{\text{threshold}} \)) to the number of packets it has received to forward. Consequently, Equation 5-6 must hold (where \( V \) is the set of all network nodes, as defined in Section 3.2.1):

\[
\sum_{\forall V \in V} T^*_{y, V} \geq (1 - \alpha_{\text{threshold}}) \sum_{\forall V \in V} F^*_{V, y}
\]  

(5-6)

Matrices \( T^\ast \) and \( F^\ast \) in Figure 5-14 include the results for the sum of the values in every row in an extra column named "Total". Values for well-behaved nodes are given in green while values for misbehaving nodes are given in red. It can be seen that misbehaving node \( V_2 \) claims to have forwarded 150 packets out of the 300 packets it claims to have received to forward. Consequently, the evaluating MN records a discrepancy between node \( V_2 \) and itself, i.e. a self-discrepancy. Self-discrepancies require a different treatment to ordinary discrepancies. If a MN were to update the expiry time of a node's self-discrepancy every time Equation 5-6 does not hold for that node, then the node could misbehave forever and its reward for being honest would be to have only one "well-updated" entry in the discrepancies table. For this reason, every time a self-discrepancy occurs a new entry is created in addition to updating all the existing self-discrepancy entries for the misbehaving node. Thus, nodes that are honest but misbehave will have their discrepancy count increase by one every time a behaviour check is performed.

Due to the manner in which our new protection scheme detects discrepancies Equation 4-3, which was introduced in Section 4.2.3, can not be used to calculate a node's probability of accusation. Equation 4-3 assumes that there can only be a single detection per behaviour check on every
A Collusion-Resistant Misbehaviour Detection and Node Accusation Approach

network node. However, in our new scheme in a single behaviour check a node can potentially exhibit multiple discrepancies with different nodes. It all depends on how many nodes have come across and communicated with the evaluated node in the last $T_{check}$ period, and how many discrepancies are detected on it. This useful characteristic of our approach allows for the accusation of a misbehaving node in a single behaviour check provided that its number of discrepancies exceeds the discrepancies limit ($L_{discrepancies}$) established for the network. Nevertheless, for the specific case of an honest misbehaving node that exhibits self-discrepancies, Equation 4-3 still applies since such a node gets a single self-discrepancy (i.e. detection) in every behaviour check, i.e. it requires of $L_{discrepancies}$ discrepancies (equivalent to $md$ misbehaviour detections) out of $ch$ behaviour checks for it to be accused of misbehaviour.

![Matrix T](image1)

![Matrix T*](image2)

![Matrix F](image3)

![Matrix F*](image4)

Figure 5-14 Matrices at MNs when there is a non-lying node dropping packets in the network

5.3.3 Addressing Collusion

The previous section has demonstrated the capability of our approach to detect liars, misbehaving liars, and honest but misbehaving nodes. This Section shows through a series of examples that this approach can also resist the coordinated attack of a predefined number of colluding nodes.

Figure 5-15 shows a network where four colluding nodes ($v_4$, $v_5$, $v_6$ and $v_7$) launch an active attack against node $v_2$ in an attempt to get it isolated from the network by the security mechanism in
place. In this attack each of the colluding nodes reports to have sent 100 UDP packets to node $v_2$. Thus, for the network of Figure 5-15 MNs will detect 4 discrepancies for $v_2$ and just 1 for nodes $v_4$, $v_5$, $v_6$ and $v_7$. No self-discrepancies are detected since the colluding nodes can simply claim that they are sources rather than intermediate nodes. As was explained in the previous section, if every behaviour check detects the same discrepancies, no new discrepancy entries are created in the discrepancies table; instead old discrepancies corresponding to the same pairs of nodes are updated. Therefore, the discrepancy count for node $v_2$ remains 4. This is a clear indication that if we desire a network resilient to attacks of up to 4 colluding nodes the limit of discrepancies $L_{\text{discrepancies}}$ must be equal to 5. In general the number of colluding nodes $k$ that our network model can withstand is given by Equation 5-7.

$$k = L_{\text{discrepancies}} - 1$$ (5-7)

Figure 5-15 Colluding nodes lying in their metrics about node $v_2$

A different type of collusion is depicted in Figure 5-16. In this case the colluding nodes $v_2$, $v_3$ and $v_4$ want to fool the security measures in place by avoiding discrepancy detections. To this end, node $v_2$ drops 60 packets out of the 100 it is supposed to forward and reports that it has actually forwarded the 100 packets. So that node $v_2$ is not detected, $v_4$ receives the 40 packets and reports that it actually received 100 packets. Now the problem is for node $v_3$ that has to justify the 40 packets it forwards instead of the 100 expected. In the figure node $v_4$ helps node $v_3$. These two nodes could agree on reporting 40 packets, but then a self-discrepancy would be detected at node

Figure 5-16 Colluding nodes in an attempt to not be detected

135
v\textsubscript{3} because it has already claimed to have received 100 packets. The other option (the one depicted in the figure) is to continue forwarding 40 packets and reporting 100. However, at some point a discrepancy will be found. For instance, in Figure 5-16 MNs detect a discrepancy for node v\textsubscript{4} and another for v\textsubscript{5}. Thus the attempt made by the three colluding nodes to avoid discrepancy detections fails. They would only stand a chance to be undetected if node v\textsubscript{5} (i.e. the destination) helped them to lie by reporting 100 received packets rather than the 40 packets actually received. But this situation would not make much sense because v\textsubscript{5} would be the only network node affected since 60 packets destined for it get lost.

5.3.4 Network Nodes Authenticity and Metrics Integrity

As has been shown in the previous sections of this Chapter, our discrepancy-based protection scheme can deal with packet dropping misbehaviour, metric lying misbehaviour and colluding nodes up to a predefined number. However, for our protection scheme to work properly it is also important to ensure that: i) nodes cannot masquerade as CHs or MNs in order to falsely accuse other nodes of misbehaviour, ii) nodes cannot masquerade as other CNs so as to report false metrics on their behalf, and iii) reported metrics are not tampered with on their way towards the MNs. Although addressing these issues fall out of the scope of this thesis, in this Section we briefly explain how security techniques such as digital signatures and data encryption can help to provide nodes authenticity and data integrity in our scheme. The idea here presented is by no means the only possible solution and we make no assertions regarding its effectiveness. Our sole intention is to prove that with the appropriate security structure in place, the proposed protection scheme can offer a fairly strong resilience against nodes attempting to masquerade as other network nodes (whether MNs, CHs or CNs) and nodes that seek to modify metric packets.

To start with, we assume that all data transfers between network nodes are encrypted (using private keys, public keys or one-way hash chains). Thus data integrity is assured and packets modified while in transit can be identified by the destination. We also assume that MNs are well known entities in the network and that all network nodes are aware of their digital signatures. Consequently, network nodes can verify whether a packet has been truly sent by a MN. Additionally, MNs are the only Certification Authority for CHs. When a node is elected as a virtual cluster’s CH it contacts a MN to be certified. If the MN approves of the election (i.e. if the MN is not suspicious of the new CH) it sends back a packet certifying or accepting the new CH. The packet is received by the new CH and broadcast to all CNs in the cluster (which can verify the authenticity of the packet, as already mentioned) to inform them that it is now an accepted CH. Cluster nodes then register with the new CH and assume it is a trusted entity. The CH and its CNs can keep a copy of the certification packet in their memory to prove to new nodes in the cluster that the CH has been accepted by a MN. However, if a MN does not approve of a CH’s
election, it sends a packet back to the CH's virtual cluster for nodes to start a new election without including the currently elected CH. This process is repeated until a valid CH is elected. Finally, in a similar fashion to CHs, cluster nodes that register with a CH are assumed to be well-behaved trusted entities, and as such they are granted access to the network.

Both CNs and CHs can be removed from the lists of trusted entities and subsequently isolated from the network if they exhibit any type of misbehaviour, e.g. dropping packets, reporting false metrics or even failing to report their metrics in a periodic manner. Nevertheless, the type of penalisation associated to each type of misbehaviour depends on the policies defined at MNs by high level entities such as applications or network administrators. Penalisations can range from simple warnings to eviction from a cluster or total isolation from the network.

When cluster nodes report their metrics to their respective CHs, the CHs keep the signed packets in their memories until they have received metrics reports from all its registered CNs or until a maximum waiting time has elapsed. Next, CHs create a new packet including all received reports, send it to the closest MN (see Section 5.3.1) and wait for an acknowledgement before clearing their memories. If an acknowledgement is not received, the metrics are retransmitted until a maximum number of retransmissions are reached. MNs receiving a metrics report send back an acknowledgement, check the integrity of all packets and the authenticity of their senders, and proceed to evaluate the network behaviour as described in Section 5.3.2.

5.3.5 Evaluation Results and Analysis

As in our previous chapters, we perform our simulations using the GloMoSim simulation package [48] [49]. The results presented for each value are the average of 20 simulation runs. The routing protocol home-zone based geo-casting (HZGC) [143], whose basic functionality was described in Section 5.2.1, is employed to do packet routing and node location discovery in all simulated networks. Unless explicitly stated otherwise, our simulation parameters takes the following values: i) nodes move according to the random waypoint mobility model with a constant speed chosen uniformly between 4 and 6 m/sec, this yields an average node speed of 5 m/sec and a standard deviation of 0.58 m/sec, ii) the wireless transmission range of every node is 100 metres iii) the node density is $5 \times 10^4 \text{ nodes/m}^2$, iv) the link capacity is 2 Mbps, v) the MAC layer protocol is IEEE 802.11 DCF, vi) the total simulation time for each scenario is 1800 seconds, and vii) 8 CBR connections are set up with ideal throughputs of 10240 bits/sec and transmitting packets of 256 bytes in size.

Additionally, clustering in the simulated networks is achieved using the virtual clustering approach of Section 5.2.1. Each virtual cluster has dimensions 200x200m$^2$ and all networks are set up such that a whole number (i.e. without fractions) of virtual clusters fit in them. Each virtual
cluster has a corresponding cluster head (CH) which is elected as described in Section 5.3.1 and using Equation 5-3. However, in our simulations we assume that all network nodes have the same processing power $PP(v_j)$ and memory capacity $MEM(v_j)$ and we omit the respective parameters in the equation. This means that CHs are elected according to their proximity to the virtual cluster centre $PVCC(v_j)$, their mobility ratio $MR(v_j)$ and their accusation likelihood $AL(v_t)$, which is directly proportional to the number of discrepancies exhibited by node $v_t$. On the other hand, the number of manager nodes (MNs) in all simulations is 5, regardless of the number of network nodes. The home-zone for MNs is always the first virtual cluster $(VC_{ID} = 1)$ in the network, and all network nodes are aware of this fact. Finally, in our simulations nodes are accused of misbehaviour and isolated from the network if their discrepancy count reaches $I_{discrepancies} = 6$.

Our first set of simulations aims at demonstrating the effectiveness of our protection scheme in large environments. We simulated 500 node networks in areas of $1000\times1000m^2$ (i.e. the node density is $5\times10^4$ nodes/m$^2$). Two types of networks are analysed: networks without misbehaving nodes and network exhibiting misbehaviour. In the latter type of networks 40% of nodes (i.e. 200 nodes) misbehave by dropping 80% of the packets that they are suppose to forward. Additionally, 50% of them (i.e. 100 nodes) lie on their reported metrics in an attempt to hide their misbehaviour. Figures 5-17, 5-18 and 5-19 show the results for the evaluated networks. Displayed throughput values in Figure 5-17 correspond to the average network throughput of the previous 180 seconds. Thus, the curves start 180 seconds when the first average network throughput is obtained.

![Figure 5-17 Average network throughput vs. simulation time (500 node network)](image)
Figure 5-17 shows the average network throughput for three different networks: networks without misbehaving nodes (No Misbehaviour), networks with misbehaving nodes but without protection (Misbehaviour Alone), and networks with misbehaviour but protected by our proposed scheme (Detection & Accusation). As expected the best and worse cases correspond to the curves No Misbehaviour and Misbehaviour Alone respectively. The network implementing our approach exhibits a low throughput during most of the first half of the simulation, but improves quickly as the simulation time crosses the 720 second mark. The results shown in the figure demonstrate that our approach effectively detects and isolates misbehaving and lying nodes alike in spite of their sheer amount. We believe that the slow improvement of the network throughput in the initial part of the simulation is due to the large amount of misbehaving nodes. Consequently, although our approach starts isolating nodes early in the simulation, as shown in Figure 5-19, enough misbehaving nodes still remain in the network to keep its throughput low.

Figure 5-18 introduces a new important parameter to evaluate the effectiveness of our approach, namely the instant network misbehaviour. We define the instant network misbehaviour as the average misbehaviour of all active network nodes (i.e. node that are actually transmitting, forwarding and receiving packets in the network) from the last behaviour check. In fact the instant network misbehaviour represents the average number of packets dropped per active network node, but expressed as a percentage of the total number of packets that each of them is supposed to forward. As can be observed from Figure 5-18, networks with no misbehaving nodes exhibit the lowest instant network misbehaviour values, while networks with misbehaving nodes but without
A Collusion-Resistant Misbehaviour Detection and Node Accusation Approach

A protection scheme in place exhibit the highest values. On the other hand, networks implementing our protection scheme exhibit high instant network misbehaviour values at the start of the simulation time, but as misbehaving nodes are detected and isolated those values gradually decrease to about 5%. Also, Figure 5-18 confirms an important characteristic of our protection scheme that was first seen in the evaluation section of Chapter 4: a network with misbehaving nodes that implements our proposed protection scheme initially behaves like a network with misbehaving nodes and no protection, but after most misbehaving nodes have been isolated from the network, the analysed network resembles one that exhibits no misbehaviour. Nevertheless, as can be seen from the figure, the instant network misbehaviour for our analysed network does not quite reach the values of a network without misbehaviour, but this can be due to the significant change in the node density. After all, the network starts with 500 nodes and ends with just about 300 nodes.

![Figure 5-19](image)

Figure 5-19 Normalised number of accused nodes, normalised throughput and instant network misbehaviour vs. simulation time (500 node network)

The purpose of Figure 5-19 is to allow the reader to see in a direct manner how the throughput and instant network misbehaviour react as our protection scheme accuses and isolates nodes from the network. The values shown the figure have been normalised and expressed in percentages. The number of accused nodes is normalised by dividing currently accused nodes by the total number of misbehaving nodes (i.e. 200 in the simulated network). Throughput values are normalised by dividing them by the ideal throughput (i.e. 10240 for our CBR connection parameters). The instant misbehaviour throughput is already expressed as a percentage of the total
number of packets an active node is supposed to forward, as explained in the previous paragraph. It can be appreciated from the figure that the instant network misbehaviour starts descending as soon as our protection scheme has accused about 30% of misbehaving nodes. The throughput however show no signs of improvement until about 80% of the misbehaving nodes have been isolated from the network (at simulation time = 720 seconds). These results reinforce the analysis made for Figure 5-17.

Our second set of results aims at demonstrating the benefits of providing our protection scheme with adaptability through policies. For this set of results we simulated 320 node networks covering areas of 800x800m² in order to maintain the node density at 5x10⁻⁴ nodes/m². With the exception of networks without misbehaving nodes, all other networks had 40% of their nodes misbehaving (i.e. 128 misbehaving nodes). Misbehaving nodes are distributed over 4 groups, each containing 25% of misbehaving nodes (i.e. 32 nodes per group) and dropping 20%, 40%, 60% and 80% of the packets they are supposed to forward respectively. Results are shown for networks using a non-adaptable protection scheme (Fixed Detection & Accusation) with misbehaviour threshold $\alpha_{\text{threshold}} = 70\%$ and networks using an adaptable protection scheme (Adaptable Detection & Accusation). Networks with no misbehaving nodes (No Misbehaviour) and networks with misbehaving nodes but without protection scheme (Misbehaviour Alone) are also included for comparison purposes.

Our adaptable approach works as follows: the protection scheme starts by enforcing a misbehaviour threshold $\alpha_{\text{threshold}} = 70\%$, at simulation time $t = 300 \text{ seconds}$ a high-level policy is introduced that requires the system to drive the instant network misbehaviour to under 10%, this triggers an algorithm that lowers $\alpha_{\text{threshold}}$ by 20%, waits for 300 seconds, checks if the goal has been achieved and if it has not, repeats the process again. In our scenario we assume that policies have set a lower limit for $\alpha_{\text{threshold}}$ of 10% to prevent the wrong accusation of well-behaved nodes. Therefore, once $\alpha_{\text{threshold}}$ reaches 10% at $t = 900 \text{ seconds}$ it remains the same until the end of the simulation. Figures 5-20, 5-21 and 5-22 show our simulation results.

Figure 5-20 demonstrates that our adaptable protection scheme offers better results in terms of the average network throughput. Whereas our adaptable approach manages to raise the throughput to about 8000 bits/sec, the non-adaptable protection scheme manages to raise it to just above 5000 bits/sec. However, the improvement offered by the non-adaptable strategy is almost half that of the adaptable one even though we expect it to accuse only about 25% of the misbehaving nodes (i.e. those dropping 80% of packets). The reason for the somewhat unexpected improvement lies in the fact that the non-adaptable scheme isolates from the network those nodes that do most of the packet dropping.
Figure 5-20 Average network throughput vs. simulation time (320 node network)

Figure 5-21 Instant Network Misbehaviour vs. simulation time (320 node network)

Figure 5-21 complements the results of Figure 5-20. In fact, not only does it confirm that a reduction in the instant network misbehaviour corresponds to an increase in the average network throughput, but also that the greater the reduction is the higher the increase is in the corresponding network throughput. Additionally, we can appreciate how similar networks implementing our protection scheme (whether adaptable or non-adaptable) are to networks without protection at the
start of the simulation and during its first minutes, when only a handful of nodes have been isolated from the network.

Figure 5-22 allows us to see the percentage of misbehaving nodes that are accused in the network as the simulation time elapses. As expected up to $t = 300$ seconds both approaches behave in exactly the same way since their misbehaviour threshold $\alpha_{\text{threshold}} = 70\%$. However, 60 seconds later at $t = 360$ seconds a small difference can be observed due to the fact that the adaptable protection scheme has started to lower the misbehaviour threshold in order to achieve the goal specified by high-level network management policies (i.e. instant network misbehaviour < 10%). The reaction of the adaptable protection scheme to each consecutive lowering of the misbehaviour threshold can also be seen at $t = 720$ seconds and $t = 990$ seconds, right after $\alpha_{\text{threshold}}$ is changed.

![Figure 5-22 Percentage of accused/isolated nodes vs. simulation time (320 node network)](image)

Our final set of results illustrate how the selection of CHs is affected by the inclusion of our parameter $\text{accusation likelihood (AL}(v_j))$ in Equation 5-3. This set of results considers networks of 80, 180, 320 and 500 nodes with areas of 400x400m$^2$ (4 virtual clusters), 600x600m$^2$ (9 virtual clusters), 800x800m$^2$ (16 virtual clusters) and 1000x1000m$^2$ (25 virtual clusters) respectively. In Figures 5-23 and 5-24 is possible to appreciate the average time that nodes spend as CHs and the average number of times that they become CHs during a single simulation, as explained below.

Figure 5-23 shows the average time that network nodes spend as cluster heads. In the figure two networks are evaluated that use different strategies to select suitable cluster heads for each virtual cluster. The first network (Normal) employs parameters virtual cluster centre $PVCC(v_j)$ and
mobility ratio \( MR(v_j) \) (as already stated we assume that all nodes have the same \( PP(v_j) \) and \( MEM(v_j) \)). The second network (Discrepancy Based) utilises the parameter \( \text{accusation likelihood} \ AL(v_j) \), which is directly proportional to the number of discrepancies exhibited by a node, in addition to \( PVCC(v_j) \) and \( MR(v_j) \). Figure 5-23 exhibits curves for both well-behaved nodes (Good) and misbehaving nodes (Bad). As it can be appreciated from the figure, when a ‘Normal’ strategy is used to select CHs, well-behaved nodes and misbehaving nodes spend about the same time in CH role. On the contrary, when a ‘Discrepancy Based’ capability function is employed, misbehaving nodes spend significantly less time as CHs while, in a parallel manner, well-behaved nodes see their average CH-role time increase to compensate for the lack of “good enough” nodes. This demonstrates that the accusation likelihood contributes towards a better selection of nodes suitable to be CHs and helps to prevent misbehaving nodes from staying long periods of time as CHs. Additionally, electing as CH a node that is about to be isolated from the network due to its misbehaviour is pointless since sooner rather than later a new selection process has to be triggered. Finally, as can be observed in Figure 5-23, all nodes seem to be less likely to stay for long periods of time as CHs as the number of nodes in the network increases.

Figure 5-23 Average time that nodes spend as CHs vs. number of network nodes (node density = \( 5 \times 10^4 \) nodes/m²)

Figure 5-24 illustrates how many times on average a node becomes CH during the course of a simulation. The figure shows different curves for well-behaved nodes (Good), misbehaving nodes (Bad) and for networks using discrepancies (Discrepancy Based) and not using them (Normal). As in the previous figure, the curves corresponding to the ‘Discrepancy Based’ function clearly
discriminate between well-behaved nodes and misbehaving ones. This improves cluster stability and system reliability in addition to boost the trust of nodes in cluster heads. Additionally, Figure 5-24 shows that, unlike the average time that nodes spend as CHs, the average number of times that a node becomes cluster head is proportional to the number of nodes in the network.

![Figure 5-24 Average number of times that network nodes become CHs vs. number of network nodes (node density = 5x10^4 nodes/m²)](image)

**5.4 Summary**

In this Chapter we have first presented a highly efficient network clustering algorithm which endows our adaptable protection scheme with the ability to scale well to large computing environments. This is a desirable characteristic for solutions targeting ubiquitous computing systems since they can typically consist of hundreds of devices embedded in the users' surroundings.

We have also introduced the concept of discrepancies, which gives nodes under evaluation the chance to tell their “own side of the story” by reporting metrics on their own behaviour. Then through a series of proof-of-concept examples we demonstrated that the discrepancies concept renders our adaptable protection scheme capable of dealing with lying nodes that report false metrics in order to hide their own misbehaviour or in an attempt to get peer nodes accused of misbehaviour. Also, we have shown through examples that our approach is resilient to attacks launch by colluding nodes, provided that the number of colluding nodes is not greater than $L_{\text{discrepancies}} - 1$. 

145
Finally, by analysing the results of simulated networks consisting of 80, 180, 320 and 500 nodes and exhibiting up to 200 misbehaving nodes, we have established the capacity of our approach to scale well to large MANET-like environments. The results have demonstrated that despite the sheer amount of misbehaving and lying nodes in the network our adaptable protection scheme virtually detects, accuses and isolates all of them. Additionally, we have also verified through simulations that by using discrepancies to determine a node’s aptness to be cluster head, the amount of misbehaving nodes that become cluster heads and the average time they spend in this role can be considerably reduced.
6 Summary, Conclusions and Future Work

6.1 Summary

The need to protect the transmission capabilities of wireless multihop communications and more specifically of ubiquitous computing environments (UCEs) was the key motivation behind the research undertaken in this thesis. To this end, we have proposed a protection scheme that effectively detects, accuses and penalises nodes exhibiting packet forwarding misbehaviour that aims at disrupting the communication process in UCEs which rely on wireless multihop transmission. Below, we detail the contributions made to the different aspects of the proposed protection scheme:

a. Collection of information for behaviour evaluation

We have proposed to collect the packets transmitted to forward and forwarded by network nodes, i.e. the nodes’ behaviour metrics, in order to determine the number of packets that they do not forward. Two main strategies for behaviour metrics collection were proposed: a limited broadcast approach and a hierarchical and organisational role-based model. Our limited broadcast approach is tailored to small scale environments and seeks to collect an evaluated node’s behaviour metrics from its local neighbourhood. On the other hand, our organisational role-based model establishes a three tier hierarchy where nodes in the middle layer (cluster heads - CHs) collect metrics from their neighbourhood or cluster (cluster nodes - CNs) and send them to top-layer nodes (manager nodes - MNs) which are in charge of detecting and accusing misbehaving nodes. The scalability of our organisational model is closely related to the clustering algorithm employed in the network.

b. Accurate detection of misbehaving nodes

In this particular aspect we believe that we have moved a step forward with respect to most work found in the literature. First of all, the proposed protection scheme does not suffer from common problems such as ambiguous collisions, receiver collisions or nodes capable of controlling their transmission power because it does not use promiscuous listening to detect misbehaving nodes.
Secondly, two methods have been proposed to assess a node's possible misbehaviour: a detection-based method and a discrepancy-based method. The detection-based method employs metrics reported by an evaluated node's neighbours to determine the fraction of packets it has failed to forward. If the fraction of packets dropped is higher than the network's misbehaviour threshold, then a misbehaviour detection occurs. The discrepancy-based method, on the other hand, utilises an evaluated node's own reported metrics in addition to those reported by its neighbours to determine whether a node is misbehaving. In this method a behaviour discrepancy occurs when two nodes disagree on their reported metrics by a fraction larger than the network's misbehaviour threshold. In our proposed protection scheme the more misbehaviour detections or behaviour discrepancies a node exhibits, the more likely it is to be truly misbehaving. We have also proved through simulations that both these strategies allow our protection scheme to discriminate adequately between misbehaving and well-behaved nodes.

c. Accurate and effective accusation of misbehaving nodes

Accusation is another aspect of our protection scheme that is not available in many of the approaches found in the literature; where avoidance rather than penalisation seems to be the common trend. It is our view, as we have expressed it in this thesis, that nodes should not be awarded with less work for their misbehaviour and at the same time let them retain their network and service access privileges. For this reason, our protection scheme has been endowed with the ability to accuse and penalise misbehaving nodes. In this work, we have favoured node isolation as an adequate penalty for misbehaving nodes that discourages intentional misbehaviour.

Accuracy in our approach is achieved by examining a node's behaviour over a period of time. Single detections or discrepancies are not enough to penalise misbehaving nodes since this approach could lead to the removal of useful well-behaved nodes from the network. Instead our protection scheme penalises only those nodes that are persistently misbehaving, i.e. exhibiting certain number of detections or discrepancies over a period of time. The effectiveness of this strategy has been tested in all chapters of this thesis with extensive sets of simulations and their corresponding results analysis.

d. Adaptability

We have proposed the use of policy based management (PBM) to make our protection scheme capable of adapting to the dynamic conditions of MANET-like UCEs. We have also presented how policies can be distributed to network nodes making use of the same organisational role-based model employed to collect behaviour metrics. Within this context, MNs have been selected as the interface between our scheme and network administrators, who define the high-level policies to be enforced in the network. CHs have been given the task of distributing the
management policies to all CNs, which are network nodes without sophisticated management capabilities that receive and enforce in a local manner polices coming from CHs and MNs. We have also demonstrated through specific simulation scenarios the advantages that our policy-based adaptable protection scheme has over our non-adaptable approach.

e. Network Clustering

Two network clustering algorithms that allow for the selection of CHs have been studied and tested in this work. The first algorithm [127] is a well-known method that creates and maintains a connected dominating set (CDS) in a fully distributed and decentralised manner. Although this algorithm has several known drawbacks, we have used it to prove the feasibility and correctness of our proposed approach. The second studied algorithm [143] is an efficient network clustering method based on the geographical location of network nodes. In addition to allowing for a better selection of CHs, this algorithm provides our protection scheme with the scalability properties required so that it can be deployed in UCEs, which typically consist of large numbers of devices and sensors. Our simulation results and their analysis demonstrated the suitability an efficiency of each method in the creation of a hypercluster (overlay) of CHs to collect behaviour metrics and distribute network management policies.

f. Resilience to lying nodes

It was shown in Chapter 5 that in our protection scheme misbehaving nodes can report false metrics as a means to launch an attack that seeks to isolate well-behaved nodes from the network. We have addressed this problem by allowing evaluated nodes to report metrics about their own behaviour and then detecting discrepancies between reported metrics. Our simulation results have shown that the discrepancy-based protection scheme effectively accuses misbehaving and lying nodes alike, while maintaining a low probability of wrongly accusing well-behaved nodes.

g. Resilience to colluding nodes

In this thesis we have identified collusion as one of the principal and most difficult problems to be tackled by protection mechanisms. In this work we have made an effort towards tackling colluding nodes that work in a cooperative manner to frame a well-behaved node for misbehaviour. Our proposed solution makes use of the discrepancies concept with a slight modification so that successive discrepancies between a same pair of nodes are counted only once towards the accusation of the nodes. We demonstrated the effectiveness of our solution through a proof-of-concept example where 4 colluding nodes attempt to remove from the network a well-behaved node.
6.2 Conclusions

This Section offers a few concluding remarks and observations within the context of the work undertaken in this thesis:

- The self-regulating nature of UCEs requires that they be able to monitor the behaviour of their devices. UCEs' heterogeneity supposes the presence of devices with limited resources, which is an incentive for nodes to misbehave by failing to forward packets in an attempt to preserve their scarce resources (e.g. selfish nodes saving battery power). Additionally, UCEs' ad hoc nature, which allows users to join and leave the network at their will, creates the potential risk of admitting malicious users that seek to disrupt the communication process by launching black-hole and grey-hole attacks. Also, it is possible that nodes misbehave for other reasons.

- Although this work has been developed in the context of wireless networks the principle employed is general enough to be applied to other types of networks. However, we make no attempt to state or predict the efficiency, effectiveness or modifications that our approach may require to work properly in such systems.

- Our proposed protection scheme does not require high density networks in which many nodes can promiscuously listen (overhear) to each others' received and transmitted packets. Instead, it uses metrics accumulated by nodes that are actively intervening in the communication process by sending, receiving and forwarding data packets.

- In general, the fewer packets a misbehaving node drops the more its behaviour resembles that of a well-behave node, as it can be appreciated from our simulation result figures. Thus, a misbehaving node could potentially avoid being detected and accused of misbehaviour by dropping a small fraction of packets. However, it would also mean that our proposed scheme would have managed to enforce a minimum level of good behaviour (or a maximum level of misbehaviour) on the misbehaving node. The simulation results indicate that misbehaving nodes should not drop more than 8% of packets above the network average to have a chance of not being detected (there is also the chance that they will be detected anyway).

- Making use of policies at the management plane to control our protection scheme gives networks administrators a powerful tool to adjust the system to a UCE's particular needs. For example, network administrators could apply different types of penalty or trigger different actions when nodes are accused of misbehaviour. If the accused node is a well-known base station, the best course of action could be to send a notification to an administrator or try restarting the base station. Isolating the misbehaving base station would most likely exacerbate the packet dropping problem detected in the UCE.
Summary, Conclusions and Future Work

Tackling colluding nodes is a complex task that is not always possible to accomplish. In this work we have managed to design a strategy to provide UCEs with resilience against nodes that collude to hide their misbehaviour or to frame well-behaved nodes for misbehaviour. However, our strategy has an upper limit for the number of colluding nodes it can resist that is directly proportional to the number of discrepancies required to accuse a node of misbehaviour. Thus, modifying the number of colluding nodes our protection scheme can handle changes the probability of nodes being accused of misbehaviour too.

6.3 Future Work and Open Issues

In this work we have addressed the specific problem of how to tackle packet forwarding misbehaviour in ubiquitous computing environments. However, there are issues that have been left unaddressed or are susceptible of improvement. Below, we propose some future research directions that we consider interesting to complement and extend work presented in this thesis.

- In Section 3.3.2 and Section 5.3.4 we showed how security services such as authenticity and data integrity can help to prevent the misuse of our proposed protection scheme, increase nodes' trust in the correctness of reported metrics, and provide confidence in the veracity of detection alert and accusation packets. However, although providing our protection scheme with the basic security services (authentication, access control, data confidentiality, data integrity, non-repudiation and service availability) is important to make its deployment possible in real environments, this field is complex and deserves its own piece of research. For this reason we propose it as an interesting area of research to further extend this thesis work.

- Our research efforts have been focused on examining the forwarding behaviour of network nodes. Nevertheless, metrics can be collected to evaluate nodes' behaviour in other areas. For instance, network nodes could maintain and report metrics on the route request packets generated by other nodes so as to identify denial of service (DoS) attacks or on the number of data packets originating at each node to detect greedy behaviour. We consider that identifying other types of misbehaviour through collection of behaviour information is a possible path for future research seeking to enhance the scope of the undertaken work.

- In this thesis policies have been used as a means to provide our protection scheme with adaptability to the dynamic conditions of UCEs. Policies express the high-level goals of a managing entity and are interpreted into low-level policies that dynamically control the operation of our protection scheme. However, there are two important areas with the context of policy based management (PBM) that we have not addressed in this work: policy conflicts and policy refinement. Policy conflicts can occur in our scheme when multiple policies and the network dynamic conditions affect simultaneously the values of the same managed object.
parameters. On the other hand, policy refinement deals with the appropriate translation of high-level policies (e.g. setting the detection rigidness of our approach) and variables (e.g. the probability of wrong accusation) to low-level algorithm configuration (e.g. the value of the misbehaviour threshold $\alpha_{\text{threshold}}$). We propose that these two open issues be addressed through future research.

- Currently it is possible to find in the literature a number of approaches that seek to protect network communication protocols against a wide variety of attacks. In particular, in Chapter 2 of this thesis we presented some strategies aiming at protecting the route discovery functionality of mobile ad hoc routing protocols. We believe that an interesting research work would be to study the suitability of such approaches to be integrated with our protection scheme, which protects the data forwarding functionality of wireless routing protocols, in order to provide a holistic protection solution for wireless multihop routing protocols.
References


153


References


References


References


References


161


References


