Luís Pedro Morais Gonçalves

SISTEMA DE DETEÇÃO DE INTRUSÕES EM AMBIENTES IoT

1 2 9 0 UNIVERSIDADE D COIMBRA

UNIVERSIDADE D
COIMBRA

Luís Pedro Morais Gonçalves

SISTEMA DE DETEÇÃO DE INTRUSÕES EM AMBIENTES IoT

VOLUME 1

Dissertação no âmbito do Mestrado em Segurança Informática, orientada pelo Professor Doutor António Jorge da Costa Granjal e apresentada à Faculdade de Ciências e Tecnologia / Departamento de Engenharia Informática.

julho de 2021

Faculdade de Ciências e Tecnologia

Departamento de Engenharia Informática

Sistema de deteção de intrusões em ambientes IoT

Luís Pedro Morais Gonçalves

Relatório da dissertação no âmbito do Mestrado em Segurança Informática, orientada pelo Professor Doutor António Jorge da Costa Granjal e apresentada à Faculdade de Ciências e Tecnologia / Departamento de Engenharia Informática.

julho de 2021

UNIVERSIDADE D
COIMBRA

Esta página foi propositadamente deixada em branco.

Agradecimentos

À minha família,
Ao meu orientador, Professor Doutor António Jorge da Costa Granjal,
A todos aqueles que me acompanham e estão sempre presentes.

Esta página foi propositadamente deixada em branco.

Resumo

O presente trabalho visa a mitigação da problemática associada à incidência de ataques *Wormhole* em redes constituídas por dispositivos Internet of Things (IoT). Ciente das limitações energéticas e computacionais deste tipo de dispositivos e dado o seu vasto domínio aplicacional, torna-se importante o desenvolvimento de estratégias que promovam a segurança desta tecnologia. Neste seguimento, foi proposto um sistema de deteção de intrusões híbrido, baseado em assinaturas, vocacionado à identificação de ataques *Wormhole* em redes operadas pelo protocolo de encaminhamento Routing Protocol for Low-Power and Lossy Networks (RPL). A estratégia de deteção desenvolvida passa pelo relacionamento da distância percorrida pelos pacotes, entre os nós remetente e destinatário, com a distância tida como referência para esse trajeto. Dado o seu princípio, o modelo pressupõe a prévia determinação da distância entre nós vizinhos, bem como, a inclusão de novos campos nos pacotes em trânsito. A avaliação do Intrusion Detection System (IDS), por recurso a simulação com a ferramenta *Cooja*, mostrou resultados bastante satisfatórios no que concerne à capacidade de deteção de ataques e identificação de nós maliciosos, apresentando ainda um impacto energético perfeitamente admissível dado o domínio em que se aplica.

Palavras-Chave

Internet of Things, Low-Power and Lossy Network, Intrusion Detection System, Ataque Wormhole, Received Signal Strength Indicator, Cooja, TmoteSky, IPv6 over Low power Wireless Personal Area Networks, Routing Protocol for Low-Power and Lossy Networks.

Esta página foi propositadamente deixada em branco.

Abstract

The present work approaches the problem associated with the incidence of Wormhole attacks in Internet of Things (IoT) environments. The energy and computational limitations of this type of devices and the vast application domain where they are used, motivate the development of strategies that promote the safety of this technology. Following this, a hybrid Intrusion Detection System (IDS), based on signatures, aimed at identifying Wormhole attacks in this type of environment was proposed. The detection strategy involves the relationship of the distance travelled by the packets, between the sender and recipient nodes, with the distance taken as a reference for this path. The operationalization of this model presupposes the prior determination of the distance between neighbouring nodes and the inclusion of new fields in the packets in transit. The proposed IDS was evaluated using simulation with the Cooja tool and showed very satisfactory results in terms of the ability to detect attacks and identify malicious nodes. Regarding the energy impact of the solution, it was perfectly acceptable given the domain in which it applies.

Keywords

Internet of Things, Low-Power and Lossy Network, Intrusion Detection System, Wormhole attack, Received Signal Strength Indicator, Cooja, TmoteSky, IPv6 over Low power Wireless Personal Area Networks, Routing Protocol for Low-Power and Lossy Networks.

Esta página foi propositadamente deixada em branco.

Conteúdo

1 Introdução **1**
1.1 Objetivos do trabalho . . . 2
1.2 Planeamento de atividades . . . 2
1.3 Estrutura do documento . . . 4

2 Contextualização Teórica **6**
2.1 Internet of Things (IoT) . . . 6
2.1.1 Domínio aplicacional . . . 6
2.1.2 Requisitos para sistemas IoT . . . 7
2.1.3 Arquiteturas . . . 8
2.2 Low-Power and Lossy Networks (LLN) . . . 9
2.3 Constrained Application Protocol (CoAP) . . . 9
2.4 IEEE 802.15.4 . . . 10
2.5 Internet Protocol Version 6 (IPV6) . . . 11
2.6 IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) . . . 12
2.7 Routing Protocol for Low-Power and Lossy Networks (RPL) . . . 13
2.7.1 Tipos de Nós . . . 14
2.7.2 Função Objetivo . . . 15
2.7.3 Métricas . . . 15
2.7.4 Mensagens protocolares . . . 16
2.7.5 Formação de DODAG . . . 18
2.7.6 Rotas Descendentes . . . 18
2.7.7 Mecanismos de gestão topológica . . . 19
2.7.8 Mecanismos de Segurança . . . 20
2.7.9 Ataques típicos ao protocolo . . . 21
2.8 Intrusion Detection Systems (IDS) . . . 29
2.9 Intrusion Prevention Systems (IPS) . . . 30
2.10 Síntese de capítulo . . . 30

3 Estado da Arte **33**
3.1 Propostas para deteção de ataques sobre o protocolo RPL . . . 33
3.2 Propostas para deteção de ataque Wormhole . . . 38
3.3 Síntese de capítulo . . . 43

4 Sistema de deteção proposto **45**
4.1 Pressupostos técnicos . . . 46
4.2 Arquitetura e funcionamento geral . . . 46
4.2.1 Módulo centralizado . . . 49
4.2.2 Módulos distribuídos . . . 51
4.3 Exemplificação teórica . . . 54
4.4 Fatores de diferenciação face a outras propostas . . . 56

4.5 Síntese de capítulo 57

5 Estratégia para validação do modelo proposto 59
5.1 Cenário de aplicação considerado 59
5.2 Ferramentas utilizadas 61
5.2.1 Sistema operativo Contiki 62
5.2.2 Simulador Cooja 62
5.2.3 Wireshark 63
5.2.4 Contiki Powertrace 63
5.3 Detalhes da simulação 63
5.4 Métricas de avaliação 64
5.5 Síntese de capítulo 68

6 Implementação do modelo 70
6.1 Módulos distribuídos 70
6.1.1 Fase de inicialização 71
6.1.2 Deteção distribuída 73
6.2 Módulo centralizado 77
6.2.1 Fase de inicialização 77
6.2.2 Deteção centralizada 78
6.3 Ataques 81
6.3.1 Ataque Wormhole por retransmissão de pacotes 82
6.3.2 Ataque Wormhole por encapsulamento 82
6.4 Síntese de capítulo 85

7 Análise dos resultados 87
7.1 Cenários de avaliação 87
7.2 Capacidade de deteção de ataques e identificação dos nós maliciosos 89
7.3 Impacto energético da solução 94

8 Conclusões e oportunidades de investigação futura 98

Esta página foi propositadamente deixada em branco.

Acrónimos

6BR 6 LoWPAN Border Router. 33

6LoWPAN IPv6 over Low power Wireless Personal Area Networks. v, vii, 1, 12, 62

ABR Area Border Router. 39

ACK Acknowledgement. 9, 10

AODV Ad-hoc On-demand Distance Vector. 14, 40

BGP Border Gateway Protocol. 14

BLE Bluetooth Low Energy. 10

BR Border router. 39, 41, 43, 46–52, 54–57, 60, 61, 63, 64, 70, 72, 75, 77, 78, 87

CoAP Constrained Application Protocol. 1, 9, 10

CON Confirmable. 9, 10

CPU Central Processing Unit. 40, 66, 95, 96

DAG Directed Acyclic Graph. 14, 20, 23, 31, 39

DAO Destination Advertisement Object. 17–19, 22, 24, 31

DAO-ACK Destination Advertisement Object Acknowledegment. 18

DIO DODAG Information Object. 15, 17–20, 22, 23, 26, 27, 38, 43, 45, 56

DIS DODAG Information Solicitation. 17, 18, 20, 22, 35

DODAG Directed Oriented Acyclic Graph. 14, 15, 17, 19, 20, 23, 24, 26

DoS Denial of Service. 34, 35

DSDV Destination-Sequenced Distance-Vector Routing. 14

DSR Dynamic Source Routing. 14

DYMO Dynamic MANET On-demand. 14

ETX Expected Transmission Count. 15, 16, 26, 34, 38

FPR False Positive Rate. 36

GPS Global Position System. 39, 43, 46, 56

HTTP Hypertext Transfer Protocol. 9

IANA Internet Assigned Numbers Authority. 16

ICMP Internet Control Message Protocol. 16

IDS Intrusion Detection System. v, vii, 2, 3, 21, 29–31, 33–35, 38, 39, 41, 45–49, 51, 52, 56, 57, 59, 63–66, 68, 71, 73, 85, 87, 89, 92–96, 98

IEEE Institute of Electrical and Electronics Engineers. 1, 10, 62, 74

IETF Internet Engineering Task Force. 13, 14

IoT Internet of Things. v, vii, 1, 2, 4, 6, 7, 15, 21, 30, 31, 35, 38, 46, 49, 59, 62, 66, 68, 98

IP Internet Protocol address. 9, 77–79

IPS Intrusion Prevention Systems. 30

IPsec IP Security Protocol. 11, 12

IPV4 Internet Protocol Version 4. 11, 12

IPv6 Internet Protocol Version 6. 11, 12, 49, 71, 73–75, 77, 85

LLN Low-Power and Lossy Network. v, vii, 4, 8, 9, 12–15, 21, 30, 38, 43, 54, 62, 63

LPM Low Power Mode. 66, 95

M2M Machine-to-Machine. 9

MAC Media Access Control. 10, 62, 71

NON Non-Confirmable. 9, 10

OLSR Optimized Link State Routing Protocol. 14

OSPF Open Shortest Path First. 14

pH Potencial Hidrogeniônico. 60

PRR Packet Reception Rate. 15, 16

QoS Quality of Service. 11

RDC Radio Duty Cycling. 62, 66, 71, 74, 81–83, 85

RFC Request for Comments. 9, 11, 12, 21

RIP Routing Information Protocol. 14

ROLL Routing Over Low power and Lossy networks. 13, 14

RPL Routing Protocol for Low-Power and Lossy Networks. v, vii, 1, 2, 4, 6, 14–22, 25, 26, 31, 33–35, 38, 40, 41, 43, 46, 48, 49, 52, 54, 57, 62, 63, 71, 72, 75, 77, 82, 83, 98

RSSI Received Signal Strength Indicator. v, vii, 39, 41, 43, 47, 48, 52, 57, 72

RST Reset. 9, 10

RTT Round Trip Time. 40

SAN Sensing Aware Nodes. 39

SO Sistema Operativo. 62

TCP Transmission Control Protocol. 11, 12, 73

TIK TESLA with Instant Key disclosure. 40

TPR True Positive Rate. 36

UDP User Datagram Protocol. 9, 10

WPAN Wirless Personal Area Network. 10

Esta página foi propositadamente deixada em branco.

Lista de Figuras

1.1 Diagrama de Gantt 1º semestre 3
1.2 Diagrama de Gantt 2º semestre 3

2.1 Domínio aplicacional IoT [4] 7
2.2 Modelo de 3 camadas [47][10] 8
2.3 Modelo de 6 camadas [30] 8
2.4 Estrutura das mensagens CoAP [44] 10
2.5 Estrutura de um pacote IPv6 [26] 11
2.6 Estrutura de um cabeçalho de extensão do tipo Hop-by-Hop [26] 12
2.7 Estrutura de uma mensagem DIO [18] 17
2.8 Estrutura de uma mensagem DAO [18] 17
2.9 Esquema Representativo da criação de um DODAG [18] 18
2.10 Esquema Representativo da propagação de mensagens DAO na determinação das rotas descendentes [31] 19
2.11 Representação esquemática do ataque Wormhole [49] 24
2.12 Exemplo DAO Inconsistency Attacks in Storing Mode, [49] 26

4.1 Lógica para a função de inicialização do módulo centralizado 50
4.2 Lógica para a função de monitorização do módulo centralizado 50
4.3 Lógica para a função de deteção de nós extremidade do módulo centralizado 51
4.4 Lógica para a função de inicialização dos módulos distribuídos 52
4.5 Lógica para a função de monitorização dos módulos distribuídos 53
4.6 Esquematização da arquitetura do sistema de deteção proposto 53
4.7 Exemplo da topologia e das estruturas auxiliares do IDS 54
4.8 Representação de um túnel malicioso entre os nós 2 e 12 da topologia 55

5.1 Exemplo de cenário real de aplicação 60
5.2 Consumo energético modular de dispositivos TmoteSky segundo o fabricante [13] 67
5.3 Output gerado pela ferramenta Powertrace 67

6.1 Conclusão do processo de inicialização do módulo distribuído 72
6.2 Advertência de ataque detetado gerado pelo módulo distribuído 74
6.3 Fluxograma representativo do funcionamento do módulo distribuído 76
6.4 Conclusão do processo de inicialização do módulo centralizado 78
6.5 Identificação de nós maliciosos por parte do módulo centralizado 79
6.6 Fluxograma representativo do funcionamento do módulo centralizado 80
6.7 Mensagem informativa de ativação de ataque no nó 5 81
6.8 Implementação do ataque Wormhole do tipo retransmissão de pacotes . . . 82
6.9 Implementação do ataque Wormhole do tipo por encapsulamento 83
6.10 Fluxograma representativo do funcionamento dos ataques considerados . . . 84

7.1 Cenário de testes constituído por 10 nós 88
7.2 Cenário de testes constituído por 20 nós 88
7.3 Cenário de testes constituído por 30 nós 89
7.4 Comportamento do modelo quando simulado no 1º cenário 90
7.5 Comportamento do modelo quando simulado no 2º cenário 90
7.6 Comportamento do modelo quando simulado no 3º cenário 91
7.7 Consumo energético total durante 30 minutos de simulação 94
7.8 Consumo energético do módulo de CPU durante 30 minutos de simulação . 96

Esta página foi propositadamente deixada em branco.

Lista de Tabelas

2.1 Tabela comparativa LLN e redes IP convencionais, adaptado de [80] 9
2.2 Síntese de ataques ao RPL 28

3.1 Propostas para deteção de ataques sobre o protocolo RPL 37
3.2 Síntese das propostas para deteção de ataque Wormhole 42

7.1 Tabela sumária de avaliação da capacidade de deteção de ataques Wormhole 91
7.2 Tabela sumária de avaliação da capacidade de identificação de nós maliciosos 92

1 Capacidade de identificação de atacantes envolvidos no ataque Wormhole do tipo por encapsulamento 108
2 Capacidade de identificação de atacantes envolvidos no ataque Wormhole do tipo por retransmissão de pacotes 108
3 Capacidade de deteção de ataques Wormhole do tipo por encapsulamento . 109
4 Capacidade de deteção de ataques Wormhole do tipo por retransmissão de pacotes 109
5 Output Powertrace no inicio de simulação para o cenário com 10 nós na presença do IDS 110
6 Output Powertrace no fim da simulação para o cenário com 10 nós na presença do IDS 110
7 Output Powertrace no inicio da simulação para o cenário com 10 nós sem a presença do IDS 111
8 Output Powertrace no fim da simulação para o cenário com 10 nós sem a presença do IDS 111
9 Output Powertrace no inicio da simulação para o cenário com 20 nós na presença do IDS 112
10 Output Powertrace no fim da simulação para o cenário com 20 nós na presença do IDS 113
11 Output Powertrace no inicio da simulação para o cenário com 20 nós na ausência do IDS 114
12 Output Powertrace no fim da simulação para o cenário com 20 nós na ausência do IDS 115
13 Output Powertrace no inicio da simulação para o cenário com 30 nós na presença do IDS 116
14 Output Powertrace no fim da simulação para o cenário com 30 nós na presença do IDS 117
15 Output Powertrace no inicio da simulação para o cenário com 30 nós na ausência do IDS 118
16 Output Powertrace no fim da simulação para o cenário com 30 nós na ausência do IDS 119

Esta página foi propositadamente deixada em branco.

Capítulo 1

Introdução

No decorrer do último século foi notório o empenho da comunidade científica e de companhias tecnológicas no desenvolvimento de aplicações que possibilitassem a informatização de infraestruturas, até então, totalmente analógicas.

O sucesso desta evolução motivou a que, rapidamente, se considerasse a hipóteses de incorporar nas funcionalidades desses novos sistemas a capacidade de estes procederem à monitorização inteligente de eventos processuais e quotidianos, através de *hardware* específico para esta função.

Esta abordagem, desde logo, despoletou a necessidade de capacitar os mais simples dispositivos a intercomunicarem entre si, por meio da Internet. Desta prática emergiu o termo Internet of Things (IoT), remetendo este para todos os dispositivos altamente restritos e que se apresentem ligados à Internet, sendo esta o meio que possibilita a interconexão dos vários componentes e coisas num sistema.

Com a rápida evolução tecnológica, somos hoje completamente dependentes deste novo conceito, estando este presente na indústria, na saúde e até na gestão de casas inteligentes.

Acontece que os próprios construtores deste tipo de *hardware*, numa tentativa de acompanhar esta tendência de mercado, começaram a desenvolver equipamentos mais pequenos, simples e com especial foco na usabilidade, eficiência e competitividade. Neste seguimento, é legítimo assumir que grande parte dos dispositivos IoT de que agora dispomos são suscetíveis a uma série de limitações, sejam elas computacionais, energéticas e/ou de memória.

De modo a respeitar os pressupostos do IoT e na necessidade de incluir este tipo de dispositivos no domínio da Internet, foi crucial proceder à alteração dos paradigmas desta imensa rede, o que se refletiu numa adaptação da pilha protocolar que a sustenta. Neste particular, foram, inclusive, propostas novas arquiteturas e protocolos, dotados de uma maior capacidade de abstração, de modo a abarcar a heterogeneidade dos dispositivos que agora os utilizam. Os modelos de 3 e 6 camadas, os protocolos Routing Protocol for Low-Power and Lossy Networks (RPL), Constrained Application Protocol (CoAP) e os *standards* IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) e Institute of Electrical and Electronics Engineers (IEEE) 801.15.4, são exemplos disso mesmo.

Dada esta rápida evolução, não é de admirar que a segurança tivesse sido transposta para um segundo plano, em prol, muitas das vezes, da usabilidade e fruto das restrições tecnológicas impostas. No entanto, o vasto domínio aplicacional dos dispositivos IoT obriga a que a pilha protocolar, aliada a mecanismos suplementares, providenciem as medidas adequadas à salvaguarda da integridade, confidencialidade, autenticidade e disponibilidade

dos dados e do próprio sistema. Não podemos ignorar que a inclusão de mais componentes num sistema, neste caso de dispositivos IoT, contribui para um aumento considerável da superfície de ataque, pelo que, é necessário o desenvolvimento de soluções eficientes, capazes de suprimir as novas ameaças, devendo estas contemplar todas as especificidades dos novos dispositivos e da pilha protocolar que os sustenta.

Na impossibilidade de cobrir todo o ambiente envolvente à tecnologia IoT, o presente documento focar-se-á no RPL, sendo este o protocolo de encaminhamento com maior relevo neste domínio. Serão abordados os ataques aos quais este está sujeito, as soluções existentes para a sua supressão, finalizando com a proposta de uma nova estratégia de deteção de ataques *Wormhole*, por intermédio de um Intrusion Detection System (IDS) desenhado especificamente para tal. Este tipo de mecanismo possibilita, através da monitorização ativa da topologia e/ou das comunicações nela existentes, a deteção de desvios ao comportamento dito normal ou o relacionamento da atividade comunicacional, interna à rede, com assinaturas de ataque pré-definidas, permitindo, em ambos os casos, a deteção dos ataques.

1.1 Objetivos do trabalho

Durante o desenvolvimento do presente trabalho espera-se atingir os seguintes objetivos:

- Clarificação dos principais conceitos associados à tecnologia IoT;
- Compreensão do protocolo de encaminhamento RPL;
- Levantamento dos principais ataques sobre o protocolo RPL;
- Sintetização das propostas existentes para deteção e supressão dos ataques referenciados;
- O desenvolvimento de um novo sistema de deteção de ataques *Wormhole* em redes operadas pelo protocolo RPL;
- Validação do sistema proposto em ambiente de simulação;
- Avaliação do modelo em termos de eficiência energética e de deteção;
- Produção de um artigo científico relativo ao sistema de deteção proposto e conclusões resultantes da sua avaliação.

1.2 Planeamento de atividades

De modo a facilitar a realização deste trabalho, procedeu-se ao planeamento detalhado das atividades a desenvolver durante a sua execução.

A primeira metade do trabalho, compreendida temporalmente entre início de Outubro de 2020 e 18 de Janeiro de 2021, data da defesa intermédia, contemplou o estudo e documentação do funcionamento de toda a pilha protocolar que sustenta a tecnologia IoT, com especial destaque para o protocolo RPL. Foram, também, estudados os ataques mais pronunciados sobre este protocolo, bem como, as técnicas existentes na literatura para a sua supressão. Findada esta tarefa, foi desenvolvido um modelo para deteção de ataques Wormhole, sendo, posteriormente, delineada a estratégia de implementação e validação,

assim como, definidas as métricas a utilizar na sua avaliação. Este conjunto de tarefas encontra-se sintetizado no diagrama de Gantt apresentado na figura 1.1, sendo neste incluindo o progresso semanal de cada uma das atividades.

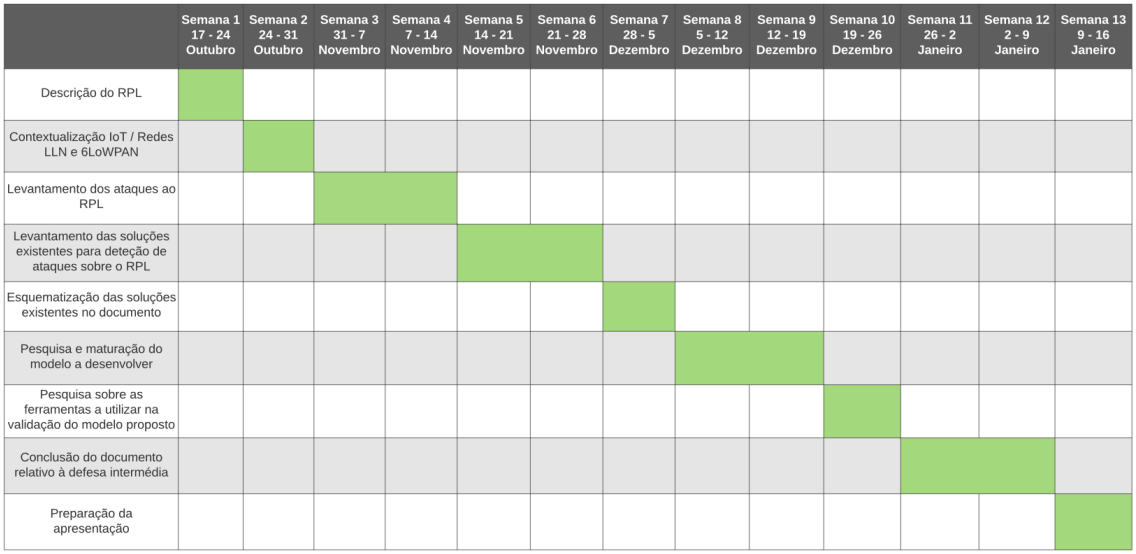

Figura 1.1: Diagrama de Gantt 1º semestre

A segunda metade do trabalho, realizado entre Janeiro e Junho de 2021, incidiu na implementação do modelo proposto e sua avaliação. Adicionalmente, procedeu-se ainda à produção de um artigo científico relativo ao funcionamento e performance do IDS desenvolvido. O planeamento prévio das atividades decorrentes destas tarefas encontra-se esquematizado em 1.2.

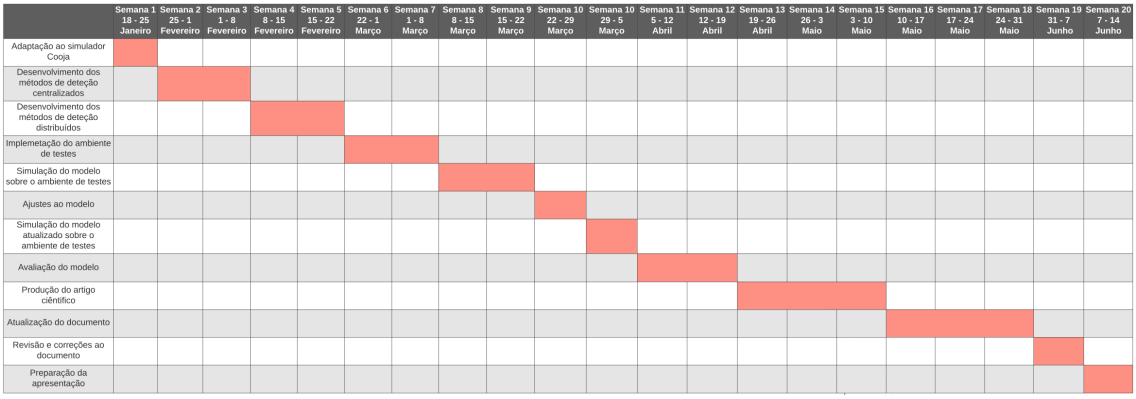

Figura 1.2: Diagrama de Gantt 2º semestre

1.3 Estrutura do documento

O presente documento encontra-se organizado em 7 secções:

Na primeira secção, contextualização, são clarificados os principais conceitos associados ao IoT e descrita a forma como esta tecnologia é operacionalizada. Além de explicados os principais protocolos envolvidos neste processo, são também abordadas as especificidades de redes Low-Power and Lossy Network (LLN). No seguimento, é dada especial relevância ao protocolo RPL, tendo este sido abordado com mais rigor e, portanto, aqui incluídos os principais ataques aos quais está sujeito e a forma como estes lhe são direcionados. Findando este capítulo, espera-se que o leitor esteja familiarizado com o tema e apto a compreender todas as propostas apresentadas nos pontos posteriores.

No segmento subsequente, a que respeita o estado de arte, são abordadas as principais técnicas para a deteção de ataques sobre o protocolo RPL, explicados em detalhe na secção anterior. Embora tenham sido referenciadas soluções associadas a múltiplas tipologias de ataque, foi dado ênfase às propostas que visam a identificação e supressão de ataques *Wormhole*.

Na subdivisão Modelo, é tratado em detalhe o funcionamento e arquitetura da nova abordagem para deteção de ataques *Wormhole*, bem como, as suas particularidades face a outras propostas já existentes. Neste seguimento, na secção posterior (estratégia de validação do modelo), documentar-se-á a forma como o modelo foi implementado e validado, sendo, portanto, referidas e explicadas as ferramentas utilizadas e as métricas de avaliação consideradas.

No 6º capítulo, serão interpretados e comentados os resultados provenientes da simulação. A avaliação aqui realizada debruçou-se no estudo da solidez do modelo na atividade de deteção de ataques, na capacidade de identificação de nós maliciosos e determinação do impacto energético anexo à solução.

Relativamente ao último capítulo, conclusões, são aqui apresentadas as principais ideias retiradas de todo o trabalho realizado, realçando a performance do novo sistema de deteção proposto, bem como, referenciadas questões emergentes que suscitem um potencial trabalho de investigação futuro.

Esta página foi propositadamente deixada em branco.

Capítulo 2

Contextualização Teórica

De modo a contextualizar o leitor, nesta secção será clarificado o conceito IoT, detalhadas as especificidades dos dispositivos passiveis desta classificação e das redes nas quais estes são integrados. Além disso, será descrita a tecnologia envolvente, que lhe serve de suporte, e a pilha protocolar que permite a sua operacionalização. Relativamente a este último ponto, é dado especial destaque ao protocolo RPL, o qual irá ser descrito em detalhe e cujas principais ameaças serão também elas abordadas.

2.1 Internet of Things (IoT)

A expressão IoT foi proposta em 1999 por Kevin Ashton durante uma apresentação na empresa Procter & Gamble [62] [30]. Dai em diante, este termo foi presença assídua em revistas e jornais científicos, ainda que, à medida que o tempo foi passando, o seu significado tenha vindo a tornar-se mais complexo e abrangente.

IoT consiste na interligação de dispositivos, bem como, na conexão dos mesmos com a Internet, com o intuito de coletar informações relativas ao meio em que se inserem [9]. Nestes dispositivos estão inclusos desde simples objetos do quotidiano, a sensores e atuadores utilizados na indústria.

Estima-se que em 2030 existam cerca de 125 biliões destes dispositivos em utilização [48], contudo, o abarcamento da pilha protocolar da Internet em "objetos" tão triviais e restritos levanta uma série de constrangimentos, alguns deles já suprimidos outros alvo ainda de investigação por parte da comunidade cientifica, na tentativa da sua superação.

2.1.1 Domínio aplicacional

A versatilidade dos sistemas IoT motiva a que, atualmente, na literatura [4], sejam considerados 5 grandes domínios aplicacionais, estando estes esquematizados na figura 2.1.

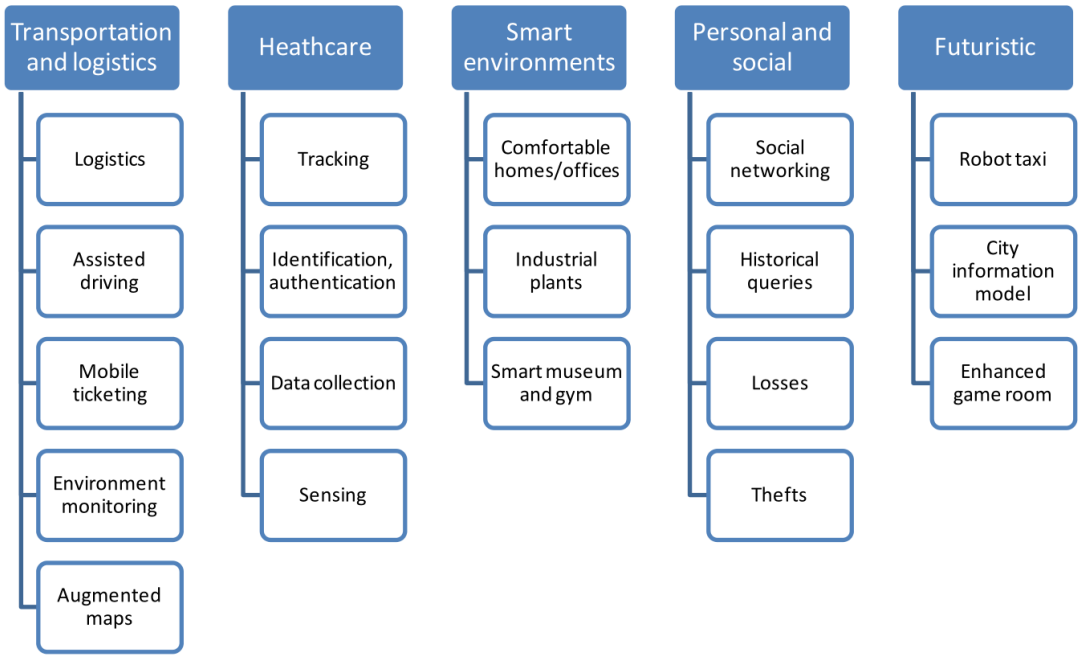

Figura 2.1: Domínio aplicacional IoT [4]

Neste vasto leque aplicacional, existem sistemas com requisitos mais aprimorados do que outros, fruto da sua empregabilidade em cenários de criticidade variável. Inerentemente, cada um destes é constituído por dispositivos com diferentes especificações, o que motiva a que as restrições associadas a estes elementos sejam, também elas, variáveis.

2.1.2 Requisitos para sistemas IoT

A dimensão e complexidade dos sistemas IoT atuais, realça a necessidade de aptidão destes a lidar com a heterogeneidade dos dispositivos que os compõem. A dinâmica acelerada destes sistemas, aliada aos problemas típicos de disponibilidade na transmissão de dados entre os vários elementos, suscita necessidades absolutas de escalabilidade, auto-organização reativa e monitorização constante [62].

Não menos importante, é garantir a interoperabilidade entre dispositivos e aplicações, seja através da atribuição de identificadores únicos a cada componente do sistema, ou mediante a utilização de formatos padronizados de mensagem.

Relativamente aos dados que transitam na rede, é absolutamente fundamental garantir a sua segurança, nomeadamente, a sua integridade, autenticidade e confidencialidade [51].

Todos os requisitos propostos devem, individual e coletivamente, assentar em soluções cuja otimização computacional e energética seja uma prioridade, contribuindo o conjunto para o desenvolvimento de sistemas suficientemente eficientes capazes de ultrapassar as limitações do hardware.

2.1.3 Arquiteturas

Os requisitos anteriores são, em parte, validados por arquiteturas que imprimem um nível de abstração tal que permite a operacionalização desta tecnologia. Dentro das várias arquiteturas propostas na literatura, destaca-se modelo de 3 camadas [47] [10], representado na figura 2.2.

Figura 2.2: Modelo de 3 camadas [47][10]

Neste modelo, a camada aplicacional promove o controlo dos dispositivos físicos por parte do utilizador final, sendo igualmente responsável pela interpretação e o processamento dos dados provenientes do meio. A camada de Rede, por sua vez, trata da transmissão dos dados desde os sensores e atuadores, contidos na camada perceção, até à camada aplicação. Por fim, a camada perceção diz respeito aos sensores e atuadores que atuam diretamente sobre o meio físico, constituindo uma topologia de rede amplamente conhecida como LLN, dadas as características dos dispositivos que as compõem.

De uma perspetiva mais tecnológica, são de resto propostos na literatura alguns modelos nos quais, para cada uma das camadas, existe uma clara correspondência com um protocolo, como é o caso do modelo de 6 camadas, proposto em [30] e [20], representado na figura 2.3 e cujos protocolos e standards nele referenciados serão apresentados no decorrer desta secção do documento.

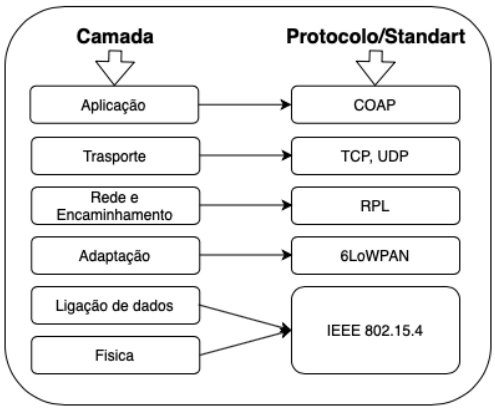

Figura 2.3: Modelo de 6 camadas [30]

2.2 Low-Power and Lossy Networks (LLN)

As LLN são redes compostas por dispositivos inteligentes, altamente restritos computacional e energicamente, interligados por meio de ligações de baixa velocidade [80]. Se tipicamente as redes convencionais são constituídas por *routers* alimentados por fontes de energia estáveis, nas LLN os dispositivos são sensores e atuadores alimentados por bateria, tornando frequente a indisponibilidade de um nó devido à falta de energia. Relativamente à memória dos dispositivos, nas redes Internet Protocol address (IP) comuns este recurso é suficiente para alocar tabelas de encaminhamento consideravelmente extensas, a par das redes LLN onde a memória é escassa e, portanto, é necessário otimizar ao máximo estas estruturas por forma a serem suportadas pelas unidades de memória dos dispositivos. No que diz respeito às ligações entre sensores, também estas são bastantes menos eficientes em redes LLN, isto porque, além de não suportarem taxas de transmissão tão elevadas, são muito mais vulneráveis a interferências provenientes do meio onde se inserem. Todas estas limitações obrigam a que a operacionalização de redes LLN seja projetada privilegiando a otimização e flexibilidade de toda a tecnologia implicada. A tabela 2.1 sumariza as principais diferenças entre estas duas tipologias de rede.

Redes IP convencionais	**Redes LLN**
Os nós são *routers*	Os nós são sensores, actuadores ou *routers*
Constituídas por centenas de dispositivos	Podem ser constituídas por milhares de dispositivos.
As ligações entre os nós são estáveis	As ligações entre os nós são instáveis sendo frequente a inoperacionalidade dos componentes.
As restrições nos nós e ligações não são um problema	Os componentes destas redes são altamente restritos

Tabela 2.1: Tabela comparativa LLN e redes IP convencionais, adaptado de [80]

2.3 Constrained Application Protocol (CoAP)

O CoAP é um protocolo aplicacional, definido no Request for Comments (RFC) 7252 [70], que permite a transação de informações entre dispositivos, contribuindo, inerentemente, para a comunicação Machine-to-Machine (M2M). Para tal, é empregue um modelo pedido/resposta sobre User Datagram Protocol (UDP), podendo as transações ser efetuadas de forma assíncrona. Na prática, trata-se de uma simplificação do protocolo Hypertext Transfer Protocol (HTTP), compreendendo os métodos *POST*, *GET*, *PUT* e *DELETE* e quatro diferentes tipos de mensagens: Confirmable (CON), Non-Confirmable (NON), Acknowledgement (ACK) e Reset (RST).

Aquando da realização de um pedido, esta operação pode ser realizada, a nível do cliente, por intermédio quer de uma mensagem CON, quer de uma mensagem NON [60]. No primeiro caso, quando o servidor recebe o pedido, se estiverem reunidas as condições para responder ao mesmo, veicula a resposta na própria mensagem de confirmação da receção do pedido (ACK). Por outro lado, se o tratamento do pedido não puder ser efetuado de imediato, é enviada ao cliente uma mensagem ACK em branco e, após realizada a ação necessária à determinação da resposta, esta é veiculada ao cliente por recurso a uma outra mensagem CON. Já em situações onde haja uma indisponibilidade permanente de resposta a um pedido, o servidor reage com uma mensagem RST. Relativamente aos pedidos direcionados ao servidor por intermédio de mensagens NON estes são também respondidos através de uma mensagem deste tipo, ou, através de uma mensagem RST, no caso de não ser possível providenciar uma resposta àquele pedido. No entanto, este

último método apenas é utilizado em situações onde as informações transacionadas não sejam críticas, uma vez que, contrariamente ao que acontece com as mensagens CON, esta estratégia não veicula garantias de que os pedidos e respostas tenham sido rececionados pelo servidor e cliente, respetivamente.

A estrutura dos pacotes UDP é transversal a todas as tipologias de mensagem definidas na especificação do CoAP [70]. Estes são constituídos por um cabeçalho com 4 bytes de extensão ao qual se seguem campos de dimensão variável, como é o caso do *Token*, podendo este variar entre 0 e 8 bytes, e dos campos *Options* e *Payload*, como clarificado na figura 2.4.

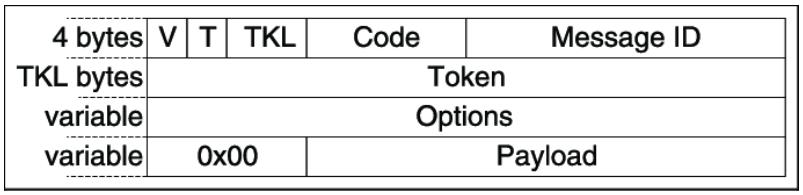

Figura 2.4: Estrutura das mensagens CoAP [44]

- **V -** Representa, em formato binário, a versão CoAP em utilização;
- **T -** Identifica o tipo de mensagem transacionada no presente pacote, (0) para CON, (1) para NON, (2) para ACK e (3) para RST;
- **TKL -** Apresenta o tamanho do *Token*, podendo este variar entre 0 e 8 bytes;
- **Code -** Identifica o cariz da mensagem, podendo esta tratar-se de um pedido (0), resposta (2), ou mesmo, de um *report* alusivo a um erro, seja da parte do servidor (5), seja do cliente (4).
- **Message ID -** Promove a inexistência de mensagens duplicadas na rede, sendo que, paralelamente, permite aferir a correspondência entre CON/NON e ACK/RST, respetivamente.

2.4 IEEE 802.15.4

O *standard* IEEE 802.15.4 [22] visa a padronização das camadas física e de ligação, também designada por Media Access Control (MAC), em redes Wirless Personal Area Network (WPAN). Este tipo de rede é composta por dispositivos com fortes limitações computacionais (poucos recursos de memória e processamento), energéticas (tipicamente alimentados por bateria) e comunicacionais (a baixa potência do sinal limita o raio de alcance das comunicações). Entre outras características, o IEEE 802.15.4 permite que a nível da camada física a rede seja reativa às condições do meio.

Note-se que o IEEE 802.15.4 não é o único standard direcionado a operar a nível da camada física e de ligação da pilha protocolar. Uma possível alternativa é o Bluetooth Low Energy (BLE), um standard que promove a comunicação de dispositivos de baixa potência (2.4 Ghz) e de curto alcance (até 100 metros), com especial foco na eficiência comunicacional, energética e operacional [50]. O BLE recorre a dispositivos *Bluetooth* menos complexos, mais baratos e eficientes que os dispositivos *Bluetooth* convencionais.

2.5 Internet Protocol Version 6 (IPV6)

O Internet Protocol Version 6 (IPv6) definido no RFC 2460 [14] é, no contexto da Internet, responsável não só pela atribuição de um endereço único a cada elemento integrante da rede como, paralelamente, por fazer incluir este identificador nos pacotes que contemplam os dados em trânsito. Este protocolo sucede ao Internet Protocol Version 4 (IPV4) [59] na tentativa de suprimir as limitações deste último que se prendem com a restrita gama de endereços disponibilizada (4.2 biliões), uma vez que, o IPV4 apenas recorre a 32 bits para endereçamento, a par do IPv6 que inclui 128 bits para a mesma função. Além disso, o IPv6 promove um encaminhamento e processamento dos pacotes mais eficiente, providencia um mecanismo de configuração topológica mais simples, oferecendo ainda suporte a uma série de serviços adicionais, como por exemplo, IP Security Protocol (IPsec) ou Quality of Service (QoS). A imagem que se segue esquematiza um pacote IPv6.

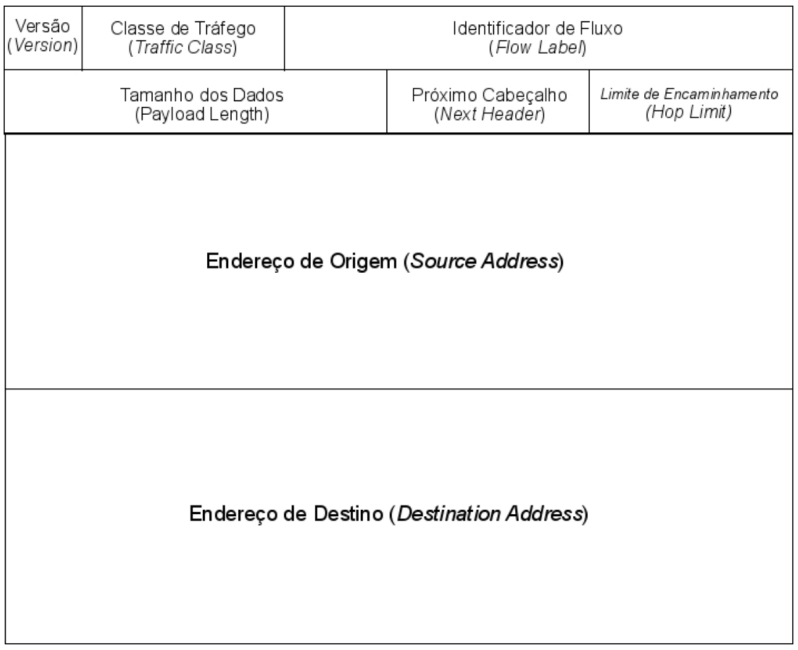

Figura 2.5: Estrutura de um pacote IPv6 [26]

É inevitável não reparar que este pacote não inclui nenhum campo dedicado a opções, no entanto, por forma a prover o protocolo de uma maior flexibilidade, mantendo o tamanho dos seus pacotes constate, as opções são incluídas através de cabeçalhos de extensão, podendo estes ser de 6 diferentes tipos, dependendo da sua funcionalidade:

- *Hop-by-Hop Options*- As informações integrantes deste tipo de cabeçalho são processadas por todos os nós intermediários no trajeto entre remetente e destinatário. Relativamente à sua estrutura, apresentada na imagem abaixo 2.6, o campo "*Próximo Cabeçalho*" remete para o elemento que posteriormente deverá ser processado, podendo este ser um outro cabeçalho de extensão, deste ou de outro tipo, ou o próprio cabeçalho Transmission Control Protocol (TCP). O campo "*tamanho do cabeçalho de extensão*" caracteriza a estrutura na medida em que corresponde à sua dimensão em unidades de 8 bytes. Por fim, no campo "*opções*", os dois primeiros bytes são alusivos à forma como o processamento do cabeçalho deve ser efetuado, mediante condições especificas, já os posteriores incluem as informações opcionais propriamente ditas.

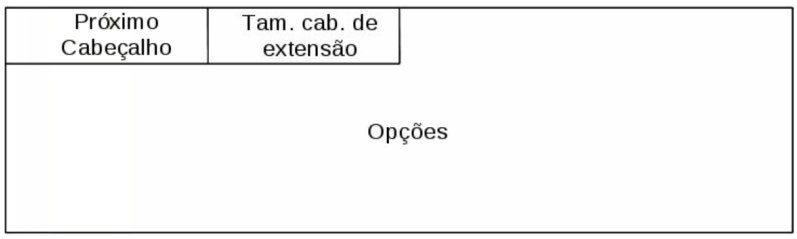

Figura 2.6: Estrutura de um cabeçalho de extensão do tipo Hop-by-Hop [26]

- *Destination Options*- Com formato semelhante ao anterior, difere do mesmo no sentido em que o campo "Opções", neste tipo de cabeçalho, apenas é tratado pelo destinatário do pacote.
- *Fragmentation*- É utilizado sempre que tamanho do pacote IPv6 excede um limite máximo. As informações anexas a este cabeçalho permitem que o destinatário identifique a posição do fragmento contido no pacote relativamente ao bloco de dados original.
- *Authentication Header e Encapsulating Security Payload*- Diretamente associados ao mecanismo de segurança IPsec, permitem que este seja utilizado de uma forma mais simples e organizada comparativamente com a mesma aplicação operada sobre IPV4.
- *Routing* - O cabeçalho de extensão do tipo *Routing* permite listar todos os nós que devem, obrigatoriamente, intermediar a entrega de um pacote.

É importante frisar que a presença do campo *Próximo Cabeçalho* é transversal às seis tipologias de cabeçalho de extensão acima mencionadas. Esta particularidade resulta do facto de um único pacote IPv6 poder incluir múltiplas extensões. Num exemplo simples de aplicação, assuma-se um pacote IPv6 constituído por dois cabeçalhos de extensão, sendo um deles do tipo *Hop-by-Hop* e o segundo do tipo *Destination Options*. Nesta situação, o cabeçalho IPv6 no campo *Próximo Cabeçalho*, definido a "00", irá mapear a primeira das duas estruturas a ser processada, neste caso, o cabeçalho de extensão *Hop-by-Hop*. Por sua vez, também este ultimo irá, seguidamente, após concluídas as atividades no seu domínio, remeter para o processamento do segundo cabeçalho de extensão existente, desta feita, com a atribuição do valor "60" ao seu respetivo campo *Próximo Cabeçalho*. Na inexistência de mais cabeçalhos de extensão, o campo *Próximo Cabeçalho*, relativo à estrutura *Destination Options* remeterá para o processamento do cabeçalho TCP.

2.6 IPv6 over Low power Wireless Personal Area Networks (6LoWPAN)

Definido no RFC 4944 [53], o *standard* 6LoWPAN propõe a utilização de pacotes IPv6 no transporte de dados sobre o IEEE 802.15.4. Este nível de abstração permite que as redes LLN se conectem à Internet, não isentando, contudo, da existência de um *Border Router* na fronteira dos dois domínios.

O principal objetivo do 6LoWPAN passa por minimizar o esforço computacional e comunicacional despendido no processamento e transmissão de pacotes IPv6. Para tal, este *standard* promove a fragmentação e compressão destes elementos, assumindo que grande

parte das informações integrantes da sua estrutura original podem ser apuradas noutras camadas da pilha protocolar.

2.7 Routing Protocol for Low-Power and Lossy Networks (RPL)

A importância de um protocolo de encaminhamento é transversal a todas as tipologias de rede, sendo este responsável pela escolha e constante otimização dos caminhos para a distribuição dos pacotes que nela fluem. O protocolo deve, igualmente, ser dotado de mecanismos que promovam a sua pronta reatividade, nomeadamente, com a determinação de rotas paralelas para o mesmo destino, de modo a promover um balanceamento eficiente da carga na rede e suprimir constrangimentos em caso de falha, ainda que o sucesso desta operação esteja dependente de tempos de convergência baixos. [80]

Em redes LLN a fragilidade dos dispositivos, que tipicamente as constituem, fomenta a constante ocorrência de erros aos quais as limitações computacionais e das ligações impedem uma ampla reação. Repare-se que responder a todos os incidentes originaria um fluxo de mensagens de controlo inaceitável, podendo mesmo colapsar toda a estrutura. Estas limitações apresentam-se, paralelamente, como uma barreira à inclusão de redundância nestas topologias.

Neste seguimento, com o objetivo de padronizar a utilização de um protocolo de encaminhamento em ambientes LLN, o grupo de trabalho Routing Over Low power and Lossy networks (ROLL) da Internet Engineering Task Force (IETF) definiu uma série de requisitos que cobrissem todas as necessidades e especificidades destas redes. A própria definição destes requisitos foi um desafio, isto porque, o multi-domínio aplicacional onde as redes LLN são empregues motiva a que nem todos os requisitos sejam de implementação obrigatória e, assim, emerge de imediato a necessidade de que o protocolo a utilizar seja dotado de uma grande modularidade, conferindo-lhe a flexibilidade que se lhe impõe. Resultaram deste trabalho os seguintes requisitos [46] [80].

- Suporte de tráfego - *Unicast*, *Multicast* e *anycast*
- Encaminhamento adaptável - o protocolo deve adaptar-se dinâmica e automaticamente às condições da rede, bem como, otimizar, sistematicamente, as rotas já existentes.
- Encaminhamento baseado em restrição - devem ser tidas em consideração as características individuais de cada nó, podendo estas condicionar a sua inclusão numa determinada rota.
- Características do tráfego - o protocolo deve assegurar a fluidez de tráfego ponto-a-ponto, multiponto-a-ponto e ponto-a-multiponto.
- Escalabilidade - O protocolo deve ser dotado dos mecanismos necessários para assegurar resposta em cenários de rede compostas por centenas de dispositivos heterogéneos.
- Configuração e gestão- A configuração topológica quer inicial quer, posteriormente, em funcionamento deve ser simples e intuitiva, não devendo existir a necessidade de proceder a qualquer alteração em situações de inclusão ou exclusão de um nó. Adicionalmente, o protocolo deve ser dotado da capacidade de isolar um nó, ou um conjunto destes, sempre que apresentem um comportamento erróneo.

- Atributo nó - É determinante a capacidade de avaliação da condição individual dos nós a fim de apurar a sua disponibilidade no encaminhamento de um pacote.

- Desempenho - O tempo de convergência é, neste contexto, o indicador de desempenho mais relevante, representado o tempo decorrente entre a deteção de uma falha num dos caminhos, a determinação de uma rota alternativa e o redireccionamento do tráfego por intermédio desta. No entanto, à semelhança de outras, esta métrica deve ser flexível, compatibilizando-se com os diferentes requisitos e características das redes onde o protocolo possa ser implementado.

- Segurança - O protocolo deve providenciar mecanismos capazes de assegurar a confidencialidade, integridade e autenticidade dos dados. Em mais detalhe na secção 2.7.8.

Se numa primeira fase o grupo de trabalho procurou a adoção e padronização de um protocolo de encaminhamento amplamente estabelecido e utilizado nas redes ditas convencionais, como por exemplo o Optimized Link State Routing Protocol (OLSR), o Routing Information Protocol (RIP), o Ad-hoc On-demand Distance Vector (AODV), o Destination-Sequenced Distance-Vector Routing (DSDV), o Dynamic MANET On-demand (DYMO), o Dynamic Source Routing (DSR), o Open Shortest Path First (OSPF), ou o Border Gateway Protocol (BGP), [80][43], rapidamente esta ideia foi abandonada, isto porque, nenhum dos protocolos em causa estava capacitado a obedecer a todos os requisitos que se impõem no domínio das LLN.

Posteriormente, a ROLL da IETF propôs o RPL [2], um protocolo baseado no conceito tecnológico de grafos acíclicos direcionados (Directed Acyclic Graph (DAG)), propositadamente desenhado para atender a todas as especificidades das redes LLN. A maior diferença deste protocolo face a outros, baseados em tipologias em árvore, passa pelo facto de ser possível que um nó se associe a múltiplos outros.

Este protocolo é direcionado ao destino, implicando a existência, na topologia, de um nó que assume o papel de raiz, sendo o responsável pela coleta dos dados originários noutros nós, hierarquicamente inferiores. Assim, a estrutura de encaminhamento do RPL recebe o nome de Directed Oriented Acyclic Graph (DODAG), sendo que, a rede pode ser constituída por mais do que um destes (DODAG), que unidos constituem uma instância RPL, identificável pelo *RPLInstanceID*.

Por sua vez, um nó pode estar associado a várias instâncias RPL, adoptando um comportamento diferente em cada uma delas. Esta particularidade, se por uma lado confere uma maior flexibilidade ao protocolo, promove um balanceamento de carga mais eficiente, contribuindo, inerentemente, para o aumento da disponibilidade do sistema [18]. No entanto, é importante frisar que um nó apenas pode estar associado a um DODAG.

A distribuição dos nós pela topologia carece da classificação individual dos mesmos por intermédio da função objetivo. Esta classificação é igualmente determinante na distinção de país, filhos e irmãos dentro da rede e, consequentemente, na deteção de *loops*.

2.7.1 Tipos de Nós

A especificação do RPL [2] define que numa rede operada por este protocolo existem três tipos de dispositivos.

Low Power and Lossy Border Routers

Este nó é, tipicamente, o nó raiz, funcionando como agregador de dados provenientes de todos os outros nós hierarquicamente inferiores, estando assim habilitado à criação de DODAG. A sua posição fronteiriça acresce-lhe a função de *Gateway* entre a rede LLN e a *Internet*.

Low Power and Lossy Routers

Podem gerar e encaminhar dados. Contudo, não são dotados da capacidade de criação de DODAGs, embora estejam aptos à transmissão de informações por intermédio de mensagens DODAG Information Object (DIO) 2.7.4.

Host

Gera tráfego, contudo não é dotado da capacidade de encaminhamento de pacotes, nem, tão pouco, da transmissão de mensagens DIO.

2.7.2 Função Objetivo

Esta função é responsável pelo relacionamento de métricas e restrições do qual resulta a atribuição de um *rank* a cada um dos nós contidos na rede. Por outro lado, é da sua responsabilidade a especificação dos critérios a ter em consideração na escolha de um pai, ou mesmo, na forma como as rotas são escolhidas e otimizadas para a distribuição dos pacotes [18].

Esta componente, assim como as métricas a si associadas, contribuem, em larga escala, para a modularidade e flexibilidade do protocolo. Repare-se que os nós podem adotar comportamentos distintos nas várias instâncias RPL em que se incluem, utilizando, em cada uma delas, uma função objetivo distinta. Paralelamente, esta abordagem permite que o protocolo em foco esteja capacitado a operar nas mais diversas áreas aplicacionais da IoT, abordadas em 2.1.1.

2.7.3 Métricas

As métricas utilizadas pela função objetivo não se encontram padronizadas na documentação do RPL [2], no entanto, existem duas métricas amplamente estabelecidas para este tipo de rede e protocolo, ainda que esta área seja, atualmente, alvo de grande escrutínio por parte da comunidade científica [75].

Expected Transmission Count (ETX)

A métrica ETX tem por objetivo a seleção do caminho que envolva um menor número de retransmissões na troca de dados entre dois nós. Assim, é imediatamente percetível que este indicador é fortemente sensível a ligações de má qualidade. Desta feita, a métrica procede a uma avaliação bilateral, calculando o Packet Reception Rate (PRR) no nó emissor e

recetor para uma janela de tamanho p, como demonstra a expressão 2.1:

$$PRR(p) = \frac{\text{Número de pacotes recebidos}}{\text{Número de pacotes enviados}} \tag{2.1}$$

Através do relacionamento do PRR das duas entidades é possível o cálculo final do ETX, através da expressão 2.2:

$$ETX = \frac{1}{\text{PRR(nó emissor} * \text{PRR(nó receptor)}} \tag{2.2}$$

Em operação, um nó que pretenda selecionar o melhor caminho através desta métrica, procede ao cálculo e avaliação do ETX de todos os seus vizinhos, optando por aquele cujo valor calculado mais se aproxime de 1, correspondendo este ao caso ótimo, onde não é necessário proceder a qualquer retransmissão de pacotes a fim de efetivar o envio de uma mensagem. Esta métrica é, naturalmente, válida na escolha de um nó pai, devendo, ainda assim, os nós candidatos a tal possuir um *rank* inferior ao de seus filhos.[19]

Node energy consumption

Esta métrica implica que um nó atente à situação energética dos seus vizinhos antes de os nomear como candidatos a seu pai. Para tal, os dispositivos são classificados de acordo com o tipo de alimentação energética que possuem: ligado, sempre que o dispositivo tenha uma fonte de alimentação energética contínua, ou com bateria, sempre que o dispositivo seja alimentado por uma fonte energética limitada. Com base na classificação anterior, esta métrica faz uma estimativa da percentagem de energia atual do dispositivo em análise. No caso de o dispositivo estar ligado a uma fonte de alimentação energética contínua, esta percentagem não é calculada, correspondendo sempre ao valor máximo (100%). Já no segundo caso, se o dispositivo for alimentado por bateria, é estimada a atual carga energética da mesma, através da seguinte expressão 2.3:

$$\text{Energia estimada} = \frac{\text{Energia actual do dispositivo}}{\text{Energia total do dispositivo}} * 100 \tag{2.3}$$

Numa situação em que os nós candidatos estejam todos ligados a fontes energéticas contínuas, a presente métrica não acrescenta qualquer valor, no entanto, caso estes sejam alimentados por bateria, ou, num misto de ambas as situações, pressupondo a existência de nós alimentados por bateria e de outros ligados a fontes energéticas contínuas, a presente métrica permite optar por aqueles que, à partida, irão estar operacionais durante um maior período de tempo.

2.7.4 Mensagens protocolares

As mensagens do controlo utilizadas pelo RPL são, na prática, mensagens com cabeçalho Internet Control Message Protocol (ICMP)v6 com o tipo definido pela Internet Assigned Numbers Authority (IANA) a 155 [31][65][21], sendo estas utilizadas nas várias atividades de manutenção protocolar.

DIO

As mensagens DIO são a principal fonte de informação topológica para os nós que compõem a estrutura e para os demais que nela pretendam ingressar. Destas constam os parâmetros necessários à determinação do *rank* do destinatário, incluindo-se aqui o *rank* do nó remetente, as métricas, restrições e função objetivo em vigor, ou ainda, a instância RPL à qual dizem respeito, bem como, o identificador do nó raiz da presente DODAG. Estas mensagens são enviadas pelo nó raiz aquando da criação de um novo DODAG, mas também sempre que exista uma alteração topológica que assim o justifique. Por exemplo, numa situação em que um nó altere o seu *rank* este deve enviar uma mensagem DIO a todos os seus vizinhos a informar desta alteração.

De modo a evitar que circulem na rede informações desatualizadas, estas mensagens incluem um número de versão, como é visível na figura 2.7.

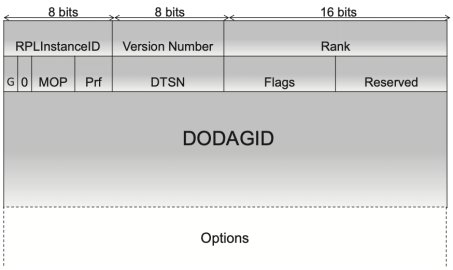

Figura 2.7: Estrutura de uma mensagem DIO [18]

DODAG Information Solicitation (DIS)

Através destas mensagens é possível que um nó solicite o envio de uma mensagem DIO por parte de um outro, de modo a que o primeiro colete as informações necessárias à sua inclusão num DODAG pré-existente.

Destination Advertisement Object (DAO)

Estas mensagens são utilizadas sempre que a rede esteja habilitada à transmissão de pacotes no sentido descendente. Neste caso, impõe-se a necessidade de definir estas rotas, partindo da difusão ascendente das informações imprescindíveis a este processo, por intermédio das presentes mensagens, cujos campos são esquematizados na figura 2.8.

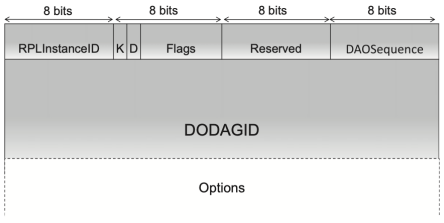

Figura 2.8: Estrutura de uma mensagem DAO [18]

Destination Advertisement Object Acknowledegment (DAO-ACK)

Estas mensagens servem para confirmação de receção de uma mensagem DAO.

2.7.5 Formação de DODAG

Numa primeira fase, o nó raiz, após calcular o seu próprio *rank*, remete aos demais uma mensagem DIO. Por sua vez, sempre que um nó recebe uma destas mensagens calcula o seu próprio *rank*, verifica se o nó que lhe enviou a mensagem é admissível como pai, de acordo com a função objetivo, métricas e restrições em vigor e, caso seja, define-o como tal. Após todas estas ações, o nó reencaminha, também ele, a mensagem DIO que recebeu, sendo esta, agora, alusiva a este último e direcionada a todos os seus vizinhos. Quando um novo nó quer ingressar na topologia este possui duas opções, ou espera a receção de uma mensagem DIO ou, no caso de esta espera ser prolongada, envia uma mensagem DIS. Repare-se que um nó recebe várias mensagens DIO, provenientes de todos os nós vizinhos. Só deste modo é possível a criação individual da lista de candidatos a seu pai. A Figura 2.9 esquematiza todo este processo.

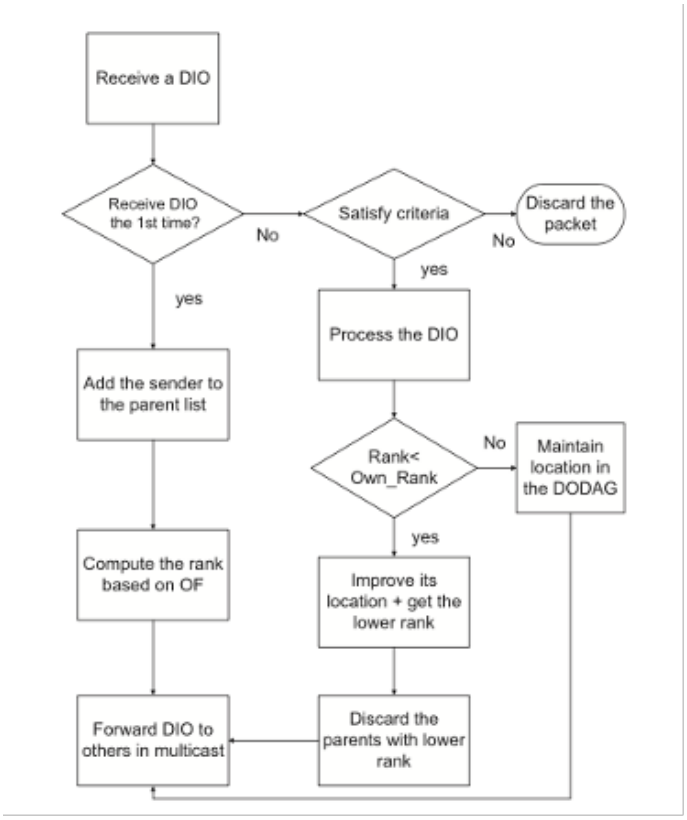

Figura 2.9: Esquema Representativo da criação de um DODAG [18]

2.7.6 Rotas Descendentes

Como já referido acima, o RPL possibilita a difusão de mensagens no sentido ascendente e descendente. Para a operação em modo descendente é necessário que a topologia esteja configurada a operar deste modo (*flag* MOP da mensagem DIO $\neq 0$). Neste caso,

é necessário proceder a configurações adicionais, nomeadamente, definir as próprias rotas descendestes. Para tal, cada nó folha envia no sentido ascendente uma mensagem DAO, onde constam as informações necessárias a que os nós hierarquicamente superiores conhecem os seus descendentes, como apresentado na figura 2.10.

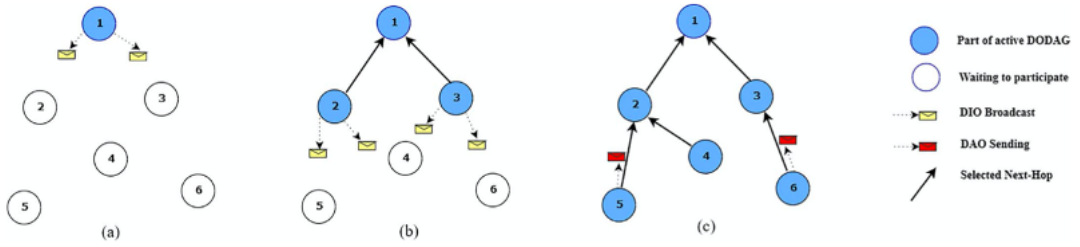

Figura 2.10: Esquema Representativo da propagação de mensagens DAO na determinação das rotas descendentes [31]

No RPL o registo das rotas descendentes pode ser feito de duas formas distintas [31]:

- *Non Storing Mode* – Todos os pacotes são encaminhados pelo nó raiz, sendo que, apenas este nó possui registo das rotas descendentes definidas. Este modo é bastante menos eficiente quando comparado com o modo com armazenamento, isto porque, obrigatoriamente, todos os pacotes têm que caminhar até à raiz DODAG e apenas de seguida são reencaminhados para o destino pretendido. Relativamente ao nó raiz é, neste modo, alvo de uma sobrecarga substancial.

- *Storing Mode* - Neste modo, todos os nós mantêm em memória as rotas descendentes existentes. Em termos de eficiência este modo apresenta vantagem sobre o anterior. No entanto, fruto das limitações dos dispositivos, em especial da escassez de memória dos mesmos, é um desafio considerável a operacionalização destas rotas através deste modo.

2.7.7 Mecanismos de gestão topológica

A formação de um *loop* é das situações mais constrangedoras numa rede, podendo provocar o colapso parcial ou integral da mesma, motivando a perda de pacotes ou atrasos na sua distribuição. O RPL possui vários mecanismos de prevenção para a ocorrência destes incidentes e, ainda assim, outros que promovem a sua correção quando não houver a possibilidade de os evitar.

Aquando da construção da topologia nenhum nó pode definir um pai cujo *rank* seja superior ao seu. Inerentemente, é de fácil compreensão que as mensagens apenas podem fluir para nós com classificação inferior em rotas ascendentes e para nós com classificação superior em rotas descendentes. Por este motivo, o RPL não processa mensagens DIO provenientes de nós com *rank* superior ao do próprio [65].

A movimentação de um nó dentro de um DODAG mostra-se como um ponto potencialmente crítico na formação de um *loop*. De forma a contornar este problema, o RPL impede que um determinado nó anuncie, dentro de um DODAG, um *rank* superior ao valor de *rank* mínimo já anunciado pelo próprio (*RankLowest*) mais uma constante definida na configuração da rede e transmitida nas mensagens DIO, (*RankMaxInc*).

Uma vez que nem sempre é possível evitar a formação de *loops*, o RPL está também capacitado à sua deteção, nomeadamente, através de informações que constam dos pacotes de dados (*RPL Packet Information*). As informações aqui contidas são atualizadas e avaliadas a cada salto, permitindo a deteção de eventuais inconsistências nos *ranks* dos emissores do pacote, verificação da congruência do fluxo de dados, se este circula no sentido ascendente ou descendente da árvore, ou mesmo, se um determinado nó não está capacitado ao reencaminhamento de um pacote, por exemplo, por não possuir rotas confiáveis para um destino [6].

Nem só os *loops* podem provocar constrangimentos na rede, como sabemos, sempre que um nó fica inacessível, é necessário iniciar diligências no sentido de desviar o tráfego e evitar que pacotes se percam, existindo para tal dois mecanismos [80]:

- Global Repair- A raiz DODAG envia uma mensagem DIO com novo número de sequência DAG, reinicializando assim toda a topologia e, inerentemente, serão corrigidas as inconsistências anteriormente detetadas.
- Local Repair- Este mecanismo não se apresenta tão invasivo como o anterior. Desta feita, quando um nó apresenta dificuldades de comunicação com os demais, o próprio envia uma mensagem a todos os seus "filhos" a informar que estes necessitam de nomear outro pai e, de seguida, envia uma mensagem DIS, de modo a ingressar novamente na topologia. Adicionalmente, também o nó filho pode inicializar este mecanismo através da alteração do seu *rank* para valores infinitos ou por alteração do campo *DODAG ID* das mensagens DIO [82].

Dadas as limitações destas redes, o colapso das mesmas poderia ser originário num excesso de mensagens DIO, como solução a este problema, o RPL implementa o Algoritmo *Trickle*, de modo a limitar a emissão destas, nomeadamente, determinando que no caso de alteração da estrutura apenas um nó o comunica por intermédio de uma destas mensagens. [18]

2.7.8 Mecanismos de Segurança

Na especificação inicial do RPL [2], embora seja feita alusão a mecanismos criptográficos e de autenticação passiveis de utilização, não é feita qualquer referência ao modo como estes devem ser implementados [65][20]. Ainda assim, o RPL está capacitado a assegurar a confidencialidade e integridade dos dados, disponibilizando 3 modos de funcionamento:

- Inseguro- todas as comunicações são realizadas sem qualquer tipo de mecanismo de segurança.
- Pré-instalado- para ingressar na rede os nós carecem da utilização das chaves criptográficas simétricas, providenciadas pelo protocolo.
- Autenticado- neste modo, adicionalmente ao que acontece no modo pré-instalado, além de haver a necessidade de que um nó utilize as chaves criptográficas pré-instaladas para ingressar na rede, é ainda necessário que estes obtenham uma segunda chave criptográfica, junto de uma entidade de autenticação, para que possam operar como rotadores [20]. Embora a especificação do RPL [2] clarifique que este modo não deva ser sustentado por chaves simétricas, não é dada qualquer orientação sobre quais as técnicas de criptografia assimétrica a utilizar.

Consciencializados da severidade dos ataques incidentes sobre o protocolo de encaminhamento de uma rede, havendo, inclusive, autores que os consideram os mais pronunciados

em ambientes LLN [43], no RFC 7416 [79] foram abordados alguns dos principais ataques ao RPL, assim como, propostos mecanismos de segurança que os atenuassem ou suprimissem. Constam também deste documento considerações às quais devem obedecer propostas de segurança que se encontrem em desenvolvimento.

Não pude deixar de constatar que dois dos três modos de segurança nativos do RPL, apresentados no RFC 6550 [2], assim como grande parte dos mecanismos propostos em [79], assentam na utilização de criptográfica assimétrica. Todavia, a complexidade computacional destas técnicas criptográficas, em contra-peso com as restrições típicas dos dispositivos IoT, suscita a que a implementação destas estratégias seja um verdadeiro desafio. Consequentemente, emerge a necessidade de desenvolvimento de abordagens de segurança alternativas, mais compatíveis com este tipo ambiente e que promovam a deteção de ataques ao normal funcionamento do RPL de forma menos desgastante, como é o caso dos IDS.

2.7.9 Ataques típicos ao protocolo

Além dos ataques transversais a todas as tipologias de rede, o RPL é vulnerável a muitos outros, especificamente desenhados para as suas características [52][82]. Nesta secção serão abordados os ataques mais frequentes sobre o protocolo, assim como, classificados de acordo com diferentes parâmetros propostos na literatura.

A classificação proposta tem em consideração o componente visado por cada ataque [49][52], assumindo as 3 seguintes categorias:

- Ataques aos recursos- Os ataques incluídos nesta categoria têm por objetivo o esgotamento dos recursos da estrutura. Dentro desta, os ataques podem, ainda, incluir-se numa de duas subcategorias:
  - Diretos- Todos os ataques em que a ação do atacante motive diretamente o constrangimento de um recurso.
  - Indiretos- Situações em que um atacante desencadeia uma série de ações que, indiretamente, comprometem um recurso, seja através de um outro nó, que não o infetado, ou de um mecanismo auto-executável do RPL, cuja condição de acionamento é validada incorretamente.
- Ataques à Topologia- Visam, como o próprio nome indica, o corrompimento da topologia da rede através da sub otimização de rotas ou do isolamento de nós. À semelhança da categoria anterior, também nesta os ataques podem incluir-se numa das seguintes subcategorias:
  - Sub otimização- Todos aqueles que motivem uma performance da rede inferior ao que seria espectável.
  - Isolamento- Sempre que o resultado do ataque implique a inacessibilidade a um nó ou a um conjunto destes.
- Ataques ao tráfego- Tem por objetivo comprometer o tráfego que circula na rede, incluindo-se nesta categoria ataques de *spoofing* e de espionagem.

Adicionalmente, todos os ataques serão classificados como internos, sempre que a operacionalização destes pressuponha a presença do atacante dentro da estrutura, com ação direta sobre as comunicações RPL existentes[41][82], ou externos, sempre que a condição anterior não seja necessária à efetivação do ataque. Relativamente aos primeiros, note-se que em grande parte das situações a sua deteção é substancialmente mais difícil, isto

porque, embora os nós adotem uma postura maliciosa, transparecem para os demais um comportamento legitimo.

Ataques aos Recursos (Diretos)

Flooding Attacks

Consiste no direcionamento, interna ou externamente, de grandes fluxos de dados sobre uma rede, esgotando os seus recursos e tornando-a indisponível. Dentro desta tipologia de ataque destacam-se o ataque *Hello Flooding*, onde o atacante envia uma mensagem DIO (comummente designada por mensagem *Hello*) onde este especifica fortes métricas de encaminhamento, tornando-o elemento preferencial na nomeação como pai de outros nós legítimos. Após se estabelecer na topologia, o atacante pode redirecionar contra a estrutura tráfego fraudulento provocando a sua congestão [49].

Outro ataque frequente dentro desta tipologia é o *DIS Attack*. Nesta modalidade, o atacante envia, sistematicamente, mensagens de solicitação(DIS) para um ou vários nós na topologia, obrigando a que estes estejam constantemente a responder-lhe com mensagens DIO e a reprogramar o seu *Trickle timer* 2.7.7, esgotando, indevidamente, recursos nestas operações [52].

Routing Table Overload Attacks in Storing Mode

Como referido na secção 2.7.5, o RPL, sempre que configurado a operar em *Storing Mode*, mantém na memória de cada nó uma tabela de encaminhamento que contém as rotas descendentes em utilização. Nesses casos, é possível que um atacante propague uma série de mensagens DAO fraudulentas pela rede que, por consequência, irão forçar a que as tabelas sejam sistematicamente atualizadas, levando ao esgotamento da capacidade dessas estruturas provocando, em situações limite, um consumo total da memória dos dispositivos [49]. Ainda que esta situação nem sempre aconteça, repare-se que quando estas tabelas são preenchidas com rotas falsas deixam de ter capacidade para armazenar as rotas legítimas, provocando atrasos ou perda de pacotes.

Local Repair Attacks

O mecanismo de reparo local, discutido em 2.7.7, é utilizado por forma a suprimir situações em que um nó filho perde contacto com seu pai. Nesta situação, o nó filho inicializa o mecanismo de reparo local de uma das seguintes formas, ou altera o seu *rank* para valores infinitos, ou altera o *DODAG ID* das mensagens DIO referentes ao próprio. Acontece que também um atacante pode, por intermédio de um nó infetado, inicializar um reparo local e, desta feita, são alocados recursos num reajuste topológico desnecessário [82].

Ataques aos Recursos (Indiretos)

Increased Rank Attacks

Em termos gerais, este ataque consiste no aumento indevido do *rank* dos nós infetados, levando esta situação à criação do *loops* [49]. Sempre que um nó anuncia um *rank* superior ao que realmente tem, provoca a que seja nomeado um dos seus filhos como seu pai [52]. Esta situação é ainda mais desastrosa se o nó pai for o único intermediário entre a raiz e os seus filhos, sendo que neste caso, além do *loop*, que inevitavelmente se gera, surgem também situações de isolamento dos nós descendentes [49]. Note-se que para que este ataque se efetive é necessário que o nó malicioso desative, localmente, o mecanismo de

deteção de *loops* [49]. Ainda que os nós vizinhos consigam resolver grande parte dos *loops* gerados através dos mecanismos disponibilizados para tal, clarificados em 2.7.7, são alocados recursos nesta tarefa que contribuem largamente para a diminuição acentuada das baterias dos dispositivos, além de todos os outros constrangimentos que desta técnica resultam, como atrasos na distribuição dos pacotes, inacessibilidade a nós, aumento do tempo de entrega dos pacotes e do fluxo de mensagens de controlo DIO.

DAG Inconsistency Attacks

Uma inconsistência DAG acontece sempre que a direção de um pacote não seja condicente com a relação de classificação entre os nós envolvidos na transação do mesmo. Ou seja, quando um nó recebe um pacote de um remetente com classificação superior e com o *bit down* 'O' definido, ou vice-versa. A ocorrência destas situações é detetada pelo protocolo e pode significar a existência de um *loop*. Quando um nó deteta este tipo de inconsistência, este pode proceder de uma das seguintes formas:

- Se o sinalizador 'R', responsável pela sinalização de inconsistências DAG, ainda não estiver definido, o nó que a deteta define-o e encaminha o pacote.
- Se o sinalizador 'R' estiver já definido, o nó que recebe o pacote descarta-o e reinicia o *Trickle Timer* 2.7.7.

Este ataque motiva um aumento do fluxo de mensagens de controlo e, consequente, congestionamento da rede, degradação energética e perda dos dados incluídos nos pacotes descartados. Repare-se que este ataque é de fácil execução, implicando apenas a alteração ou adição dos sinalizadores presentes nos cabeçalhos dos pacotes [49]. Em [40] e [43] é ainda feita referência a uma variante deste ataque em que o atacante impede que o sinalizador 'R' seja acionado, mesmo que a inconsistência seja real e se verifique.

Version Number Attacks

Como abordado na secção 2.7.4 e 2.7.7 o número de versão é um identificador presente no cabeçalho das mensagens DIO que evita que um nó sustente as decisões protocolares em informações topológicas desatualizadas, sendo este parâmetro, em situações normais, unicamente modificável pela raiz DODAG. Desta feita, um atacante pode incrementar este parâmetro nas mensagens DIO levando a que todos os seus vizinhos acreditem estar desatualizados e iniciem diligências no sentido de reverter esta situação, resultando na constante reconstrução da topologia [52], provocando, por sua vez, uma diminuição na performance da rede e potenciando a formação de *loops* [82].

Denial of Service Attack

Este tipo de ataque é comum a todas as tipologias de rede. A ideia passa pelo sequestro de nós legítimos para a partir deles gerar tráfego fraudulento (pacotes IPv6 UDP), com o intuito de provocar o congestionamento da rede [52]. Habitualmente, este tipo de ataque faz uso de um largo conjunto de nós infetados o que dificulta a sua deteção.

Ataques à topologia (Sub otimização)

Routing Table falsification Attacks In Storing Mode

Este ataque é semelhante ao *Routing Table Overload Attack in Storing Mode*, seja por visar a mesma estrutura ou por ambos implicarem o funcionamento do protocolo em *storing mode*. Contudo, o objetivo deste último é diferente. O que acontece é que, embora as

mensagens DAO sejam também adulteradas e, portanto, resultem na criação de rotas descendentes falsas, o processo não é repetido exaustivamente. Assim, embora não se proceda à saturação da estrutura, as inconsistências na tabela de encaminhamento são suficientes para provocar o congestionamento da rede, atrasos na propagação de pacotes e utilização de caminhos sub otimizados [49]. A prática mais comum consiste em um nó malicioso anunciar rotas para um outro nó que não se encontra na sua Sub DODAG, ou seja, que não seja atingível por seu intermédio.

Sinkhole Attacks

Neste ataque, um nó infetado anuncia na topologia ser detentor de melhores rotas para o encaminhamento dos pacotes até a raiz. Com esta ação, o nó malicioso irá tornar-se elemento central na distribuição dos pacotes dos restantes nós, podendo, assim, realizar uma série de outros ataques com consequências nefastas sobre os mesmos [52]. O facto deste ataque não implicar, obrigatoriamente, a inoperacionalidade da rede dificulta a sua deteção [83][49].

Wormhole Attacks

Este ataque pressupõe, tipicamente, a existência de pelo menos dois nós maliciosos na topologia que mantêm entre si uma ligação privada através da qual partilham dados. Esta prática motiva a que os nós que se encontrem nas imediações dos nós infetados assumam, erradamente, nós distantes como seus vizinhos. Para melhor compreensão deste ataque atente-se a figura 2.11. Os nós A e B são nós maliciosos que estabelecem entre si uma ligação privada, por sua vez, os pacotes e mensagens de controlo topológico que são direcionados, legitimamente, a A são, de seguida, partilhados com B através dessa mesma ligação. Desta forma, os nós que se encontram na sub DODAG de A (22) ficarão convencidos que os nós da sub DODAG de B (31,32,33) são seus vizinhos e vice-versa, o que não se verifica [49].

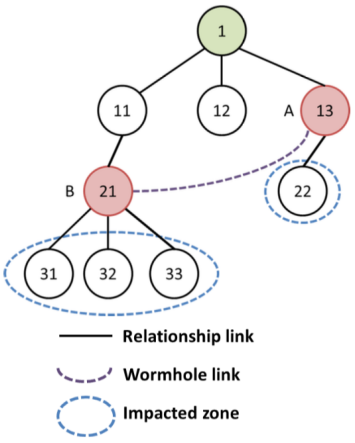

Figura 2.11: Representação esquemática do ataque Wormhole [49]

Internamente, um ataque *Wormhole* pode ser incluído numa das seguintes 3 tipologias [27]:

- Oculto - Nesta modalidade embora os pacotes sejam remetidos por intermédio de um túnel malicioso, não é efetuada qualquer alteração no cabeçalho dos mesmos. Assim, a identificação dos nós extremidade do túnel é de extrema dificuldade, uma vez que, estes desempenham um papel impercetível na perspetiva do nó vítima;
- Exposto - Nesta modalidade os nós maliciosos procedem à modificação do cabeçalho

dos pacotes que circulam pelo túnel, atualizando-os com as suas próprias informações. Desta forma, estes nós simulam um comportamento legítimo, não deixando transparecer a sua responsabilidade no direcionamento do ataque.

- Meio Exposto - Trata-se de uma abordagem híbrida, implicando, deste modo, que um dos nós extremidade do túnel malicioso modifique os cabeçalhos dos pacotes vítima com as sua informações, a par da outra extremidade que não o fará.

Relativamente às técnicas para direcionamento deste tipo de ataque, são diversas, pelo que, passo a clarificar as mais comuns e relevantes [5]:

- Por encapsulamento - Sempre que um nó legítimo propaga um pacote, seja este alusivo às suas informações de encaminhamento ou de dados, se no decurso do trajeto este atingir um nó malicioso é, por intermédio do túnel por este criado, remetido para a outra extremidade, onde o outro nó envolvido no ataque o difundirá na sua vizinhança. Repare-se que nesta situação os nós próximos de ambas as extremidades passarão a considerar-se, indevidamente, vizinhos, dado que, ambos pressupõe que a distância entre eles é diminuta e que a respetiva rota incluí um número reduzido de saltos, tendo sido a contagem dos mesmos negligenciada.
- Canal fora da banda - Esta é, talvez, a abordagem deste tipo de ataque menos recorrente, visto carecer da inclusão de *hardware* específico para a efetivação do mesmo, seja um cabo dedicado ou uma ligação sem fio, de alta qualidade, por onde o túnel malicioso se irá estabelecer.
- Transmissão de alta potência - De forma semelhante ao que acontece na técnica por encapsulamento, as informações de encaminhamento de nós legítimos são difundidas para zonas longínquas. Esta técnica diverge da anterior no sentido em que um único nó pode direcionar o ataque. Para tal, os dados são modelados num sinal de alta potência, atingindo assim os nós mais distantes.
- Retransmissão de pacotes - A retransmissão de pacotes implica que todos aqueles provenientes de nós legítimos, localizados na zona de vizinhança de um nó malicioso, sejam difundidos sem que os seus cabeçalhos sejam atualizados, criando a errada sensação de que os nós envolvidos na transação são vizinhos.
- Desvios ao protocolo - Nesta técnica, os nós maliciosos não respeitam os intervalos temporais nos quais devem enviar as suas informações topológicas. Deste prática resulta a que em determinadas situações a informação relativa a um nó distante atinja mais rapidamente um nó vítima do que as informações de nós legítimos que se encontram, efetivamente, mais próximos.

Routing Information Replay Attacks

O presente ataque visa o corrompimento das tabelas de encaminhamento, nomeadamente, através da coleta e posterior difusão de informações topológicas de outros nós. O ataque foi proposto em [64], contudo, os autores não clarificaram a potencialidade deste ataque sob o protocolo RPL, dispondo este de mecanismos que promovem a constante atualização das informações topológicas que circulam na rede.

Worst Parent Attacks

Em [39] foi proposto o *Worst Parent Attack*. Este ataque implica que, na fase de escolha de um pai 2.7.5, o nó opte pela pior opção dada a função objetivo em vigor, contribuindo para uma operação sub otimizada da rede e consequente baixo desempenho da mesma [49].

Selective Forwarding Attacks

Este ataque é bastante semelhante ao ataque *Grayhole*, contudo, se nesse ataque a sabotagem dos pacotes a encaminhar é aleatória, neste ataque, apenas são encaminhadas as mensagens de controlo do RPL barrando todo o restante tráfego [52]. O facto de as mensagens de controlo não serem comprometidas motiva a que a sua deteção seja difícil.

Neighbor Attacks

Neste ataque, o nó malicioso sempre que recebe uma mensagem DIO apenas a reencaminha, não a atualizando, previamente, com as suas informações como seria suposto. Note-se que se o nó boicota este processo na fase de difusão de mensagens DIO os seus vizinhos interpretaram-nas como sendo referentes ao penúltimo remetente e, por sua vez, o destinatário final considera que este nó é seu vizinho, o que não acontece nem, tão pouco, este está a seu alcance [52].

ETX Manipulation

A métrica ETX, abordada em 2.7.3, pode ser utilizada por um atacante de modo a posicionar-se privilegiadamente na topologia para, de seguida, proceder a um outro ataque. Note-se que ao diminuir fraudulentamente o valor ETX de um nó infetado este irá ser elemento preferencial na nomeação de um nó pai por parte de outros nós, colocando-se numa posição hierarquicamente superior, onde o fluxo de dados é maior [82].

Ataques à topologia (Isolamento)

DAO Inconsistency Attacks In Storing Mode

Existem diversas situações que implicam que rotas descendentes deixem de ser válidas e, portanto, é necessário que o protocolo de encaminhamento desenvolva esforços no sentido de encontrar soluções que permitam a entrega dos pacotes aos nós abrangidos por essas rotas, agora inoperacionais. Assim, quando um nó deteta um erro de encaminhamento, ou seja, não consegue fazer fluir o pacote para o destino pretendido, aciona no pacote a *flag* 'F' sinalizando esta situação e, de seguida, retorna o pacote ao nó que lho havia, previamente, remetido. Este, por sua vez, quando recebe um pacote nestas circunstâncias, remove a *flag* e procura o seu envio por um caminho alternativo, descartando a rota de utilização futura. Desta feita, um atacante pode acionar a *flag* 'F', indevidamente, deixando inacessíveis os nós hierarquicamente inferiores, sem caminho alternativo para a raiz DODAG, como é o caso, na figura 2.12, dos nós 31, 32 e 33 vítimas de um ataque deste tipo por parte do nó 21. Este tipo de ataque, quando direcionado por vários nós, pode comprometer a utilização de todas as rotas descendentes previamente estabelecidas [49].

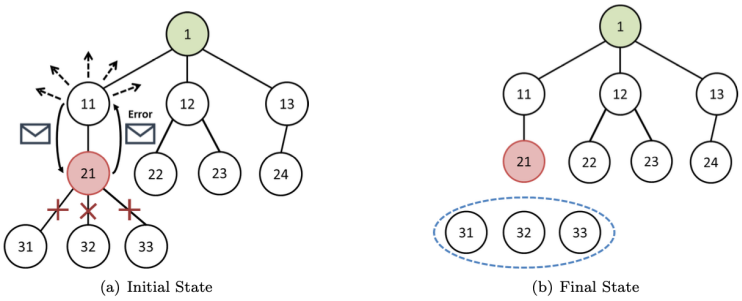

Figura 2.12: Exemplo DAO Inconsistency Attacks in Storing Mode, [49]

Blackhole Attacks

O *Blackhole* ataque consiste em um nó malicioso descartar todo o tráfego que lhe é direcionado. Este ataque é tão mais grave quando mais alto estiver o nó infetado na topologia, sendo que, nessas posições, a possibilidade de coleta e adulteração de dados é bastante mais substancial [49]. Existe uma variante deste ataque denominado de *Grayhole* ataque que, embora o princípio seja o mesmo, o tráfego não é integralmente descartado [83]. O boicote ao encaminhamento dos pacotes é, neste último, intermitente e aleatório.

Ataques ao tráfego (Espionagem)

Sniffing Attacks

Este ataque visa a coleta indevida dos dados que circulam na rede. Para tal, um atacante pode fazer uso de um nó comprometido ou do próprio meio de comunicação partilhado [49]. Além das consequências óbvias de acesso indevido a dados, é importante frisar que este ataque pode ser incluído num outro de maiores dimensões, sendo que, numa primeira fase, mediante o estudo das mensagens intercetadas, é possível a determinação das características e configurações da rede em foco.

Traffic Analysis Attacks

Os ataques ao tráfego não implicam o acesso ao conteúdo dos pacotes, passam sim por determinar as características e padrões no fluxo de encaminhamento dos mesmos por forma a aferir a arquitetura da rede e modo de operação da mesma. Este ataque não se mostra tão invasivo como o anterior, no entanto, é igualmente determinante na fase de *scouting* a uma rede [82] [52]. Com as informações resultantes desta técnica facilmente o atacante poderá optar pela melhor estratégia para o direcionamento de um outro tipo de ataque no futuro. De referir que a sua ação passiva dificulta a sua deteção.

Ataques ao tráfego (Usurpação de Identidade)

Decreased Rank Attacks

A manipulação indevida do *rank* de um nó infetado é o meio mais simples para mover um nó na topologia, isto porque, quanto mais baixo for o *rank* de um nó, mais próximo este se irá situar do nó raiz. A facilidade desta operação resulta do facto de que o *rank* é especificado nas mensagens DIO e, assim, basta adulterar o campo relativo a este indicador para que o nó infetado transite para uma posição superior na topologia [49] [52]. Repare-se que o facto de o nó estar próximo da raiz é uma vantagem no direcionamento de outros ataques, pois aí o fluxo de dados que transitam pelos nós é bastante superior.

Identity Attacks

Esta categoria contempla dois ataques de especial relevo [82][52]:

- *Sybil*- O atacante forja sistematicamente a identidade de diferentes nós tornando a sua deteção extremamente difícil.
- *Spoofing*- também designado por ataque de ID, o *Spoofing* passa pelo usurpação de identidade de um outro nó legítimo. Sendo que, num caso extremo, é possível que um atacante obtenha total controlo da rede assumindo a identidade do nó raiz.

Nome	Breve discrição	Classificação do ataque (Interno ou Externo)
Ataques aos recursos (Diretos)		
Flooding Attacks	São direcionados grandes fluxos de dados na rede, esgotando os seus recursos deixando-a inoperacional. Contempla o ataque *Hello flooding* onde o atacante numa primeira fase envia mensagens DIO fraudulentas de modo a posicionar-se estrategicamente na topologia para, posteriormente, proceder à injeção de grandes quantidades de dados na mesma [49]. O outro ataque, DIS ataque, passa pela constante propagação de mensagens DIS pela topologia motivando o sistemático reajuste do *Trickle timer* e emissão de mensagens DIO como resposta por parte dos nós legítimos, esgotando recursos desnecessariamente [52]	Interno e Externo
Routing Table Overload Attacks in Storing Mode	Quando o RPL é operado em *storing mode*, a propagação de mensagens DAO fraudulentas motiva a constante atualização das tabelas de encaminhamento descendente o que provoca a saturação deste recurso e a indisponibilidade do mesmo ao armazenamento das rotas legitimas [49].	Interno
Local Repair Attacks	Através da adulteração do rank, para valores infinitos, ou do DODAG ID são motivadas, indevidamente, ações de reparo local da topologia [82].	Interno
Ataques aos recursos (Indiretos)		
Increased Rank Attacks	É incrementado, maliciosamente, o rank de um nó forçando a que este nomeie um dos seus filhos como seu pai [52], situação da qual resulta um loop e, em situações especificas, o isolamento de um conjunto do nós.	Interno
DAG Inconsistency Attacks	São modificados os sinalizadores nos cabeçalhos dos pacotes de dados forçando a que estes sejam descartados [49]. Uma outra variante deste ataque impossibilita a definição dos sinalizadores anteriores ainda que a inconsistência se verifique [43] [40].	Interno
Version Number Attacks	É adulterado o número de versão de modo a obrigar a um reajuste da topologia através de um reparo global [82].	Interno
Denial of Service Attack	Consiste no sequestro de nós na topologia a partir dos quais são gerados grandes volumes de tráfego, provocando a indisponibilidade de todo o sistema por falta de recursos [52].	Interno
Ataques à topologia (Sub-otimização)		
Routing Table falsification Attacks In Storing Mode	Através de mensagens DAO fraudulentas, as tabelas de roteamento descendente são preenchidas com informações erradas criando inconsistências na topologia [49]	Interno
Sinkhole Attacks	Um nó malicioso anuncia, fraudulentamente, melhores métricas para o encaminhamento dos pacotes até à raiz a fim de centralizar em si um maior número de pacotes provenientes de outros nós legítimos[83].	Interno
Worhole Attacks	Dois ou mais nós comprometidos estabelecem entre si um canal dedicado para a partilha de pacotes criando inconsistências na topologia [49].	Interno
Rountig Information Replay Attacks	Coleta e posterior difusão de informações topológicas de outros nós [64]	Interno
Worst Parent Attacks	Adulteração da função objetivo de modo a ser escolhido o pior pai [39].	Interno
Selective Forwarding Attacks	É negligenciada a propagação de pacotes de dados por parte dos nós maliciosos, ainda que os pacotes de controlo continuem a ser processados e reencaminhados de forma correta [52].	Interno
Neighbor Attacks	Um nó malicioso propaga as mensagens DIO sem previamente atualizar os campos com as suas informações, motivando que os nós vizinhos fiquem com uma perceção errada da topologia [52] [82].	Interno e Externo
ETX Manipulation	Manipulação da métrica ETX com o objetivo de obter uma posição privilegiada na topologia [82].	Interno
Ataques à topologia (Isolamento)		
Blackhole Attacks	Um nó malicioso descarta todos os pacotes que lhe são direcionados impedindo que estes cheguem ao destinatário [83]	Interno
DAO Inconsistency Attacks In Storing Mode	Acionamento fraudulento da Flag F, nos pacotes de dados, inviabilizando a futura utilização das rotas descendentes intervenientes na transmissão do mesmo [49].	Interno
Ataques ao tráfego (Espionagem)		
Sniffing Attacks	Análise do conteúdo dos pacotes que circulam na rede [49].	Interno e Externo
Traffic Analysis Attacks	Análise dos padrões de transmissão de mensagens a fim de determinar informações sobre a topologia [49].	Interno e Externo
Ataques ao tráfego (Usurpação de identidade)		
Decreased Rank Attacks	O atacante diminui, ficticiamente, o rank de um nó infetado por forma a obter uma posição privilegiada na topologia [52]	Interno
Sybil	Um nó comprometido assume a identidade de múltiplos outros contidos na rede [82].	Interno
Spoofing	Um nó faz-se passar por um outro, num caso extremo pode obter controlo total da rede, simulando a identidade da raiz DODAG [52].	Interno

Tabela 2.2: Síntese de ataques ao RPL

A tabela 2.2 sumariza as principais características de cada um dos ataques acima descritos, classificando-os como internos ou externos e agrupando-os de acordo com o componente visado por cada um deles (tráfego, topologia ou recursos).

2.8 Intrusion Detection Systems (IDS)

O termo IDS foi proposto em [3] decorria o ano de 1980, remetendo para um mecanismo que, mediante a análise das comunicações, atividade interna e eventos ocorrentes num sistema ou rede, é capaz de proceder à deteção de ataques. Atualmente, a utilização deste tipo de tecnologia é frequente na maior parte dos sistemas informáticos com requisitos sólidos de segurança. Repare-se que a capacidade de análise interna dos fluxos de dados se apresenta como um excelente complemento às clássicas estratégias de segurança por recurso a *firewall*, podendo estas, inclusive, operar em conjunto com o IDS, por forma a desempenhar um papel ativo na supressão dos agentes maliciosos envolvidos nos ataques reportados.

A evolução dos IDS motivou a que, à data, estes sejam classificados nas mais diversas categorias de acordo com as suas características [8].

Relativamente ao componente sobre o qual operam estes podem ser *host based*, no caso de apenas monitorizarem um dispositivo de uma rede ou sistema, ou, *network based*, no caso destes serem empregues na monitorização integral de um sistema ou rede constituída por múltiplos *hosts*.

No que toca à sua arquitetura, os IDS podem ser classificados da seguinte forma:

- Centralizada - Todas as atividades decorrentes da operação do IDS são incluídas num único dispositivo, dedicado ou não, posicionado local ou remotamente. As principais vantagens desta abordagem prendem-se com o facto de a sobrecarga implícita das atividades deste componente não comprometerem a performance dos demais elementos do sistema ou rede. No entanto, esta centralização motiva a existência de um ponto único de falha.

- Distribuída - Todos os componentes do sistema ou rede no qual o IDS está instalado desempenham um papel ativo na deteção dos ataques. Esta abordagem imprime uma maior resiliência em caso de falha de um dos componentes, alem de que, os eventos decorrentes de um ataque podem, deste modo, ser avaliados de diferentes perspetivas, podendo, no entanto, esta prática resultar em inconsistências de classificação. Em contrapartida, denota-se uma sobrecarga generalizada dos componentes do sistema/rede, sendo estes, em vários cenários, altamente restritos.

- Híbrida - Este tipo de abordagem combina aspetos dos dois métodos anteriores, embora as atividades de coleta de dados estejam a cargo de todos os elementos, as tarefas de análise dos mesmos, tipicamente mais desgastantes, são da responsabilidade de um componente central dotado de mais recursos.

Já no que diz respeito ao esquema de deteção este pode ser:

- Baseado em assinatura - Os ataques são detetados sempre que haja uma correspondência dos elementos coletados e em avaliação com uma ou mais assinaturas de ataque previamente definidas. Estes IDS são precisos e eficazes, em contrapartida,

mostram-se pouco eficientes na deteção de ataques cujas assinaturas ainda não estejam definidas.

- Baseado em anomalia - Numa primeira fase é criado um perfil que reflete o funcionamento do sistema ou rede em condições normais, por comparação, sempre que detetado um desvio a este perfil é reportado um ataque. Tem como vantagem a possibilidade de identificação de ataques inovadores, assumindo que todos eles produzem comportamentos erróneos no sistema ou rede vítima. Ainda assim, obrigam a uma configuração morosa aquando da sua implementação, fruto do tempo de aprendizagem necessário à definição dos perfis. Este tipo de esquema de deteção resulta, normalmente, em altas taxas de falsos positivos, sendo este o maior inconveniente da presente abordagem.

- Baseado em especificações - Partem do mesmo pressuposto do esquema de deteção anterior, no entanto, a atividade da rede ou sistema sob monitorização não é relacionada com um perfil, mas sim com um conjunto de características e definições protocolares, previamente definidas. A maior dificuldade neste tipo de IDS passa pela definição de limiares de desvio, precisos e assertivos, a partir dos quais é gerado o alerta de ataque. Mesmo assim, estes IDS apresentam taxas de falsos positivos mais satisfatórias que outros baseados em anomalia, todavia, ainda superiores àquelas verificadas nos modelos baseados em assinatura.

- Híbrido - Este esquema procura incluir as características mais promissoras de cada um dos anteriores por formar a aumentar a eficácia na deteção.

2.9 Intrusion Prevention Systems (IPS)

Um sistema de prevenção de intrusões é um mecanismo de controlo, similar a um IDS, mas que, além de proceder à deteção de potenciais intrusões, realiza uma série de diligências, em tempo real, no sentido de que a ameaça detetada não se efetive [25].

Embora não seja o tema do presente trabalho, é uma estratégia de segurança com grande potencialidade e, portanto, é importante a clarificação deste conceito dado o contexto. Note-se que, este tipo de mecanismo é frequentemente empregue juntamente com um IDS, visto serem tecnologias complementares. Em muitos cenários, os ataques reportados por um IDS são suprimidos por intervenção de um Intrusion Prevention Systems (IPS), visto este apresentar uma postura ativa na rede ou sistema onde se insere.

2.10 Síntese de capítulo

No presente capítulo foram discutidos os principais aspetos relacionados com o conceito IoT. A inclusão de dispositivos triviais no domínio da Internet é uma tendência em crescimento, comprovável através do vasto domínio aplicacional onde esta tecnologia se insere aos dias de hoje, seja na indústria, na agricultura, ou mesmo, nas futuristas casas e cidades inteligentes.

Se a potencialidade deste tipo de dispositivos é imensa, não pude deixar de abordar todas as suas limitações, em particular, as resultantes das restrições energéticas e computacionais típicas deste *hardware*, que, por sua vez, se estendem às redes compostas por estes dispositivos, comummente designadas por LLN.

O ponto acima remete para a necessidade de desenvolver um conjunto de estratégias que permitam ultrapassar todos estes constrangimentos. A nível protocolar, a operacionalização de dispositivos IoT assenta num modelo standard de 6 camadas, das quais são integrantes protocolos especificamente desenhados para atender a todos os requisitos inerentes a esta tecnologia.

Dado o tema do presente trabalho, neste capítulo foi dado especial relevo ao protocolo de encaminhamento RPL. Baseado em grafos acíclicos direcionados (DAG), este protocolo promove a organização hierárquica dos nós na topologia através do relacionamento de métricas e restrições por intermédio de funções objetivo capacitadas não só à determinação do rank de cada nó, como ao encaminhamento do tráfego até ao respetivo destino pela rota mais conveniente.

À semelhança de outros, também o RPL é vulnerável a uma série de ataques, sejam eles de adulteração ou supressão do tráfego, como o ataque *blackhole*, *grayhole*, ou o ataque de encaminhamento seletivo. Os ataques podem ainda visar a inativação dos mecanismos de controlo protocolar, destacando-se neste âmbito o ataque de reparo local ou o ataque por inconsistência de mensagens DAO. Por fim, os ataques podem também incidir na desorganização da topologia, isto porque, normalmente, os nós maliciosos tendem a posicionar-se perto da raiz, uma vez que, ai os fluxos de dados são superiores. Destaca-se aqui o ataque *sinkhole*, *decreased rank*, ou o ataque de adulteração das tabelas de encaminhamento. Relativamente ao ataque *Wormhole*, este implica o estabelecimento indevido de um canal entre dois nós maliciosos através do qual são remetidos os pacotes vítima, de onde resulta a falsa sensação de que os nós legítimos que se encontram nas duas extremidades são vizinhos.

Ciente do impacto dos vários ataques documentados, sintetizados na tabela 2.2, e em especial do ataque Wormhole, foi ainda, neste capítulo, clarificado o principio de funcionamento de um IDS, dada a aptidão deste mecanismo à deteção de ataques na tipologia de rede em foco. Sucintamente, esta ferramenta de segurança procede a uma monitorização constante da atividade topológica com o intuito de identificar padrões passiveis de associação com um evento malicioso o que, inerentemente, permite a deteção dos ataques.

Esta página foi propositadamente deixada em branco.

Capítulo 3

Estado da Arte

No presente capítulo pretende-se a clarificação das propostas tecnológicas, já existentes na literatura, que visem a supressão de alguns dos ataques abordados em 2.7.9, com especial destaque para a deteção de ataques *Wormhole*. A estrutura desta secção passará por uma descrição textual de cada uma das propostas seguida de uma tabela comparativa de todas elas. De modo a facilitar a leitura e compreensão, as proposta de foro mais generalizado serão apresentadas de seguida, a passo que, as técnicas associadas à deteção do ataque *Wormhole* serão agrupadas numa subsecção específica.

3.1 Propostas para deteção de ataques sobre o protocolo RPL

O SVELTE é um IDS híbrido em tempo real, baseado em anomalias, fruto de inconsistências a nível das especificações do RPL, apto à deteção de ataques *Sinkhole* e encaminhamento seletivo, proposto em [63]. A operacionalização desta tecnologia parte do pressuposto da utilização de 3 módulos, o primeiro, *6LowPAN Mapper*, é responsável pela coleta de informações RPL que possibilitam a representação da rede em 6 LoWPAN Border Router (6BR). O segundo módulo, por sua vez, é responsável pela análise das informações providenciadas pelo primeiro e deteção dos ataques, nomeadamente, através das evidências por estes deixadas, seja a nível do grafo topológico, disponibilidade dos nós, ou inconsistências no gráfico de encaminhamento. Por fim, o terceiro módulo adota uma postura de *firewall* distribuída, podendo proceder à supressão do tráfego malicioso e ao bloqueio de nós invasores, no caso de um qualquer evento se enquadrar nos padrões pré introduzidos manualmente pelo gestor de rede. Desta feita, assume-se a existência de dois sub módulos em cada nó da topologia, um destinado à coleta de dados e o segundo responsável pela atividade da *firewall*. Como vantagens, destaca-se o eficiente consumo energético desta solução. O facto das tarefas que exigem mais recursos computacionais serem realizadas no domínio do 6BR motiva uma diminuição da sobrecarga de outros nós da topologia com recursos limitados, o que é um fator determinante neste tipo de ambientes. Segundo os autores, o SVELTE apresenta uma alta taxa de verdadeiros positivos, sem, ainda assim, ser provocado um *overhead* substancial. O caráter modular da proposta possibilita a expansão da tecnologia à deteção de mais tipologias de ataques. Como desvantagem, apresenta-se a alta taxa de falsos positivos, justificada no artigo por eventuais erros de representação da topologia em 6BR, resultantes de inconsistências temporais ou extravio de pacotes inerentes às medições de classificação. Outro ponto crítico passa pela necessidade de estabelecimento dos módulos do IDS em posições estratégicas, uma vez que, a deteção de um nó malicioso está dependente da comunicação deste com um pai legítimo.

A validação desta proposta passou pela simulação em *Cooja* de nós *Tmote Sky* operados por *contiki OS*. De modo a garantir a consistência dos resultados, cada teste foi repetido 10 vezes, tendo sido, posteriormente, calculada a média e desvio padrão dos valores obtidos.

A potencialidade do SVELTE motivou, em [72], a sua expansão, tendo-lhe sido incluído um módulo de deteção baseado na métrica ETX. Como referido na secção 2.7.9, a métrica ETX pode ser alvo de manipulação indevida por parte de um atacante, de modo a este conseguir posicionar-se estrategicamente numa zona topológica onde o direcionamento de outros ataques seja mais impactante. De modo a suprimir a ocorrência desta prática foi adicionado ao SVELTE [63] um módulo que visa a deteção de eventuais desvios no valor ETX anunciado e efetivo. Recorde-se que no RPL o valor ETX de um nó pai deve ser menor que o valor ETX apresentado por um qualquer nó seu filho, sendo esta a condição de validação deste novo módulo e verificada individualmente pelo *6LowPAN Mapper*. Adicionalmente, os autores propuseram a inclusão de informações geográficas dos nós, isto para, em caso de não funcionamento dos outros módulos, ser, ainda assim, possível proceder à deteção de ataques. A abordagem utilizada neste último ponto passa pela associação da distância que separa dois nós ao alcance dos mesmos de modo a apurar a autenticidade destes, carecendo da inclusão de *hardware* específico para este efeito. A validação desta proposta foi realizada por via de simulação, sendo esta realizada em *Contiki OS* no simulador *Cooja* onde foram implementados nós *Tmote Sky*. Como resultados, os autores alegam uma satisfatória precisão, nomeadamente, alta a taxa de verdadeiros positivos aliada a uma boa eficiência energética e computacional, em particular quando os módulos de deteção são utilizados em conjunto. Relativamente às desvantagens, são herdadas do modelo original do SVELTE [63], sendo estas alto *overhead* e alta taxa de falsos positivos.

O InDReS [78] apresenta-se como um IDS distribuído, baseado em especificação, capaz de detetar ataques *Sinkhole*. O seu modelo de operação é baseado na topologia de árvore de *Cluster*. Desta feita, o *Cluster* principal adota um comportamento de monitor. Este tipo de nós, contabilizam o número de pacotes descartados pelos nós adjacentes e no caso desse valor ultrapassar um limiar, pré-definido, é gerado um alerta e, posteriormente, isolado o nó malicioso. Na proposta o InDReS é demonstrado através de uma implementação em NS-2 constituída por 150 nós e cujos resultados demonstram uma alta taxa de eficácia, baixa taxa de falsos positivos, consumo energético eficiente e baixo *overhead*. As falhas mais pronunciadas deste modelo passam pela existência de uma série de elementos centrais de falha, em concreto os nós líder. Além disso, é notória a dificuldade em definir um limiar ótimo a utilizar no processo de deteção em especial em redes dinâmicas.

A proposta apresentada em [32] trata-se de uma *freamwork* dedicada à deteção de ataques Denial of Service (DoS). O IDS nela referenciado adota uma tipologia centralizada, sendo baseado em assinaturas. O sistema proposto monitoriza, ativamente, todos as comunicações existentes sobre a rede e, desta forma, sempre que detetado um desvio é gerado um alarme. Uma particularidade deste sistema passa pelo facto de cada possível ataque sofrer uma dupla verificação, sendo que, depois de gerado o alerta é novamente verificada a efetivação do mesmo por intermédio de informações veiculadas por outro elemento de monitorização da rede. No caso do ataque realmente se efetivar, são coletadas as informações sobre o mesmo, de modo a criar uma nova assinatura que facilite a deteção desse tipo de ataque numa ocorrência futura. As vantagens desta abordagem são a alta taxa de verdadeiros positivos e baixa taxa de falsos positivos. Como desvantagens apresentam-se o decréscimo funcional em redes dinâmicas e a inexistência de mecanismos claros e eficientes de atualização das assinaturas.

Em [33], os autores propõem a utilização de um IDS centralizado, baseado em assinatura, para a deteção de ataques DoS. O esquema de deteção passa pela implementação de um nó sonda, encarregue pelo *snifing* dos pacotes que circulam na rede. As informações daqui resultantes serão processadas e correlacionadas com padrões de ataque pré-definidos, por intermédio do *Suricata* (IDS *Open Source*). De modo a evitar o congestionamento da rede, bem como, a assegurar a operação do IDS proposto em caso de interferência nas transmissões sem fios, a conexão entre a sonda e o módulo *Suricata* é feita através de uma ligação dedicada por cabo. Como vantagens desta proposta apresentam-se as taxas de deteção bastante satisfatórias, bem como, a possibilidade de expansão à deteção de outros ataques, nomeadamente, através da definição de novas assinaturas. Inclusive, os autores propõem a integração desta tecnologia com outras como o *SVELTE* [63]. A nível de limitações destaca-se o *overhead* provocado sobre a rede e a passividade na deteção de ataques internos, em parte devido à arquitetura centralizada do mesmo. Não posso deixar de frisar que esta proposta apenas foi validada através de um exemplo hipotético, não existindo qualquer simulação que a sustente em termos de desempenho e usabilidade. Não fica também claro a forma como as assinaturas são atualizadas durante a execução do IDS.

No artigo [11] é proposto um IDS para a deteção de ataques *sinkhole* em redes estáticas e móveis. A estratégia de deteção assenta na metodologia de reputação e confiança *Watchdog* aliada a *clustering* dinâmico, o que, por sua vez, implica o agrupamento dos nós hierárquica e dinamicamente. O mecanismo de disparo para a ocorrência de um alerta é acionado no caso do número de mensagens de entrada e saída de um dispositivo não ser condicente. Repare-se que, no modelo em estudo, os nós possuem dupla função sendo não só responsáveis pelo encaminhamento de mensagens mas também pela monitorização topológica. Este sistema foi validado por recurso a simulação em *Cooja*, a qual compreendeu 50 nós estáticos e móveis. A título comparativo, o SVELTE [63] foi também simulado no mesmo cenário, no entanto, apresentou resultados inferiores face à alta taxa de precisão decorrente de baixas taxas de falsos positivos e negativos do presente modelo, não obstante, ainda, de algumas limitações, nomeadamente, no que toca ao reduzido número de tipologias de ataques detetáveis e à sobrecarga dos componentes do sistema, ainda que esta solução não tenha sido claramente estudada quanto ao seu impacto sobre dispositivos de baixa potência.

Com o objetivo de identificar nós maliciosos numa topologia, em [56] é proposto um IDS cuja principal particularidade se relaciona com o facto de este ser bastante leve e dotado de uma eficiência energética notória, conferindo-lhe uma mais-valia aquando de operação em ambientes IoT. Esta particularidade é conseguida com a adoção de técnicas de deslocamento auxiliar e decisão antecipada. Por ser um IDS baseado em assinatura permite um alargamento do escopo de deteção, ainda que, dependente da intervenção do gestor de rede. No entanto, na simulação da proposta por recurso a um *Raspberry Pi* e sensores *Omnivision OV5647*, foi possível apurar que, embora eficiente energética e computacionalmente, a abrangência na tipologia de ataques detetados é, em todo o caso, diminuta.

Para a deteção de ataques de *Rank*, *Sinkhole*, reparo local, vizinho, e DIS, em [42] foi proposto um sistema de deteção baseado em especificação e assente em *Extended Finite State Machine*. De forma sucinta, a solução compreende uma fase inicial onde o RPL é simulado sob condições normais e uma segunda fase onde a informação recolhida da anterior simulação é transposta para os algoritmos de deteção que, por sua vez, com base em desvios ao perfil, previamente criado, procedem à deteção dos ataques. Os autores do artigo procedem à validação da proposta por simulação em *Cooja*, compreendendo esta 100 nós, distribuídos aleatoriamente, de entre os quais se destacam 1 nó concentrador e 11 nós chefes de *cluster*, sendo estes responsáveis pela coleta e análise das informações

relativas aos restantes nós da rede. Como resultado, foi possível apurar uma alta taxa de precisão e deteção (True Positive Rate (TPR) de 100% e False Positive Rate (FPR) de 6,78%), denotando-se-lhe, ainda, uma boa capacidade de escalabilidade e eficiente degradação energética que contrasta com a ineficiência temporal no processamento e sobrecarga comunicacional. De frisar que a precisão do sistema decresce à medida que o tempo de execução do mesmo aumenta.

Outro modelo de destaque para a deteção de comportamentos anormais na topologia de rede em estudo é proposto em [77]. A abordagem descrita no documento passa pela correspondência do padrão de bits de modo a obter características dos *payloads* dos pacotes que circulam na rede, por forma a classificá-los como normais ou anormais, presumindo-se nesta situação a existência de um ataque a decorrer. Para validação da proposta, os autores recorrem à coleta dos pacotes de um cenário real, relativos aos dados transmitidos entre uma câmara de vídeo e uma estação meteorológica, pelo que foi possível agrupar os pacotes em 3 conjuntos distintos e detetar com sucesso 4 diferentes tipos de ataque, com uma taxa de falsos positivos baixa, embora com um significativo *overhead* computacional.

De forma a facilitar a comparação entre as várias propostas acima descritas, a tabela 3.1 sumariza as principais características de cada uma delas, referindo, individualmente, o esquema de deteção e arquitetura utilizada, ataques passiveis de deteção, principais vantagens e desvantagens e a estratégia de validação proposta pelos autores.

Artigo	Esquema de deteção	Arquitetura	Ataques	Vantagens	Desvantagens	Estratégia de avaliação
[63]	Anomalia	Hibrido	Sinkhole; Encaminhamento seletivo	Boa gestão de recursos, dada a centralização em 6BR; Bom PDR; Pouco overhead; Extensível a outros ataques; Alta taxa de verdadeiros positivos; Eficácia Energética	Inconsistências provocadas pela dessincronização temporal das medições podem provocar altas taxas de falsos positivos; Substancial consumo de recursos; Baixa precisão de deteção; Não é considerada mobilidade dos nós nem ataques coordenados; Dificuldade no posicionamento dos módulos do IDS.	Simulação em contiki OS/ Cooja
[72]	Anomalia	Hibrido	Sinkhole; Encaminhamento seletivo; Ataques sobre a métrica ETX;	Alta taxa de eficácia; Alta taxa de positivos verdadeiros; Consumos energético e de recursos eficiente.	Overhead provocado; Falsos Positivos; Não é considerada mobilidade dos nós ou ataques coordenados.	Simulação em contiki OS/ Cooja
[78]	Especificação	Distribuído	Sinkhole	FP baixa; Eficiência energética, Taxa de eficácia alta; Overhead	Inutilidade em redes dinâmicas; Elementos centrais de falha (nó líder); Dificuldade em definir um limiar justo; Não é deteção em tempo real; Dependência do administrador de rede.	Simulação em NS2 com 150 nó e comparação com o INTI
[32]	Assinatura	Centralizada	DOS	Boas taxas de deteção e poucos falsos positivos; Possibilidade de expansão a outros ataques com novas inclusão de novas assinaturas; Possibilidade de integração com outras propostas	Overhead provocado; Não é clara a forma como as assinaturas são atualizadas; Deteção de ataques internos; Decréscimo funcional em redes dinâmicas.	Criação de uma Testbed submetida a pentest em contiki OS.
[11]	Híbrido	Distribuído	Sinkhole	Alta precisão; Baixos falsos positivos; Baixos falsos negativos; Admite a mobilidade dos nós	Apenas deteta uma tipologia de ataques; Não foi estudado o impacto deste modelo sobre dispositivos de baixa potência; Imprime sobre a rede um overhead computacional considerável	Contiki OS no simulador Cooja
[56]	Assinatura	Distribuído	Ataques convencionais	Leveza e simplicidade; Eficácia na deteção;	Pouca diversidade na tipologia de ataques detetáveis.	Raspberry Pi e sensores Omnivision OV5647
[33]	Assinatura	Centralizada	DOS	Boas taxas de deteção e poucos falsos positivos; Possibilidade de expansão a outros ataques com novas assinaturas; Possibilidade de integração com outras propostas	Overhead provocado; Não é clara a forma como as assinaturas são actualizadas; Deteção de ataques internos; Decréscimo funcional em redes dinâmicas.	Modelo Hipotético, não é realizada nenhuma simulação.
[42]	Especificação	Híbrido	Ataques de Rank; Sinkhole; Vizinho; DIS; Reparo Local	Eficiente em termos energéticos e computacionais; Boa taxa de eficácia; Escalabilidade.	Overhead comunicacional; Decréscimo de eficiência quando operado por longo períodos de tempo.	Contiki OS no simulador Cooja
[77]	Anomalia	-	Ataques convencionais	Leve e eficiente; É possível proceder à expansão do escopo de deteção	Overhead comunicacional.	Cenário real pela análise de pacotes entre uma estação meteorológica e uma câmara vídeo.

Tabela 3.1: Propostas para deteção de ataques sobre o protocolo RPL

3.2 Propostas para deteção de ataque Wormhole

O DAWN [28], trata-se de um IDS distribuído para a deteção de ataques *Wormhole*, cuja principal especificidade passa por este sistema não carecer de informações geográficas dos nós, de suposições de sincronização global ou de qualquer tipo de *hardware* adicional. A metodologia de deteção proposta passa pelo relacionamento da métrica ETX do nó remetente e destinatário durante a troca de mensagens. Numa primeira fase, sempre que um nó recebe um pacote de um outro, cujo ETX é muito superior ao seu, emite um relatório que identifica o nó suspeito. No caso desta situação ser reportada por vários nós vizinhos do nó suspeito, este será isolado. O facto dos relatórios onde são referenciados os nós suspeitos transitarem livremente pela rede, motivou a que os autores propusessem a que estes fossem encriptados por intermédio de técnicas de criptografia simétrica, de modo a não serem comprometidos por atacantes internos. O artigo propõe ainda a simulação do modelo apresentado através de um simulador de eventos discretos, desenvolvido na linguagem C, onde foram obtidas taxas de deteção satisfatórias, mas com um custo de comunicação e computacional elevado.

Em [37] é proposto um IDS distribuído, especialmente desenhado para a deteção de ataques *Wormhole* em redes operadas pelo protocolo RPL. A ideia passa pela análise das mensagens DIO e utilização de informações nelas contidas para o cálculo de dois novos atributos, através dos quais é possível proceder à deteção do agente malicioso. O primeiro desses indicadores, *Rank Diff*, corresponde ao módulo da diferença entre o *Rank* do nó remetente da mensagem DIO e a o *Rank* do nó que a rececionou. O segundo indicador, *Rank Threshold*, corresponde ao módulo da diferença entre o *rank* do nó pai do destinatário e o *rank* do próprio destinatário. Posto isto, sempre que o *Rank Diff* seja superior ao *Rank Threshold* é gerado um alerta, sinalizando o nó remetente envolvido. Esta estratégia apresenta boa precisão, exatidão e *recall*, sendo ainda, relativamente fácil de implementar. No entanto, não são considerados ambientes de deteção dotados de mobilidade e não são, também, avaliados os impactos energéticos e o atraso ponta a ponta provocados sobre a rede onde a presente solução é implementada.

Em [73] são apresentados 3 novos sistemas centralizados, baseados em *Machine Learning*, para a deteção de ataques *Wormhole* em redes IoT, operadas pelo protocolo de encaminhamento RPL. A primeira proposta, KM-IDS, recorre ao agrupamento, inteligente e não supervisionado, dos nós da topologia em vários *clusters*, designados por zonas seguras. Desta forma, no caso de um nó se predispor a ser vizinho de um outro, esta relação apenas se efetiva se ambos pertencerem à mesma zona segura. A segunda proposta recomenda o uso de uma árvore de decisão, baseada em aprendizagem, através da qual é veiculada uma distância segura que deve ser respeitada no estabelecimento de novas relações entre os nós. A terceira abordagem, compreende as duas anteriores numa só, que opera em dois estágios. No primeiro, o agrupamento *K-means* permite a delineação das várias zonas seguras e num segundo estágio é feito uso da árvore de decisão. Esta solução híbrida permite a diminuição dos falsos positivos, fruto da dupla validação, e confere flexibilidade ao modelo. No artigo, é ainda descrita uma avaliação experimental de cada uma das soluções para uma implementação em C++ das mesmas. Posto isto, o KM-IDS apresentou taxas de deteção que variam entre os 70 e 93%, o IDS baseado em árvore de decisão, apresentou taxas de deteção entre 71 e 80% e, por fim, a proposta híbrida, que compreende as duas anteriores, apresentou taxas de deteção entre 71 e 75%, ainda que, para esta última se tenha notado um aumento considerável na precisão, dada a diminuição do número de falsos positivos.

Numa tentativa de mitigar a ocorrência de ataques *Wormhole* sobre redes LLN, em [1] é proposto um esquema de deteção baseado em duas técnicas, *Area Border Router* (ABR)

e *Sensing Aware Nodes* (SAN). O facto de serem utilizadas duas técnicas distintas e independentes aumenta não só a capacidade de deteção como, também, a precisão do modelo. A técnica Area Border Router (ABR), como o próprio nome indica, é operada a partir do Border router (BR), compreendendo uma fase de inicialização onde, por conversão do valor Received Signal Strength Indicator (RSSI), são determinadas as distâncias entre os vários nós constituintes da topologia. Com base neste indicador, sempre que a distância entre dois nós vizinhos seja superior a um limiar, pré-estabelecido, correspondente ao alcance dos dispositivos, é gerado um alerta e solicitado a que os nós vítima procedam à sanitização da sua lista de vizinhos. O método Sensing Aware Nodes (SAN) pressupõe a existência de nós na topologia com características específicas e *hardware* adicional (Global Position System (GPS)), possuindo, estes elementos, conhecimento relativo à sua localização geográfica, e à localização dos seus semelhantes, também eles SAN e com os quais mantêm contacto. A distribuição destes nós pela topologia deve ser homogénea, o que dificulta, muitas vezes, a operacionalização desta técnica. Cada SAN, compreende uma série de nós, ditos normais, que se encontram nas suas imediações e dos quais mantém informações posicionais, resultantes da conversão do valor RSSI, coletado aquando da troca de mensagens com os próprios. Sempre que um SAN envia uma mensagem de verificação, os nós integrantes da sua instância irão responder-lhe. Se no decorrer desta operação o SAN receber uma resposta proveniente de um nó externo ao seu domínio, este solicita informações ao SAN responsável pelo elemento em causa. O facto dos módulos SAN deterem conhecimento relativo à posição exata de outros nós com as mesmas funções e destes possuírem documentada uma estimação da distância aos nós integrantes da sua instância, permite mapear, inequivocamente, um nó malicioso. Para validação desta estratégia os autores propuseram uma simulação do modelo, por intermédio da ferramenta *Cooja*, ao que foi possível apurar alta precisão e uma taxa de deteção entre 90 a 95 %, ainda que, com um *overhead* comunicacional considerável e uma degradação energética notória. Note-se que, o funcionamento pleno deste sistema de deteção requer a utilização de um número considerável de nós SAN, posicionados estrategicamente e dotados de *hardware* adicional e específico para a determinação das suas posições.

No artigo [35], os autores propõem um modelo para inibição de ocorrência de ataques *Wormhole*, baseado em autenticação e em árvores *Merkle*, ainda que, se saiba à priori, que esta abordagem imprime um custo comunicacional considerável com a raiz. A ideia fulcral, por de trás desta proposta, passa por, após estabelecida a estrutura DAG, construir uma árvore *Merkle* que, na prática, é uma cadeia *Hash* de IDs e chaves de elementos filhos para cada relacionamento filho-pai definidos na pré-estabelecida DAG. Desta forma, é criada, como que, uma cadeia de confiança que inviabiliza o ingresso de novos elementos na topologia não respeitantes desta estrutura. A necessidade de utilização de técnicas criptográficas imprime ao sistema de deteção um alto *overhead* comunicacional, que se reflete num significativo atraso ponta a ponta, tendo estes resultados sido obtidos pelos autores numa simulação executada em NS-2.

O DelPHI, proposto em [23], trata-se de um IDS que, através da contagem de saltos e monitorização dos atrasos em caminhos disjuntos entre o emissor e recetor, procede à detenção de ataques *Wormhole*. Repare-se que, em operação normal, a cada salto o atraso na propagação de um pacote é sensivelmente o mesmo ao longo do caminho, o que leva os autores a assumir que em situações de ataque esse atraso por salto seja claramente superior, uma vez que, à partida, as rotas maliciosas incluem um maior número de saltos sendo, habitualmente, a sua contagem adulterada. Esta estratégia, embora válida, apresenta limitações quando os *links* maliciosos são de alta velocidade, ou quando existe congestionamento da rede. Para validação da proposta, foi realizada, pelos autores, uma simulação em LBNL NS onde foi possível apurar taxas de deteção entre os 85 e 95%, ainda que não tenha sido

avaliado o impacto computacional e energético do sistema, quando operado em redes com recursos limitados. Um sistema semelhante foi proposto em [36], compreendendo este um módulo adicional de autenticação por forma aos nós ilegítimos não estarem capacitados a adulterar ou interpretar as mensagens topológicas dos nós legítimos da rede. A nível criptográfico é utilizada a cifra de César por intermédio de segredo partilhado entre os intervenientes. Esta solução, na simulação em NS-2 levada a cabo pelos autores, mostrou resultados semelhantes aos da técnica anterior, no entanto a adição de um módulo confere-lhe um aumento no consumo de recursos e no tempo de execução.

Em [24] é introduzido o conceito de trela para a supressão do ataque *wormhole*. Os autores definem por trela toda e qualquer informação adicionada a um pacote por forma a restringir a propagação do mesmo. No artigo é proposta a utilização de trelas geográficas e temporais para proceder à deteção do ataque. Assim, as trelas geográficas são responsáveis por evitar que a distância entre o remetente e o destinatário ultrapasse um valor pré-definido. Obviamente que esta abordagem obriga a que ambos os nós envolvidos na comunicação sejam dotados de ferramentas que lhes veiculem informações geográficas relativas à sua posição, para que, aquando da receção de um pacote o destinatário esteja capacitado a avaliar se a distância entre si e o remetente é admissível. Por sua vez, as trelas temporais implicam que os pacotes que circulam na rede estejam associados a um tempo limite de vida, ao fim do qual, serão descartados, assumindo que o tempo de propagação de um pacote é proporcional à distância percorrida e assumindo como velocidade máxima de transmissão a velocidade da luz. Este último mecanismo necessita que os relógios dos vários dispositivos constituintes da rede estejam perfeitamente sincronizados, podendo esta particularidade ser vista como um inconveniente. A necessidade de troca de informações geográficas e temporais entre nós levanta mais um desafio, a autenticidade das mensagens. Cientes deste problema os autores propõem a utilização de mecanismos criptográficos leves como o TESLA with Instant Key disclosure (TIK), baseado em criptografia assimétrica e desenhado especificamente, pelos autores, para o efeito. Os dados resultantes da avaliação deste sistema de deteção apontam para uma clara sobre carga do Central Processing Unit (CPU) e da memoria, condicionando a aplicação da presente estratégia em dispositivos mais restritos.

Em [67] é referido um sistema híbrido para deteção de ataque *wormhole* baseado na contagem do número de saltos e Round Trip Time (RTT). Embora esta estratégia tivesse sido já proposta em [61] para redes operadas pelo protocolo de encaminhamento AODV, foi agora, no presente artigo, adaptada para atuar sobre o protocolo de encaminhamento RPL. A estratégia passa pela utilização de módulos distribuídos que coletam o RTT para todos os nós da topologia seus vizinhos por forma a ser calculado o RTT médio, informação esta partilhada, posteriormente, com o módulo central que pelo relacionamento das duas métricas (RTT e Nº de saltos) procede à deteção do ataque, ou seja, para cada rota, multiplica o RTT médio pelo número de saltos necessários a que o pacote atinja o nó destinatário, resultando desta operação o tempo máximo de transmissão para aquele trajeto. Deste modo, sempre que, no decorrer de uma troca de pacotes, por intermédio daquela rota, este valor seja ultrapassado é gerado um alerta. No artigo é documentada uma implementação da proposta, por recurso a *Contiki OS* no simulador *Cooja*, onde são avaliados os impactos do modelo em termos de sobrecarga energética, comunicacional, de memória e de CPU, considerados insignificantes pelos autores. O maior constrangimento do sistema está associado ao facto de as limitações de memória dos dispositivos impossibilitarem a adoção desta estratégia em redes compostas por muitos nós.

O CHA-IDS, proposto em [55] é um dos mais polivalentes IDS na deteção de ataques sobre o protocolo RPL, sendo capaz de proceder à deteção de ataques *Sinkhole*, *Hello Flooding* e *Wormhole*, que ocorram de forma isolada ou combinados. O ponto de partida para a deteção passa pela coleta e análise dos dados contidos nos cabeçalhos comprimidos. A esta fase segue-se e identificação dos recursos mais significativos, os quais serão testados individualmente, quanto à sua legitimidade por 6 algoritmos de *machine learning* distintos (MPL, J48, SVM, *Navie Bayes*, *Logistic* e *Random Forest*). Aquando da deteção de atividade maliciosa, as assinaturas contidas no BR são atualizadas. Em comparação à proposta SVELTE [63], por simulação em *Cooja/Contiki OS* esta é ultrapassada em larga escala em termos de taxa de deteção. Contudo, a proposta em estudo apresenta sérios constrangimentos no que toca ao consumo energético e de memória.

A proposta de Pongle e Chavan apresentada em [58], trata-se de um IDS do tipo híbrido, baseado em anomalias, especificamente desenhado para a deteção de ataques *Wormhole* em redes estáticas. Numa fase inicial, onde se pressupõe a inexistência de quaisquer ataques, é feito um levantamento das distâncias entre nós vizinhos, por conversão do RSSI apurado nas mensagens trocadas, especificamente para este efeito, entres estes. Esta informação ficará armazenada a nível do BR, para que, aquando de um nó reportar uma alteração da sua lista de vizinhos, este possa aferir se o novo vizinho do nó se encontra num raio de transmissão admissível, dado o alcance dos dispositivos. Em caso de inconsistência os dois nós são notificados, desencadeando esta notificação uma troca de mensagens que servirá para estimar a distância que os separa do nó malicioso, a fim de delinear um raio no qual é provável que os nós atacantes se encontrem. Note-se que este mecanismo não está capacitado a determinar quais os pacotes que transitaram pelo túnel malicioso e que a deteção dos nós, também eles maliciosos, é algo morosa e por vezes pouco assertiva, se existirem discrepâncias entre o valor estimado para a distância entre nós e o valor real da mesma. Além disso, se os nós atacantes se encontrarem numa zona onde exista uma forte densidade de nós legítimos é difícil aferir com exatidão o autor do ataque. Esta proposta foi validada por recurso à ferramenta *Cooja* que simulou nós *TmoteSky* a operar sobre o protocolo RPL. Como resultado foi notório um baixo consumo energético e de memória, ainda que, anexo de um *overhead* comunicacional considerável. As maiores condicionantes do modelo prendem-se com a alta taxa de falsos positivos e baixa precisão, aliadas à não consideração de mobilidade dos nós e à diminuta variedade de tipos de ataque detetados.

A tabela 3.2, sumariza as principais características, pontos positivos e negativos de cada um dos modelos de deteção de ataque *Wormhole* abordados.

Artigo	Esquema de deteção	Arquitetura	Ataques	Vantagens	Desvantagens	Estratégia de avaliação
[28]	Especificação	Distribuído	Wormhole	Boas taxas de deteção	Custo comunicacional e computacional elevando, em parte devido à utilização de técnicas criptográficas.	Simulador de eventos discretos em C.
[37]	Especificação	Distribuído	Wormhole	Consegue detetar variados tipos de ataque; Não carece de hardware adicional; Fácil de implementar.	Não é funcional para nós dotados de mobilidade; Não é realizada uma analise de performance.	-
[73]	Assinatura	Centralizado	Wormhole	Boa precisão no modelo híbrido.	Alta taxa de falsos positivos, embora esta diminua na proposta híbrida tendo um impacto positivo na precisão; Na proposta híbrida a taxa de deteção baixa consideravelmente.	C++
[23]	Anomalia	Híbrido	Wormhole	Boa taxa de deteção em cenários específicos.	Consumo energético e computacional; Inviável na deteção de ataques quando realizados por intermédio de ligações de alta velocidade; O congestionamento da rede pode provocar, indevidamente, um alerta.	NS-2
[24]	Anomalia	Distribuído	Wormhole	Boas taxas de deteção; Boa precisão; Redundância na deteção	Carece de hardware adicional seja para a sincronização dos relógios ou para a determinação da posição geográfica dos nós; Sobrecarga dos nós com recursos limitados.	-
[67]	Anomalia	Híbrido	Wormhole	Fácil de implementar; Boas taxas de deteção e precisão em ambientes específicos.	Utilização inviável em redes compostas por muitos nós devido às típicas limitações de memoria destes componentes; Falha em ataques direcionados por intermédio de ligações de alta velocidade; Decréscimo funcional em redes congestionadas.	Contiki OS no simulador Cooja
[58]	Anomalia	Híbrido	Wormhole	Eficiência Energética e computacional;	Baixa eficácia; Alta taxa de falsos positivos; Deteção de uma única tipologia de ataque; Não deteta os pacotes que transitaram por intermédio de um túnel malicioso.	Simulação em contiki OS/ Cooja com nós Telos B
[55]	Híbrido	Centralizado	Sinkhole; Wormhole; Hello Flooding.	Eficiência na deteção; Vasto escopo de tipologias de ataque detetáveis; Atualização dinâmica das assinaturas em BR; Deteção de ataques isolados e combinados	Constrangimentos a nível energético e de memoria	Contiki OS no simulador Cooja
[1]	Anomalia	Híbrido	Wormhole	Redundância na deteção de ataques e, portanto, mais precisão e resiliência a falhas.	Dificuldade na distribuição homogénea dos nós SAN pela topologia.	Contiki OS no simulador Cooja
[35]	Anomalia	Híbrido	Wormhole	Alta taxa de deteção e de precisão	A utilização de técnicas criptográficas imprime na rede atraso ponta a ponta e overhead comunicacional em particular com o nó raiz	NS-2

Tabela 3.2: Síntese das propostas para deteção de ataque Wormhole

3.3 Síntese de capítulo

A título de conclusão, ficou claro que embora exista um vasto leque de modelos para a deteção de ataques sobre o protocolo RPL, nenhum deles está capacitado à identificação integral de todas as tipologias de ataque existentes. Isto porque, os vetores de ataque a estas associados são extremamente desuniformes. Desta feita, também as evidencias deixadas por cada um dos ataques apenas são percetíveis pela avaliação de diferentes informações topológicas e protocolares. Ora, a monitorização constante de todas estas implica uma sobrecarga inaceitável sobre os dispositivos da rede. No domínio da variedade de ataques detetáveis, destacam-se as propostas [55], [63], [72], capazes de proceder à deteção de 3 diferentes tipos de ataque a par da proposta [42], cujo modelo descrito está qualificado a identificar 5 diferentes tipos de ataque.

Relativamente ao ataque *Wormhole*, foi possível apurar a existência de diversas técnicas para a sua deteção, podendo estas ser agrupadas em 3 grupos distintos.

No primeiro desses grupos inserem-se todas as soluções que se baseiem na localização geográfica dos nós da topologia. A utilização deste indicador é possível, em grande parte das soluções propostas, por inclusão de *hardware* GPS específico [24]. Já noutras, é utilizada a conversão do valor RSSI em distância [58] [1]. Ainda que esta última técnica não implique *hardware* adicional, a precisão do indicador não é tão satisfatória. No segundo grupo, inserem-se todas as estratégias que façam uso de técnicas criptográficas [23] [35], esta abordagem imprime um custo computacional considerável sobre os nós altamente restritos, uma vez que, as técnicas criptográficas existentes não foram projetadas para operar a nível deste tipo de dispositivos. No último grupo inserem-se as restantes estratégias, cujo esquema de deteção passa pela representação da topologia, tipicamente a nível do BR, ou por aglomeração dos nós constituintes em *Clusters* [73] [1]. Esta abordagem, embora viável, provoca um acréscimo comunicacional entre os nós líderes de *cluster*.

Por outro lado, enquanto que algumas das técnicas dão especial enfoque à validação das mensagens DIO outras procuram monitorizar as comunicações da topologia na integra. Obviamente que coletar e processar todos os pacotes de uma rede provoca um maior *overhead* computacional e maior atraso ponta a ponta, no entanto, este método de monitorização mostra-se mais viável na deteção de ataques *Wormhole*, isto porque, é possível a deteção de mais derivações deste ataque e de forma mais ágil, nomeadamente no que diz respeito à identificação dos nós maliciosos, não sendo, ainda assim, estas propostas isentas de limitações provocadas por congestionamentos na rede [67] [23] ou adulterações protocolares [37] [28].

As tabelas 3.1 e 3.2, esquematizam, de forma resumida, cada uma das propostas estudadas. Sendo a primeira relativa aos modelos existentes para a deteção da generalidade dos ataques sobre o protocolo RPL e a segunda, incidente, especificamente, na deteção do ataque Wormhole, são um excelente ponto de comparação das várias técnicas presentes na literatura.

Da perspectiva da estrutura geral do documento, o capitulo que agora finda e anteriores, permitem a contextualização do leitor perante a problemática da ocorrência de ataques sobre o protocolo de encaminhamento em redes LLN, com especial destaque para o ataque *Wormhole* associado ao protocolo RPL. Embora tenham sido referenciadas algumas propostas de segurança, ficou claro que nenhuma destas é dotada de total eficácia e eficiência na deteção deste tipo de ataque, tendo sido, precisamente, esta a motivação ao desenvolvimento do sistema de deteção de ataques *Wormhole* que de seguida será apresentado.

Esta página foi propositadamente deixada em branco.

Capítulo 4

Sistema de deteção proposto

Durante o estudo das propostas existentes para a deteção de ataques *Wormhole*, ficou claro que cingir o esquema de deteção à avaliação das condições de envio de mensagens DIO restringe as modalidades deste tipo de ataque detetáveis, propostas em 2.7.9. Repare-se que embora grande parte dos ataques *Wormhole* sejam direcionados através da partilha fraudulenta de mensagens DIO, provenientes da extremidade oposta do túnel malicioso, isto nem sempre é prática obrigatória. Se dois nós maliciosos partilharem pacotes, mesmo sendo estes de dados, entre as extremidades de um túnel, é suficiente para que os nós de ambas a regiões fiquem convencidos de que são vizinhos.

Como motivação, é importante clarificar que de um cenário de ataque deste tipo podem resultar uma série de situações potencialmente comprometedoras do normal funcionamento da rede. Num primeiro ponto, a partir do momento em que os nós se associam indevidamente a um outro geograficamente distante é expectável que os pacotes que sejam difundidos por intermédio dessa relação topologia fraudulenta sofram um atraso na sua entrega. Paralelamente, dependendo da periodicidade deste tipo de ataques, se os nós vítima forem forçados a sistematicamente alterar as sua tabelas de encaminhamento, fruto da alteração dos seus vizinhos, torna-se evidente o aumento de recursos envolvidos nesta atividade, sendo estes altamente restritos neste tipo de dispositivo. Por outro lado, e associado ao ponto anterior, não podemos descorar o facto de nas tarefas de reorganização topológica estar implícito um incremento do fluxo de pacotes na rede, podendo este ser suficiente quer para provocar atrasos generalizados na propagação dos pacotes legítimos quer para, em situação limite de congestionamento, levar ao colapso da rede.

Obviamente que o próprio mecanismo de deteção imprime um custo temporal nas comunicações internas da rede, contribuindo igualmente para o aumento do consumo energético dos nós. No entanto, como em todas as soluções de segurança, é necessário proceder a uma avaliação consciente do custo benefício da adoção da estratégia, balanceado o decréscimo na prestação, resultante da inclusão do IDS, com benefício que deste advém, na deteção de ataques cujo impacto é, potencialmente, mais nefasto para a topologia.

No presente capítulo será proposto um IDS em tempo real para a deteção de ataques *Wormhole*. Trata-se de uma abordagem híbrida, baseada em assinatura e que direciona a sua monitorização não só a mensagens DIO mas a todo o tráfego que circula na rede em que opera. Desta forma, além de identificados mais ataques dentro da classe Wormhole, é possível referenciar os pacotes que transitaram no túnel e, inerentemente, com maior facilidade se consegue proceder à identificação dos agentes maliciosos envolvidos na operacionalização do mesmo.

4.1 Pressupostos técnicos

Na impossibilidade de desenvolver uma estratégia de deteção de ataques *Wormhole* transversal a todos os ambientes aplicacionais da IoT, o sistema proposto parte dos seguintes pressupostos:

- Rede estática - não são considerados ambientes cujos nós constituintes sejam dotados de mobilidade. Esta imposição deriva do facto de a estratégia de deteção do sistema proposto ter por base a distância aproximada dos nós vizinhos. Denote-se que no caso de se assumir a mobilidade dos nós, sempre que um destes alterasse a sua posição, estava implícita uma atualização das distâncias documentadas pelo IDS. Esta tarefa, por sua vez, imprimiria, sobre a rede, um elevado custo computacional, comunicacional e energético, altamente indesejáveis neste tipo de ambiente, dadas as limitações dos componentes.
- Operada pelo protocolo RPL em *Non Storing Mode* - Sendo o IDS em desenvolvimento baseado numa arquitetura híbrida, é vantajoso que as tarefas que careçam de mais recursos sejam realizadas a nível do BR, visto este componente ser, tipicamente, sujeito a menos restrições. Por conseguinte, a obrigatoriedade do RPL operar em *Non Storing Mode* origina a que, como referido na secção 2.7.6, a maior parte do fluxo de dados passe pela raiz, neste caso o BR. Além disso, a monitorização dos pacotes neste ponto da rede permite uma prévia deteção de ataques *Wormhole*.
- Período de inicialização isento de ataques - A coleção das distâncias aproximadas entre nós vizinhos apenas é possível após estabelecido o grafo de encaminhamento por intermédio do RPL. Repare-se que só depois desta ação cada um dos nós integrantes da topologia conhecerá a sua posição hierárquica no grafo de encaminhamento e as várias relações detidas com outros nós, nomeadamente o conhecimento dos seus vizinhos. Sendo este indicador de extrema importância, é vital que aquando do seu levantamento este seja autêntico. Caso contrário, o sistema de deteção iria basear-se em dados fraudulentos, sendo complacente com eventuais futuros ataques.

Devo frisar que grande parte destes pressupostos apresentam uma ínfima componente restritiva do ponto de vista aplicacional, uma vez que, em múltiplos domínios, a validação dos mesmos é meramente configuracional, não apresentado qualquer impacto funcional no sistema. Na secção 5.1 é apresentado um cenário real de aplicação, não obstante, ainda assim, da existência de muitos outros.

4.2 Arquitetura e funcionamento geral

O mecanismo de deteção proposto é baseado numa arquitetura híbrida, isto porque, a deteção dos ataques contempla atividades desenvolvidas a nível dos nós distribuídos e correlacionadas com outras executadas no domínio do BR. Relativamente ao método de deteção, este é baseado em assinaturas, sendo estas representativas das inconsistências geradas pelo ataque e associadas à distância percorrida pelos pacotes em trânsito.

Numa primeira fase, aquando da inicialização da rede e após estabelecido o grafo de encaminhamento, por intermédio do protocolo RPL, é iniciada a coleta da distância entre nós vizinhos, criando, localmente, em cada um dos nós, uma tabela na qual cada entrada compreende a identificação de um vizinho e a distância até este. De modo a isentar a necessidade de inclusão, nos dispositivos, de *hardware* GPS específico, esta distância é

estimada por conversão do valor RSSI, aferido no conjunto de mensagens trocadas, especificamente para este efeito, entre cada par de nós vizinhos, durante a fase de inicialização. Em alternativa, o preenchimento destas estruturas pode ser feito manualmente, visto as redes onde o sistema de deteção opera serem estáticas, o que implica a imutabilidade das tabelas durante o período de funcionamento do mesmo.

De forma análoga, também o BR possuirá uma estrutura onde constam as distâncias a cada um dos seus vizinhos, no entanto, adicionalmente, esta estrutura incluí as distâncias aos restantes nós integrantes da rede. Note-se que a distância a estes últimos não é medida de forma linear, mas pelo somatório das distâncias que separam os nós intermediários vizinhos contemplados no trajeto entre o BR e o nó em foco. De forma prática, se a rota entre um nó C e o BR incluir 3 saltos (C-B-A-BR), a distância entre remetente e destinatário corresponde ao somatório da distância entre C e B, B e A e A e BR, estando estas previamente definidas nas tabelas locais dos nós envolvidos.

Devo frisar que a presente abordagem apenas é possível pelo facto do IDS desenvolvido operar, unicamente, em redes estáticas. Além disso, é perfeitamente admissível que na tabela de distâncias do BR existam mais que uma entrada alusiva ao mesmo nó, significando, apenas, que existem várias rotas legítimas e paralelas para atingir esse elemento. Assume-se que esta fase de inicialização é executada em condições controladas, nas quais a rede não é vítima de qualquer ação maliciosa, só deste modo é possível garantir a solidez e autenticidade dos dados recolhidos.

Findando a fase de inicialização, o IDS estará apto a proceder à monitorização da rede onde foi instalado. Para este efeito, são adicionados a todos os pacotes que nela circulam dois novos campos, correspondentes a dois novos indicadores, "distância percorrida" e "variável de estado". O primeiro deles é incrementado a cada salto com a distância que separa os dois nós envolvidos no mesmo, estando esta definida nas tabelas locais acima caracterizadas. Já o segundo, embora nativamente definido a "0", é modificável pelos nós intermediários, sempre que estes detetam, no momento da receção do pacote, que este não lhe foi remetido por nenhum dos seus vizinhos. Nestas situações o presente campo é alterado por forma a tomar o valor "1", sendo o pacote, de seguida, remetido para o domínio do BR. Note-se que nestas condições, no trajeto até ao elemento central, os nós intermediários não devem alterar qualquer campo no pacote vítima, devendo apenas proceder às atividades protocolares necessárias ao seu encaminhamento.

Por sua vez, quando o BR recebe um pacote cujo campo "variável de estado" está definido a "1", significa que é muito provável que este tenha sido alvo de um ataque. Posto isto, este elemento é responsável pelo relacionamento da distância percorrida pelo pacote, incluída no campo apropriado da estrutura, com a distância a que se encontra do remetente original do mesmo, definida na sua tabela local, a fim de determinar a distância que o separa do nó atacante e assim identificá-lo.

$$\text{Distância ao nó malicioso} = |\text{Distância ao remetente} - \text{Distância percorrida pelo pacote}| \tag{4.1}$$

A expressão 4.1 representa, precisamente, a operação matemática, realizada no domínio do BR, para determinação da distância a que se encontra do nó malicioso. Repare-se que com este indicador é possível realizar uma consulta na tabela local e encontrar a entrada correspondente que permite a identificação inequívoca do atacante.

Todavia, também os pacotes com o campo "variável de estado" definido a "0" são processados no domínio do BR, no entanto, nestas situações, não é realizada a operação de

identificação do nó malicioso, uma vez que nada indicia que o pacote em processamento tenha sido sujeito a um ataque. Ainda assim, de forma similar ao que é feito pelos nós distribuídos, é verificado se o pacote recebido foi remetido por um dos vizinhos do BR, bem como, atualizada a distância percorrida pelo pacote, no caso BR não ser o seu destinatário final.

Note-se que, do ponto de vista funcional do modelo, a necessidade de operação do protocolo de encaminhamento RPL em *Non Storing Mode*, prende-se com o facto de só assim ser possível a identificação dos nós maliciosos. Isto porque, neste modo de operação é sabido, à priori, que independentemente dos nós remetente e destinatário envolvidos na transação de um pacote, o seu trajeto compreende obrigatoriamente duas etapas. Na primeira, o pacote é difundido do remetente até ao BR, e na segunda, este é remetido do BR até ao nó destinatário, fruto da centralização do conhecimento das rotas descendentes no domínio do nó raiz, neste caso o BR, como clarificado em 2.10. Esta ocorrência apenas não se verifica na situação especifica em que o destinatário seja atingido no primeiro segmento do trajeto, ou seja, na fase ascendente.

Da perspetiva da tipologia de ataques *Wormhole* detetáveis, se na modalidade por retransmissão de pacotes as ações acima descritas procedem à identificação integral dos agentes maliciosos envolvidos, na modalidade por encapsulamento é sabido que, tipicamente, cada ataque contempla dois atacantes. Desta feita, nesta situação, cada um dos elementos é identificado de forma independente.

Embora os ataques *Wormhole* sejam normalmente direcionados por agentes externos à rede, a partir do momento em que estamos a incluir novos campos nos pacotes sabemos que também a informação a estes anexa pode ser alvo de adulteração, por intermédio de outras tipologias de ataque referenciadas em 2.7.9. Este tipo de incidente pode, obviamente, comprometer a solidez do modelo de deteção proposto, ainda que, dificilmente este se torne complacente com ataques. Para isso, seria necessário que o atacante detivesse conhecimento de todas as estruturas anexas ao IDS, implicando o comprometimento de todos os nós da topologia. Mesmo assim, existem na literatura diversas propostas para supressão deste tipo de ataque, seja pela aplicação de soluções criptográficas nos pacotes [74], ou por configuração dos modos de segurança nativos do RPL, abordados em 2.7.8.

Conversão da força do sinal em distância

Como já explicado, por forma a aferir a distância entre dois nós vizinhos, sem que esta operação implique a inclusão de *hardware* adicional nos dispositivos, recorreu-se à conversão do valor RSSI em distância.

Esta operação é feita através da formula matemática apresentada em 4.2, onde "d" representa a distância que separa emissor e recetor, "A" o valor absoluto do RSSI, providenciado pelo fabricante dos dispositivos, e n o coeficiente de atenuação que varia entre 1.6 e 6, de acordo com as condições do meio [7].

$$d = 10^{\frac{-\ RSSI\ -\ A}{10\ *\ n}} \tag{4.2}$$

Note-se que a distância obtida se trata de uma estimação, pelo que, é expectável a existência de desvios entre o valor real da distância e o valor estimado para a mesma, resultantes da atenuação da força do sinal, provocada pelo meio e por quaisquer obstáculos que neste se incluam. No entanto, no modelo de deteção proposto este tipo de inconsistência torna-se admissível, visto que as distâncias estimadas são assumidas como absolutas durante toda a operação do IDS.

Campos adicionais nas mensagens

A necessidade de inclusão de dois novos indicadores nos pacotes que circulam na rede obrigou a uma reflexão sobre qual das estruturas da pilha protocolar melhor se enquadraria nesta operação. Após ponderadas as várias hipóteses, a escolha passou pela adição dos campos no cabeçalho de extensão do tipo *Hop-by-Hop* do pacote IPv6 [14].

Num primeiro ponto, é importante perceber que esta estrutura é processada numa zona da pilha protocolar que inclui, paralelamente, a atividade associada ao protocolo de encaminhamento RPL, admitindo a utilização preferencial do modo *route-over* por parte dos sistemas operativos vocacionados para dispositivos IoT, como clarificado em 5.2.1.

Num segundo ponto, os cabeçalhos em causa são especificamente empregues em situações onde o processamento das informações nestes contida deva ser realizado por todos os nós visados no decurso da distribuição de um pacote.

Por fim, o recurso aos pacotes IPv6, designadamente aos seus cabeçalhos de extensão, restringe a atividade do IDS à camada de rede, motivando a que a operação do mecanismo de segurança seja completamente transparente e independente da camada aplicacional, promovendo, nesta última, uma maior flexibilidade na escolha do protocolo a utilizar.

A estrutura e mais detalhes do pacote IPv6 e seus respetivos cabeçalhos poderá ser consultada em 2.6. No que concerne à forma como foi efetuada a adição dos campos nesta estrutura, esta será minuciosamente explicada na secção 6.

4.2.1 Módulo centralizado

No domínio do BR encontram-se definidas três funções de extrema relevância no funcionamento do sistema de deteção proposto.

Uma delas relaciona-se com a fase de inicialização deste módulo e é responsável pela definição da estrutura auxiliar a si associada.

As duas outras, têm como função a monitorização e atualização dos pacotes, assim como, a identificação dos nós maliciosos.

Função de inicialização central

No início da operação do IDS é necessário definir, a nível do BR, a tabela com a distância a cada um dos nós na topologia. Como tal, a presente função aguarda o envio de pacotes por parte dos nós distribuídos a fim de, com base no campo "distância percorrida", destes integrante, preencher a respetiva entrada alusiva ao remetente.

Adverte-se que esta operação pressupõem que os nós distribuídos tenham já a suas estruturas auxiliares devidamente preenchidas.

A Figura 4.1 esquematiza o pseudocódigo desta função, ajudando na sua compreensão.

```
Função de inicialização BR(er){

    er # número de nós integrantes da rede
    T[er]; # tabela onde ficarão armazenadas as distâncias a cada um dos nós contidos na rede,
             a cada entrada corresponde ip e respetiva distância
    cronometro = 180
    contador = 0;
    inicia cronometro();
    enquanto cronometro != 0{
        enquanto contador != er{
            quando recebe um pacote{
                se o remetente não estiver em T{
                    T[contador].ip = ip remetente;
                    T[contador].distância = distância percorrida pacote
                    contador++;
                }
                se o contador == er{
                    imprime mensagem sucesso();
                    exit();
                }
            }
        }
        imprime mensagem de erro();
    }
}
```

Figura 4.1: Lógica para a função de inicialização do módulo centralizado

Função de monitorização central

À semelhança do que acontece nos módulos distribuídos, também o BR procede à atualização dos campos "distância percorrida" nos pacotes que por ele passam, sendo esta ação executada por intermédio da presente função. Adicionalmente, também nesta fase, é verificado se o pacote recebido é proveniente de um dos seus vizinhos, sendo que, no caso de não ser, ou ainda, se o campo do pacote "variável de estado" estiver definido a "1", é chamada a função responsável pela identificação de nós extremidade.

A imagem que se segue, 4.2, é alusiva ao pseudocódigo deste componente operacional, permitindo o entendimento de todas as ações, por si executas.

```
Função monitorização BR(){
    T # tabela com a distância a cada um nos nós inseridos na rede

    quando recebida uma mensagem{
        se variavel_estado == 0{
            se o remetente da mensagem for vizinho{
                se o BR for destinatario final do pacote{
                    exit();
                }
                nexthop = determina nexthop();
                para cada elemento em T{
                    se elemento.ip == nexthop{
                        distância percorrida += elemento.distância;
                        exit();
                    }
                }
            }
            se o remetente da mensagem não for vizinho{
                Função deteção nós extremidade BR(id remetente, distância percorrida);
                imprime mensagem de advertência();
            }
        }
        se variavel_estado == 1{
            Função deteção nós extremidade BR(id remetente, distância percorrida);
        }

}
```

Figura 4.2: Lógica para a função de monitorização do módulo centralizado

Função de deteção de nós extremidade

Sempre que é executada esta função está implícito que o pacote em processamento foi alvo de um ataque. Isto porque, esta é responsável por identificar os nós atacantes envolvidos no mesmo.

Numa primeira fase, é percorrida a estrutura que contém a distância a todos os elementos da rede, a fim de determinar a distância a que o BR se encontra do remetente do pacote. Com base nesta informação, e por relacionamento com a distância percorrida pelo pacote, é determinada a distância a que o atacante se situa. Por sua vez, com este indicador, é possível percorrer novamente a estrutura anterior, encontrar a entrada correspondente e reportar a identidade do agente malicioso.

À semelhança do que foi feito nas outras funções, é apresentado na figura 4.3 o pseudocódigo associado a esta função.

```
Função deteção nós extremidade BR(ip remetente, distância percorrida){
    T# tabela com a distância a cada um nos nós inseridos na rede

    para cada elemento em T{
        se elemento.ip = ip remetente{
            float distância remetente = elemento.distância;
            break;
        }
    distância no malicioso = mod(distância no malicioso - distância remetente);
    para cada elemento em T{
        se elemento.distância = distância no malicioso{
            id nó malicioso = elemento();
            break();
        }
    }
    imprime mensagem a identificar o nó malicioso(id nó malicioso);
}
```

Figura 4.3: Lógica para a função de deteção de nós extremidade do módulo centralizado

4.2.2 Módulos distribuídos

Estando neste sistema de deteção os módulos distribuídos associados a cada um dos nós da rede, à exceção do BR, estes têm um papel fundamental na definição das estruturas de suporte à deteção (tabelas de distâncias), monitorização dos eventos ocorrentes na topologia e, inerentemente, na deteção de ataques. Desta feita, dos nós distribuídos são integrantes 2 funções.

Uma delas, é utilizada no preenchimento das tabelas com a distância a cada um dos vizinhos, sendo que, apenas é executada num momento inicial e quando a definição das estruturas auxiliares destes módulos não seja feita manualmente.

A segunda função, tem por objetivo a atualização do indicador "distância percorrida" nos pacotes em trânsito, assim como, a averiguação da legitimidade do remetente no envio de pacotes, na medida em que este tem que ser, obrigatoriamente, seu vizinho.

Função de inicialização local

Ao longo de toda a operação do IDS, é imprescindível que a estrutura individual de cada nó distribuído, alusiva à distância aos seus vizinhos, esteja devidamente preenchida. Como

tal, esta função é responsável por, no momento de inicialização do IDS, determinar a distância a cada um dos nós vizinhos e documentá-la na respetiva tabela de distâncias local. Para isso, com base na lista de vizinhos providenciada pelo protocolo RPL, são trocadas mensagens a fim de estimar a distância entre estes, por conversão do valor RSSI.

Note-se que é vital que dois nós vizinhos documentem a mesma distância para o trajeto entre ambos. Mais uma vez, se o gestor de sistema optar por definir manualmente estas estruturas a presente função não carece de ser executada.

A imagem 4.4 apresenta o pseudocódigo desta função.

```
Função de inicialização Nós(Tdag,p){
    nv # número de vizinhos
    LV # lista de vizinhos associada ao protocolo RPL
    TV[nv] #tabela onde ficarão armazenadas as distâncias a cada um dos nós vizinhos do nó,
             a cada entrada corresponde ip e respetiva distância

    i=0;
    para cada elemento em LV{
        TV[i].ip = elemento;
        envia pacote em branco(elemento);
        i++;
    }
    quando recebe um pacote{
        se o pacote esta em branco{
            para cada elemento e TV{
                se elemento.ip == remetente && elemento.distância != NULL{
                    envia pacote(remetente, elemento.distância);
                }
                se elemento.ip == remetente && elemento.distância == NULL{
                    distância elemento = calcula distância(rssi);
                    envia pacote(remetente, distância elemento);
                }
            }
        }
        se o pacote não está em branco{
            para cada elemento e TV{
                se elemento.ip == remetente{
                    elemento.distância = campo distância percorrida do pacote;
                }
            }
        }
    }

}
```

Figura 4.4: Lógica para a função de inicialização dos módulos distribuídos

Função de monitorização local

A presente função possui, neste contexto, múltiplas funcionalidades. Num primeiro ponto, aquando da receção de um pacote, é responsável por determinar se o nó que lho enviou é seu vizinho, para, no caso de não o ser definir o campo "variável de estado" no pacote com o valor "1" e, de seguida, remeter este elemento para o domínio do BR.

Por sua vez, se o nó remetente for legitimo, ou seja, seu vizinho, a função procede à verificação do campo "variável de estado" pois no caso de este se encontrar definido a "1" nenhuma ação deve ser executada sobre o pacote e este apenas deve seguir caminho rumo ao BR.

Por fim, se o pacote tiver sido remetido por um nó legitimo e se o seu campo "variável de estado" for igual a "0", no caso do nó que o está a processar não ser o destinatário final da mensagem, este deve determinar o *nexthop* e proceder à atualização do campo "distância percorrida" com a distância correspondente.

Todas estas atividades encontram-se esquematizadas no pseudocódigo da função representado na figura 4.5.

```
função de monitorização dos nós distribuídos(){
    TV # tabela local com as distâncias a cada um dos vizinhos

    quando recebida uma mensagem{
        se o remetente da mensagem for vizinho{
            se variavel_estado == 0{
                se o nó for destinatario final do pacote{
                    exit();
                }
                nexthop = determina nexthop();
                para cada elemento em TV{
                    se elemento.ip == nexthop{
                        distância percorrida += elemento.distância;
                        exit();
                    }
                }
            }
            se variavel_estado == 1{
                envia pacote para o BR();
            }
        }
        se o remetente da mensagem não for vizinho{
            variável estado = 1;
            envia pacote para o BR();
        }
    }
}
```

Figura 4.5: Lógica para a função de monitorização dos módulos distribuídos

A imagem 4.6 representa, esquematicamente, a arquitetura do sistema de deteção proposto, distinguindo os módulos distribuídos do módulo centralizado assim como as funções integrantes de cada um destes.

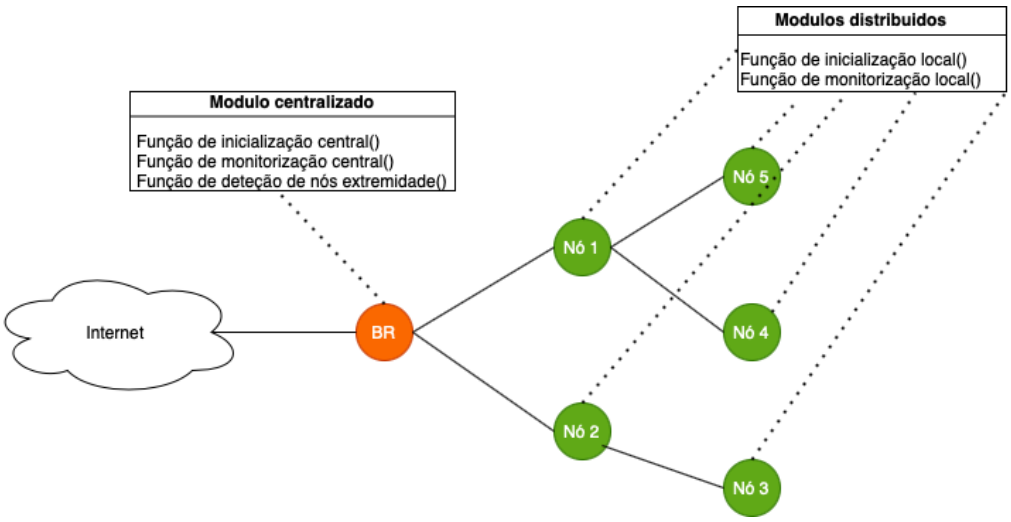

Figura 4.6: Esquematização da arquitetura do sistema de deteção proposto

4.3 Exemplificação teórica

Para esta exemplificação, assuma-se uma rede LLN estática, operada pelo protocolo de encaminhamento RPL em *Non Storing Mode*, de acordo com os requisitos definidos em 4.1. A topologia em causa é constituída por 14 nós, correspondendo o nó identificado com o número 1 ao BR.

De modo a facilitar a compreensão do funcionamento do sistema proposto, considerar-se-á que este já se encontra inicializado e, portanto, seja as estruturas anexas aos nós distribuídos, seja a estrutura auxiliar inerente ao módulo centralizado estão já definidas, como demonstra a imagem 4.7.

Tendo o exemplo por objetivo o entendimento da estratégia de deteção e identificação dos nós maliciosos, os elementos 2 e 12 representam um ataque Wormhole do tipo por encapsulamento, o que implica que estes nós operacionalizam entre si um túnel, a partir do qual direcionam o tráfego legitimo da rede, como apresentado na imagem 4.8.

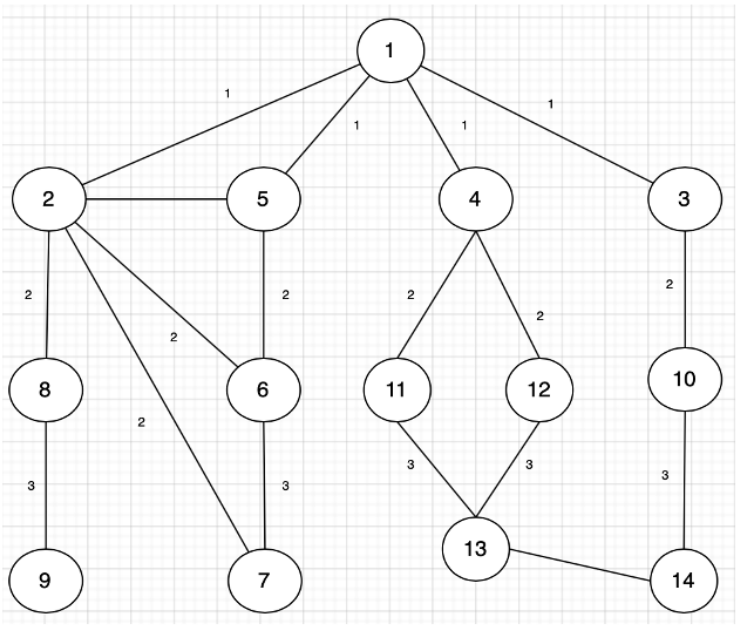

Tabela geral Nó 1 (BR)	Distância
2	4.5m
3	4.75m
4	6m
5	5m
6	8m
7	9.5m
8	10m
9	14m
10	11.25m
11	9.5 m
12	8.5 m
13	9.5 m
14	10 m

Tabela local Nó 2	Distância
1	4.5m
6	4.2m
7	5m
8	5.5m
5	4.5m

Tabela local Nó 8	Distância
2	5.5m
9	4m

Tabela local Nó 9	Distância
8	4m

Tabela local Nó 7	Distância
2	5m
6	3.5m

Tabela local Nó 13	Distância
11	1.5 m
12	1 m
14	0.5

Tabela local Nó 6	Distância
2	4.2m
5	3m
7	3.5m

Tabela local Nó 3	Distância
1	4.75m
10	6.5m

Tabela local Nó 11	Distância
4	3.5m
13	1.5m

Tabela local Nó 10	Distância
3	6.5m
14	4m

Tabela local Nó 12	Distância
4	2.5m
13	1 m

Tabela local Nó 4	Distância
1	6m
11	3.5m
12	2.5m

Tabela local Nó 5	Distância
1	5m
2	4.5m
6	3m

Tabela local Nó 14	Distância
10	4m
13	0.5m

Figura 4.7: Exemplo da topologia e das estruturas auxiliares do IDS

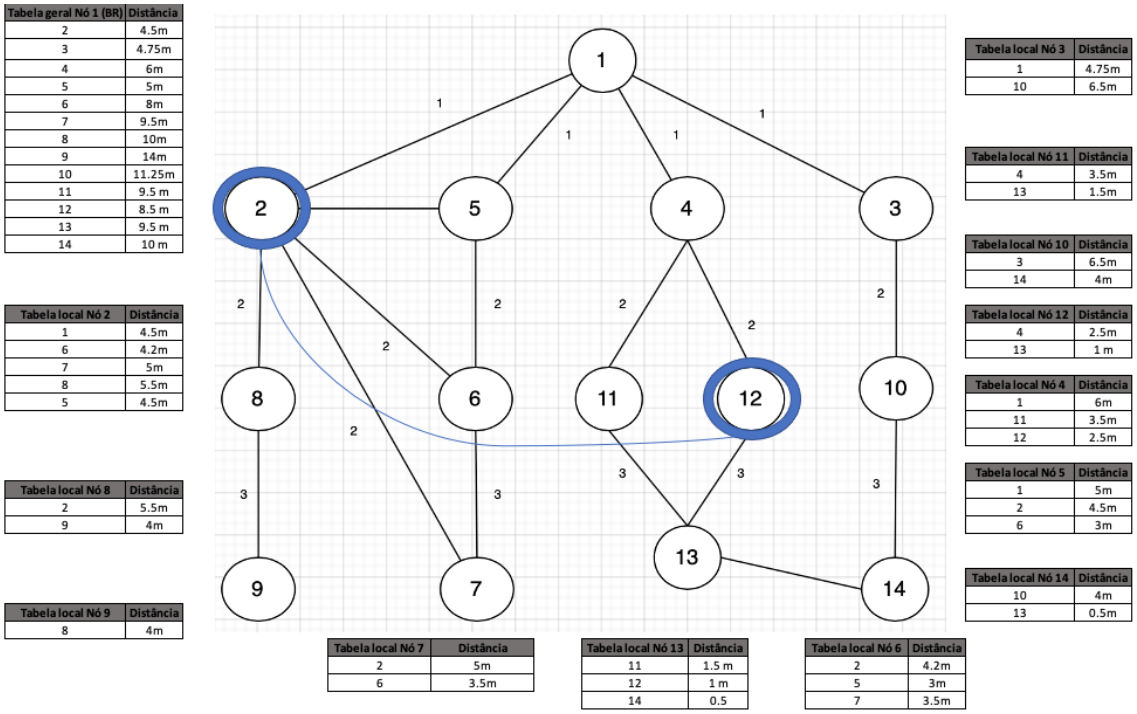

Figura 4.8: Representação de um túnel malicioso entre os nós 2 e 12 da topologia

Monitorização

Suponha-se que de forma legítima o nó 9 pretende enviar uma mensagem para o nó 13. Desta forma, em circunstâncias normais, o nó 9 começa por remeter o pacote original ao nó 8, incrementando, no campo desta estrutura dedicado ao efeito, a distância correspondente a este primeiro salto, 4 metros.

De forma semelhante, o nó 8, no momento de receção do pacote, irá proceder, mais uma vez, à atualização do campo distância percorrida, incrementando-o com o valor 5.5m, correspondente à distância que o separa do nó 2, sendo este o elemento atingível no decurso do próximo salto, para onde, findando esta tarefa, o pacote será direcionado.

Repare-se que quando o pacote atinge o nó malicioso (2), a distância presente no campo distância percorrida é de 9.5m. No entanto, tratando-se o nó 2 de uma das extremidades envolvidas na operacionalização de um ataque *Wormhole* por encapsulamento do tipo oculto, não efetuará qualquer atualização sobre o pacote, remetendo-o, unicamente, para a outra extremidade do túnel previamente criado, ou seja, para o nó 12. Este, por sua vez, fará a distribuição do pacote a um dos seus vizinhos, considerando-se, neste exemplo, que esta foi feita ao nó 4, ainda que, também no decorrer desta ultima transação não tenha sido feita qualquer atualização dos campos do pacote em trânsito. Já o nó 4, quando receciona o pacote vítima constata que este lhe foi remetido pelo nó 8, tendo este sido o último nó legitimo a intermediar o pacote na outra extremidade do túnel. O facto de o nó 8 e o nó 4 não serem vizinhos, motiva a que este último altere o campo do pacote "variável de estado" para o valor "1" e o remeta de seguida para o BR, dados os fortes indícios de ocorrência de ataque.

Identificação dos nós extremidade

No momento da receção do pacote vítima, transacionado pelo nó 4, o BR imediatamente constata que o campo "variável de estado" está definido a "1", suscitando a que a função de identificação de nós extremidade seja imediatamente executada, sendo a sua primeira ação a pesquisa na tabela local pela distância a que o nó 9, remetente original do pacote, se encontra da raiz. Distando este 14m do BR, a distância a que o nó malicioso se encontra é dada pelo módulo da subtração da distância ao remetente com a distância percorrida pelo pacote, justamente, mod(14-9.5) = 4.5. Numa pesquisa posterior na mesma estrutura, é facilmente observável que o único nó que se encontra a uma distância do BR de 4.5m é o nó 2, sendo, portanto, o atacante. Note-se que apenas uma das extremidades do túnel foi identificada, a outra, por sua vez, será mapeada quando iniciar um ataque no seu domínio, por outras palavras, quando remeter um pacote para a extremidade oposta do túnel.

Note-se que caso não tivesse sido dirigido nenhum ataque sobre a topologia durante a distribuição do pacote, este teria sido difundido pela rota (9-8-2-1-4-12-13). Por se tratar de uma rota legitima, todos os saltos seriam realizados entre nós vizinhos o que, em momento algum, iria motivar a definição da "variável de estado" com o valor "1". Consequentemente, embora o pacote transitasse pelo BR, este, corretamente, apenas iria atualizar o campo distância percorrida, não efetuando qualquer ação a fim de determinar a identidade dos nós maliciosos, pois nada indiciaria que estes existissem.

4.4 Fatores de diferenciação face a outras propostas

Embora existam na literatura diversas propostas para a deteção de ataques *Wormhole*, o presente IDS é dotado de características diferenciadoras.

Após uma análise das propostas existentes, clarificadas no capítulo 3, verificamos que aquelas que se baseiam na distância entre nós podem ser divididas em dois grandes grupos. Primeiramente, destaco aquelas que carecem da inclusão de hardware GPS nos dispositivos da rede, de modo a determinar a localização geográfica dos nós. No segundo grupo, encontram-se as propostas que realizam a mesma tarefa por intermédio da conversão da força do sinal.

Enquadrando-se a proposta apresentada neste segundo grupo, difere das demais no esquema de deteção dos ataques e na forma como identifica os nós maliciosos.

Assim, enquanto que as demais medem a distância de forma linear, esta, para além de a medir entre nós vizinhos, mede também a distância de cada nó ao BR, através do somatório das distâncias individuais, correspondendo, na prática, à distância do trajeto. De realçar que estas ficam armazenadas numa tabela no domínio do BR. Relativamente aos outros nós, também estes mantêm documentadas as distâncias aos seus vizinhos, algo que não acontece na literatura. Além disso, este modelo inova no facto de não analisar apenas mensagens DIO, fazendo uma avaliação integral do tráfego da rede, através da inclusão de dois novos campos nas mensagens. Esta ação permite não só a deteção de mais modalidades de ataque *Wormhole*, como, também, identificar os pacotes que transitaram por intermédio de um túnel malicioso. Outro dos fatores que diferencia esta proposta é o facto de que, aquando da deteção de um ataque, esta mapeia diretamente os nós maliciosos, enquanto que em outras a identificação destes elementos é realizada, frequentemente, por recurso a modelos probabilísticos.

4.5 Síntese de capítulo

No capítulo que agora termina foi clarificado o funcionamento do novo sistema de deteção de ataques *Wormhole* proposto.

Ainda que seja um IDS talhado a operar sobre condições ambientais e protocolares especificas, nomeadamente, pela obrigatoriedade da rede em que se insere ser estática e operada pelo protocolo de encaminhamento RPL em *Non Storing Mode*, considera-se que a solução tenha bastante potencial.

A título sumário, a estratégia de deteção passa pela inclusão de dois novos indicadores nos pacotes que circulam na rede, por forma a determinar a distância por eles percorrida entre remetente e destinatário, para posterior comparação com os valores de distância referência para cada trajeto, calculados no domínio do BR, sendo que, esta verificação apenas se efetiva se no decorrer da propagação de um pacote, um dos nós envolvidos detetar que este não lhe foi remetido por nenhum dos seus vizinhos. Obviamente que para tal é necessária a definição de novas estruturas onde serão armazenadas as distâncias entre nós, sendo estas apuradas num momento inicial, por estimação resultante da conversão do valor RSSI, determinado nas mensagens trocadas entre nós. Ainda assim, não existe nenhum vínculo a que o processo de definição das estruturas auxiliares seja feito por estimação, uma vez que nada impede que o gestor de rede procede a uma definição manual das mesmas, dada a obrigatória estaticidade da rede.

Esta página foi propositadamente deixada em branco.

Capítulo 5

Estratégia para validação do modelo proposto

A fim de avaliar a prestação e solidez do sistema de deteção proposto foi realizada uma simulação computacional do modelo.

Tratando-se a representatividade de um ambiente real um dos objetivos de qualquer simulação e consciente do impacto deste fator nos resultados obtidos, é importante, desde já, a definição de um cenário de aplicação real onde a inclusão do IDS proposto faça sentido, sendo esta a referência a ter em consideração na transposição para a simulação virtual.

5.1 Cenário de aplicação considerado

Com o avanço tecnológico, à semelhança do que acontece noutras áreas, também no setor agrícola se denota um aumento da utilização de dispositivos IoT nos processos de cultivo. Esta prática justifica-se pelo facto de tal abordagem permitir minimizar o custo de produção e, paralelamente, incrementar a produtividade.

Um exemplo concreto de utilização de dispositivos IoT na agricultura é o processo de produção em estufa de frutas e hortícolas, onde estudos realizados demonstraram que a inclusão tecnológica permite uma diminuição de 60% da mão de obra, 80% do consumo de água e 70% da quantidade de pesticidas utilizados [84].

A imagem que se segue, 5.1, representa a forma como a tecnologia é integrada neste tipo de situações.

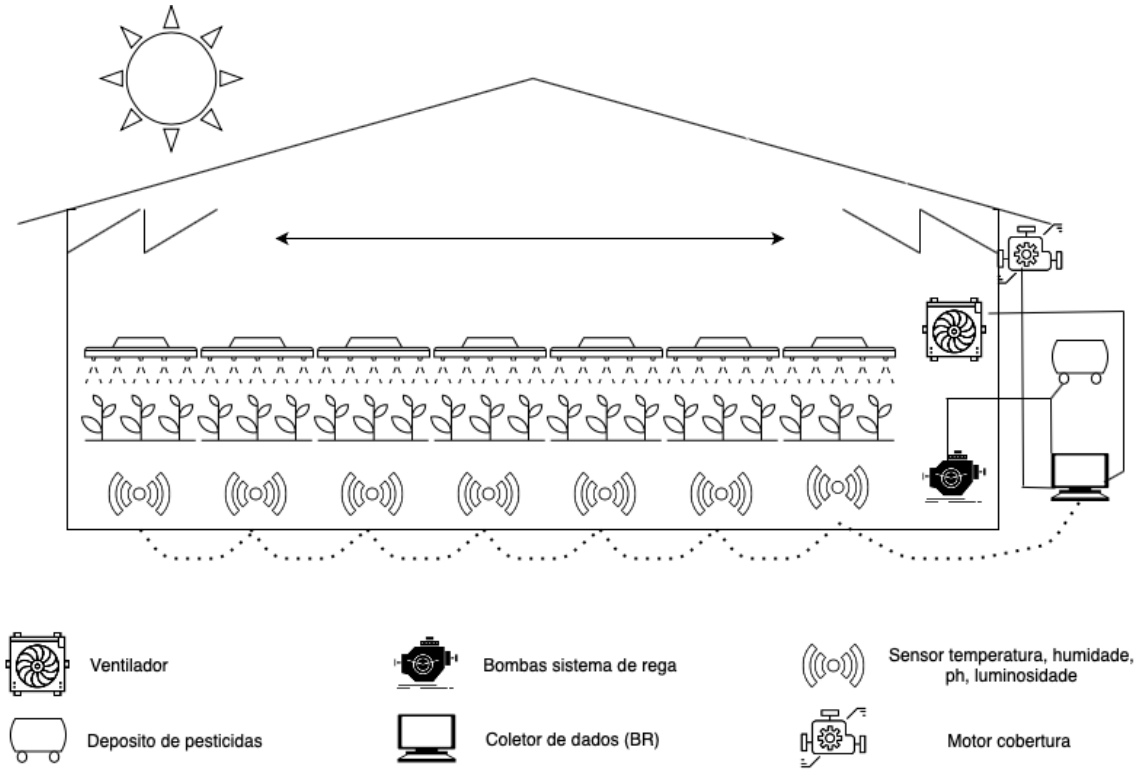

Figura 5.1: Exemplo de cenário real de aplicação

Repare-se que no interior da estufa estão instalados vários sensores, distribuídos por toda a área de cultivo, a fim de coletar do meio informações como o Potencial Hidrogeniônico (pH), luminosidade, humidade e temperatura. Com base nestes indicadores, correlacionados no domínio do coletor central, que aqui assume função de BR, são tomadas decisões que, por sua vez, irão efetivar ações nos vários sistemas anexos [71]. Por exemplo, se os sensores de pH reportarem valores fora de um intervalo pré-determinado, são adicionadas, a nível das bombas do sistema de rega, soluções mais ou menos alcalinas e pesticidas por forma a corrigir a acidez do estrato. De igual forma, quando os indicadores de pH retomarem a valores aceitáveis esta operação é suspensa. Já se os sensores de temperatura de uma dada secção reportarem valores inadequados, são diligenciadas ações que os revertam, através da ativação ou desativação do sistema de ventilação. No que concerne à luz, e dada a relevância deste fator na produtividade, com base no valor da luminosidade, lido pelos sensores, é ajustado o posicionamento da cobertura com o auxílio do sistema mecânico dedicado ao efeito. Por fim, sendo a água outro elemento altamente impactante na qualidade do produto, o sistema de rega é acionado sempre que os valores providenciados pelos sensores de humidade assim o exigiam. A supressão da irrigação ocorre, justamente, assim que os valores de humidade atinjam o valor desejado.

Retenha-se que a maximização da produção implica que todos os fatores ambientais, anteriormente mencionados, oscilem em intervalos bem delineados. No caso da humidade, por exemplo, a produção de limões e vegetais obriga a que esta se mantenha entre os 70 a 80%, já a temperatura deve ser, no mesmo cenário, mantida entre os 29 e os 32 graus centígrados [54]. Ainda assim, não podemos deixar de considerar a existência de outras espécies cujo processo de produção é mais sensível às alterações do meio, como é o caso do cogumelo [12]. Além disso, é recorrente encontrar estufas onde a variedade de espécies produzidas é imensa e, portanto, cada uma delas é cultivada numa câmara especifica, obrigando a que o

relacionamento das variáveis a nível do BR seja particionado de acordo com a área à qual os dados pertencem.

Dada a sensibilidade dos produtos que estão a ser produzidos, obviamente que o sistema de monitorização e controlo, integrado no meio de produção, deve obedecer a uma série de requisitos, seja a disponibilidade, a integridade, ou mesmo, o tempo de resposta às alterações do meio [34].

Neste seguimento, é percetível que o direcionamento de um ataque *Wormhole* pode ser altamente prejudicial para a produção, isto porque, através de um ataque deste tipo é possível provocar um atraso na propagação de mensagens, ou mesmo, a inoperacionalização de um conjunto de sensores.

Num exemplo prático, supondo uma unidade de produção de cogumelos, sendo este produto altamente vulnerável a oscilações ambientais, e admitindo que o esquema de produção segue o modelo acima descrito, se durante o processo de rega um ataque *Wormhole* provocar atrasos na propagação de mensagens na rede de forma reiterada é o suficiente para que a produção seja afetada, isto porque, a inativação das bombas do sistema de irrigação está dependente dos dados recebidos dos vários sensores, esta questão torna-se ainda mais delicada se transpusermos este cenário de ataque para um momento de fertilização ou correção da acidez do solo, o que provocaria a que este deixasse de ser fértil passando a ser tóxico. Note-se que ainda que este tipo de ocorrências nem sempre impliquem a perda das colheitas, a qualidade do produto é fortemente impactada, ainda no exemplo da produção de cogumelo, o calibre dos exemplares influencia o seu preço de venda, por esse motivo, aquando da apanha, é necessário proceder à separação de acordo com o tamanho. Assim, se parte da produção sofreu uma perturbação durante o período de desenvolvimento, a mão de obra necessária na separação do produto é muito superior, já o seu preço de venda é, nestas circunstâncias, claramente inferior.

Se até então apenas foram dados exemplos em que o impacto do ataque *Wormhole* incidiu sobre a qualidade do produto, não podemos deixar de considerar que dependendo da quantidade de ataques efetivados e da sua periodicidade possa ser atingida uma situação limite em que as próprias mensagens protocolares para reajuste topológico causem a saturação e o inerente colapso da rede. De forma análoga, ainda que a rede consiga absorver todo o tráfego, se os dispositivos forem alimentados por bateria, a sua degradação será claramente superior e, portanto, não se tratando de ambientes constantemente monitorizados, qualquer uma das situações anteriores pode culminar na perda total da produção.

No que concerne à extensão das estufas dadas como exemplo, esta pode ser variável, no entanto, habitualmente estas possuem entre 100 a 300 metros de extensão e integram sensores a cada 10 metros, sendo as condições do meio aferidas em intervalos que variam entre 5 a 60 segundos.

É importante frisar que o cenário proposto tráta-se de uma mera possibilidade aplicacional do modelo, podendo este ser integrado em muitos outros, sejam eles voltados para o setor agrícola, industrial ou mesmo no domínio quotidiano.

5.2 Ferramentas utilizadas

Definido o cenário real a ter em consideração para representação em ambiente virtual, foi necessário proceder à escolha do software que melhor se enquadra no tipo de simulação desejada. Nesta escolha pesaram o tipo de dispositivos a simular e o sistema operativo que lhes servirá de suporte. Adicionalmente, nesta secção, serão ainda abordadas outras

ferramentas auxiliares, utilizadas quer na fase de implementação, quer, posteriormente, na fase de testes.

5.2.1 Sistema operativo Contiki

O *Contiki OS* trata-se de um Sistema Operativo (SO), *open source*, desenvolvido em 2003 por Adam Dunkels. O facto de ter sido especificamente desenhado para operar em dispositivos altamente restritos, faz com que se enquadre, perfeitamente, na operacionalização de dispositivos IoT [81].

Este SO é baseado num modelo de programação orientado a eventos, podendo estes ser do tipo síncronos ou assíncronos. No primeiro caso, na prática, os eventos funcionam como que uma chamada de função, no sentido em que são imediatamente vinculados a um processo no qual efetivam as ações desejadas. Já os eventos assíncronos, contrariamente aos anteriores, são processados de acordo com a sua posição na fila de eventos.

Dada a natureza modular do *Contiki OS* e estando na presença de um *Kernel* híbrido, torna-se possível a operação em *multithreading*, nomeadamente, por recurso a *protothreads* responsáveis pela gestão dos pedidos remetidos por cada *thread* à biblioteca geral do *Kernel* [17].

Relativamente à gestão de memoria, esta é feita, entre outros, através do módulo *Menaged Memory Allocator*, capacitado à alocação dinâmica e eficiente de memoria no decurso do funcionamento do SO.

Vocacionado a operar em dispositivos altamente restritos e conectáveis com a *Internet*, não é de admirar que o *Contiki OS* assente numa pilha protocolar 6LoWPAN standard, similar ao modelo de 6 camadas abordado em 2.1.3. Contudo, são-lhe reconhecidas algumas particularidades, em especial, na camada de ligação de dados que é, em *Contiki*, composta por 3 sub-camadas [16]:

- MAC - Encarregue pelo endereçamento e retransmissão de pacotes perdidos
- Radio Duty Cycling (RDC) - Responsável pelo acionamento e inativação do dispositivo de acordo com as solicitações aquando da receção e envio de pacotes.
- Framer - Procede à organização do *frames* de acordo com o especificado no IEEE 802.15.4 2.4

No que toca ao protocolo de encaminhamento, o *Contiki OS* permite a operação em *mesh-under* e *route-over*, se no primeiro modo o protocolo de encaminhamento, neste caso o RPL, opera a partir da camada de adaptação (6LoWPAN), no segundo, o RPL é implementado na camada de rede [69]. É importante referir que este ultimo é o modo de funcionamento padrão deste SO, sendo o utilizado no decorrer da implementação.

5.2.2 Simulador Cooja

O *Cooja* é uma ferramenta de simulação incluída no ambiente de desenvolvimento *Instant Contiki* [57] que permite não só o desenvolvimento em *Contiki OS*, como, também, a simulação de redes LLN, nomeadamente, com a inclusão de uma série de *hardware* característico destes ambientes [68].

5.2.3 Wireshark

Trata-se um programa que permite a análise dos fluxos de dados que circulam numa rede [38]. A ferramenta em causa, permite, inclusive, a organização do tráfego de acordo com o protocolo correspondente. A informação é disponibilizada ao utilizador por intermédio de uma interface gráfica, simples e intuitiva. A utilização desta ferramenta teve especial relevância na fase de implementação do IDS, uma vez que nos permite aferir o conteúdo dos pacotes e, deste modo, detetar eventuais afastamentos ao modelo de funcionamento idealizado.

5.2.4 Contiki Powertrace

O *Powertrace* trata-se de uma ferramenta, também ela incluída no ambiente de desenvolvimento *Instant Contiki* [57], que permite estimar o consumo energético de redes LLN, seja a nível global, seja, individualmente, para cada nó constituinte [15]. Esta ferramenta terá especial pertinência no estudo do impacto energético do sistema de deteção proposto.

5.3 Detalhes da simulação

Por forma a validar o modelo de deteção de ataques *Wormhole* proposto, foram realizadas várias simulações por intermédio da ferramenta *Cooja* 5.2.2, cada uma delas representativa de um de 3 ambientes, que passo a caracterizar.

- Ambiente 1 - Será representativo de uma rede LLN composta por 10 nós do tipo *Tmote Sky*, associados a um outro que assume a função de BR, não padecendo este das mesmas limitações energéticas e comunicacionais dos dispositivos anteriores. Esta rede será operada pelo protocolo de encaminhamento RPL em modo *Non Storing Mode*, não sendo considerada qualquer alteração posicional dos nós no período de simulação.
- Ambiente 2 - Representativo de uma rede LLN, a operar nas mesmas condições do ambiente 1, mas sendo agora constituída por 20 nós *Tmote Sky*, mais uma vez, associados a um outro elemento com recursos ilimitados e com função de BR.
- Ambiente 3 - Herda as características operacionais dos dois anteriores mas inclui 30 nós *Tmote Sky* e um BR.

O motivo da escolha de nós *Tmote Sky* prende-se com o facto destes validarem os pressupostos de dispositivo de baixa potência, por terem capacidade sensorial e serem frequentemente incluídos em redes de sensores e ainda pelo facto de serem suportados pela ferramenta *Cooja* [29] [13]. Relativamente ao BR, este será simulado por intermédio de um computador dada a sua isenção restritiva.

Adicionalmente, cada um dos ambientes criados foi simulado para três diferentes cenários operacionais:

- Na ausência de ataques e sem o IDS em operação - Findando a inicialização protocolar e, desta forma, definida a árvore de encaminhamento, estabelecidas as relações topológicas resultantes desta, será simulado tráfego comunicacional legitimo entre nós aleatórios. Paralelamente, será aferido o consumo energético da rede, por forma a ser

usado como referência na avaliação comparativa do impacto energético do modelo proposto.

- Na ausência de ataques e com o IDS em operação - Depois de definida a árvore de encaminhamento e estabelecidas as relações hierárquicas entre os nós incluídos no ambiente, procede-se à inicialização do sistema de deteção proposto. Decorrido o período necessário à efetivação desta ação, espera-se que os módulos locais tenham as suas tabelas com as distâncias aos nós vizinhos devidamente preenchidas, bem como, que o módulo centralizado (BR) possua definida a tabela com a distância a cada um dos nós da topologia. Seguir-se-á a este processo a inclusão de tráfego legitimo na rede em intervalos de 30 segundos, sendo que, neste cenário, de forma contraria ao que acontece no anterior, todos os pacotes que circulam na rede irão ser atualizados e monitorizados pelos módulos do IDS que os intercetam a cada salto. Também neste caso é estimado o consumo energético, que, posteriormente, será comparado com o mesmo indicador, aferido no mesmo ambiente, mas nas condições do cenário anterior.

- Na presença de ataques e com o IDS em operação - Neste cenário, à semelhança do anterior, o IDS estará ativo e, portanto, todo o tráfego gerado será monitorizado. Contudo, durante o período de execução da simulação serão direcionados diversos ataques *Wormhole*, seja por encapsulamento ou por retransmissão de pacotes. Os nós maliciosos estarão colocados de forma aleatória na topologia e a determinado momento iniciam os ataques para os quais estão programados. Deste modo, será possível averiguar o número de ataques detetados correta e incorretamente pelo sistema e avaliar a sua capacidade de identificação dos nós maliciosos envolvidos em cada ataque.

5.4 Métricas de avaliação

Os dados em bruto, provenientes das várias simulações realizadas e do *output* gerado pelas ferramentas utilizadas na fase de avaliação, dificultam a sua análise, suscitando assim a necessidade de definir métricas que os correlacionem e facilitem a sua interpretação. Desta feita, serão consideradas 12 métricas, clarificadas, individualmente, de seguida. Dado o contexto, as 11 primeiras são habitualmente utilizadas em problemas de classificação binária tendo, portanto, sido utilizadas na avaliação quer da capacidade de deteção do IDS, quer da capacidade de identificação dos nós maliciosos. Por fim, a última métrica a ser mencionada visa o estudo impacto energético inerente à inclusão do sistema de deteção proposto.

Verdadeiros Positivos (VP)

Dado o contexto do trabalho, representa o número de nós maliciosos referenciados de forma assertiva. Já no estudo da capacidade de deteção do modelo, a métrica possui um significado diferente, relacionando-se agora com o número de ataques reportados corretamente.

Verdadeiros Negativos (VN)

Número de não ataques reportados corretamente. Já na perspetiva da avaliação da capacidade de identificação de nós maliciosos, o presente indicador reporta o número de nós legítimos identificados como tal e, portanto, não associados a qualquer ataque.

Falsos Positivos (FP)

Número de ataques reportados indevidamente. Na avaliação da capacidade identificativa

do IDS, a métrica remete para o número de nós legítimos incorretamente identificados como maliciosos.

Falsos Negativos (FN)

Número de ataques não reportados ou atacantes não identificados, dependendo da componente de avaliação em que se aplica.

Taxa de Falsos Positivos (TFP)

Providência a relação entre o número de ataques reportados de forma indevida relativamente ao número total de não ocorrência de ataques. De igual forma, aquando do estudo da capacidade de identificação de nós maliciosos, corresponde ao número de nós legítimos identificados incorretamente como maliciosos, em função do número total destes elementos presentes na topologia 5.1.

$$\text{TFP} = \frac{\text{FP}}{\text{FP} + \text{VN}} \tag{5.1}$$

Taxa de Falsos Negativos (TFN)

Afere a relação entre o número de ataques não reportados e o número total de ataques efetivos. Quando utilizada na avaliação da componente de identificação dos nós maliciosos, estabelece a relação entre o número de atacantes não identificados face ao número total de atacantes presentes na topologia 5.2.

$$\text{TFN} = \frac{\text{FN}}{\text{FN} + \text{VP}} \tag{5.2}$$

Taxa de Verdadeiros Negativos (TVN)

Trata do relacionamento do número de não ataques reportados de forma correta relativamente ao número total de ocorrências não associadas a atividade maliciosa. De igual forma, relaciona o número de nós classificados como legítimos com a totalidade destes elementos inclusos no ambiente 5.3.

$$\text{TVN} = \frac{\text{VN}}{\text{VN} + \text{FP}} \tag{5.3}$$

Precisão

Permite apurar, de entre todos os ataques reportados, a porção destes que realmente se efetivaram. No que concerne à identificação dos nós maliciosos, a presente métrica permite apurar de entre todos os elementos referenciados como atacante pelo IDS, a porção destes que efetivamente adotam este comportamento 5.4.

$$\text{Precisão} = \frac{\text{VP}}{\text{VP} + \text{FP}} \tag{5.4}$$

Recall (TPR)

Indica a porção de ataques corretamente detetados em função do número total de ataques efetivos. No contexto dos objetivos de avaliação propostos, pode, igualmente, remeter para relação entre o número de nós maliciosos identificados pelo modelo e o número total de

elementos com este comportamento inseridos na rede sob monitorização 5.5.

$$\text{TPR} = \frac{\text{VP}}{\text{VP} + \text{FN}} \tag{5.5}$$

Exatidão

Esta métrica avalia a capacidade de o sistema de deteção classificar corretamente as situações de ataque e, por sua vez, classificar situações de não ocorrência de ataque como tal. Este indicador pode ser calculado através da expressão 5.6. Paralelamente, permite averiguar a assertividade do modelo na caracterização dos elementos integrantes como malicioso ou legitimo.

$$\text{Exatidão} = \frac{\text{VP} + \text{VN}}{\text{VP} + \text{VN} + \text{FP}+\text{FN}} \tag{5.6}$$

F-measure

Trata-se da média harmónica entre a precisão e o *recall*. Na prática, em cenário ótimo, o resultado desta métrica é 1, significando que a precisão e *recall* têm valores de 100%, implicando assim uma taxa de deteção/identificação máxima (100%) e nenhum falso positivo reportado 5.7.

$$\text{F-Measure} = 2 * \frac{\text{Precisão} * \text{TPR}}{\text{Precisão} + \text{TPR}} * 100 \tag{5.7}$$

Consumo energético

Dadas as limitações dos dispositivos nos quais o sistema de deteção proposto irá operar, torna-se pertinente proceder a uma avaliação do impacto energético da solução. Ao incluir processamento adicional em cada um dos nós, é expectável que a degradação da energia das suas baterias seja superior, quando alimentados por este meio. Desta feita, é necessário concluir se este impacto é admissível, ou, caso contrário, se contribui para uma diminuição abrupta do tempo de operacionalidade dos dispositivos. Neste seguimento, será realizada uma análise comparativa acerca do consumo energético dos dispositivos na presença do IDS e, posteriormente, sem que este esteja incluído, tendo em consideração o mesmo ambiente de simulação.

Com o objetivo de otimizar o consumo energético dos dispositivos IoT, os fabricantes desenvolveram estratégias que promovem a modularidade operacional dos mesmos. Desta forma, sempre que um módulo funcional do dispositivo não esteja em utilização pode ser desativado, processo este gerido a nível protocolar pela camada RDC 5.2.1. Assim sendo, à semelhança de outros dispositivos IoT, o *Tmote Sky* possui 4 modos de operação [13]:

- CPU - Implica que apenas operações de CPU estejam a ser executadas, permitindo, por isso, que toda a componente rádio esteja inativa, seja para operações de escuta ou transmissão.
- Low Power Mode (LPM) - É o modo que imprime um menor consumo energético, dada a inatividade do módulo de rádio (transmissão e escuta) e de CPU.
- Transmissão (TX) - Apenas o módulo rádio para operações de transmissão de dados está ativo.

- Escuta (RX) - Pressupõe que apenas o módulo rádio para operações de escuta de pacotes esteja em funcionamento.

Obviamente que a cada um dos módulos anteriores estão associados consumos energéticos diferentes. No entanto, esse consumo é documentado individualmente pelo fabricante, em situação de operação normal, como podemos ver na tabela abaixo 5.2.

	MIN	NOM	MAX	UNIT
Supply voltage	2.1		3.6	V
Supply voltage during flash memory programming	2.7		3.6	V
Operating free air temperature	-40		85	°C
Current Consumption: MCU on, Radio RX		21.8	23	mA
Current Consumption: MCU on, Radio TX		19.5	21	mA
Current Consumption: MCU on, Radio off		1800	2400	µA
Current Consumption: MCU idle, Radio off		54.5	1200	µA
Current Consumption: MCU standby		5.1	21.0	µA

Figura 5.2: Consumo energético modular de dispositivos TmoteSky segundo o fabricante [13]

Na prática, o *output* da ferramenta *Powertrace*, apresentado na figura 5.3, corresponde ao intervalo de tempo que cada nó operou em cada um dos módulos acima referidos.

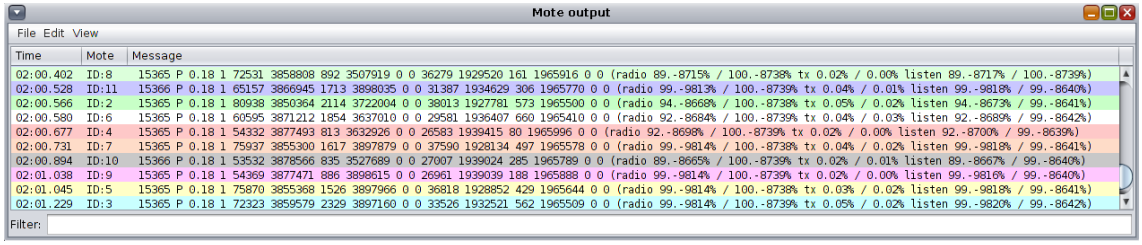

Figura 5.3: Output gerado pela ferramenta Powertrace

Através do relacionamento desta informação com as especificações energéticas veiculas pelo fabricante 5.2, é possível estimar o consumo energético individual de cada módulo operacional de um dispositivo, através da expressão 5.8, ou mesmo, o seu consumo energético total durante o período de simulação, pelo somatório das 4 componentes [58], como mostra a expressão 5.9.

$$Energia\ (mJ) = \frac{\Delta(s)\ Componente\ *\ Corrente\ associada\ à\ componente}{RTIMER\_SECOND} *\ Voltagem \quad (5.8)$$

$$Energia\ total(mJ) = \frac{\Delta(\mathrm{s})\ CPU\ *\ 1.8 + \Delta(\mathrm{s})\ LPM\ *\ 0.0545 + \Delta(\mathrm{s})\ Tx\ *\ 19.5 + \Delta(\mathrm{s})\ Rx\ *\ 21.8}{4096\ *\ 8} * 3 \quad (5.9)$$

5.5 Síntese de capítulo

Em suma, através da simulação em ambiente virtual de 3 diferentes ambientes com 10, 20 e 30 nós do tipo *TmoteSky*, simulados sobre diferentes cenários de operação, espera-se ser possível averiguar a capacidade de deteção do modelo, a sua eficácia na identificação dos nós maliciosos e, por fim, determinar o seu impacto energético. Por este motivo, se as métricas associadas à problemática da classificação binária foram apuradas por recurso aos cenários operacionais nos quais são direcionados ataques Wormhole do tipo por retransmissão de pacotes e por encapsulamento, o impacto energético é aferido por comparação, em cada ambiente, do consumo energético total, nos cenários com e sem a presença do IDS.

Por forma a aumentar a representatividade das simulações foi, previamente, descrito um cenário de aplicação real relacionado com a inclusão de dispositivos IoT no processo de produção em estufa de frutas e hortícolas. Neste meio, os dispositivos utilizados possuem funcionalidade critica na gestão dos sistemas de rega, ventilação, fertilização e controlo de luminosidade. Por este motivo, as preocupações de segurança são imensas, dados os efeitos nefastos sobre a produção em caso de ataque. Assim, o número de dispositivos simulados em cada ambiente, bem como, o seu modo de funcionamento, têm em consideração o dimensionamento típico destas estruturas de produção e o intervalo de tempo decorrente entre cada uma das várias atividades sensoriais em cenário real.

Esta página foi propositadamente deixada em branco.

Capítulo 6

Implementação do modelo

Depois de definidas as ferramentas a utilizar, estabelecido o cenário real tido como referência à simulação, é agora necessário proceder à implementação, propriamente dita, do sistema de deteção nos dispositivos virtuais.

A realização desta tarefa compreendeu 3 etapas: implementação do módulo distribuído nos nós *TmoteSky*, implementação do módulo centralizado, a nível do BR, e implementação das tipologias de ataque *Wormhole* tidas em consideração.

Note-se que a avaliação do sistema de deteção proposto e a representatividade do ambiente simulado está dependente da existência de tráfego resultante da atividade operacional legitima dos elementos inclusos na topologia. Por este motivo, foi necessária a definição dos métodos responsáveis pelo envio sistemático de mensagens representativas da atividade sensorial e, portanto, originárias em cada um dos vários nós distribuídos e endereçadas ao BR. Neste sentido, no ficheiro *client.c*, integrante da camada aplicacional dos nós distribuídos, a cada intervalo de 30 segundos é gerado um evento (*etimer_set(&periodic, SEND_INTERVAL)*), responsável pelo envio de um pacote (*ctimer_set(&backoff_timer, SEND_TIME, send_packet, NULL)*) e pelo restabelecimento do respetivo temporizador (*etimer_reset(&periodic)*).

Na presente secção, cada uma das etapas, acima referidas, será documentada e descrita detalhadamente.

6.1 Módulos distribuídos

Recorde-se que as principais funções dos módulos distribuídos passam pela identificação de todos os pacotes recebidos por um nó, cujo remetente não faça parte da sua vizinhança, notificação do módulo centralizado de tal ocorrência e atualização da distância percorrida nos pacotes que transitam na rede. Dada a inclusão deste módulo em todos os nós da topologia, à exceção do BR, e considerado a natureza restritiva destes dispositivos (*TmoteSky*) é importante garantir a eficiência de todas as operações.

De uma perspetiva protocolar, a implementação do módulo distribuído recai, exclusivamente, sobre a camada de rede, ainda que, apoiada em algumas variáveis provenientes de camadas inferiores. Consequentemente, a inclusão deste módulo implicou a definição de novas estruturas e funções no ficheiro *tcpip.c*.

Encontrando-se o *ContikiOS* a operar em modo *route-over*, são intrínsecas à camada de

rede as atividades protocolares de encaminhamento, por parte do RPL. No que toca às variáveis externas à camada e necessárias ao funcionamento do módulo distribuído do IDS, a mais notória é o endereço MAC do remetente dos pacotes, definido no ficheiro *sicslowmac.c* integrante da camada RDC.

Ainda que o sistema de deteção proposto pudesse ter sido implementado numa camada superior, tal abordagem não foi considerada. Repare-se que quanto mais "baixa" for a implementação do sistema de deteção, mais transparente será a sua operação de uma perspetiva aplicacional. Consequentemente, ao apostar numa implementação a nível da camada de rede estamos a providenciar uma maior flexibilidade no protocolo aplicacional a utilizar, sendo as atividades decorrentes desta atividade protocolar completamente independentes das do IDS proposto no presente documento.

Também do ponto de vista da eficiência e complexidade computacional, a definição dos métodos anexos ao módulo distribuído na camada de rede traz enumeras vantagens. Isto porque, no caso de o nó que se encontra a processar um pacote não ser o destinatário final, não há necessidade, no seu domínio, que o processamento se estenda à camada aplicacional, o que não só economiza recursos como, ao mesmo tempo, promove uma reação mais antecipada em caso de ataque.

Estando todas as operações do módulo distribuído do IDS a ser implementadas a nível da camada de rede, considerou-se pertinente que os campos adicionais dos pacotes (*distancia_percorrida* e *variavel_estado*) fossem incluídos num cabeçalho também ele processado nesta zona da pilha protocolar. A opção passou pela utilização dos cabeçalhos de extensão do pacote IPv6. Se o leitor mais atento se está neste momento a questionar sobre o porquê de não ter sido implementado o módulo distribuído na camada de adaptação, por exemplo no ficheiro *sicslowpan.c*, a resposta é simples. Observe-se que nas situações de progressão ascendente da pilha protocolar, ou seja, aquando da receção de um pacote, no momento do processamento do mesmo na camada de adaptação, as decisões de encaminhamento ainda não foram tomadas e, portanto, ainda não foi determinado o *nexthop*, o que impossibilita a incrementação da distância percorrida correspondente a esse trajeto.

Por todos os motivos acima mencionados, a camada de rede e o ficheiro *tcpip.c*, sendo este o elemento mais centralizado desta camada, mostrou-se a melhor opção posicional para desenvolvimento dos métodos associados ao módulo distribuído.

6.1.1 Fase de inicialização

Sendo vital ao funcionamento do IDS a definição das distâncias a que se encontra cada um dos vizinhos de um nó, foi necessário, previamente, identificar esses mesmos vizinhos. O facto do presente modelo de deteção ser vocacionado a operar em redes estáticas implica que os vizinhos de um nó sejam imutáveis durante o período de funcionamento da rede, salvo indisponibilidade anormal de um dos elementos constituintes, durante o período de execução.

Acontece que embora a identificação dos nós vizinhos seja da responsabilidade do protocolo de encaminhamento RPL, sendo estes documentados numa estrutura protocolar apropriada (*ds6_neighbors*), foi necessário transpor esta informação para uma outra estrutura dedicada ao IDS. Isto porque, em resultado de um ataque, a estrutura *ds6_neighbors* pode ser adulterada e, portanto, o ponto de referência do sistema de deteção deve ser estático e imutável. Além disso, a utilização da estrutura *ds6_neighbors* por parte do sistema de deteção implicaria o reajuste da mesma, uma vez que, a cada elemento é necessário que corresponda uma distância. Neste seguimento, foi definida no ficheiro *tc-*

pip.c a estrutura *tabela_vizinhos*, constituída unitariamente por elementos *nos_vizinhos*, representativos disso mesmo e dotados dos atributos *uip_ipaddr_t endereço* e *float distancia*. Note-se que embora esta estrutura seja declarada no inicio da simulação, apenas é inicializada com *uip_ds6_nbr_num()* elementos (número de vizinhos de cada nó) e devidamente preenchida quando o processo de inicialização do RPL estiver completo. Por este motivo, a fase de inicialização do sistema de deteção apenas começa 30 segundos após o início da simulação e espera-se que neste período não ocorram quaisquer ataques. De uma perspetiva mais técnica, este compasso de espera é operacionalizado pelo método *PROCESS_WAIT_UNTIL(&inicializazao)*, sendo *inicializacao* um cronometro (*static struct etimer inicializacao*) , definido em *ContikiOS* como *etimer_set(&inicializacao, 30*CLOCK_SECOND)*.

Estando os vizinhos identificados e definidos na *tabela_vizinhos* é agora necessário definir, individualmente, o segundo atributo, a distância a que estes se encontram. Para isso, à medida que são rececionadas mensagens de um nó vizinho, no caso de na *tabela_vizinhos* a distância correspondente a esse elemento ainda não estar definida, é coletado o valor RSSI da mensagem e estimada, a partir deste, a distância ao vizinho em causa.

Não sendo o foco do presente trabalho a avaliação da solidez dos resultados da estimação da distância com base no RSSI, mas crente no bom desempenho da técnica, de acordo com o documentado por vários autores [7], [66], [76], [45], para efeitos de simulação os valores RSSI de cada nó foram definidos no ficheiro *project_conf.h* sendo estes os utilizados na estimação da distância por recurso à equação 4.2

Findando o processo de inicialização, acima descrito, ou seja, quando a todas as entradas da *tabela_vizinhos* corresponder uma distância, é gerado um pacote com destino ao BR (*uip_packet_sendto()*) e impressa uma mensagem informativa do sucesso da operação, como podemos ver na Figura 6.1, neste caso, no campo *mote output* da ferramenta *Cooja*.

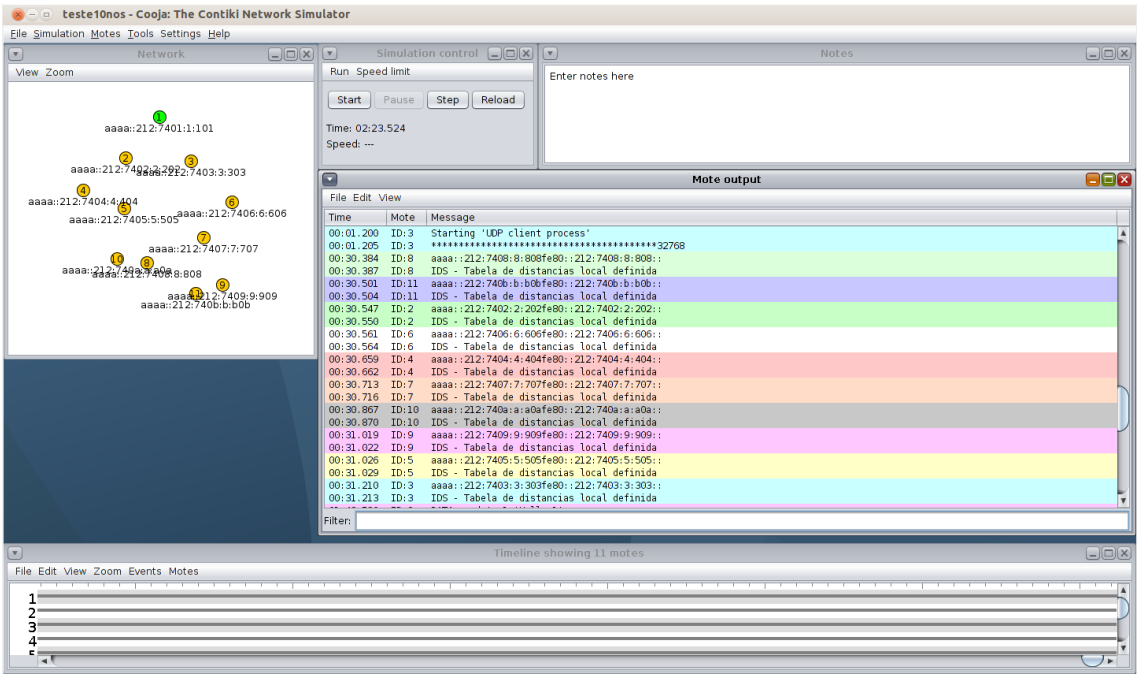

Figura 6.1: Conclusão do processo de inicialização do módulo distribuído

6.1.2 Deteção distribuída

Estando as estruturas auxiliares dos módulos distribuídos do IDS devidamente criadas e preenchidas, estamos em condições de proceder à monitorização da rede.

Como referido na parte introdutória da presente secção, os campos *distancia_percorrida* e *variavel_estado* foram incluídos no cabeçalho IPv6. Tendo sido previamente fundamentado o motivo desta escolha, é importante descrever, detalhadamente, a forma como esta associação foi feita, para, no seguimento, entender a restante implementação e funcionamento do modelo.

Sempre que uma mensagem é gerada no domínio de um nó, na camada de rede, o ficheiro *tcpip.c* tem a responsabilidade de adjudicar aos métodos definidos no ficheiro *uip6.c* o empacotamento do pacote TCP dentro de um pacote IPv6. Na presente implementação do modelo, nesta fase, é adicionado ao pacote IPv6, recém-criado, um cabeçalho de extensão do tipo *hop-by-hop*.

Esta adição é feita através da definição do campo *next-header* do pacote IPv6 com o valor "00". Já o cabeçalho de extensão *hop-by-hop*, propriamente dito, é composto, também ele, pelo campo *next-header*, tendo-lhe a este sido atribuído o valor 6, sendo que, o pacote que se lhe segue é do tipo TCP. Este cabeçalho inclui ainda o campo "extensão", definido a "1", uma vez que, corresponde ao tamanho do campo *options* em unidades de 8 bytes. O último campo desta estrutura, *options*, neste caso com um tamanho de 8 bytes é, justamente, a zona onde serão armazenadas as variáveis *distancia_percorrida* e *variavel_estado*, que no decorrer da operação do IDS serão definidas.

Ainda no âmbito da atividade protocolar da camada de rede, o ficheiro *tcpip.c* tem a responsabilidade de, por recurso aos métodos definidos no ficheiro *uipds6route.c*, determinar o elemento da rede que visa ser atingido no decorrer do primeiro salto do pacote agora criado, sendo esse elemento designado de *nexthop*. A ação que se segue, no domínio do sistema de deteção e levada a cabo pela função *tcpip_ipv6_output*, passa por percorrer todos os elementos contidos na *tabela_vizinhos* a fim de encontrar a entrada correspondente ao *nexthop* (*uip_ipaddr_cmp(&tabela_vizinhos[k].ip, nexthop)*), por forma a transpor a distância a este associada para a variável *distancia_percorrida*, (*distancia_percorrida = tabela_cliente[k].distancia;*) que é, neste momento, definida no pacote, à semelhança da (*variavel_estado*), ainda que, a esta seja atribuído o valor "0", simbolizando que até ao momento nenhuma ocorrência de ataque foi registada.

Supondo que a distância associada ao *nexthop* na *tabela_vizinhos* são 17 metros, o campo *options* do cabeçalho de extensão *hop-by-hop* do pacote IPv6 será definido como "017", sendo o dígito "0" relativo à *variavel_estado*. Daqui em diante, o pacote fará o seu normal percurso descendente da pilha protocolar *ContikiOS*, até ser transmitido pela interface rádio. Notem que este conjunto de operações apenas é executado pelo nó que gera o pacote, não se repetindo nos posteriores nós que o virão a processar durante o trajeto, isto porque, posteriormente, o processamento irá recair, unicamente, sobre as variáveis agora definidas no pacote (*distancia_percorrida* e *variavel_estado*), sendo a primeira atualizada a cada salto e a segunda reajustada mediante os possíveis eventos maliciosos ocorridos durante o trajeto.

Não se limitando a atividade topológica de um nó à difusão dos pacotes gerados no seu domínio, será, de seguida, clarificado o modo de funcionamento dos módulos distribuídos do IDS no decurso da atividade de encaminhamento de pacotes em trânsito.

Desta feita, aquando da receção de um pacote na interface rádio, a camada *framer* trata da organização dos *frames*, de modo a tornar o pacote compatível com a especificação IEEE 802.15.4 [22], 2.4. Ora, na camada contigua (RDC), nomeadamente no ficheiro *sicslowmac.c*, dada a já organização da informação, é possível identificar o último remetente intermediário do pacote, isto porque, nos restantes cabeçalhos da pilha protocolar, o remetente identificado é o nó que gerou e solicitou o envio do mesmo. Por este motivo, no ficheiro *sicslowmac.c*, é armazenado o endereço deste ultimo remetente na variável *(uint8_t)param_src_addr*.

Posteriormente, quando o processamento do pacote atinge a camada de rede, a primeira ação a ser executada passa pela verificação da correspondência entre a variável *(uint8_t×)param_src_addr* e o endereço de cada elemento contido na *tabela_vizinhos*. Caso não exista uma correspondência, a variável *ataque_detetado*, definida localmente no ficheiro *tcpip.c* é incrementada a 1, simbolizando que o pacote em processamento não foi remetido por um nó vizinho. Nesta situação, é alterado o primeiro dígito do campo *options* do cabeçalho *hop-by-hop* (ipv6), correspondente à *variavel_estado*, para "1".

No decorrer do processamento dos pacotes nesta situação, embora o *nexthop* seja determinado para efeitos de encaminhamento, a distância correspondente ao percurso até ao mesmo não é incrementada na variável *distancia_percorrida* do cabeçalho de extensão IPv6 do pacote em trânsito. Adicionalmente, sempre que a *variavel_estado* é alterada de "0" para "1" é impressa uma mensagem de advertência para um potencial ataque, como mostra a figura 6.2.

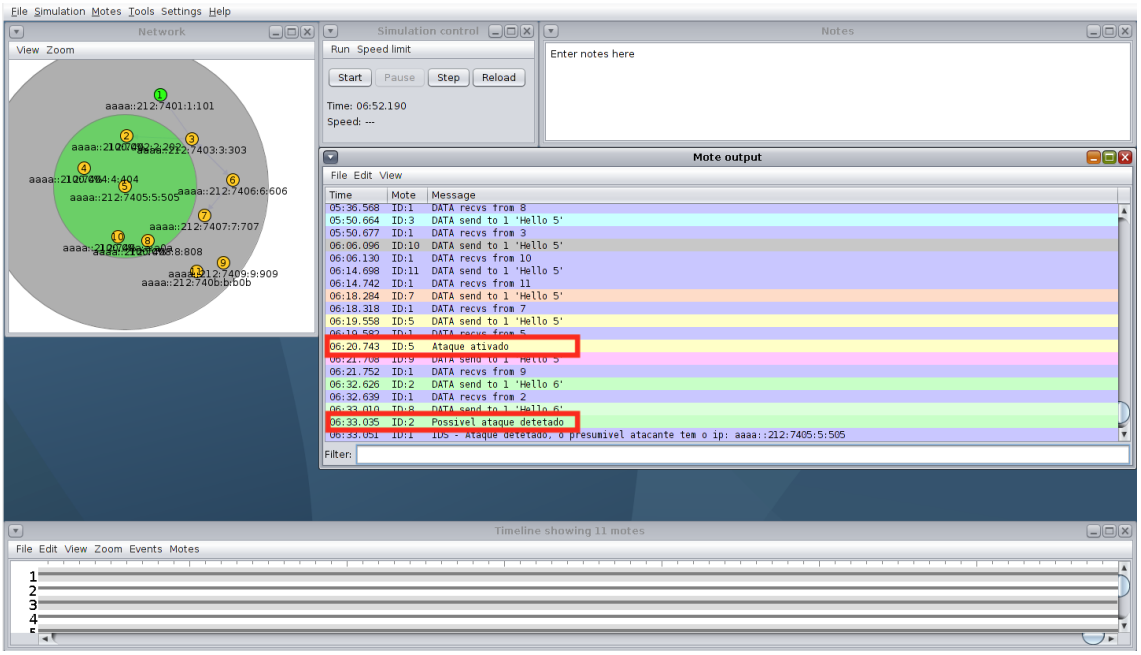

Figura 6.2: Advertência de ataque detetado gerado pelo módulo distribuído

Por outro lado, no caso de o pacote que está a ser processado ter sido remetido por um vizinho, e tendo-se por isso verificado a correspondência entre a variável *(uint8_t×)param_src_addr* e um qualquer elemento da *tabela_vizinhos*, a variável local *ataque _detetado* não é alterada, prosseguindo-se o processamento do pacote por parte do módulo de deteção distribuído, nomeadamente, com a avaliação da *variavel_estado*, contida no cabeçalho de extensão IPv6.

Se esta for igual a "1", significa que um possível ataque foi reportado por um nó que anteriormente processou o pacote em causa. Nesta situação, não é feita nenhuma alteração ao mesmo, sendo unicamente remetido para o BR (*NETSTACK_RADIO.send*), por intermédio da rota determinada pelo RPL, (*uip_ds6_route_lookup(&UIP_IP_BUF->bripaddr)*).

Por fim, no caso de um pacote ter sido remetido por um vizinho (flag ataque_detetado=0) e da análise da *variavel_estado* resultar a igualdade *variavel_estado*=0, então, se o pacote tiver como destino final o nó em causa, este segue o normal processamento a nível da camada aplicacional, caso seja um nó intermediário, é determinado o *nexthop* (*nexthop = uip_ds6_defrt_choose()*) e, com base neste, é percorrida a *tabela_vizinhos*, encontrada a correspondência e incrementada a variável *distancia_percorrida* no campo *options* do cabeçalho de extensão do pacote IPv6 com a distância correspondente. Pegando no exemplo do início desta descrição, se o campo estiver definido a "017" e a distância correspondente ao próximo salto na *tabela_vizinhos* for 10m, o valor resultante será "027".

Por forma a facilitar a compreensão da abordagem utilizada na implementação do módulo que agora acabo de descrever, na figura 6.3, é apresentado um fluxograma do funcionamento do mesmo, sendo neste clarificada toda a dinâmica operacional desta componente, bem como, apresentadas as zonas da pilha protocolar onde cada operação é realizada.

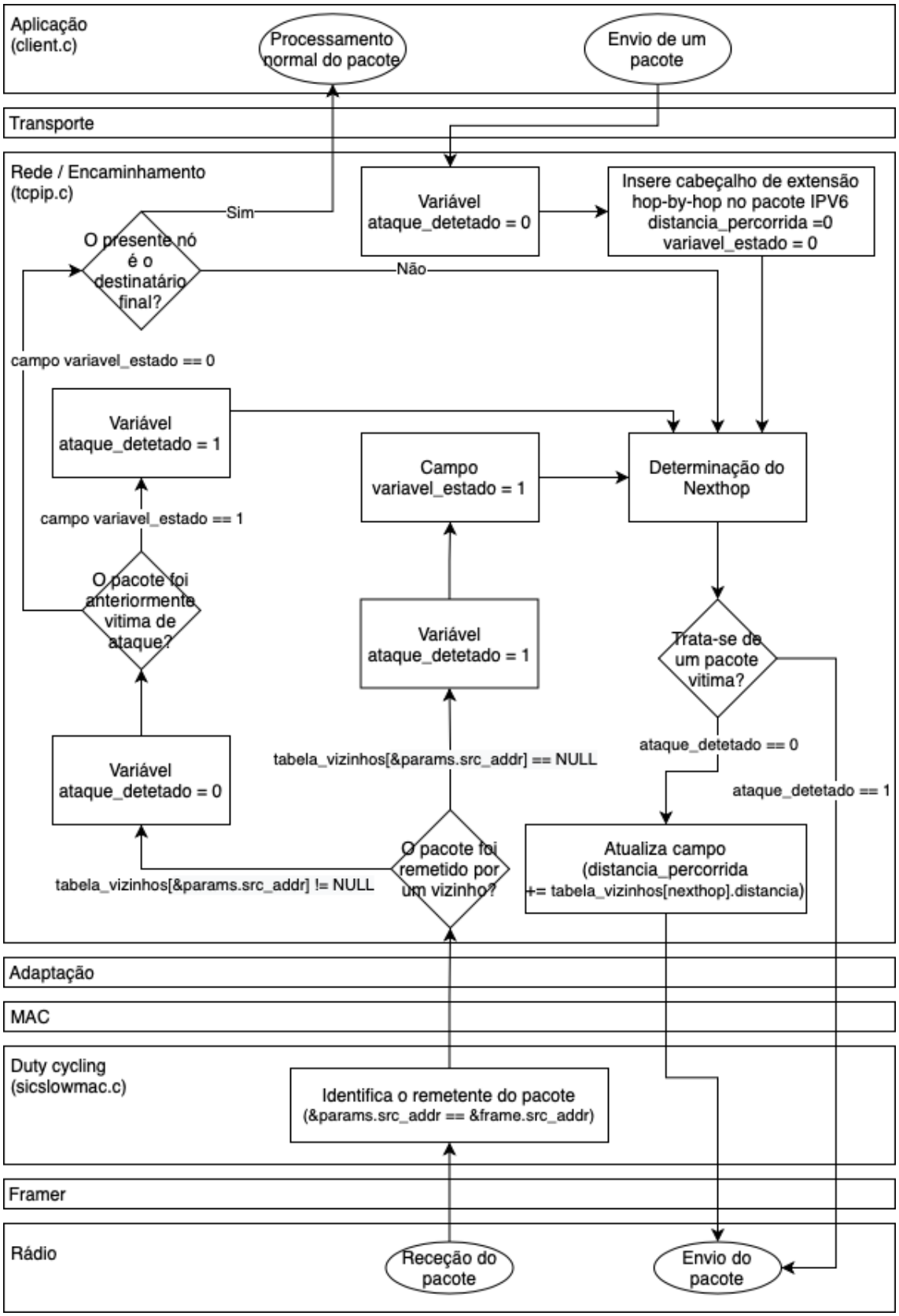

Figura 6.3: Fluxograma representativo do funcionamento do módulo distribuído

6.2 Módulo centralizado

Sendo este um componente essencial ao bom funcionamento do modelo de deteção proposto, irá ser abordada, em detalhe, a estratégia para a sua implementação. Retenha-se que todas as variáveis e funções inerentes a este módulo apenas são incluídas no domínio do BR.

6.2.1 Fase de inicialização

À semelhança do que acontece nos módulos distribuídos, o módulo centralizado carece da definição de uma estrutura composta não só pela distância a cada um dos seus vizinhos, mas, também, pela distância aos restantes nós da topologia.

A fase de inicialização deste módulo tem por objetivo a definição e preenchimento da estrutura *tabela_geral_distancias* no ficheiro *server.c*, sendo esta composta por um número de entradas equivalentes ao número de nós da topologia, definido manualmente no ficheiro *project_conf.h*. Cada entrada desta estrutura compreende os atributos uip_ipaddr_t endereço e float distancia.

Se no processo de inicialização dos módulos distribuídos é necessário o arranque prévio do RPL, no módulo centralizado este momento inicial está dependente da conclusão do processo de inicialização dos módulos distribuídos. Assim, após reunidas as condições, cada nó distribuído envia uma mensagem ao BR (módulo centralizado) sendo esta, no decorrer do trajeto, atualizada com a distância correspondente a cada salto, não se esperando, durante este período, a ocorrência de qualquer ataque.

No ficheiro *server.c*, integrante da camada aplicacional, foi ainda incluída a variável *int contador*, inicializada a zero e alvo de incrementação de uma unidade sempre que o BR recebe uma mensagem de um mote cujas informações (*endereço* e *distância*) ainda não estejam atribuídas na *tabela_geral_distancia*. Isto porque, nestas situações o BR adiciona uma nova entrada na estrutura em causa, à qual faz associar o IP do remetente (*UIP_IP_BUF->srcipaddr*) e a distância percorrida, incluída no respectivo cabeçalho de extensão *hop-by-hop* IPv6. Repare-se que o tratamento do pacote IPv6 é realizado na camada de rede e, portanto, esta informação fica armazenada na variável temporária *float distancia_percorrida*, definida e atribuída no ficheiro *tcpip.c*.

No momento em que a variável *contador* é igual ao número de nós incluídos na topologia, significa que a *tabela_geral_distancias* está devidamente preenchida. Nesta fase, é impressa uma mensagem a informar do sucesso da operação, como apresentado na Figura 6.4. Ainda assim, se decorridos 3 minutos desde o início da simulação e este processo ainda não tiver findado, é igualmente apresentada uma mensagem de advertência neste sentido.

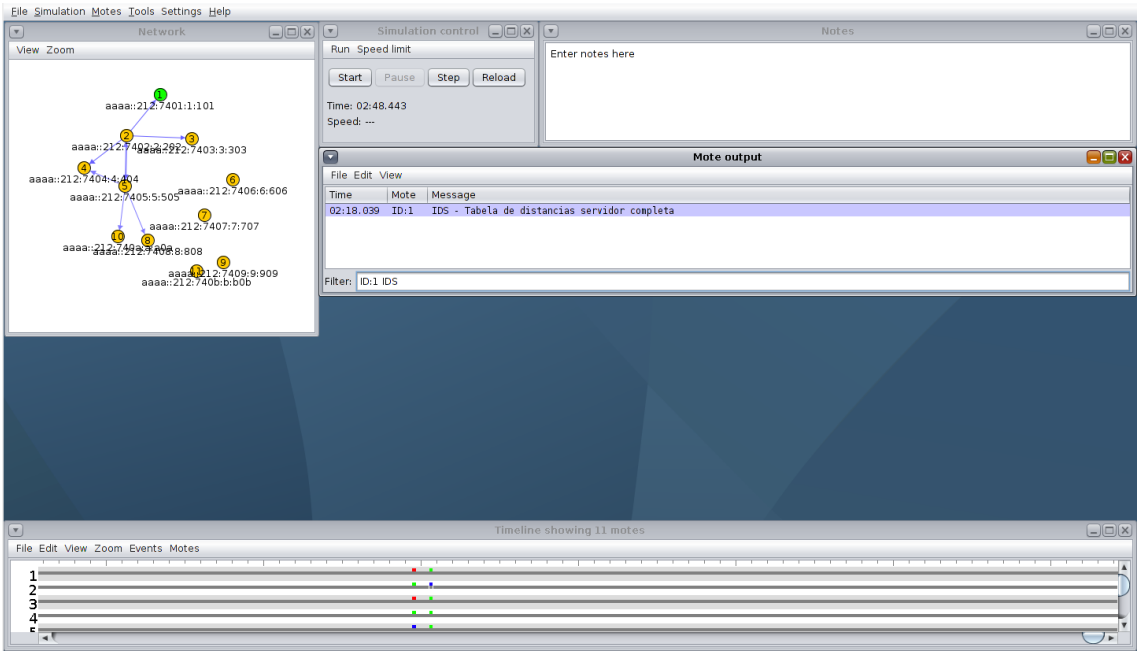

Figura 6.4: Conclusão do processo de inicialização do módulo centralizado

6.2.2 Deteção centralizada

Ainda que os ataques possam ser detetados pelos módulos distribuídos, apenas o módulo centralizado tem a capacidade de identificar os nós maliciosos envolvidos nos mesmos.

Partindo deste pressuposto, embora este módulo partilhe funcionalidades com os módulos distribuídos, possui, adicionalmente, métodos e especificidades face aos anteriores.

Nesta perspetiva, naturalmente que à receção de um pacote, a primeira operação a realizar é a averiguação da proximidade ao ultimo remetente do mesmo, por outras palavras, é verificado se este elemento é seu vizinho, sendo esta ação realizada de forma similar ao que é feito nos módulos distribuídos, pelo relacionamento da variável *(uint8_t)param_src_addr* com o endereço de cada um dos elementos contidos, neste domínio, na *tabela_geral_distancias*.

A maior singularidade deste módulo incorre, precisamente, na fase de análise do campo *variavel_estado*, uma vez que, no caso de este se encontrar definido a "0", apenas é determinado o *nexthop* e atualizado o campo *distacia_percorrida*, nos mesmos moldes do que é feito a nível dos módulos distribuídos. No entanto, se a *variavel_estado* se encontrar definida a "1", significa que, anteriormente, um nó distribuído classificou um evento como possível de associar a um ataque. Para identificação do nó malicioso envolvido, são utilizadas duas novas variáveis, definidas localmente no ficheiro *server.c*, sendo estas *float distancia_remetente* e *float distancia_atacante*. Numa primeira fase, é percorrida a estrutura *tabela_geral_distancias*, por forma a encontrar a entrada cujo IP corresponda ao endereço do remetente original do pacote em processamento (*UIP_IP_BUF->srcipaddr*). A distância associada a este elemento será então armazenada na variável recém criada *distancia_remetente*, (*distancia_remetente* =tabela_geral_distancias[k].distancia). Já à variável *distancia_atacante* será alusiva ao resultado da operação matemática mod(*distancia_remetente - distancia_percorrida*). Depois desta última definida, é novamente feita uma pesquisa na *tabela_geral_distancias*, a fim de encontrar o elemento cuja distância ao BR seja igual a *distancia_atacante*.

Todo este processo culmina com a apresentação do IP do elemento em causa, sendo este o nó malicioso, na consola *Mote output* da ferramenta *Cooja*, como demonstra a Figura 6.5.

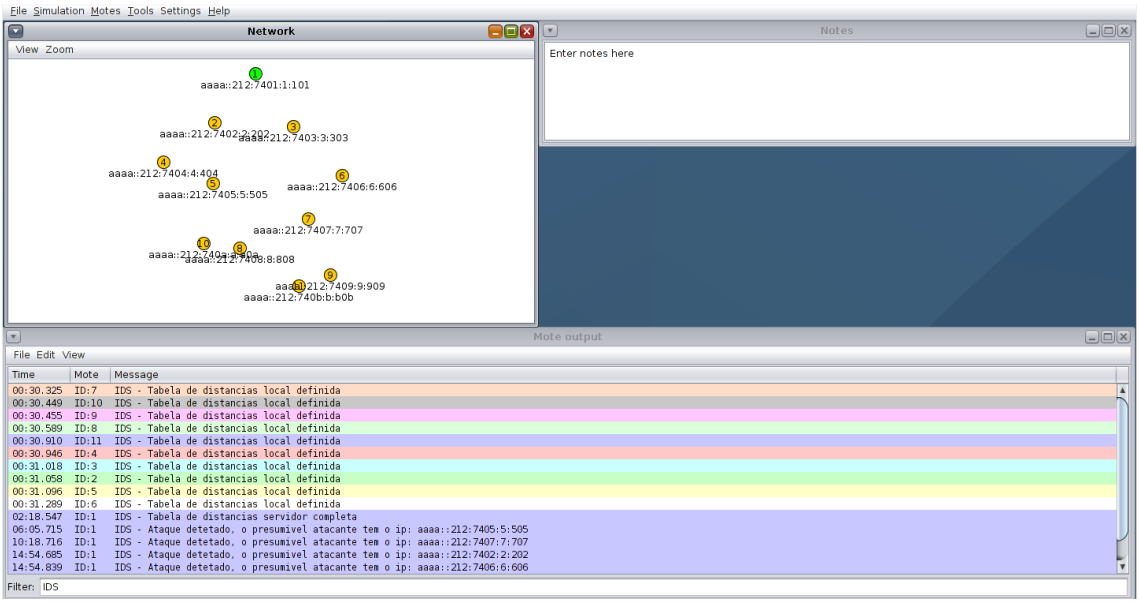

Figura 6.5: Identificação de nós maliciosos por parte do módulo centralizado

A Figura 6.6 sintetiza o funcionamento e implementação do módulo centralizado. O fluxograma apresentado esquematiza, de uma perspetiva interna, o comportamento deste componente no decurso das atividades de receção e envio de pacotes. Além disso, e nos mesmos moldes, é também clarificada a forma como este módulo procede à identificação dos nós maliciosos.

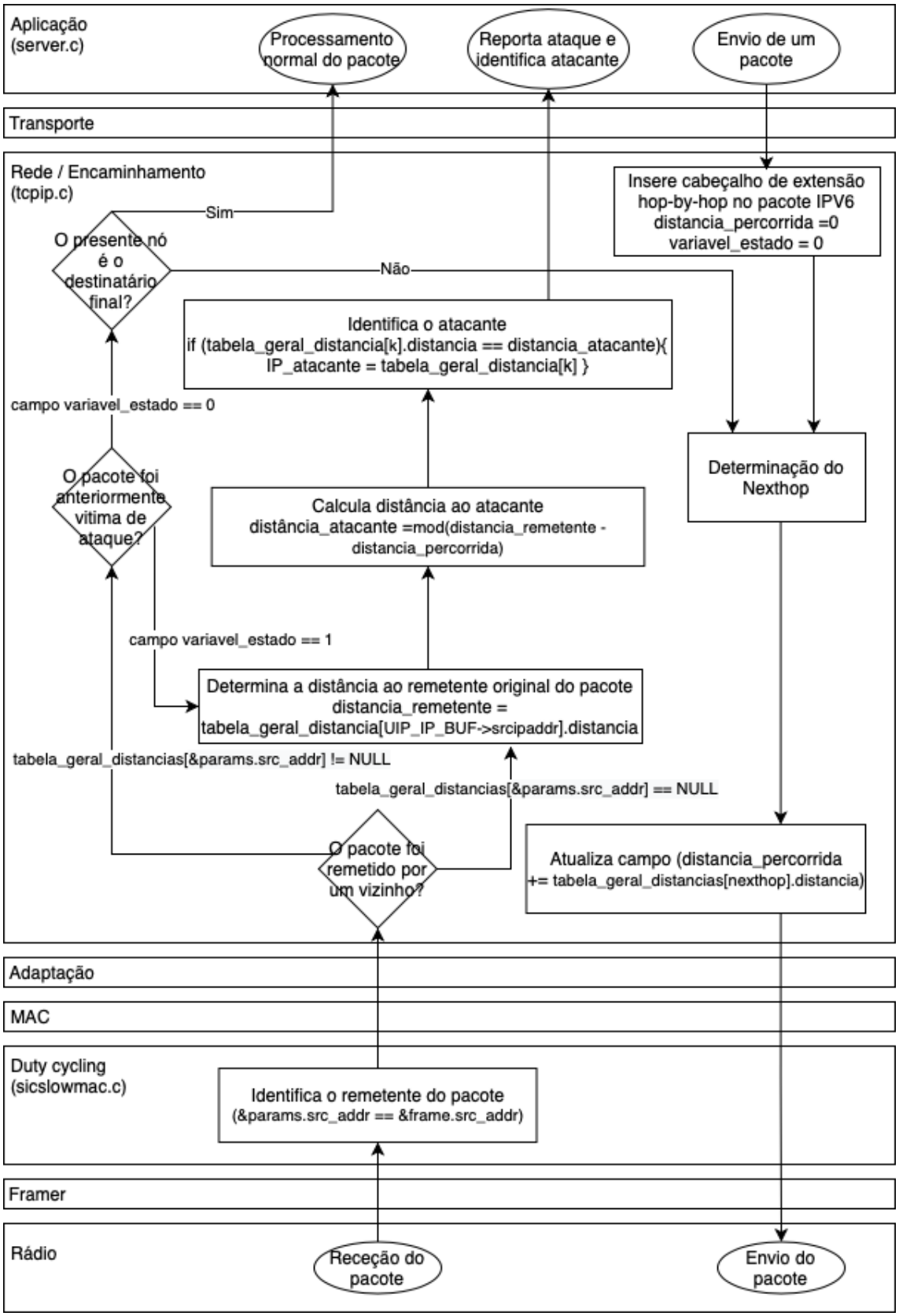

Figura 6.6: Fluxograma representativo do funcionamento do módulo centralizado

6.3 Ataques

Para efeitos de simulação foram desenvolvidos dois tipos de ataque *Wormhole*: por encapsulamento e por retransmissão de pacotes, ambos abordados na secção 2.7.9. Nas duas tipologias consideradas, a abordagem a seguir implicou que o desafio fosse encarado da perspetiva de um atacante. Posto isto, o principal objectivo, além da operacionalização propriamente dita dos ataques, foi que estes fossem dotados da máxima discrição, de modo a ultrapassar as malhas deste e de outros sistemas de deteção.

No seguimento dos objetivos traçados, optou-se por incluir o código malicioso numa zona mais baixa da pilha protocolar, isto para, no momento do ataque, o nó malicioso realizar o mínimo de operações sobre o pacote vítima a fim de sobre este deixar as mais ínfimas evidências da sua adulteração.

Assim sendo, os códigos de ataque foram implementados na camada RDC, nomeadamente, a nível do ficheiro *sicslowmac.c*.

De modo a tornar mais prática a ativação e desativação dos ataques durante o período de simulação, foi também alterada, ainda que superficialmente, a camada aplicacional, no ficheiro *client.c* e unicamente no domínio dos nós distribuídos.

A primeira ação a realizar foi, justamente, a definição de uma variável global no ficheiro *sicslowmac.c*. A variável em causa, *int i_ am_ atacker*, é inicializada a zero, no entanto, a sua incrementação em uma unidade está dependente da ocorrência de um evento do tipo *button_ sensor* no ficheiro *client.c*. Na prática o que acontece é que quando o botão do dispositivo é premido a variável toma o valor de um se estiver a zero, de dois se estiver a um e de zero, novamente, se estiver a dois. Este procedimento permite, portanto, o acionamento e inativação do tipo de ataque desejado e é, precisamente, a única operação realizada a nível aplicacional, ainda que, na verdade, não faça realmente parte do código malicioso por de trás dos ataques.

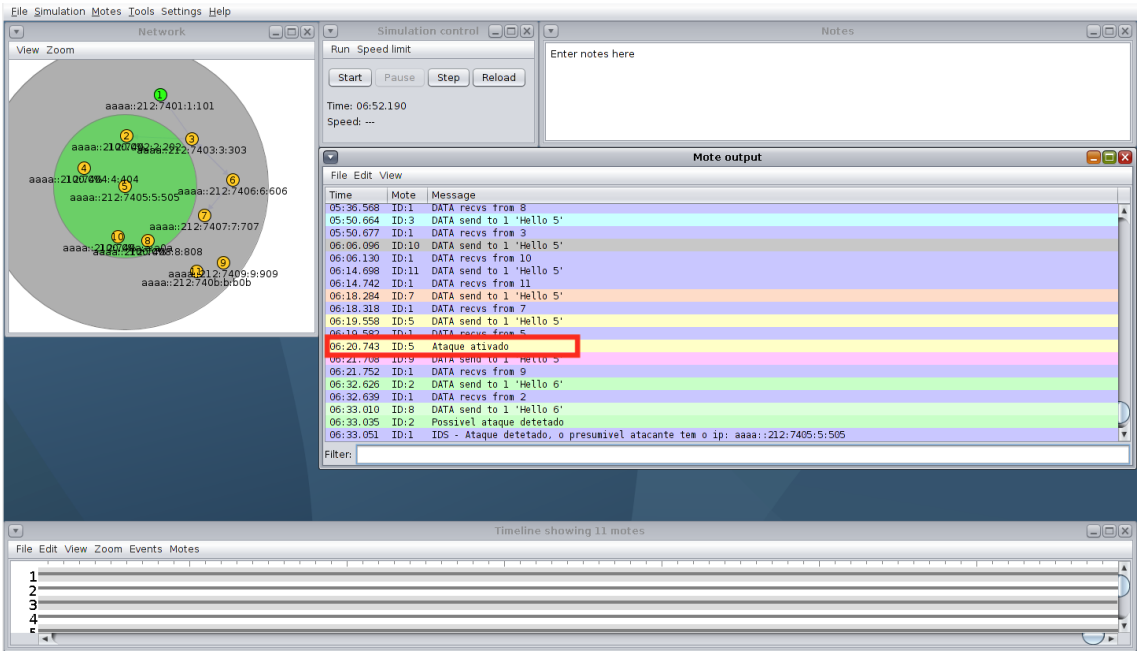

Figura 6.7: Mensagem informativa de ativação de ataque no nó 5

6.3.1 Ataque Wormhole por retransmissão de pacotes

Neste tipo de ataque, sempre que o botão do dispositivo seja premido e decorrente desse evento a variável *i_am_attacker* fique definida a "1", todos os pacotes que atinjam a camada RDC são imediatamente retornados à interface de radio de modo a serem novamente difundidos na rede sem que tenham sido devidamente processados pelo protocolo RPL. Desta forma, como apresentado no exemplo da figura 6.8, o nó malicioso (nó 2) quando recebe um pacote não atualiza o campo correspondente à sua identificação enquanto remetente o que, inerentemente, provoca a que quando o pacote vítima seja processado pelo próximo nó legitimo (nó 3) este considere que lhe foi remetido por um outro nó que nem tão pouco é seu vizinho, o nó 1.

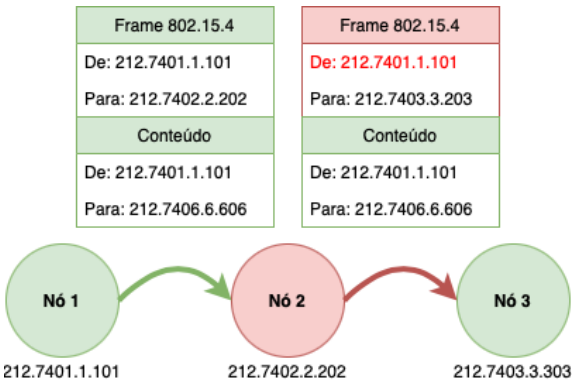

Figura 6.8: Implementação do ataque Wormhole do tipo retransmissão de pacotes

Numa discrição mais rigorosa, sempre que o ficheiro *sicslowmac.c* processa um pacote avalia se *i_am_attacker==1*. No caso desta condição se verificar, assegura-se que o endereço do remetente permanece inalterado no pacote (*(uint8_t*)&params.src_addr)==&frame.src_addr*) sendo, de seguida, encaminhado novamente para a interface rádio (*NETSTACK_RADIO.send*).

6.3.2 Ataque Wormhole por encapsulamento

Como o próprio nome indica, e no seguimento da explicação providenciada em 2.7.9, a tipologia de ataque *Wormhole* em foco requer o estabelecimento de um túnel malicioso através do qual o tráfego legitimo da rede será enviado a fim de provocar uma desorganização topológica constante. A potencialidade deste tipo de ataque é proporcional à extensão do túnel malicioso utilizado para o efeito. Neste sentido, ainda que o ataque pudesse ter sido operacionalizado por apenas 2 nós maliciosos optou-se por adicionar um 3º, com o objetivo de incrementar a extensão do túnel criado, isto porque, ciente da dificuldade, em cenário real, de utilizar um canal dedicado para o efeito, a extensão do túnel malicioso está dependente do alcance dos dispositivos e, consequentemente, do número de elementos envolvidos no ataque. No entanto, não é espectável que o sistema de deteção, à semelhança dos demais, identifique todos os nós maliciosos, mas apenas aqueles que configurem os nós extremidade do túnel, isto porque, os restantes operam de forma passiva e completamente impercetível. A imagem que se segue, 6.9 ajuda na compreensão da estratégia de implementação utilizada.

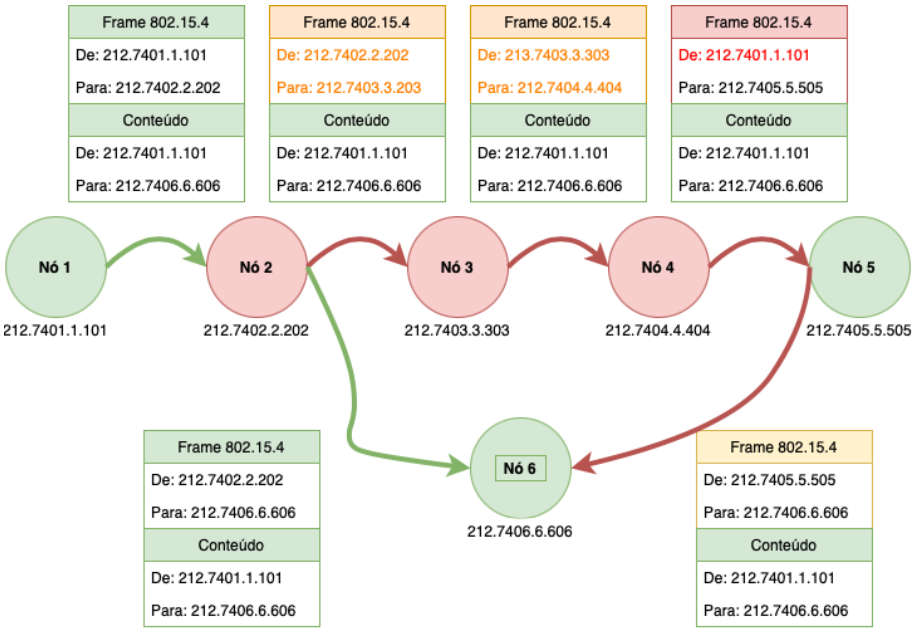

Figura 6.9: Implementação do ataque Wormhole do tipo por encapsulamento

Note-se que o primeiro nó malicioso (nó 2), aquando da receção de um pacote proveniente de um nó legitimo, impede que este seja processado pelo protocolo de encaminhamento RPL, como tal, este é imediatamente remetido para o segundo nó envolvido no ataque (nó 3), que, por sua vez, realiza exatamente a mesma operação e remete o pacote para o terceiro e último nó malicioso (nó 4), responsável por nova adulteração do pacote a fim de modificar o campo identificativo do último remetente, fazendo-lhe corresponder o endereço do último nó legitimo pelo qual o pacote foi processado (nó 1). Além disso, nesta fase é também alterado no pacote em trânsito o campo destinatário (*nexthop*), ao qual corresponderá o endereço de um qualquer vizinho legitimo deste último nó malicioso, neste caso o nó 5. Note-se que quando o pacote vítima atinge o nó 5 este irá considerar que o nó 1 é seu vizinho, além disso, como resultado, temos também que o pacote não foi encaminhado pela rota ótima (nó 1, nó 2, nó 6), mas sim por uma rota bastante mais extensa (nó 1, nó 2, nó 3, nó 4, nó 5, nó 6).

Numa abordagem mais técnica, quando um pacote atinge a camada RDC de um nó malicioso, no ficheiro *sicslowmac.c*, à semelhança do que acontece para a tipologia de ataque anterior, é verificado se a variável *i_am_attacker* está definida, neste caso, a "2". Caso esteja, é averiguado se o endereço do nó em causa corresponde a algum dos endereços atribuído às variáveis *(uint8_t)attacker1*, *(uint8_t)attacker2* ou *(uint8_t)attacker3*, definidas manualmente pelo utilizador no mesmo ficheiro. Mediante a correspondência, se o nó se enquadrar como *attacker1* ou *attacker2*, então este adopta uma postura de nó extremidade do túnel criado pela trilogia. Deste modo, se o endereço do remetente do pacote em processamento for igual ao endereço atribuído à variável *attacker3* este campo é alterado por forma a corresponder ao endereço do último nó remetente legitimo, já o destinatário do mesmo pacote, é definido com o endereço de um qualquer elemento da estrutura *ds6_neighbors*, constituída pelos vizinhos do presente nó malicioso. Todavia, se o endereço do remetente do pacote recebido não corresponder a nenhuma das variáveis, significa que é um pacote legitimo e que a atividade maliciosa contra este terá agora início. Para tal, o campo remetente do pacote é definido com o endereço do nó malicioso que o está a processar. Já o destinatário, corresponderá ao endereço contido na variável *attacker3*, de

modo a que o pacote seja transmitido para o nó intermediário do ataque.

Por fim, resta explicar o funcionamento do algoritmo na situação em que o endereço do nó seja equivalente ao endereço contido na variável *attacker3*. Nesta situação, sendo o elemento intermediário, interno ao túnel malicioso, a sua única função passa por difundir os pacotes remetidos por *attacker1* para o domínio do *attacker2*, ou vice-versa. O modo de funcionamento deste elemento, permite a inclusão de bilateralidade na operacionalização do ataque, uma vez que, este pode ter início em ambas as extremidades do túnel. A Figura 6.10 procura esquematizar a implementação e funcionamento das duas tipologias de ataque *Wormhole* consideradas.

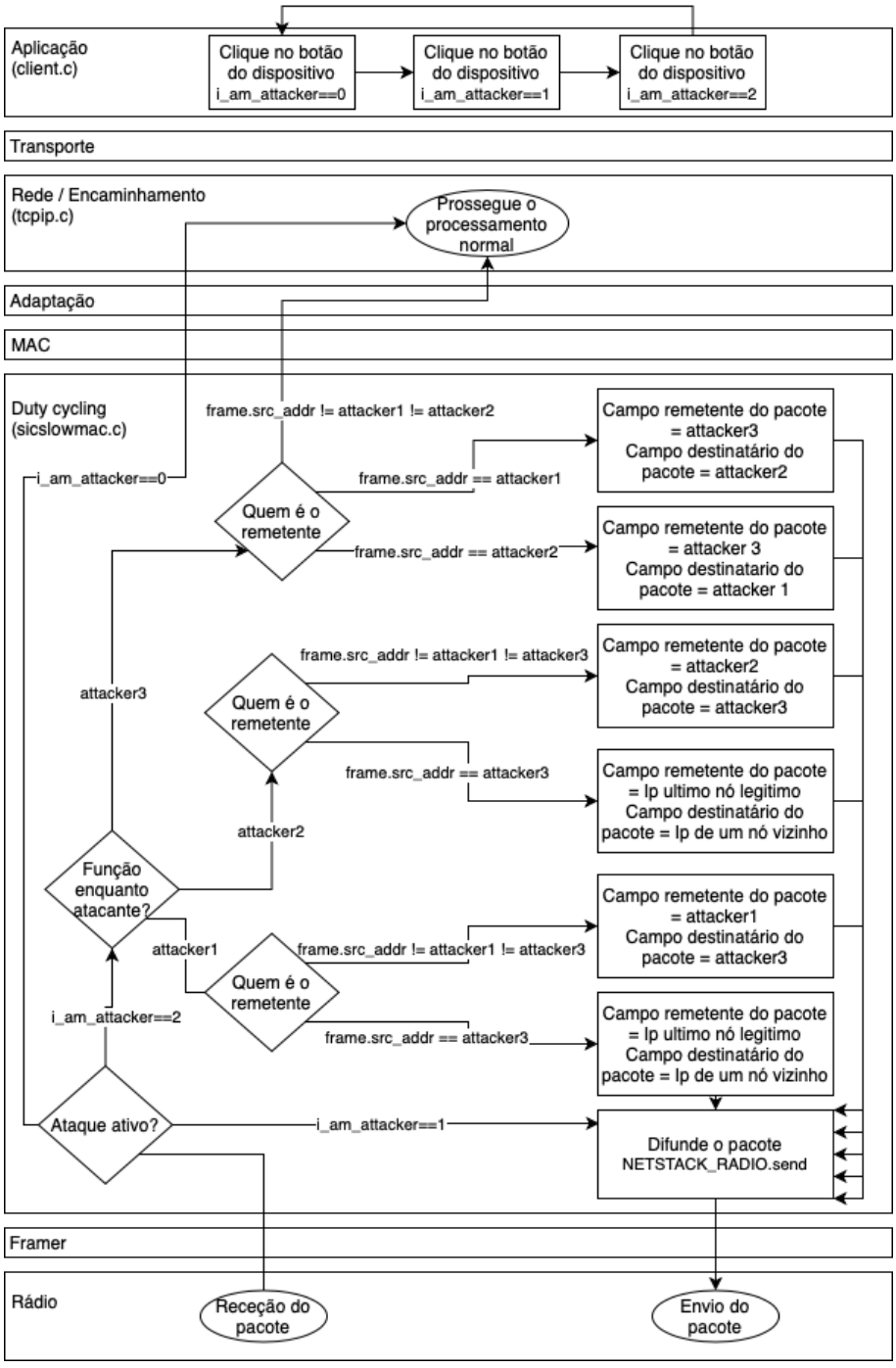

Figura 6.10: Fluxograma representativo do funcionamento dos ataques considerados

6.4 Síntese de capítulo

Na secção que agora finda, foram abordados, numa vertente mais técnica, as etapas decorrentes da implementação do sistema de deteção, bem como, clarificada a forma como os ataques *Wormhole*, tidos em consideração, foram direcionados sobre a topologia.

Dada a sensibilidade dos dispositivos inseridos na rede simulada, foi dado especial enfoque a todos os aspetos que contribuem para uma maior eficiência e modularidade da solução proposta.

Não menos importante, foi descrito o procedimento para inclusão de tráfego legitimo na rede, nomeadamente, por recurso a eventos gerados a cada intervalo de 30 segundos, responsáveis pelo envio de pacotes e pelo restauro do temporizador.

Em ambos os módulos do IDS (centralizado e distribuído) os métodos e estruturas a estes associados foram incluídos na camada de rede, a nível do ficheiro *tcpip.c*. Repare-se que este é um dos aspetos que mais contribui para a modularidade do sistema proposto, dada a transparência e independência com que o sistema de deteção opera da perspetiva da camada aplicacional.

Embora os dois módulos compartilhem funções, nomeadamente a de incrementação da distância percorrida a cada salto e avaliação da proximidade do remetente dos pacotes, o módulo centralizado possui métodos específicos, definidos nos ficheiros *tcpip.c* e *server.c*, que lhe permitem a identificação dos nós maliciosos. Inevitavelmente se a estrutura auxiliar nos módulos distribuídos apenas contempla a distância a cada um dos nós vizinhos, a estrutura anexa ao módulo centralizado remete para a totalidade dos nós inseridos na topologia. A necessidade de inclusão de novos indicadores nos pacotes que transitam na rede, implicou a adição de um cabeçalho de extensão do tipo *hop-by-hop* no pacote IPv6, isto porque, o processamento desta estrutura é realizado na mesma zona protocolar em que o IDS opera, a camada de rede. Além disso, são igualmente neste domínio realizadas as tarefas de encaminhamento, a partir das quais é possível determinar o *nexthop*, elemento necessário à incrementação da distância percorrida pelos pacotes de acordo com a entrada correspondente nas estruturas auxiliares de ambos os módulos.

Relativamente aos ataques direcionados sobre a rede, para fins de avaliação do sistema de deteção, foram desenvolvidas duas tipologias distintas, ambas operacionalizadas no domino da camada RDC, no ficheiro *sicslowmac.c*. Se na modalidade por retransmissão de pacotes estes são imediatamente retornados para a camada de radio para nova difusão, na modalidade por encapsulamento, são utilizados 3 nós maliciosos em conluio, sendo que ao pacote vítima é adulterado o campo correspondente ao endereço do próximo salto a fim deste coincidir com o segundo elemento envolvido no ataque, que, por sua vez, irá realizar a mesma operação sendo agora o pacote vítima remetido para o 3º e último nó malicioso que o difundirá na sua vizinhança. Atente-se que ao recorrer a 3 nós nesta modalidade de ataque torna-se possível a criação de um túnel malicioso mais extenso, do qual resulta um ataque mais impactante.

Esta página foi propositadamente deixada em branco.

Capítulo 7

Análise dos resultados

Nesta secção são apresentados os resultados da avaliação realizada ao modelo de deteção proposto. Esta avaliação visou o apuramento da capacidade de deteção de ataques por parte do IDS, aferiu a sua assertividade na identificação de nós maliciosos e analisou o impacto energético anexo à solução.

De forma a garantir a solidez dos valores apresentados, será ainda, no decorrer deste capítulo, clarificada a forma como estes foram obtidos.

As conclusões emergentes da análise dos resultados incluem uma forte componente autocrítica, procurando identificar eventuais pontos de falha e justificá-los de acordo com a dinâmica operacional do sistema de deteção.

7.1 Cenários de avaliação

De modo a simplificar a interpretação dos resultados, é importante clarificar, de uma perspetiva mais prática e visual, os cenários tidos em consideração em cada uma das simulações realizadas.

Começando pelos testes realizados para aferição do desempenho na deteção e identificação de nós maliciosos, foram considerados 3 cenários, cada um deles simulado para um intervalo de 30 minutos, estando o IDS em pleno funcionamento.

O primeiro, representado na figura 7.1, é constituído por 10 nós do tipo *TmoteSky* associados ao BR. Durante a simulação, mais concretamente ao minuto 6, o nó 5 adotará um comportamento malicioso, devido ao acionamento do ataque por retransmissão de pacotes. Com o objetivo de dificultar o trabalho do IDS, o ataque estará ativo durante 2 minutos, tendo sido, posteriormente, desativado. Este procedimento de ataque foi novamente executado ao minuto 10 sendo, desta vez, o atacante o nó 7.

Ainda neste primeiro cenário, no decurso do minuto 14, foi incluída a tipologia de ataque por encapsulamento, sendo os nós 2, 3 e 6 responsáveis pela criação do túnel malicioso.

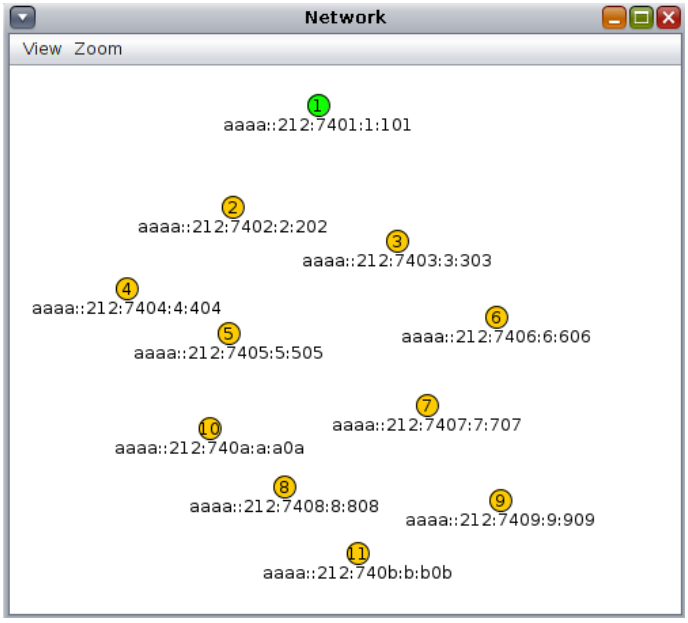

Figura 7.1: Cenário de testes constituído por 10 nós

O segundo cenário é constituído por 20 nós do tipo *TmoteSky*, dos quais, os nós 7, 8, 19 e 3, aos minutos 3, 4, 6 e 7, respetivamente, operacionalizam 4 ataques *Wormhole* do tipo por retransmissão de pacotes.

Por sua vez, no mesmo cenário, os nós 5, 12 e 6, ao minuto 9, os nós 10, 5 e 12 ao minuto 12 e os nós 19, 10 e 4, ao minuto 17, configuram, cada um destes conjuntos, 1 ataque do tipo por encapsulamento. Este cenário encontra-se representado na figura 7.2.

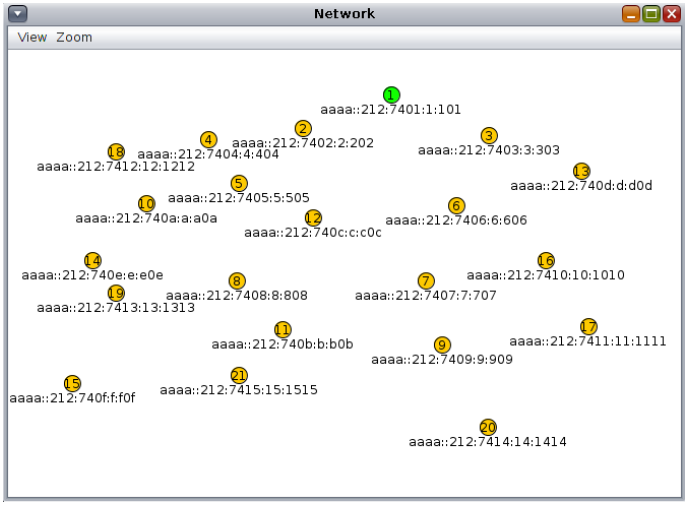

Figura 7.2: Cenário de testes constituído por 20 nós

Por fim, foi simulado um ambiente com 30 nós do tipo *TmoteSky*, dos quais os nós 4, 19, 17, 27 e 10 operacionalizam 5 ataques do tipo por retransmissão de pacotes e as trilogias 16, 17, 24; 26,16,7; 10,18,28; 5,4,12 e 17,26,6 retratam a operação de 5 ataques *Wormhole* do tipo por encapsulamento, como apresentado na figura 7.3.

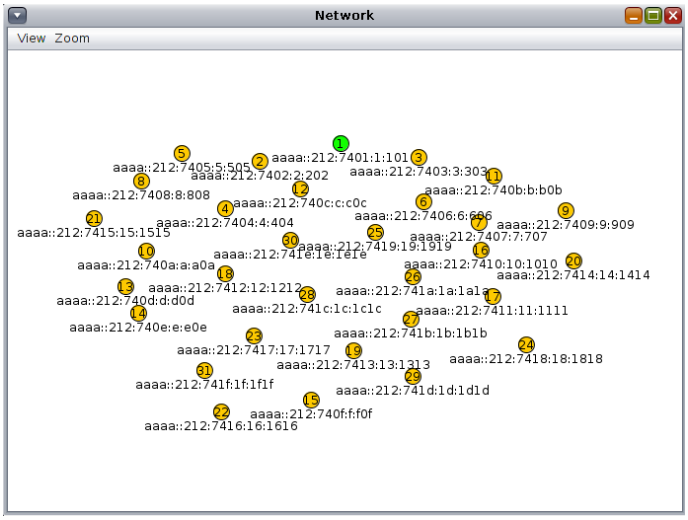

Figura 7.3: Cenário de testes constituído por 30 nós

De uma perspetiva geral, foram simulados, no conjunto dos 3 ambientes, 11 ataques por retransmissão de pacotes, num total de 11 nós maliciosos e 9 ataques por encapsulamento, estando associados a esta modalidade de ataque um total de 27 nós maliciosos.

Por fim, com o objetivo de avaliar o impacto energético do sistema de deteção proposto, foram novamente utilizados os 3 ambientes com 10, 20 e 30 nós, tendo sido, cada um destes, simulado para dois diferentes cenários, com e sem o IDS em funcionamento.

É importante reter que no decorrer desta fase de avaliação não foi incluído qualquer nó malicioso nas topologias, isto porque, não é desejável que os dados energéticos recolhidos sejam sujeitos a perturbações, nomeadamente, resultantes da atividade protocolar adicional motivada pela ocorrência de um ataque.

7.2 Capacidade de deteção de ataques e identificação dos nós maliciosos

A avaliação do modelo proposto compreendeu duas vertentes. A primeira delas, passa pela averiguação da solidez do mesmo na atividade de classificação dos eventos ocorridos na rede que constituam um ataque. A segunda, por sua vez, procura avaliar o desempenho do IDS na identificação dos nós maliciosos envolvidos em cada um dos ataques reportados.

No decorrer desta secção será avaliado, individualmente, o comportamento do sistema de deteção em cada um dos ambientes simulados, culminando esta avaliação com o relacionamento de todos os dados obtidos e determinação das métricas propostas em 5.4.

Começando pelo primeiro cenário, onde, como já referido, estão incluídos 2 ataques do tipo por retransmissão de pacotes e 1 ataque do tipo por encapsulamento, o presente IDS foi capaz de identificar de forma assertiva todos os ataques, assim como, identificar todos os atacantes a estes associados. Estes resultados podem ser comprovados por observação da figura 7.4.

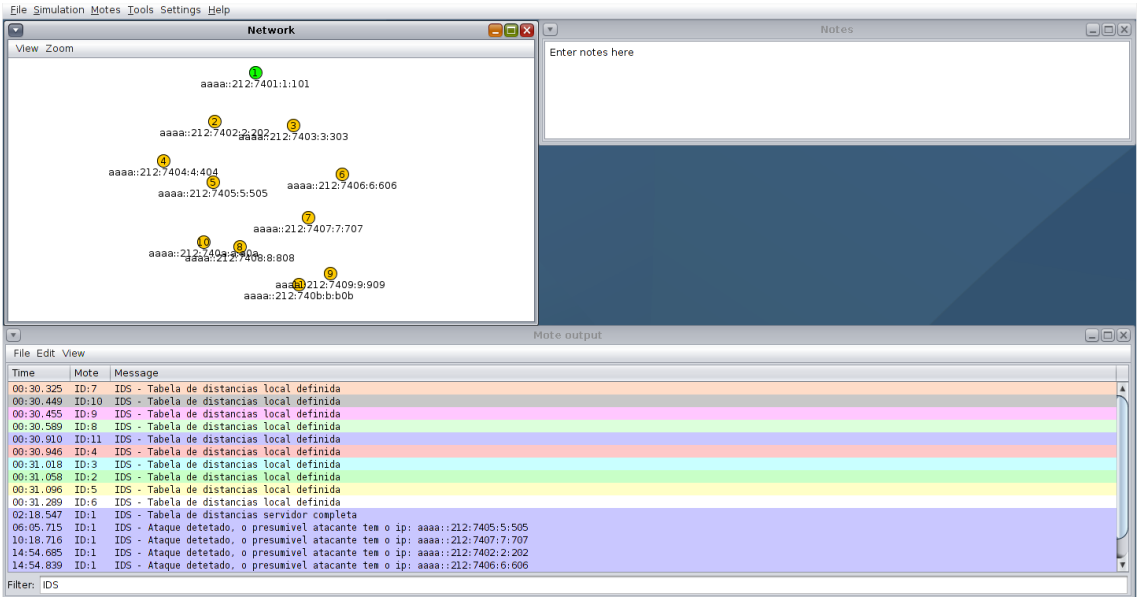

Figura 7.4: Comportamento do modelo quando simulado no 1º cenário

No segundo cenário, todos os 4 ataques por retransmissão de pacotes foram reportados, tal como, os 3 ataques por encapsulamento. No que diz respeito à capacidade de identificação dos nós maliciosos, os 3 associados aos ataques por retransmissão de pacotes foram, inequivocamente, referenciados. Já para os nós maliciosos envolvidos nos ataques por encapsulamento, dos 6 nós inseridos na rede com este comportamento, apenas 5 foram reportados. O caso de falha corresponde ao ataque estabelecido por intermédio do túnel criado entre os nós 10 e 12, onde uma das extremidades, o nó 12, não foi identificada. No entanto, o ataque foi reportado, assim como o outro agente envolvido, o nó 10, foi corretamente referenciado, como é perceptível na imagem 7.5

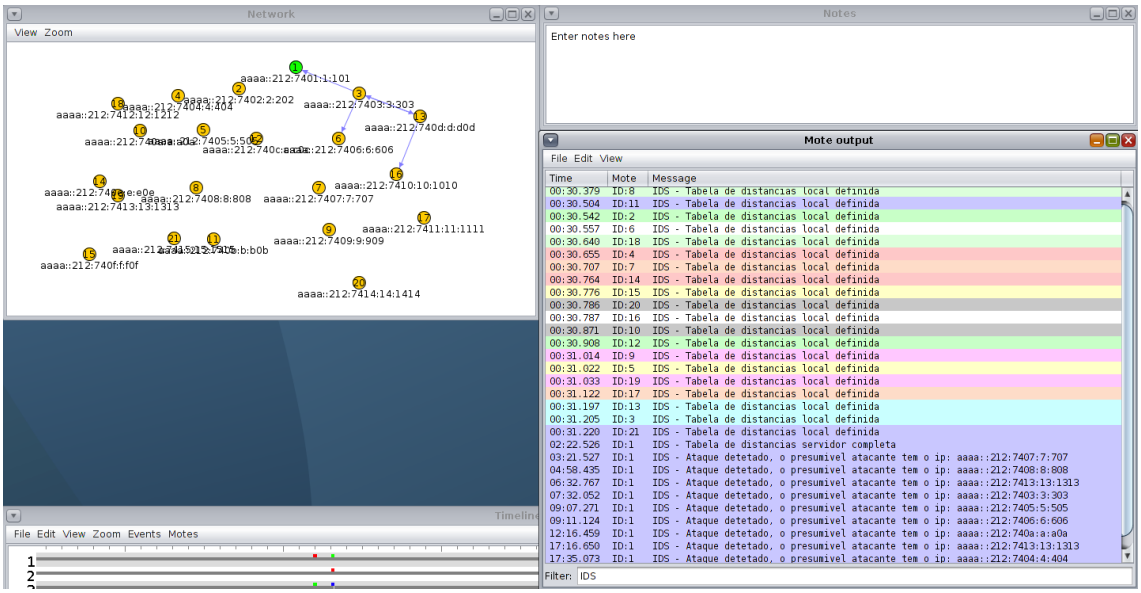

Figura 7.5: Comportamento do modelo quando simulado no 2º cenário

Por fim, no 3º cenário, integrante de 5 ataques do tipo por retransmissão de pacotes e 5 do tipo por encapsulamento, mais uma vez, todos os ataques foram identificados. Relativamente à identidade dos atacantes, o sistema de deteção conseguiu que esta fosse reportada na generalidade dos casos. Apenas um dos nós envolvidos no ataque por encapsulamento, estabelecido entre os nós 16 e 24 não o foi, nomeadamente, o nó 24, como demonstra a Figura 7.6.

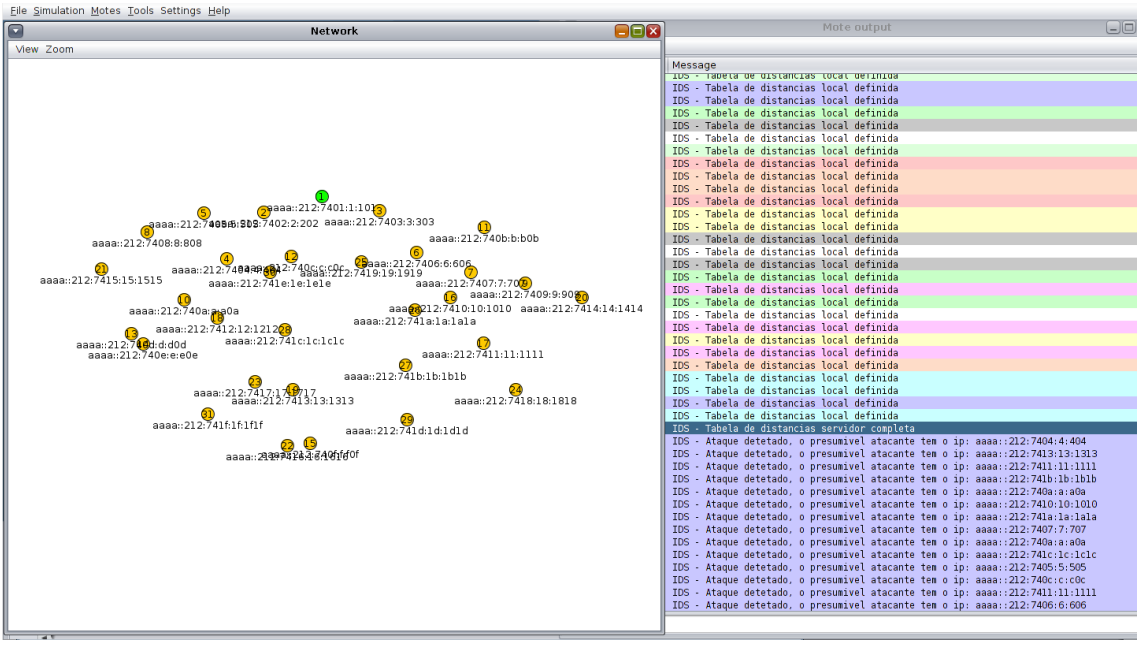

Figura 7.6: Comportamento do modelo quando simulado no 3º cenário

Da agregação da informação veiculada pela simulação dos 3 ambientes e do posterior relacionamento da mesma pelas formulas associadas a cada métrica, propostas em 5.4, para a problemática da capacidade de deteção de ataques, foi possível apurar os resultados contidos na Tabela 7.1.

Deteção de ataque				
Cenário	**1**	**2**	**3**	**Total**
Nº de ataques	3	7	10	20
Verdadeiros Positivos	3	7	10	20
Falsos Positivos	0	0	0	0
Falsos Negativos	0	0	0	0
Taxa de Falsos Positivos	0	0	0	0
Taxa de Falsos Negativos	0	0	0	0
Taxa de Verdadeiros Negativos	1	1	1	1
Precisão	1	1	1	1
Recall	1	1	1	1
Exatidão	1	1	1	1
F-Measure	100	100	100	100

Tabela 7.1: Tabela sumária de avaliação da capacidade de deteção de ataques Wormhole

Naturalmente que tendo sido todos os ataques corretamente reportados e na inexistência de quaisquer falsos positivos, os resultados não poderiam ter sido mais satisfatórios. Esta afirmação é facilmente comprovável com a observação direta das métricas Precisão, Exatidão, *F-measure* e *Recall*. Repare-se que todos estes indicadores apresentam valores máximos, implicando que nos testes realizados nenhuma ocorrência suscitou um comportamento erróneo por parte do IDS.

No que concerne à capacidade de identificação dos nós maliciosos envolvidos nos ataques, pelo relacionado dos dados obtidos, foi possível calcular as métricas presentes na Tabela 7.2.

Identificação dos atacantes				
Cenário	**1**	**2**	**3**	**Total**
Nº de atacantes	4	10	15	29
Verdadeiros Positivos	4	9	14	27
Verdadeiros Negativos	6	10	15	31
Falsos Positivos	0	0	0	0
Falsos Negativos	0	1	1	2
Taxa de Falsos Positivos	0	0	0	0
Taxa de Falsos Negativos	0	0.10	0.07	0.07
Taxa de Verdadeiros Negativos	1	1	1	1
Precisão	1	1	1	1
Recall	1	0.90	0.93	0.93
Exatidão	1	0.95	0.97	0.97
F-Measure	1	94.74	96.55	96.43

Tabela 7.2: Tabela sumária de avaliação da capacidade de identificação de nós maliciosos

Da análise dos dados é percetível que no que respeita à atividade de identificação de nós maliciosos o IDS apresenta um *recall* de 93%, aliado a uma Exatidão e *f-measure* de 97 e 96%, respetivamente, situando-se as restantes métricas consideradas no valor máximo, ou ótimo, se assim lhe quisermos chamar. Este facto resulta da incapacidade de identificação de um nó extremidade associado a um ataque *Wormhole* do tipo por encapsulamento no cenário 3 e por situação idêntica ocorrida no cenário 2, o que justifica também os 2 falsos negativos documentados.

Embora esta inconsistência apenas se tenha verificado em 2 dos 29 nós maliciosos integrantes dos 3 ambiente simulados, considerou-se pertinente a sua particularização, tendo sido possível concluir que ambas as situações erróneas refletem o mesmo caso de falha, permitindo-nos, inclusive, a sua generalização.

O sistema de deteção em estudo é incapaz de proceder à identificação de um nó extremidade, associado a um ataque *Wormhole* do tipo por encapsulamento, nas circunstâncias especificas em que esse nó seja um nó folha, ou seja, não possua nenhum outro nó a si associado hierarquicamente inferior. Repare-se que nem o nó 12 no cenário 2 nem o nó 24 no cenário 3 possuem qualquer nó nestas condições, não tendo por isso sido referenciados como atacante, ainda que o fossem.

Do ponto de vista funcional do modelo, esta situação é justificada pelo facto de no caso de um nó não possuir descendência, em nenhum momento este irá intermediar qualquer transação e, portanto, nenhum pacote será remetido por si para a outra extremidade do túnel malicioso. Na prática, o que acontece é que o túnel entre os nós 10 e 12 no cenário 2 e o túnel estabelecido entre os nós 16 e 24 no cenário 3, têm um comportamento unidirecional,

no sentido em que apenas os pacotes remetidos por 10 e 16, respetivamente, são alvo de ataque, uma vez que, o tráfego no sentido oposto é inexistente, implicando que as respetivas extremidades opostas (12 e 24) adotem um comportamento passivo durante o ataque.

Ainda assim, se os nós 12 e 24 remetessem os pacotes originários no seu domínio por intermédio dos túneis maliciosos por si criados, estes seriam, obviamente, identificados como atacantes pelo presente IDS. No entanto, embora admissível, em cenário real, este comportamento representa uma abordagem de ataque pouco criativa, no sentido em que a exposição do atacante seria uma constante, dai não ter sido a forma escolhida para implementação desta tipologia de ataque *Wormhole*.

Não posso deixar de frisar que ainda que nas duas situações anteriores não tenha sido possível proceder à identificação de um dos nós envolvidos na criação do túnel malicioso, os ataques foram corretamente detetados. Além disso, a identificação de um único nó envolvido na operacionalização de ataques *Wormhole* do tipo por encapsulamento é suficiente para a sua supressão. Neste seguimento, se atentarmos os casos concretos de falha nos cenários 2 e 3, tendo sido em ambos o ataque reportado assim como o nó extremidade com comportamento ativo no decurso do mesmo identificado, é possível suspender a operação de ambos os túneis.

Numa alusão ao cenário de aplicação real, descrito em 5.1, fica claro que a inclusão do mecanismo de segurança em foco seria altamente benéfico, dada a sua solidez na deteção de ataques *Wormhole* e a sua assertividade no mapeamento dos nós maliciosos envolvidos nos mesmos. Desta forma, seria possível a contenção prematura de um um potencial ataque, antes mesmo das suas consequências serem percetíveis no meio de cultura.

Não sendo vitais à compreensão de nenhuma das componentes da avaliação levada a cabo, encontram-se no apêndice 8 os quadros representativos da aplicação individual das métricas propostas para cada tipo de ataque *Wormhole* tido em consideração, por encapsulamento e por retransmissão de pacotes.

7.3 Impacto energético da solução

Os resultados obtidos mostraram haver um ligeiro aumento do consumo energético nos cenários em que o IDS está em funcionamento, comparativamente com o mesmo ambiente na ausência deste mecanismo, como demonstra o gráfico da Figura 7.7. Ressalve-se que os dados utilizados na criação deste elemento, documentados no apêndice 8, provêm do relacionamento do output da ferramenta *Powertrace* nos moldes do que foi clarificado em 5.4.

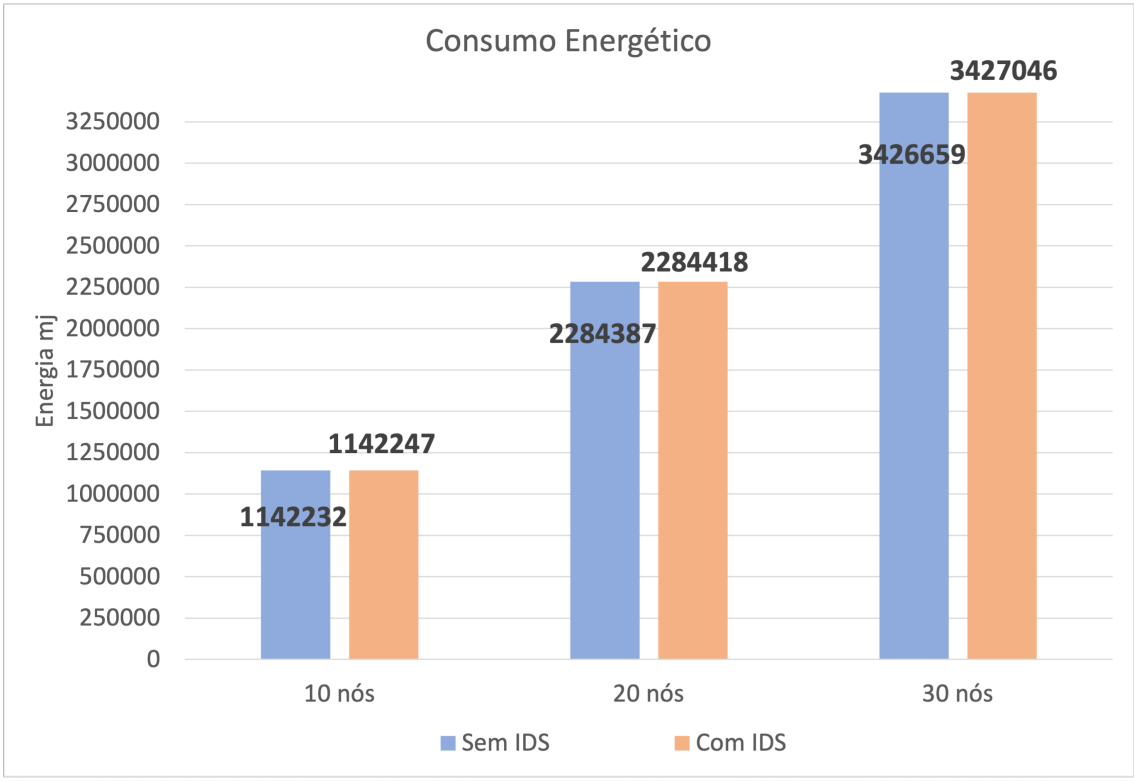

Figura 7.7: Consumo energético total durante 30 minutos de simulação

Do ponto de vista do seu significado, o gráfico apresentado na figura 7.7 representa o consumo energético total de cada um dos 3 ambientes, nos cenários com e sem o IDS em funcionamento, aferido decorridos 30 minutos de simulação.

Numa analise mais rigorosa aos dados apresentados, podemos concluir que nos cenários com 10 e 20 nós, com o IDS em funcionamento, o consumo energético total é 0.001% superior ao indicador equivalente aferido nos mesmos ambientes, mas na ausência do sistema de deteção. Já no cenário constituído por 30 nós, o aumento do consumo energético na presença do IDS é de 0.01%. Posto isto, em média, a inclusão do sistema de deteção proposto tem associado um custo energético adicional de 0.004%.

Como era expectável, o impacto energético da solução não é linear nem tão pouco proporcional ao número de elementos constituintes de uma rede. Obviamente que à medida que incluímos mais elementos numa topologia, maior será o consumo energético a esta associado. Contudo, não se verifica que o impacto energético do modelo proposto siga esta tendência de proporcionalidade, ainda que, o número de nós da topologia seja uma variável a considerar.

Neste seguimento, é possível afirmar que as variações do impacto energético apurado para os 3 ambientes simulados estão relacionadas com duas variáveis. A primeira delas, já abordada, é o número de nós que constituem as topologias onde o IDS está a operar, isto porque, quantos mais nós integrantes das mesmas, maior será a estrutura auxiliar anexa ao módulo centralizado. Repare-se que o preenchimento desta estrutura requer um esforço computacional adicional, motivado pelo processamento das variáveis necessárias à operação e pela própria alocação dos elementos à respetiva tabela. Posto isto, se partirmos do pressuposto que ambas as atividades são energeticamente dispendiosas, obviamente que o incremento do período de tempo que os nós incorrem nestas operações, resulta num aumento do seu consumo energético total. Esta constatação é igualmente válida nas operações de consulta, realizadas aquando da atualização da distância percorrida pelos pacotes, ou no decurso da identificação de nós maliciosos. Note-se que dependendo da dimensão da estrutura, o número de comparações necessárias a fim de encontrar um elemento específico é variável.

O segundo elemento contributivo é justamente a densidade das topologias. Uma topologia mais densa implica que os nós constituintes possuam um maior número de vizinhos. Assim sendo, e no seguimento do raciocínio anterior, é possível afirmar que a definição das estruturas associadas aos módulos distribuídos, bem como, a sua consulta irá ser mais desgastante, no sentido em que estas terão mais elementos. Inerentemente, também o impacto energético será, nesta situação, superior.

Seria imprudente realizar uma avaliação a este modelo sem dar a atenção necessária à componente operacional mais impactada com a inclusão deste sistema de deteção. Numa alusão à secção 5.4, recorde-se que a atividade operacional de um dispositivo é dividida em 4 módulos: CPU, LPM, Escuta e Transmissão.

Num primeiro ponto, e dada a morfologia do modelo de deteção proposto, é possível afirmar que os módulos de Escuta e Transmissão sofrem um impacto diminuto, isto porque, em fase alguma da operação do IDS, este recorre ao envio e inerente escuta de pacotes adicionais na rede, excetuando o período de inicialização, com menor relevo se a atribuição das estruturas for feita manualmente, não tendo, por isso, esta fase sido considerada durante a avaliação realizada.

Desta feita, as atenções voltam-se para os módulos de CPU e LPM, sendo que, o estudo deste último é menos pertinente, uma vez que depende dos 3 restantes. Claro está que quanto mais tempo um nó estiver a executar atividades no domínio do módulo de CPU, Escuta ou Transmissão, menos tempo estará em modo de baixa potência (LPM).

Face ao exposto, é imediato que o impacto do IDS se faz sentir, essencialmente, no módulo de CPU. Desta feita, este elemento será agora estudado em mais detalhe.

O gráfico da Figura 7.8 é representativo do consumo energético por parte do módulo de CPU, aferido decorridos 30 minutos de simulação de cada um dos três ambientes considerados, com e sem o IDS em funcionamento.

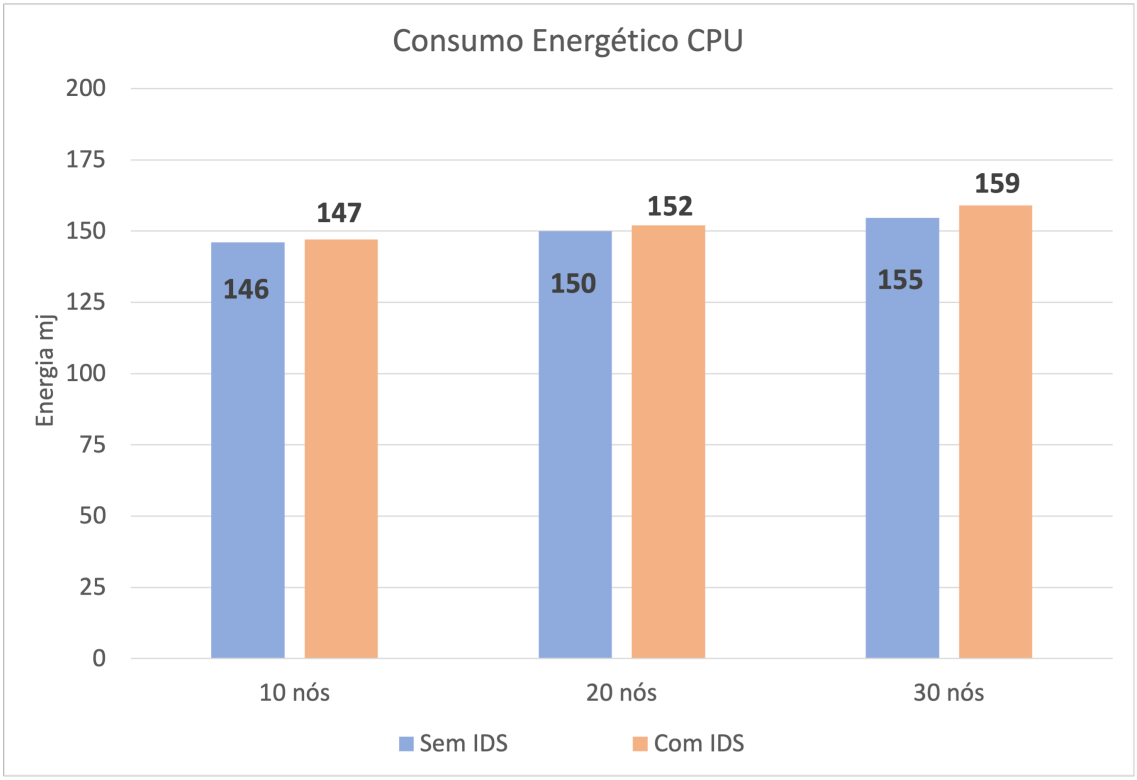

Figura 7.8: Consumo energético do módulo de CPU durante 30 minutos de simulação

A análise destes resultados vem evidenciar que a definição e consulta das estruturas auxiliares, assim como, o processamento dos campos adicionais nos pacotes são as atividades mais contributivas para o aumento do consumo energético quando incluído o IDS numa rede. Note-se que todas estas operações incidem diretamente sobre o módulo de CPU. Fica igualmente claro que o número de nós e a densidade destes elementos são as variáveis com maior influência na oscilação do impacto energético nos ambientes simulados. Ainda assim, não podemos desassociar a proporcionalidade existente entre o número de nós e o número de mensagens que circulam na rede. Repare-se que quantas mais mensagens são trocadas, mais pacotes são processados a nível dos módulos distribuídos e centralizado do IDS, sendo os nós hierarquicamente superiores os mais impactados com esta situação.

De uma perspetiva mais prática, do ponto de vista energético, os resultados obtidos viabilizam a inclusão da solução proposta no cenário real de aplicação descrito em 5.1.

Numa clara extrapolação teórica, se assumirmos que um dispositivo, alimentado por bateria e integrante deste tipo de ambiente, tem um período de operacionalidade de 90 dias, transpondo-o para um cenário na presença do sistema de deteção proposto, esta seria reduzida, sensivelmente, em 8 horas e 40 minutos. Esta redução é claramente admissível, principalmente quando equacionadas as perdas resultantes de um potencial ataque.

Esta página foi propositadamente deixada em branco.

Capítulo 8

Conclusões e oportunidades de investigação futura

Abordar nos dias que correm o termo IoT remete não só para a tecnologia, propriamente dita, mas para toda uma tendência evolucionária de informatização a nível global.

A transversalidade aplicacional da Internet das coisas enfatiza a necessidade de aprofundar o estudo em novas estratégias de segurança que contribuam para o enraizar e desenvolvimento desta área.

O contributo do trabalho aqui documentado passou pelo desenvolvimento de um novo modelo para a deteção de ataques *Wormhole* em ambientes IoT.

Este tipo de ataque, com enfoque no protocolo de encaminhamento RPL, apresenta-se como altamente nefasto para a rede sobre a qual é direcionado. Se de forma mais imediata o ataque em causa provoca uma desorganização topológica constante, em momento algum podemos deixar de considerar o efeito cascata posterior. Note-se que, as constantes alterações hierárquicas dos nós e da sua vizinhança, motivam a um aumento das mensagens protocolares de reajuste topológico, podendo estas ser determinantes para a ocorrência de um congestionamento na rede que provoque uma situação de colapso da mesma, ou sérios atrasos na propagação de mensagens legitimas.

A estratégia para deteção de ataques *Wormhole* proposta, passa pela avaliação da proveniência dos pacotes e pela inclusão de novos campos nas mensagens que circulam na rede, por forma a ser possível determinar a distância percorrida pelos pacotes no trajeto entre o respetivo emissor e recetor. Esta distância será, posteriormente, comparada com a distância tida como referência para esse percurso, a fim de encontrar desvios que possam estar associados a um ataque e permitam proceder à identificação dos nós maliciosos envolvidos.

A avaliação realizada ao modelo, por intermédio da ferramenta *Cooja*, demonstrou resultados bastante satisfatórios. Do ponto de vista da capacidade de deteção, o IDS proposto foi capaz de reportar todos os ataques *Wormhole* direcionados sobre os vários ambientes simulados, sendo estes do tipo por retransmissão de pacotes e encapsulamento, sem que se tenha verificado qualquer falso positivo, motivando a que as métricas Exatidão, Precisão, *Recall* e *F-measure* tenham apresentado valores máximos (100%).

No que respeita à capacidade de identificação dos nós maliciosos, o sistema de deteção em estudo apenas não consegui identificar 2 atacantes do total de 29 incluídos nas várias simulações realizadas. Ainda assim, foi possível concluir que esta situação de falha se deve à inexistência de descendência dos nós extremidade do túnel malicioso, associado a

um ataque *Wormhole* do tipo por encapsulamento. No entanto, mesmo aquando desta ocorrência, o ataque é reportado, bem como, identificado um dos dois nós envolvidos no mesmo, sendo o suficiente, nestas condições de ataque, para a supressão do evento malicioso. Sumariamente, da avaliação desta componente resultou um *Recall* de 93%, Precisão de 100%, Exatidão de 97%, *F-measure* de 96% e nenhum falso positivo reportado.

Do ponto de vista da eficiência energética, a avaliação ao modelo, por recurso à ferramenta *Powertrace*, mostrou que a inclusão do sistema de deteção contribui para um aumento do consumo energético total da rede de 0.004%, sendo este valor perfeitamente admissível nos cenários de operação considerados.

O trabalho realizado abre portas a uma série de questões potenciadoras de investigação futura, que visem, por exemplo, a sua continuidade. Repare-se que com a adição de novos campos nos pacotes, torna-se possível a utilização dos indicadores a estes associados para a deteção de outras modalidades de ataque, como por exemplo, *Neighbor attack*, *Sybil* ou *Routing Information Replay Attack*. No que concerne ao domínio aplicacional do sistema de deteção, considero ainda pertinente o desenvolvimento de estratégias que possibilitem a operação deste modelo em redes dotadas de mobilidade.

Referências

[1] M. S. Ahsan, M. N. M. Bhutta, and M. Maqsood. Wormhole attack detection in routing protocol for low power lossy networks. In *2017 International Conference on Information and Communication Technologies (ICICT)*, pages 58–67, 2017.

[2] Roger Alexander, Anders Brandt, JP Vasseur, Jonathan Hui, Kris Pister, Pascal Thubert, P Levis, Rene Struik, Richard Kelsey, and Tim Winter. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, March 2012.

[3] J P Anderson. Computer security threat monitoring and surveillance. *Technical Report James P Anderson Co Fort Washington Pa*, page 56, 1980.

[4] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, pages 2787–2805, 10 2010.

[5] Marianne Azer, Sherif El-Kassas, and Magdy El-Soudani. A full image of the wormhole attacks-towards introducing complex wormhole attacks in wireless ad hoc networks. *arXiv preprint arXiv:0906.1245*, 1, 06 2009.

[6] Lorenzo Bartolozzi, Tommaso Pecorella, and Romano Fantacci. ns-3 RPL module: IPv6 Routing Protocol for Low power and Lossy Networks. In *WNS3 - Workshop on ns-3*. ACM, 6 2012.

[7] Ugur Bekcibasi. Increasing rssi localization accuracy with distance reference anchor in wireless sensor networks. *Acta Polytechnica Hungarica*, 11:103–120, 08 2014.

[8] Bruno Bogaz Zarpelão, Rodrigo Miani, Cláudio Kawakani, and Sean Alvarenga. A survey of intrusion detection in i nternet of things. *Journal of Network and Computer Applications*, 84, 02 2017.

[9] Alessio Botta, Walter Donato, Valerio Persico, and Antonio Pescapè. Integration of cloud computing and internet of things: A survey. *Future Generation Computer Systems*, 56, 10 2015.

[10] Muhammad Burhan, Rana Asif Rehman, Byung-Seo Kim, and Bilal Khan. Iot elements, layered architectures and security issues: A comprehensive survey. *Sensors*, 18, 08 2018.

[11] C. Cervantes, D. Poplade, M. Nogueira, and A. Santos. Detection of sinkhole attacks for supporting secure routing on 6lowpan for internet of things. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 606–611, 2015.

[12] Bipasha Chakravarty et al. Trends in mushroom cultivation and breeding. *Australian Journal of Agricultural Engineering*, 2(4):102, 2011.

[13] Moteiv Coorperation. Ultra low power ieee 802.15.4 compliant wireless sensor module. Technical report, moteiv, 2006.

[14] Steve Deering, Robert Hinden, et al. Internet protocol, version 6 (ipv6) specification, 1998.

[15] Adam Dunkels, Joakim Eriksson, Niclas Finne, and Nicolas Tsiftes. Powertrace: Network-level power profiling for low-power wireless networks, 04 2011.

[16] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki-a lightweight and flexible operating system for tiny networked sensors. In *29th annual IEEE international conference on local computer networks*, pages 455–462. IEEE, 2004.

[17] Caglar Durmaz, Moharram Challenger, Orhan Dagdeviren, and Geylani Kardas. Modelling contiki-based iot systems. In *6th symposium on languages, applications and technologies (SLATE 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[18] Olfa Gaddour and Anis Koubâa. RPL in a nutshell: A survey. *Computer Networks*, 56(14):3163–3178, 2012.

[19] Omprakash Gnawali and P Levis. The Minimum Rank with Hysteresis Objective Function. RFC 6719, September 2012.

[20] Jorge Granjal, Edmundo Monteiro, and Jorge Sá Silva. Security for the internet of things: A survey of existing protocols and open research issues. *IEEE Communications Surveys & Tutorials*, pages 1–1, 07 2015.

[21] Mukesh Gupta and Alex Conta. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. RFC 4443, March 2006.

[22] Jose A. Gutierrez, Edgar H. Callaway, and Raymond Barrett. *IEEE 802.15.4 Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensor Networks*. IEEE Standards Office, USA, 2003.

[23] Hon Sun Chiu and King-Shan Lui. Delphi: wormhole detection mechanism for ad hoc wireless networks. In *2006 1st International Symposium on Wireless Pervasive Computing*, pages 6 pp.–6, 2006.

[24] Y. . Hu, A. Perrig, and D. B. Johnson. Packet leashes: a defense against wormhole attacks in wireless networks. In *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, volume 3, pages 1976–1986 vol.3, 2003.

[25] Zakira Inayat, Abdullah Gani, Nor Badrul Anuar, Muhammad Khurram Khan, and Shahid Anwar. Intrusion response systems: Foundations, design, and challenges. *Journal of Network and Computer Applications*, 62:53–74, 2016.

[26] IPv6.br. Cabeçalho ipv6, May 2012.

[27] Mohit Jain and Himanshu Kandwal. A survey on complex wormhole attack in wireless ad hoc networks. In *Proceedings of the 2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies*, ACT '09, page 555–558, USA, 2010. IEEE Computer Society.

[28] Shiyu Ji, Tingting Chen, Sheng Zhong, and Subhash Kak. Dawn: Defending against wormhole attacks in wireless network coding systems. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 664–672. IEEE, 04 2014.

[29] Michael Johnson, Michael Healy, Pepijn Van de Ven, Martin J Hayes, John Nelson, Thomas Newe, and Elfed Lewis. A comparative review of wireless sensor network mote technologies. In *SENSORS, 2009 IEEE*, pages 1439–1442. IEEE, 2009.

[30] Kamaldeep, M. Dutta, and J. Granjal. Towards a secure internet of things: A comprehensive study of second line defense mechanisms. *IEEE Access*, 8:127272–127312, 2020.

[31] Patrick Olivier Kamgueu, Emmanuel Nataf, and Thomas Djotio Ndie. Survey on RPL enhancements: A focus on topology, security and mobility. *Computer Communications*, 120:10–21, 2018.

[32] P. Kasinathan, C. Pastrone, M. A. Spirito, and M. Vinkovits. Denial-of-service detection in 6lowpan based internet of things. In *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 600–607, 2013.

[33] Prabhakaran Kasinathan, Gianfranco Costamagna, Hussein Khaleel, Claudio Pastrone, and Maurizio A Spirito. Demo: An ids framework for internet of things empowered by 6lowpan. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1337–1340, 11 2013.

[34] Mohamed Rawidean Mohd Kassim. Iot applications in smart agriculture: Issues and challenges. In *2020 IEEE Conference on Open Systems (ICOS)*, pages 19–24, 2020.

[35] F. I. Khan, Taeshik Shon, Taekkyeun Lee, and Kihyung Kim. Wormhole attack prevention mechanism for rpl based lln network. In *2013 Fifth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 149–154, 2013.

[36] S. Khobragade and P. Padiya. Detection and prevention of wormhole attack based on delay per hop technique for wireless mobile ad-hoc network. In *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*, pages 1332–1339, 2016.

[37] Gu-Hsin Lai. Detection of wormhole attacks on ipv6 mobility-based wireless sensor network. *EURASIP Journal on Wireless Communications and Networking*, 2016, 11 2016.

[38] Ulf Lamping and Ed Warnicke. Wireshark user's guide. *Interface*, 4(6), 2004.

[39] A. Le, J. Loo, A. Lasebae, A. Vinel, Y. Chen, and M. Chai. The impact of rank attack on network topology of routing protocol for low-power and lossy networks. *IEEE Sensors Journal*, 13(10):3685–3692, 2013.

[40] A. Le, J. Loo, Y. Luo, and A. Lasebae. Specification-based ids for securing rpl from topology attacks. In *2011 IFIP Wireless Days (WD)*, pages 1–3, 2011.

[41] A. Le, J. Loo, Y. Luo, and A. Lasebae. The impacts of internal threats towards routing protocol for low power and lossy network performance. In *2013 IEEE Symposium on Computers and Communications (ISCC)*, pages 000789–000794, 2013.

[42] Anhtuan Le, Jonathan Loo, Kok Chai, and Mahdi Aiash. A specification-based ids for detecting attacks on rpl-based network topology. *Information*, 7(2):25, May 2016.

[43] Anhtuan Le, Jonathan Loo, A. Lasebae, Mahdi Aiash, and Yuan Luo. 6lowpan: A study on qos security threats and countermeasures using intrusion detection system approach. *International Journal of Communication Systems*, 09 2012.

[44] Jung Lee, Kyung Kim, and Hee Youn. Enhancement of congestion control of constrained application protocol/congestion control/advanced for internet of things environment. *International Journal of Distributed Sensor Networks*, 12, 11 2016.

[45] Chui Yew Leong, Thinagaran Perumal, Kwan Wei Peng, and Razali Yaakob. Enabling indoor localization with internet of things (iot). In *2018 IEEE 7th Global Conference on Consumer Electronics (GCCE)*, pages 571–573. IEEE, 2018.

[46] Olga León, J. Hernández-Serrano, and Miguel Soriano. Securing cognitive radio networks. *International Journal of Communication Systems*, 23:633 – 652, 05 2010.

[47] Jie Lin, Wei Yu, Nan Zhang, Xinyu Yang, Hanlin Zhang, and Wei Zhao. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal*, PP:1–1, 03 2017.

[48] IHS Markit. The internet of things: a movement, not a market. *IHS Market*, 2017.

[49] Anthea Mayzaud, Remi Badonnel, and Isabelle Chrisment. A taxonomy of attacks in rpl-based internet of things. *International Journal of Network Security*, 18(3):459–473, 2016.

[50] Konstantin Mikhaylov, Nikolaos Plevritakis, and Jouni Tervonen. Performance analysis and comparison of bluetooth low energy with ieee 802.15. 4 and simpliciti. *Journal of Sensor and Actuator Networks*, 2(3):589–613, 2013.

[51] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10, 09 2012.

[52] I. Mishra, D. Sharma, and S. Jain. A detailed classification of routing attacks against rpl in internet of things. *International Journal of Advance Research, Ideas and Innovations in Technology*, 3, 2017.

[53] Gabriel Montenegro, Jonathan Hui, David Culler, and Nandakishore Kushalnagar. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944, September 2007.

[54] Jirapond Muangprathub, Nathaphon Boonnam, Siriwan Kajornkasirat, Narongsak Lekbangpong, Apirat Wanichsombat, and Pichetwut Nillaor. Iot and agriculture data analysis for smart farm. *Computers and Electronics in Agriculture*, 156:467–474, 01 2019.

[55] M. N. Napiah, M. Y. I. Bin Idris, R. Ramli, and I. Ahmedy. Compression header analyzer intrusion detection system (cha - ids) for 6lowpan communication protocol. *IEEE Access*, 6:16623–16638, 2018.

[56] Doohwan Oh, Deokho Kim, and Won Woo Ro. A malicious pattern detection engine for embedded security systems in the internet of things. *Sensors (Basel, Switzerland)*, 14:24188–24211, 12 2014.

[57] Fredrik Osterlind and Adam Dunkels. Contiki cooja hands-on crash course: Session notes. *Sics. Se,(July)*, 2009.

[58] Pavan Pongle. Real time intrusion and wormhole attack detection in internet of things. *International Journal of Computer Applications*, 121:1–9, 07 2015.

[59] Jon Postel et al. Internet protocol, 1981.

[60] R. A. Rahman and B. Shah. Security analysis of iot protocols: A focus in coap. In *2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC)*, pages 1–7, 2016.

[61] V. K. Raju and K. V. Kumar. A simple and efficient mechanism to detect and avoid wormhole attacks in mobile ad hoc networks. In *2012 International Conference on Computing Sciences*, pages 271–275, 2012.

[62] P P Ray. A survey on Internet of Things architectures. *Journal of King Saud University - Computer and Information Sciences*, 30(3):291–319, 2018.

[63] Shahid Raza, Linus Wallgren, and Thiemo Voigt. Svelte: Real-time intrusion detection in the internet of things. *Ad Hoc Networks*, 11, 05 2013.

[64] Anass Rghioui and Anass Khannous. Denial-of-service attacks on 6lowpan-rpl networks: Issues and practical solutions. *Journal of Advanced Computer Science & Technology*, 3:143, 09 2014.

[65] Jeferson Rodrigues Cotrim, Estefânia Arata, and João Kleinschmidt. *Roteamento para Internet das Coisas-Protocolos, Mobilidade e Segurança.* 09 2017.

[66] Mohd Ezanee Rusli, Mohammad Ali, Norziana Jamil, and Marina Md Din. An improved indoor positioning algorithm based on rssi-trilateration technique for internet of things (iot). In *2016 International Conference on Computer and Communication Engineering (ICCCE)*, pages 72–77. IEEE, 2016.

[67] C. Samuel, B. M. Alvarez, E. Garcia Ribera, P. P. Ioulianou, and V. G. Vassilakis. Performance evaluation of a wormhole detection method using round-trip times and hop counts in rpl-based 6lowpan networks. In *2020 12th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, pages 1–6, 2020.

[68] Anuj Sehgal. Using the contiki cooja simulator. *Computer Science, Jacobs University Bremen Campus Ring*, 1:28759, 2013.

[69] Mohamed AM Seliem, Khaled MF Elsayed, and Ahmed Khattab. Performance evaluation and optimization of neighbor discovery implementation over contiki os. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pages 119–123. IEEE, 2014.

[70] Zach Shelby, Klaus Hartke, and Carsten Bormann. The Constrained Application Protocol (CoAP). RFC 7252, June 2014.

[71] Jeetendra Shenoy and Yogesh Pingle. Iot in agriculture. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 1456–1458, 2016.

[72] Dharmini Shreenivas, Shahid Raza, and Thiemo Voigt. Intrusion detection in the rpl-connected 6lowpan networks. In *Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security*, IoTPTS '17, page 31–38, New York, NY, USA, 2017. Association for Computing Machinery.

[73] P. Shukla. Ml-ids: A machine learning approach to detect wormhole attacks in internet of things. In *2017 Intelligent Systems Conference (IntelliSys)*, pages 234–240, 2017.

[74] Saurabh Singh, Pradip Kumar Sharma, Seo Yeon Moon, and Jong Hyuk Park. Advanced lightweight encryption algorithms for iot devices: survey, challenges and solutions. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–18, 2017.

[75] Sharwari Solapure and Harish Kenchannavar. Design and analysis of rpl objective functions using variant routing metrics for iot applications. *Wireless Networks*, 26, 08 2020.

[76] Biljana Risteska Stojkoska, Ivana Nižetić Kosović, and Tomislav Jagušt. How much can we trust rssi for the iot indoor location-based services? In *2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–6. IEEE, 2017.

[77] D. H. Summerville, K. M. Zach, and Y. Chen. Ultra-lightweight deep packet anomaly detection for internet of things devices. In *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*, pages 1–8, 2015.

[78] M. Surendar and A. Umamakeswari. Indres: An intrusion detection and response system for internet of things with 6lowpan. In *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 1903–1908, 2016.

[79] Tzeta Tsao, Roger Alexander, Mischa Dohler, Vanesa Daza, Angel Lozano, and Michael Richardson. A Security Threat Analysis for the Routing Protocol for Low-Power and Lossy Networks (RPLs). RFC 7416, January 2015.

[80] Jean-Philippe Vasseur and Adam Dunkels. *Interconnecting Smart Objects with IP: The Next Internet*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2010.

[81] A Velinov and A Mileva. Running and Testing Applications for Contiki OS Using Cooja Simulator. *International Conference on Information Technology and Development of Education*, 01(August):279–285, 2016.

[82] A. Verma and V. Ranga. Security of rpl based 6lowpan networks in the internet of things: A review. *IEEE Sensors Journal*, 20(11):5666–5690, 2020.

[83] Linus Wallgren, Shahid Raza, and Thiemo Voigt. Routing attacks and countermeasures in the rpl-based internet of things. *International Journal of Distributed Sensor Networks*, 2013, 08 2013.

[84] Libin Wang, Haitao Lan, Dan Guo, qingfeng zhang, David Roland-Holst, and Jan Hinrichs. *Internet Plus Agriculture: A New Engine for Rural Economic Growth in the People's Republic of China*. Asian Development Bank, 10 2018.

Apêndices

Esta página foi propositadamente deixada em branco.

Resultados auxiliares para avaliação da capacidade de deteção de ataques e identificação de nós maliciosos

Ataque Wormhole por encapsulamento				
Cenário	**1**	**2**	**3**	**Total**
Nº de atacantes	2	6	10	18
Verdadeiros Positivos	2	5	9	16
Verdadeiros Negativos	8	14	20	42
Falsos Positivos	0	0	0	0
Falsos Negativos	0	1	1	2
Taxa de Falsos Positivos	0	0	0	0
Taxa de Falsos Negativos	0	0.17	0.10	0.11
Taxa de Verdadeiros Negativos	1	1	1	1
Precisão	1	1	1	1
Recall	1	0.83	0.9	0.89
Exatidão	1	0.95	0.97	0.97
F-Measure	1	90.91	94.74	94.12

Tabela 1: Capacidade de identificação de atacantes envolvidos no ataque Wormhole do tipo por encapsulamento

Ataque Wormhole por retransmissão de pacotes				
Cenário	**1**	**2**	**3**	**Total**
Nº de atacantes	2	4	5	11
Verdadeiros Positivos	2	4	5	11
Verdadeiros Negativos	8	16	25	49
Falsos Positivos	0	0	0	0
Falsos Negativos	0	0	0	0
Taxa de Falsos Positivos	0	0	0	0
Taxa de Falsos Negativos	0	0	0	0
Taxa de Verdadeiros Negativos	1	1	1	1
Precisão	1	1	1	1
Recall	1	1	1	1
Exatidão	1	1	1	1
F-Measure	100	100	100	100

Tabela 2: Capacidade de identificação de atacantes envolvidos no ataque Wormhole do tipo por retransmissão de pacotes

Ataque Wormhole por encapsulamento				
Cenário	**1**	**2**	**3**	**Total**
Nº de ataques	1	3	5	9
Verdadeiros Positivos	1	3	5	9
Falsos Positivos	0	0	0	0
Falsos Negativos	0	0	0	0
Taxa de Falsos Positivos	0	0	0	0
Taxa de Falsos Negativos	0	0	0	0
Taxa de Verdadeiros Negativos	1	1	1	1
Precisão	1	1	1	1
Recall	1	1	1	1
Exatidão	1	1	1	1
F-Measure	1	1	1	1

Tabela 3: Capacidade de deteção de ataques Wormhole do tipo por encapsulamento

Ataque Wormhole por retransmissão de pacotes				
Cenário	**1**	**2**	**3**	**Total**
Nº de ataques	2	4	5	11
Verdadeiros Positivos	2	4	5	11
Verdadeiros Negativos	8	16	25	49
Falsos Positivos	0	0	0	0
Falsos Negativos	0	0	0	0
Taxa de Falsos Positivos	0	0	0	0
Taxa de Falsos Negativos	0	0	0	0
Taxa de Verdadeiros Negativos	1	1	1	1
Precisão	1	1	1	1
Recall	1	1	1	1
Exatidão	1	1	1	1
F-Measure	1	1	1	1

Tabela 4: Capacidade de deteção de ataques Wormhole do tipo por retransmissão de pacotes

Output Powertrace

Powertrace (10 nós) - Inicio de simulação				
No C/IDS	**CPU**	**LPM**	**Transmit**	**Listening**
2	42349	1923157	1454	1756327
3	37175	1928679	1521	1931892
4	27672	1938152	732	1666672
5	38475	1927097	1020	1932401
6	30052	1935762	1033	1671759
7	37251	1928246	959	1932460
8	36057	1929479	732	1541743
9	27012	1938825	617	1932806
10	26460	1939603	548	1561641
11	32941	1933144	1246	1932430
Media	33544.4	1932214.4	986.2	1786013.1

Tabela 5: Output Powertrace no inicio de simulação para o cenário com 10 nós na presença do IDS

Powertrace (10 nós) - Após 30m de simulação				
No C/IDS	**CPU**	**LPM**	**Transmit**	**Listening**
2	1122743	57853940	16408	58757857
3	984550	57989333	15016	58934686
4	788929	58189580	4639	58679335
5	1087855	57887478	11549	58938509
6	845764	58131616	12732	58676504
7	1082104	57894798	11171	58939171
8	1037623	57937885	5073	58553690
9	792978	58185244	5035	58945105
10	787290	58189618	4829	58573863
11	919515	58060141	6654	58943298
Media	944935.1	58031963.3	9310.6	58794201.8

Tabela 6: Output Powertrace no fim da simulação para o cenário com 10 nós na presença do IDS

Powertrace (10 nós) - Inicio de simulação				
No S/IDS	**CPU**	**LPM**	**Transmit**	**Listening**
2	42192	1923339	1452	1756327
3	37024	1928841	1525	1931892
4	27404	1938306	732	1666672
5	38312	1927267	1020	1932400
6	29777	1935920	1032	1671758
7	37075	1928421	962	1932459
8	35880	1929653	731	1541744
9	26756	1938975	617	1932809
10	26224	1939754	550	1561640
11	32781	1933306	1243	1932431
Media	33342.5	1932378.2	986.4	1786013.2

Tabela 7: Output Powertrace no inicio da simulação para o cenário com 10 nós sem a presença do IDS

Powertrace (10 nós) - Após 30m de simulação				
No S/IDS	**CPU**	**LPM**	**Transmit**	**Listening**
2	1118297	57854989	16104	58757567
3	982514	57993387	14929	58934591
4	783738	58194232	4355	58679051
5	1084321	57888988	11139	58938096
6	841886	58131842	12487	58676274
7	1081156	57893446	10491	58938471
8	1034013	57939117	5072	58553695
9	786821	58190374	4611	58944668
10	781614	58195820	4604	58573645
11	918597	58060871	6656	58943298
Media	941295.7	58034306.6	9044.8	58793935.6

Tabela 8: Output Powertrace no fim da simulação para o cenário com 10 nós sem a presença do IDS

Powertrace (20 nós) - Inicio de simulação				
No C/IDS	**CPU**	**LPM**	**Transmit**	**Listening**
2	52844	1912649	3070	1754705
3	41339	1924496	2167	1931241
4	44305	1921251	1859	1665536
5	45879	1919783	1260	1932156
6	37019	1928811	1432	1671355
7	28691	1937114	1039	1932379
8	32637	1933207	1054	1541419
9	27606	1938211	858	1932563
10	47071	1918873	1904	1567709
11	26086	1939984	617	1933063
12	27608	1938432	733	1689181
13	27828	1938233	801	1900112
14	27187	1938885	711	1477768
15	21870	1943970	619	1900296
16	31778	1934285	712	1607045
17	22243	1943514	617	1867515
18	27233	1938802	652	1575873
19	36511	1929331	1021	1867108
20	21743	1944023	447	1413789
21	29962	1936095	620	1867517
Media	32872	1932997.45	1109.65	1751416.5

Tabela 9: Output Powertrace no inicio da simulação para o cenário com 20 nós na presença do IDS

Powertrace (20 nós) - Após 30m de simulação				
No C/IDS	**CPU**	**LPM**	**Transmit**	**Listening**
2	1250372	57723794	33299	58740632
3	1046998	57931302	23270	58926294
4	1158516	57815999	21772	58662019
5	1205980	57769276	8707	58941310
6	1010029	57967757	18450	58670771
7	831825	58145450	10815	58939404
8	968679	58010439	13038	58546479
9	812449	58166006	8661	58942174
10	1251619	57723825	17328	58568576
11	808749	58170168	7333	58942705
12	792069	58186271	4727	58701764
13	789002	58189118	4794	58912553
14	785153	58191116	4712	58490059
15	664583	58305937	4939	58912287
16	931485	58043581	7979	58616523
17	660733	58309450	4931	58879853
18	793001	58184988	4664	58588301
19	1083640	57889676	11778	58872564
20	664233	58305728	4445	58426516
21	941357	58038436	8662	58875810
Media	922523.6	58053415.9	11215.2	58757829.7

Tabela 10: Output Powertrace no fim da simulação para o cenário com 20 nós na presença do IDS

Powertrace (20 nós) - Inicio de simulação				
No S/IDS	**CPU**	**LPM**	**Transmit**	**Listening**
2	52671	1912836	3068	1754702
3	41169	1924667	2167	1931243
4	44132	1921438	1856	1665534
5	45695	1920017	1260	1932156
6	36846	1928985	1435	1671356
7	28495	1937268	1038	1932377
8	32459	1933395	1052	1541419
9	27514	1938197	857	1932565
10	46877	1919080	1898	1567702
11	25855	1940142	618	1933065
12	27383	1938591	735	1689182
13	27579	1938389	801	1900113
14	26896	1939049	710	1477765
15	21634	1944198	616	1900295
16	31606	1934436	712	1607048
17	22028	1943752	616	1867515
18	27012	1938961	649	1575875
19	36334	1929515	1019	1867108
20	21603	1944200	551	1413942
21	29793	1936267	619	1867517
Media	32679.05	1933169.15	1113.85	1751423.95

Tabela 11: Output Powertrace no inicio da simulação para o cenário com 20 nós na ausência do IDS

Powertrace (20 nós) - Após 30m de simulação				
No S/IDS	**CPU**	**LPM**	**Transmit**	**Listening**
2	1244466	57727779	33174	58740499
3	1047543	57931168	23218	58926254
4	1157929	57814765	21497	58661756
5	1205156	57772683	8362	58940956
6	1011196	57967139	18197	58670491
7	825984	58147357	10263	58938832
8	964104	58015164	12223	58545660
9	809522	58163218	7503	58940999
10	1251493	57723319	17195	58568463
11	802564	58173693	7253	58942612
12	786641	58191424	4436	58701467
13	785167	58193312	4647	58912406
14	781546	58196512	4705	58490052
15	662785	58311200	4894	58912247
16	928409	58047388	7504	58616046
17	656825	58315896	4537	58879482
18	787779	58189402	4511	58588149
19	1083408	57890693	11770	58872566
20	658055	58315758	4239	58426075
21	938908	58040826	8588	58875733
Media	919474	58056434.8	10935.8	58757537.3

Tabela 12: Output Powertrace no fim da simulação para o cenário com 20 nós na ausência do IDS

Powertrace (30 nós) - Inicio de simulação				
No C/IDS	**CPU**	**LPM**	**Transmit**	**Listening**
2	63462	1902235	3712	1754051
3	49310	1916521	3371	1930028
4	49122	1916672	1295	1666105
5	34459	1931379	1101	1932320
6	55104	1910553	2642	1670137
7	39132	1926389	1362	1932055
8	27759	1938025	893	1541581
9	27951	1937867	939	1932481
10	36311	1929481	792	1568821
11	22708	1943069	697	1932980
12	31176	1934888	733	1689181
13	26001	1940061	619	1900295
14	25774	1940278	549	1477929
15	25685	1940382	616	1900295
16	37331	1928405	952	1606804
17	27492	1938515	858	1867271
18	26711	1939324	652	1575873
19	37299	1928478	1100	1867029
20	21749	1944016	447	1413789
21	22216	1943556	620	1867516
22	25736	1940305	550	1369886
23	32205	1933838	961	1834415
24	21792	1943983	550	1368629
25	27987	1938049	799	1834575
26	32958	1933119	951	1557152
27	27385	1938626	860	1801749
28	38166	1927867	1774	1582178
29	21962	1943818	619	1801993
30	45742	1920026	2101	1685273
31	22183	1943590	618	1769213
Média	32762.2667	1933110.5	1124.43333	1721053.47

Tabela 13: Output Powertrace no inicio da simulação para o cenário com 30 nós na presença do IDS

Powertrace (30 nós) - Após 30m de simulação				
No C/IDS	**CPU**	**LPM**	**Transmit**	**Listening**
2	1484745	57488789	46267	58727973
3	1009723	57965750	16581	58932589
4	1361692	57614113	15234	58667821
5	950311	58026108	10257	58938788
6	1407670	57565974	10123	58678910
7	1562896	57411277	40339	58931854
8	809178	58165291	7648	58550506
9	1219133	57754466	100934	58919461
10	1078836	57895545	10528	58574915
11	757167	58220766	34054	58919901
12	909821	58070105	4453	58701476
13	784016	58194515	4641	58912401
14	781849	58196663	4683	58489932
15	782522	58195427	4522	58912670
16	1489716	57486107	9569	58618405
17	973974	58000523	57793	58879218
18	787231	58189951	4508	58588143
19	1081154	57892509	10480	58873493
20	985627	57986950	55130	58427257
21	657870	58315339	4466	58879844
22	780783	58197423	4438	58382446
23	934495	58040865	7541	58843642
24	788780	58184725	3843	58355432
25	782969	58195149	4669	58846869
26	1122626	57855797	59025	58537823
27	1103560	57873674	14534	58724733
28	1006523	57972486	18033	58581385
29	743411	58232860	29157	58737689
30	1143019	57832873	20041	58683694
31	660399	58313333	4535	58781491
Media	998056.533	57977845.1	20600.8667	58720025.4

Tabela 14: Output Powertrace no fim da simulação para o cenário com 30 nós na presença do IDS

Powertrace (30 nós) - Inicio de simulação				
No S/IDS	**CPU**	**LPM**	**Transmit**	**Listening**
2	63269	1902482	3711	1754050
3	49125	1916699	3373	1930028
4	48880	1916885	1295	1666104
5	34290	1931562	1099	1932318
6	54915	1910796	2643	1670136
7	38961	1926572	1361	1932052
8	27536	1938208	892	1541582
9	27845	1937861	937	1932482
10	36126	1929668	792	1568822
11	22495	1943347	696	1932982
12	31002	1935061	735	1689182
13	25783	1940225	617	1900296
14	25562	1940447	550	1477929
15	25444	1940549	614	1900294
16	37155	1928593	950	1606803
17	27244	1938689	858	1867271
18	26483	1939489	649	1575875
19	37116	1928675	1101	1867028
20	21603	1944200	551	1413942
21	21999	1943805	619	1867517
22	25489	1940475	549	1369885
23	32034	1934018	963	1834415
24	21578	1944260	549	1368630
25	27729	1938213	798	1834577
26	32758	1933302	952	1557149
27	27140	1938802	860	1801750
28	37997	1928064	1777	1582177
29	21749	1944083	617	1801993
30	45567	1920225	2102	1685277
31	21969	1943809	618	1769213
Média	32561.4333	1933302.13	1127.6	1721058.63

Tabela 15: Output Powertrace no inicio da simulação para o cenário com 30 nós na ausência do IDS

Powertrace (30 nós) - Após 30m de simulação				
No S/IDS	**CPU**	**LPM**	**Transmit**	**Listening**
2	1484870	57487873	45704	58727416
3	1012711	57960679	16998	58933034
4	1365019	57608586	15777	58668355
5	948838	58025916	10870	58939398
6	1282784	57692482	10121	58678907
7	1326031	57649439	17731	58909097
8	815917	58160148	8220	58551083
9	875268	58098067	30016	58847883
10	1080178	57897435	10924	58575323
11	750253	58224418	29967	58915716
12	912578	58067252	4729	58701753
13	787935	58190108	4800	58912546
14	785856	58190980	4835	58490071
15	785041	58192196	4535	58912675
16	1198378	57778208	5630	58614435
17	922519	58055212	5204	58826073
18	792600	58185389	4667	58588305
19	1080190	57896104	10858	58873886
20	752585	58218826	3275	58375302
21	660322	58309729	4574	58879931
22	783633	58194186	4276	58382271
23	936349	58038001	7993	58844100
24	743723	58231045	29932	58381633
25	788422	58189219	4798	58846988
26	1171874	57804879	36271	58514986
27	1287805	57688961	93335	58804271
28	1003698	57973451	18722	58582080
29	994077	57975331	80780	58789497
30	1143957	57829965	19817	58683488
31	662238	58308515	4630	58781576
Media	971188.3	58004086.7	18332.9667	58717736

Tabela 16: Output Powertrace no fim da simulação para o cenário com 30 nós na ausência do IDS